Ministério da
Ciência e Tecnologia

INPE-15346-TDI/1382

USO DE HONEYPOTS PARA O ESTUDO DE SPAM E PHISHING

Klaus Steding-Jessen

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Antonio Montes Filho e Nandamudi Lankalapalli Vijaykumar, aprovada em 8 de julho de 2008

Registro do documento original:
<http://urlib.net/sid.inpe.br/mtc-m18@80/2008/08.18.19.02>

INPE
São José dos Campos
2008

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6911/6923
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

Ministério da
Ciência e Tecnologia

INPE-15346-TDI/1382

USO DE HONEYPOTS PARA O ESTUDO DE SPAM E PHISHING

Klaus Steding-Jessen

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Antonio Montes Filho e Nandamudi Lankalapalli Vijaykumar, aprovada em 8 de julho de 2008

Registro do documento original:
<http://urlib.net/sid.inpe.br/mtc-m18@80/2008/08.18.19.02>

INPE
São José dos Campos
2008

Dados Internacionais de Catalogação na Publicação (CIP)

S31u Steding-Jessen, Klaus.
Uso de honeypots para o estudo de Spam e phishing / Klaus Steding-Jessen. – São José dos Campos: INPE, 2008.
204p. ; (INPE-15346-TDI/1382)

Tese (Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2008.

1. Honeypots. 2. Spam. 3. Proxies abertos. 4. Phishing. 5. Estudo. I.Título.

CDU 004.056:004.773

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de Doutor(a) em

Computação Aplicada

Dr. Stephan Stephany

Presidente / INPE / SJCampos - SP

Dr. Antonio Montes Filho

Orientador(a) / CENPRA / Campinas - SP

Dr. Nandamudi Lankalapalli Vijaykumar

Orientador(a) / INPE / SJCampos - SP

Dr. Rafael Duarte Coelho dos Santos

Membro da Banca / INPE / SJCampos - SP

Dr. Alberto Camilli

Convidado(a) / USP / São Paulo - SP

Dr. Paulo Sergio da Motta Pires

Convidado(a) / UFRN / Natal - RN

Aluno (a): Klaus Steding-Jessen

São José dos Campos, 08 de julho de 2008

"As idéias estão no chão, você tropeça e acha a solução."

Titãs, em "A Melhor Forma"

AGRADECIMENTOS

Ao longo deste trabalho várias pessoas foram fundamentais, tanto do ponto de vista pessoal como técnico/científico. Gostaria de registrar aqui o meu agradecimento:

À Cristine Hoepers, que sem dúvida foi fundamental em todo o período do doutorado. Sem ela, esse projeto não teria nem começado;

À minha Vó e a minha Mãe, por todo o seu apoio;

Ao Prof. Vijay, que prestou uma ajuda inestimável ao longo deste trabalho;

A todo o pessoal do CERT.br pelo entusiasmo e ajuda em várias etapas desse trabalho;

Aos membros do Consórcio Brasileiro de Honeypots;

Ao Prof. Antonio Montes, pela ajuda na criação da primeira *Honeynet* no Brasil, no INPE, no início de 2002;

Ao meu empregador, o NIC.br, por seu apoio ao permitir que eu conciliasse trabalho com o doutoramento;

e a todos que, direta ou indiretamente, me ajudaram em todos esses anos.

RESUMO

Este trabalho propõe uma infra-estrutura extensível de sensores, baseada em *honeypots*, para estudar o problema do *spam* e do *phishing*, de modo a obter dados mais detalhados sobre o problema. Esta infra-estrutura permite a correlação desses dados com aqueles capturados por outros sensores, também com base em *honeypots*. Um protótipo desta infra-estrutura foi implementado e teve enfoque em obter dados sobre o abuso de *relays* e *proxies* abertos, a obtenção de endereços de *email* em *sites* Internet, a coleta de URLs enviadas através de mensagens de *pop-up* e a correlação de todos estes dados com atividades relacionadas com *spam*, capturadas pelo Consórcio Brasileiro de *Honeypots*. Este protótipo esteve em operação por diversos meses e coletou dados sobre vários aspectos do problema do *spam*, permitindo a obtenção de um conjunto de métricas que auxiliam a compreensão da situação no Brasil. Os resultados da operação deste protótipo mostram a intensidade do abuso de *relays* e *proxies* abertos em redes brasileiras, a origem e o destino destes *spams*, os indícios de envio a partir de máquinas infectadas e as características do *harvesting* de endereços de *email*. Como resultado da análise destes dados são apresentadas propostas de mitigação para os problemas observados.

USE OF HONEYPOTS FOR THE STUDY OF SPAM AND PHISHING

ABSTRACT

This work presents an extensible honeypot-based infrastructure to study the spam and phishing problem in order to obtain more detailed data on it. This infrastructure allows the correlation of the former data with data captured by other sensors also based on honeypots. A prototype of this infrastructure was implemented with the aim of obtaining data about the following: abuse of open relays and open proxies, email address harvesting, pop-up spam, and the correlation of these data with spam-related activities captured by the Brazilian Honeypots Alliance. This prototype was in operation for several months and collected data on several aspects of the spam problem. This allowed the generation of metrics to help understand the spam problem in Brazil. The obtained results show the magnitude of open relays and open proxies abuse in Brazilian networks, the source and the destination of these spams, the evidence of spam being sent from infected computers, and the characteristics of email harvesting. As a result of the analysis, some mitigation techniques for the observed problems are proposed.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

ADSL	–	*Asymmetric Digital Subscriber Line*
AfriNIC	–	*African Network Information Centre*
APNIC	–	*Asia Pacific Network Information Centre*
ARIN	–	*American Registry for Internet Numbers*
ARP	–	*Address Resolution Protocol*
AS	–	*Autonomous System*
ASN	–	*Autonomous System Number*
BCP	–	*Best Current Practice*
BGP	–	*Border Gateway Protocol*
CC	–	*Country Code*
CERT	–	*Service Mark* da *Carnegie Mellon University*, usada pelo CERT *Coordination Center*
CERT.br	–	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CERT/CC	–	CERT *Coordination Center*
CGI.br	–	Comitê Gestor da Internet no Brasil
CIDR	–	*Classless Inter-Domain Routing*
CSIRT	–	*Computer Security Incident Response Team*
DF	–	*Don't Fragment*
DMZ	–	*Demilitarized Zone*
DNS	–	*Domain Name System*
DNSBL	–	*DNS Blacklist*
DPN	–	Domínio de Primeiro Nível
DSL	–	*Digital Subscriber Line*
ESP	–	*Email Service Provider*
GMT	–	*Greenwich Mean Time*
gTLD	–	*Generic Top-Level Domain*
HTML	–	*HyperText Markup Language*
HTTP	–	*HyperText Transfer Protocol*
HTTP:BL	–	*HTTP Black List*
IAB	–	*Internet Architecture Board*
IANA	–	*Internet Assigned Numbers Authority*
IEEE	–	*Institute of Electrical and Electronics Engineers*
IESG	–	*The Internet Engineering Steering Group*
IETF	–	*The Internet Engineering Task Force*
IP	–	*Internet Protocol*
IPv4	–	*Internet Protocol Version 4*
IPv6	–	*Internet Protocol Version 6*
ISP	–	*Internet Service Provider*

LACNIC – *Latin American and Caribbean Internet Addresses Registry*
LAN – *Local Area Network*
MAAWG – *Messaging Anti-Abuse Working Group*
MD5 – *Message-Digest Algorithm 5*
MDA – *Mail Delivery Agent*
MSA – *Mail Submission Agent*
MTA – *Mail Transfer Agent*
MUA – *Mail User Agent*
MX – *Mail Exchange*
NIC.br – Núcleo de Informação e Coordenação do Ponto BR
NTP – *Network Time Protocol*
PHP – *PHP Hypertext Preprocessor*
RFC – *Request for Comments*
RIPE NCC – *RIPE Network Coordination Centre*
RIR – *Regional Internet Registry*
SHA-1 – *Secure Hash Algorithm 1*
SMTP – *Simple Mail Transfer Protocol*
SPF – *Sender Policy Framework*
SQL – *Structured Query Language*
SSH – *Secure Shell*
SYN – *Synchronize Control Flag*
TCP – *Internet Transmission Control Protocol*
TTL – *Time to Live*
UDP – *Internet User Datagram Protocol*
URI – *Uniform Resource Identifier*
URL – *Uniform Resource Locator*

1 INTRODUÇÃO

O *spam* é uma das formas de abuso da Internet que mais tem crescido, atualmente sendo responsável por uma parte significativa do volume de *email* trafegando na rede (HAYES, 2003; MAAWG, 2008). Além disso, o *spam* tem sido amplamente utilizado para enviar mensagens relacionadas com *phishing*, isto é, que procuram induzir o usuário ao fornecimento de dados pessoais e financeiros, além da disseminação de códigos maliciosos (MILLETARY, 2005; SHOSTACK; STEWART, 2008).

O termo *spam* é utilizado para descrever o envio indiscriminado de mensagens não solicitadas ou com conteúdo inapropriado, especialmente em grandes quantidades e com teor comercial (SHIREY, 2007; HAMBRIDGE; LUNDE, 1999). A popularização deste termo para denotar o envio em massa de *emails* não solicitados remonta a um incidente ocorrido em abril de 1994, quando uma mensagem não solicitada foi enviada, simultaneamente, para seis mil grupos de *Usenet News* (ZAKON, 1997; HAYES, 2003), embora a preocupação com o recebimento de *emails* não solicitados já data de 1975 (POSTEL, 1975).

Com a popularização da Internet, o acesso ao *email*, inicialmente limitado a comunidades mais restritas, como a comunidade acadêmica, aumentou significativamente entre o público em geral. E, como o custo para o envio de *email* é muito baixo, comparado ao da correspondência convencional, isto tem servido de incentivo para o envio de *emails* comerciais em grandes quantidades (CERF, 2005; SHOSTACK; STEWART, 2008).

Atualmente, os métodos amplamente usados na tentativa de reduzir o recebimento de *spams* envolvem uma ou mais das seguintes ações: bloqueio do *spams* na entrada da rede local, por exemplo através de listas de bloqueio; uso de *software* de filtragem de *spam*; e criação e aplicação de leis anti-*spam* (KOLSTAD, 2005; XIE et al., 2006). Porém, dado o volume que o spam ainda representa em relação às mensagens legítimas na Internet, essas técnicas não parecem estar sendo muito efetivas (MAAWG, 2008; KOLSTAD, 2005).

Existe, também, um aumento da sofisticação dos *softwares* de envio de *spam*, o que torna as técnicas existentes de bloqueio ainda menos eficientes e a rastreabilidade do *spammer* mais difícil (CRANOR; LAMACCHIA, 1998; MILLETARY, 2005). Um exemplo disso é o crescimento na utilização de máquinas infectadas por códigos maliciosos,

como *bots*, para o envio de *spam* e *phishing* (MILLETARY, 2005; PATHAK et al., 2008).

Embora exista a percepção de que o *spam* hoje corresponde a uma parcela significativa do total de *emails* em circulação e contribua para a disseminação de códigos maliciosos e *phishing*, a maioria dos números divulgados ainda provém de pesquisas informais ou de empresas que vendem produtos ou serviços anti-*spam* (HAYES, 2003). Um dos maiores desafios para a criação de mecanismos efetivos para a mitigação do problema ainda é a ausência de dados mais precisos sobre sua dimensão e sobre os mecanismos utilizados para a disseminação de *spam* e de códigos maliciosos por *email*.

Este trabalho utiliza *honeypots* para estudar o problema do *spam* e do *phishing*, com o objetivo de obter dados sobre os diversos aspectos, como: o abuso de *relays* e *proxies* abertos, tradicionalmente usados para o envio de *spam* e outras atividades maliciosas (HOEPERS et al., 2003); e o abuso de máquinas de usuários finais, principalmente aquelas conectados via banda larga, que estão vulneráveis e tem o potencial de serem infectadas por códigos maliciosos e controladas remotamente para o envio de *spam* (IANELLI; HACKWORTH, 2005; SAUVER, 2005; HOLZ, 2005). Mais especificamente, é definida uma infra-estrutura extensível de sensores, baseada em *honeypots*, para a captura e análise de *emails* abusivos ou maliciosos, coletando dados sobre: o abuso de *relays* e *proxies* abertos; o abuso de aplicações *Web* para envio de *spam*; o uso de *harvesting* pelos *spammers* e a efetividade de algumas técnicas de ofuscação de endereços de *email*; e o envio de *phishing* e *malware* via *email*. Além disso, esta infra-estrutura permite a correlação desses dados com aqueles capturados por outros sensores, também com base em *honeypots*.

Um protótipo desta infra-estrutura esteve em operação por alguns meses e coletou dados sobre diversos aspectos do problema do *spam*, permitindo a obtenção de um conjunto de métricas que auxiliam a compreensão da situação no Brasil. Os resultados da operação deste protótipo, discutidos em detalhes no capítulo 4, têm como destaque métricas sobre a intensidade do abuso de *relays* e *proxies* abertos em nossas redes, a origem e o destino destes *spams*, os indícios de envio a partir de máquinas infectadas e as características do *harvesting* de endereços de *email*.

Como resultado do trabalho desenvolvido nesta tese, podem-se destacar as seguintes contribuições:

- a implementação de mecanismos mais eficientes de coleta de *spam*, através do uso de *honeypots*, possibilitando o auxílio a outras iniciativas de estudo e combate ao *spam* e *phishing*;
- o estudo do problema do *spam* não apenas sob o ponto de vista do destinatário do *spam*, mas também em outros pontos de abuso da infra-estrutura de rede;
- o aumento da compreensão sobre o problema de coleta de endereços de *email* por *spammers* e o estudo da efetividade de algumas estratégias de ofuscação de endereços;
- o melhor entendimento sobre o abuso das redes brasileiras para o envio de *spam*;
- a proposta de estratégias de mitigação para os problemas observados.

Os demais capítulos desta tese estão organizados como segue. No capítulo 2 são apresentados conceitos que servem de base para o trabalho desenvolvido. Neste capítulo também são revistos alguns trabalhos envolvendo o estado da arte na área de estudo de *spam* e *phishing* através do auxílio de *honeypots*. No capítulo 3 é detalhado o trabalho proposto. No capítulo 4 são apresentados os resultados da análise dos *spams* coletados nos módulos implementados no protótipo. As conclusões deste trabalho, bem como possíveis trabalhos futuros, são discutidas no capítulo 5. As siglas usadas nesse trabalho são expandidas na página 21. Um glossário com alguns dos termos usados também está disponível, na página 107. Nos apêndices, iniciando na página 111, estão disponíveis artigos, publicados em revistas e congressos, decorrentes dessa tese, bem como uma listagem de palestras e minicursos ministrados. Também estão disponíveis os códigos fonte de alguns dos programas escritos para os módulos implementados no protótipo. O índice está disponível ao final do documento.

2 FUNDAMENTAÇÃO E TRABALHOS RELACIONADOS

Este capítulo apresenta conceitos que servem de base para o trabalho desenvolvido nesta tese. Serão definidos *honeypots* e seus tipos, discutido como os serviços de Internet são atualmente abusados para o envio de *spam* e descritos os métodos anti-*spam* mais difundidos. Também serão discutidos alguns trabalhos relacionados com esta tese, com enfoque em suas abordagens e nas limitações que apresentam.

2.1 *Honeypots*

Um *honeypot* é um recurso computacional de segurança, devidamente monitorado, que é dedicado a ser sondado, atacado ou comprometido (SPITZNER, 2002; PROVOS; HOLZ, 2007).

O valor dos *honeypots* baseia-se no fato de que tudo o que é observado é suspeito e potencialmente malicioso. Desse modo, são ferramentas de segurança com baixo número de falsos-positivos, que fornecem informações de alto valor e em um volume bem menor do que outras ferramentas tradicionais de segurança, como por exemplo *Intrusion Detection Systems* (IDS) (PROVOS; HOLZ, 2007; HONEYNET, 2004).

Do ponto de vista do nível de acesso que um atacante tem ao sistema e aos recursos disponíveis, os *honeypots* podem ser classificados em *honeypots* de alta interatividade e *honeypots* de baixa interatividade (PROVOS; HOLZ, 2007; HONEYNET, 2006).

2.1.1 *Honeypots* de Alta Interatividade

Nos *honeypots* de alta interatividade, os atacantes interagem com sistemas operacionais, aplicações e serviços reais. Uma vez atacados, esses sistemas são utilizados para observar o comportamento dos invasores, possibilitando análises detalhadas das ferramentas por eles utilizadas, de suas motivações e das vulnerabilidades exploradas.

Vários *honeypots* de alta interatividade podem ser combinados numa rede de *honeypots* denominada *honeynet* (PROVOS; HOLZ, 2007).

Um risco desse tipo de *honeypot*, e em menor grau, de *honeypots* em geral, é que uma vez comprometido ele seja utilizado pelo invasor para perpetrar atividades nocivas. Exemplos desse tipo de atividade incluem varreduras e comprometimentos de outras redes, envio de *spam* e *phishing* e hospedagem de material ilícito.

Para minimizar esse risco, *honeypots* de alta interatividade devem possuir mecanismos que impeçam, uma vez comprometidos, que sejam utilizados para atacar outras redes (HONEYNET, 2004; PROVOS; HOLZ, 2007). Tais mecanismos normalmente incluem técnicas de contenção de tráfego malicioso de saída tais como filtragem de pacotes, limitação de banda, limitação do número de sessões, modificação do conteúdo de pacotes, normalização e redirecionamento de tráfego (STEDING-JESSEN et al., 2003).

Esse tipo de *honeypot* é normalmente usado quando o objetivo é estudar o comportamento dos invasores, suas motivações, além de analisar em detalhes as ferramentas utilizadas e vulnerabilidades exploradas. Seu uso, contudo, demanda tempo, pessoal qualificado e técnicas de contenção eficientes (HONEYNET, 2006).

2.1.2 *Honeypots* de Baixa Interatividade

Em um *honeypot* de baixa interatividade são instaladas ferramentas para emular sistemas operacionais e serviços com os quais os atacantes ou os códigos maliciosos podem interagir. *Honeypots* deste tipo oferecem baixo risco de comprometimento e são indicados para redes de produção, quando não há pessoal ou *hardware* disponível para manter uma *honeynet* ou quando o risco de um *honeypot* de alta interatividade não é aceitável.

Usos típicos de *honeypots* de baixa interatividade incluem: identificação de varreduras e ataques automatizados, obtenção de assinaturas de ataques, identificação de tendências e coleta de códigos maliciosos (PROVOS; HOLZ, 2007).

Nesta tese, o objetivo foi a utilização de *honeypots* de baixa interatividade, que simulem o comportamento de computadores com problemas de configuração ou infectados por códigos maliciosos, e que sejam, conseqüentemente, abusados para envio de mensagens não desejadas.

2.2 Abuso de Serviços para o Envio de *Spam*

Esta seção descreve os métodos mais usados pelos *spammers* para abusar serviços legítimos ou máquinas mal configuradas para o envio de *spam* e para ofuscar a sua real origem.

2.2.1 Abuso de *Relays* Abertos

O protocolo SMTP (*Simple Mail Transfer Protocol*), normalmente associado à porta 25/TCP (*Transmission Control Protocol*), tem por objetivo a transmissão de *emails* de modo confiável e eficiente (KLENSIN, 2001). Um *relay* SMTP é um sistema que recebe um *email* de um cliente SMTP e o retransmite para outro servidor SMTP para entrega final ou para adicional retransmissão. Normalmente esse serviço de *relay* é seletivo, funcionando apenas para clientes autorizados.

Servidores SMTP mal configurados, comumente denominados de servidores com *relay* aberto, possibilitam a entrega de mensagens a partir de qualquer origem para quaisquer destinatários (LINDBERG, 1999). Esse tipo de servidor é abusado para o envio de *spams* por dificultar a detecção da real origem dos *emails*, além de poder ser usado como um mecanismo para burlar listas de bloqueio (PATHAK et al., 2008).

2.2.2 Abuso de *Proxies* Abertos

Um *proxy* é um servidor que atua como intermediário entre um cliente e outro servidor. Normalmente é utilizado para aumentar a performance de acesso a determinados serviços ou permitir que mais de uma máquina se conecte à Internet compartilhando um mesmo endereço IP (*Internet Protocol*). Desse modo, pode-se dizer que um serviço de *proxy* age como um intermediário, fazendo conexões em nome de outros clientes (FIELDING et al., 1999).

Quando um *proxy* está mal configurado, ele permite o redirecionamento indiscriminado de conexões de terceiros para quaisquer endereços IP e portas, sendo denominado *proxy* aberto. *Proxies* abertos são também intencionalmente instalados por códigos maliciosos, como *bots* e cavalos-de-tróia. Os dois protocolos mais usados para implementar *proxies* são o HTTP (FIELDING et al., 1999) e o SOCKS (LEECH et al., 1996).

Spammers continuamente realizam varreduras na Internet à procura de computadores que estejam com *proxies* abertos (KRAWETZ, 2004). Uma vez localizados, estes computadores são então explorados para efetuar conexões para os servidores SMTP dos destinatários do *spam*. Os *spammers* utilizam estes *proxies* abertos para obter anonimato (CHUVAKIN, 2003; KRAWETZ, 2004).

2.2.3 Entrega Direta de Mensagens

Spammers, fraudadores e códigos maliciosos abusam de computadores em redes residenciais, que estejam com *proxies* abertos, para entregar *emails* diretamente ao MTA (*Mail Transfer Agent*) do destinatário (MAAWG, 2005). A Figura 2.1 ilustra, em linhas tracejadas, como *spammers* subvertem o caminho legítimo de uma mensagem, que passaria pelo MTA do provedor do usuário, representado na Figura pelas linhas contínuas.

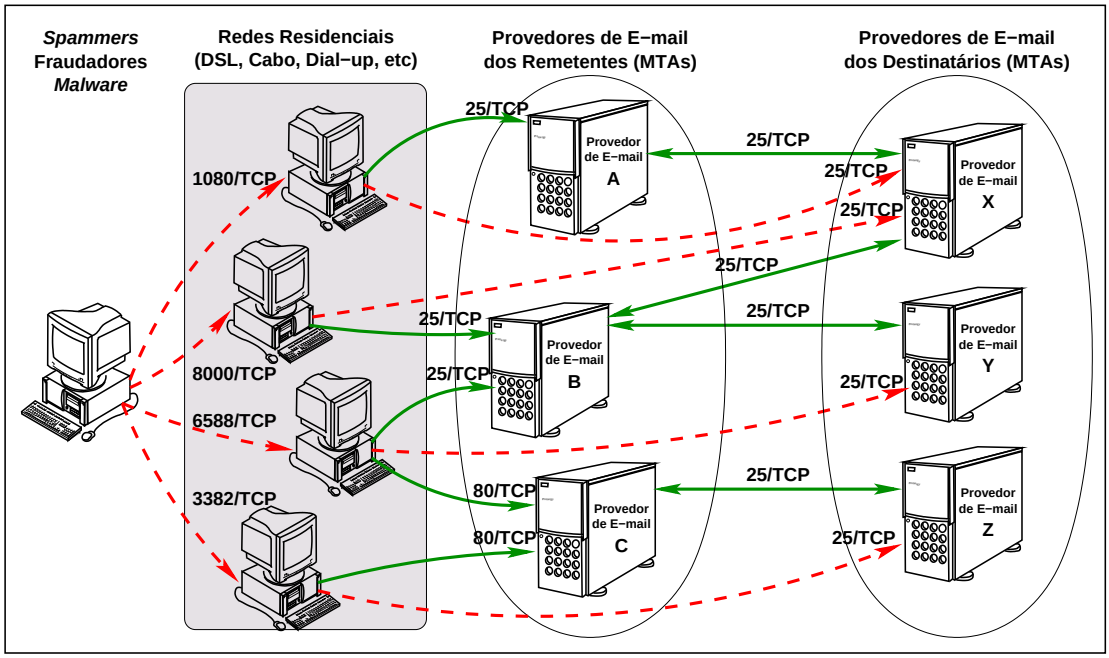


Figura 2.1 - Cenário de abuso de redes residenciais para entrega direta de *email*.

Ao conectar-se diretamente na porta 25/TCP do MTA de destino, o *spammer* burla mecanismos de controle de vazão, dificulta a filtragem de mensagens com base na origem ou no volume e, principalmente, dificulta o rastreio do *spam*.

2.3 Métodos Anti-Spam

Esta seção descreve inicialmente os métodos mais utilizados na tentativa de reduzir o recebimento de *spam*. Estes métodos são aplicados nas redes de destino do *spam* e podem ou não ser baseados em conteúdo da mensagem. Ao final, são apresentados alguns métodos que se destinam a evitar que o *spam* saia da rede de origem e são

propostas medidas que tornem economica ou legalmente inviável o seu envio (XIE et al., 2006; KOLSTAD, 2005).

2.3.1 Métodos Baseados em Conteúdo das Mensagens

Estes métodos baseiam-se no conteúdo das mensagens, o que inclui o cabeçalho e o corpo, para identificá-las como *spam* ou contendo conteúdo malicioso.

Filtros de Endereços de *Email* são filtros baseados em um conjunto de endereços de *email* que estão permitidos (*whitelist*) ou proibidos (*blacklist*) a enviar mensagens. Uma vez que endereços de *email* podem ser forjados, esta técnica pode ser facilmente burlada por *spammers*.

Filtros Heurísticos são filtros que tentam classificar mensagens em função de características que são raras em *emails* legítimos, mas comuns em *spams*. Tais características incluem presença de certas palavras e URLs, entre outras.

Filtros de Classificação de Texto são filtros que se valem de algoritmos de classificação de texto para tentar diferenciar *spam* de mensagens legítimas. Muito populares nessa categoria são os filtros Bayesianos.

2.3.2 Métodos Não Baseados em Conteúdo das Mensagens

Estes métodos baseiam-se em identificar características do remetente do *spam*, como endereços IP de origem, ou características relacionadas com a entrega das mensagens, como taxa de envio de *emails* ou violação dos padrões do protocolo SMTP.

DNSBLs são *blacklists* distribuídas, que mantém endereços IP originadores de *spam*, que podem ser acessadas através de consultas DNS. Essas consultas são normalmente efetuadas pelo MTA ao qual o *email* se destina, com o objetivo de verificar se o IP de origem da conexão está listado em alguma *blacklist*.

Registros de Autorização via DNS consiste em uma classe de técnicas que tenta identificar endereços de *email* forjados, utilizados em mensagens não solicitadas. Uma rede publica registros DNS informando quais são os MTAs autorizados a enviar *emails* em seu nome ou chaves de criptografia com as quais os cabeçalhos das mensagens são assinados. A rede que recebe uma

mensagem pode, então, verificar essas informações através de consultas DNS. Soluções nessa categoria incluem SPF (WONG; SCHLITT, 2006), *Domain Keys* (ALLMAN et al., 2007) e *Sender ID* (LYON; WONG, 2006).

Desafio-Resposta – neste método, mensagens oriundas de endereços que ainda não constem em uma *whitelist* são retornadas ao remetente, em conjunto com um desafio que deve ser resolvido. A idéia é que a solução do desafio não possa ser resolvida rapidamente de forma automatizada, o que obriga a intervenção de uma pessoa. Isto reduz a velocidade de envio, tornando inviável o envio de *spams* para redes que utilizem estas soluções. Após o recebimento de uma resposta correta, o endereço do remetente é então autorizado. Este procedimento não é amplamente utilizado por ter grande impacto na facilidade de uso, além dos problemas com relação à acessibilidade.

Greylisting – esse método se baseia na diferença entre o comportamento esperado de MTAs legítimos e de ferramentas de envio de *email* em massa. Um MTA legítimo, implementado de acordo com a definição do protocolo SMTP, possui filas para o reenvio de mensagens quando ocorre um erro de caráter temporário. Já muitas ferramentas de envio de *spam* e máquinas comprometidas por códigos maliciosos, possuem implementações simplificadas do protocolo SMTP e, geralmente, não tentam o reenvio caso a primeira tentativa não tenha êxito. A técnica de *greylisting*, implementada no MTA de destino, ao verificar que um endereço IP de origem está se conectando pela primeira vez, retorna um erro não definitivo como resposta. Caso o MTA de origem tente novamente, depois de um tempo mínimo pré-estabelecido, a mensagem é aceita e o endereço IP é colocado em uma *whitelist*.

Atraso na Recepção – este mecanismo procura identificar IPs de origem que estejam entregando mensagens com uma taxa de vazão muito elevada. Quando é detectada uma taxa de recebimento acima de valores pré-estabelecidos, o MTA de destino passa a atrasar a comunicação com o MTA de origem.

Análise do Comportamento da Conexão SMTP – técnicas que tentam separar *spams* de mensagens legítimas levando em conta o comportamento de conexões SMTP por emissor e sua aderência ao protocolo SMTP. Quaisquer comportamentos que sejam considerados maliciosos ou que não estejam de acordo com a definição do protocolo podem causar a rejeição da

mensagem.

2.3.3 Métodos de Prevenção na Origem

As técnicas descritas acima, tanto as baseadas em conteúdo das mensagens com as baseadas no emissor dessas mensagens, são técnicas aplicáveis no destino. Outra classe de técnicas, descritas a seguir, se destinam a limitar o envio de *spam* ainda na origem.

Gerência de Porta 25/TCP e *Mail Submission* – o protocolo SMTP foi concebido como um protocolo de transferência de mensagens entre servidores de *email*. Porém, tem sido usado também como protocolo de submissão de mensagens entre cliente e servidor. Uma proposta para diferenciar o tráfego de submissão (de um cliente de *email* para um MTA) do tráfego de mensagens entre MTAs é o protocolo de *Message Submisssion* (GELLENS; KLENSIN, 2006). Este protocolo fornece um meio para distinguir a submissão do transporte de mensagens, permitindo assim a aplicação de políticas diferentes para cada tipo de conexão. Uma política que pode ser adotada é o bloqueio, em redes residenciais, de conexões de saída com destino a porta 25/TCP, impedindo a entrega direta de *emails* a partir destas redes.

Custos por Mensagem – têm sido propostas técnicas anti-*spam* baseadas em custos por mensagem, que visam mover o custo de envio de *spam* do destinatário para o remetente. Essas técnicas assumem que o valor a ser pago pelo envio de *emails* por um usuário legítimo seria baixo, mas para um *spammer* o valor acumulado seria alto o suficiente para que o envio de *emails* em massa se tornasse economicamente inviável.

Legislação – nos últimos anos têm crescido o número de países que têm implantado alguma legislação anti-*spam* (BUETI, 2005; KOLSTAD, 2005). Estas leis procuram definir penalidades que coíbam as ações dos *spammers*. Entretanto, elas não parecem estar tratando do problema de forma adequada (BUETI, 2005).

O conjunto dos métodos anti-*spam* tradicionais, contudo, têm apresentado uma eficácia relativa no combate ao *spam*. A seção seguinte discute outras técnicas utilizadas no estudo do problema.

2.4 Trabalhos Relacionados

Nesta seção serão discutidos alguns trabalhos envolvendo o estado da arte na área de estudo de *spam* e *phishing* com o auxílio de *honeypots*, bem como discutidas as limitações de algumas abordagens.

2.4.1 Módulos de *Proxy* e *Relay* Aberto do Honeyd

O `Honeyd` é um *daemon* de rede que implementa um *framework* para a criação de *honeypots* virtuais, permitindo a emulação de diversos computadores, sistemas operacionais e serviços de rede (PROVOS, 2004). Este programa tem sido muito usado na implementação de *honeypots* de baixa interatividade.

De maior interesse para este trabalho é o seu uso para o estudo e melhoria de filtros anti-*spam*, através de dois módulos que acompanham o `Honeyd`:

Emulador de *Relay* Aberto – emula um servidor SMTP que aceita retransmitir qualquer mensagem. As mensagens, entretanto, são armazenadas e nunca entregues aos seus destinatários. Este módulo pode ser configurado para responder ao primeiro *email* enviado a partir de cada endereço IP. Isto visa possibilitar a resposta a *emails* de teste, porventura enviados por *spammers* para verificar se o *relay* realmente está funcionando.

Emulador de *Proxy* Aberto – emula o comportamento de um *proxy* aberto, implementando um subconjunto do protocolo HTTP (FIELDING et al., 1999), muito utilizados para implementar serviços de *proxy*. Requisições de uso do comando CONNECT, com destino à porta de SMTP de um servidor remoto são redirecionadas ao emulador de SMTP local, descrito no item anterior.

Utilizando estes dois módulos é possível capturar um grande número de *emails* para estudo e também submissão a projetos colaborativos de criação de assinaturas para filtragem de *spam* (PROVOS, 2004).

Uma limitação da utilização destes módulos está no mecanismo de resposta a *emails* de teste. Ao responder a todos os primeiros *emails* vindos de cada endereço IP, o módulo pode efetivamente entregar *spams* a destinatários, e não somente a endereços de teste sob controle do *spammer*. Adicionalmente, por não levar em conta outras

técnicas para identificar mensagens de teste, este método pode deixar de responder a *emails* de teste que sejam enviados em outros momentos de uma campanha de *spam*.

Outra limitação é o sistema possuir suporte somente a *proxies* que utilizem o protocolo HTTP, o que limita a captura de *spams* que abusam de outros protocolos, como o SOCKS (LEECH et al., 1996).

2.4.2 *HoneySpam*

HoneySpam é uma arquitetura, baseada em *honeypots*, implementada em uma rede corporativa como um recurso de identificação e bloqueio de *spam* (ANDREOLINI et al., 2005). Suas principais funcionalidades são:

Diminuição na Velocidade do Processo de *Harvesting* – o processo de *harvesting* é o processo de coleta de endereços de *email* públicos em *sites* na Internet. Como parte da arquitetura, são criadas páginas *Web* com um grande número de *links* entre si, com o objetivo de retardar o processo ao forçar o *spammer* a visitar repetidamente as mesmas páginas.

Identificação de Endereços IP Originadores de *Spam* – as páginas HTML geradas contém endereços de *email* criados dinamicamente, de um domínio criado especificamente para este fim. O domínio possui um emulador SMTP, do próprio sistema, que aceita mensagens destinadas a qualquer endereço. Assim que esses endereços começam a receber *spam*, essa informação pode ser usada para implementar listas de bloqueio aos IPs originando as mensagens. O outro objetivo de fornecer aos *spammers* endereços de *email* inválidos é poluir suas bases de dados de *emails*, diminuindo a sua utilidade.

Emulação de *Proxies* e *Relays* Abertos – o sistema usa os emuladores de *relay* aberto e *proxy* aberto HTTP, implementados pelo `Honeyd` (PROVOS, 2004) e descritos em mais detalhes na seção 2.4.1. Conexões a estes serviços emulados são registradas e essa informação é usada para efetuar bloqueio do tráfego vindo desses endereços IP.

Por utilizar o `Honeyd`, o *HoneySpam* possui as mesmas limitações discutidas na seção anterior. Adicionalmente, o sistema foi desenvolvido com um objetivo muito espe-

cífico: ser instalado em uma rede em particular para detectar e bloquear endereços IP originadores de *spam*. O bloqueio do tráfego de *spam* reduz a utilidade desta arquitetura para a sua captura e estudo.

2.4.3 Projeto *Honey Pot*

O projeto *Honey Pot* é um sistema distribuído para identificar *spammers* e os *spambots* por eles usados para fazer *harvesting* de *emails* em *websites* (PRINCE et al., 2005; HONEY POT, 2008). O principal objetivo do projeto é diminuir o *spam* através da diminuição do *harvesting* de endereços de *email*.

O sistema possui um *software* que é instalado em páginas *Web* por voluntários que se cadastram junto ao projeto. Este programa gera páginas com conteúdo único para cada acesso. Quando uma página gerada pelo sistema é acessada, informações relativas ao visitante, tais como endereço IP e campo "*referer*"[1], são transferidas para um servidor central do projeto. Este servidor, então, armazena essas informações e gera um endereço de *email* único, que é associados com o IP que fez o acesso, e então transferido de volta para o *site* do voluntário e exibido na página dinâmica. *Links* para estas páginas são inseridos em páginas legítimas dos voluntários, de modo que não sejam visíveis para visitantes regulares, mas sim para os programas que percorrem páginas *Web*, acessando o seu conteúdo e recursivamente seguindo todos os seus *links* (*crawlers*).

Deste modo, se os endereços de *email* gerados dinamicamente começarem a receber *spam*, é possível determinar qual IP fez o *harvesting*. Também é possível ter certeza que a mensagem é um *spam*, pois os endereços de *email* não tem outra finalidade a não ser seu uso como armadilhas, ou *spam traps*. Servidores SMTP do projeto são responsáveis por receber os *spams* direcionados a estes endereços de *email*.

Através da análise dos *spams* recebidos, o projeto tenta determinar:

- quais endereços IP realizam o processo de *harvesting* e quais enviam *spam* para os endereços coletados;
- o tempo médio entre o *harvesting* de endereços de *email* e o posterior recebimento de *spam*;

[1] A palavra "*referer*" é o termo utilizado no padrão HTTP (FIELDING et al., 1999), embora seja um erro de ortografia da palavra em inglês "*referrer*".

- a classificação do *spam* recebido:
  - *spam* "regular";
  - *spam* relacionado com fraude e *phishing*.
- a distribuição geográfica dos IPs envolvidos.

Os dados coletados são também compartilhados com grupos de pesquisa na área de *spam* e utilizados para auxiliar processos legais contra *spammers*. Adicionalmente, cada voluntário recebe uma análise das atividades de *harvesting* e dos *spams* recebidos em *emails* coletados em seu *site*.

Mais recentemente este projeto criou um novo serviço para seus membros: uma lista de bloqueio baseada em DNS, a `http:BL` (LANGHEINRICH, 2008). Ao contrário de listas de bloqueio tradicionais, normalmente utilizadas por servidores de *email*, a `http:BL` tem por objetivo ser usada por servidores *Web*. Para cada IP efetuando acesso, o servidor *Web* pode fazer uma consulta DNS para a `http:BL`, e caso o IP esteja listado como *harvester*, o servidor pode decidir em exibir o conteúdo de modo parcial ou ainda bloquear o acesso inteiramente.

Uma limitação deste projeto é que ele contempla apenas a identificação de *harvesting* em páginas HTML. Não há nenhum mecanismo para identificar *harvesting* em endereços de *email* contidos em arquivos de outros formatos.

Uma outra questão a se considerar é o impacto em termos de performance para usuários da lista `http:BL`. *Sites* com um grande volume de acesso podem ter degradação no tempo de exibição das páginas, devido à latência das consultas DNS. Também é importante considerar que, se o *harvesting* estiver sendo feito por uma máquina infectadas por um *bot*, um usuário legítimo pode ser bloqueado.

2.4.4 Comportamento de *Spammers* do Ponto de Vista de Rede

Em (RAMACHANDRAN; FEAMSTER, 2006) é descrito um estudo que observou 17 meses de tráfego de *spam* para um domínio criado exclusivamente para coletar *emails*, sem usuários legítimos. Os principais objetivos foram investigar se é possível identificar conexões de *spammers* somente do ponto de vista de comportamento de rede e capturar informações para auxiliar o desenvolvimento de filtros anti-*spam* mais efetivos.

Além da coleta dos *mails*, também foram armazenadas as informações sobre os anúncios de rotas BGP das redes de origem do *spam* e sobre o *fingerprint* passivo das conexões originadoras de *spam*, de modo a identificar o sistema operacional de origem. Todos esses dados foram comparados com traços de rede da infra-estrutura de comando e controle de uma *botnet*, conhecidamente usada para o envio de *spam*, e *logs* de *emails* considerados legítimos por um grande provedor de *email*.

Algumas conclusões deste estudo foram:

- a origem da grande maioria do *spam* recebido concentrava-se em porções reduzidas do espaço de endereçamento IP;
- a maioria do *spam* recebido originou-se em máquinas com sistema operacional Windows®;
- uma pequena parcela dos *spams* originou-se de blocos de rede que estão alocados mas não em uso e ligados a um número de AS não alocado. Estas redes tiveram anúncios de roteamento de curta duração que coincidiram com o recebimento de *spams* delas originados. Segundo os autores, isso levanta a suspeita de seqüestro (*hijacking*) desses endereços e ASs.

Uma limitação deste projeto é o fato da coleta de dados estar concentrada na rede destino do *spam*. Mesmo havendo a observação do tráfego de comando e controle de uma *botnet*, o estudo não obteve dados das características de rede associadas ao abuso de serviços para o envio de *spam*.

2.4.5 Comportamento de *Spammers* Visto a Partir de *Relays* Abertos

Em (PATHAK et al., 2008), os autores descrevem a implementação de um *honeypot* que emulou o comportamento de um *relay* aberto, com o objetivo de captura e estudo de *spam*. Este sensor coletou dados por um período de 3 meses e capturou 40 milhões de *spams* no período.

Os autores argumentam que técnicas tradicionais de captura de *spam* normalmente são implementadas no destino das mensagens, o que limita a visão dos dados a uma única organização. O método proposto, de observar o tráfego de *spams* num ponto mais intermediário da rede apresenta grandes vantagens, por possibilitar uma

visão mais ampla ao capturar *spams* de múltiplas origens e com destino a múltiplas organizações.

A partir da análise das mensagens capturadas, os autores dividem os originadores de tráfego em duas categorias:

***High-Volume Spammers* (HVS):** *spammers* originando uma grande quantidade de mensagens, a partir de um número pequeno de IPs distintos;

***Low-Volume Spammers* (LVS):** *spammers* controlando de maneira centralizada um grande número de máquinas, provavelmente comprometidas, cada uma enviando um volume pequeno de mensagens.

O trabalho define uma heurística que separa o tráfego gerado por HVSs e LVSs em função do número de conexões e do número de destinatários por conexão de um mesmo endereço IP de origem.

Uma grande limitação deste trabalho é observar apenas o tráfego em *relays* abertos, sem incluir *proxies* abertos.

2.4.6 Estudo de *Phishing* usando *Honeypots* de Alta Interatividade

Membros do *German Honeynet Project* e do *UK Honeynet Project* documentaram múltiplos ataques de *phishing* ocorridos em alguns de seus *honeypots* de alta interatividade (RESEARCH ALLIANCE, 2005).

Neste estudo, foi possível acompanhar as seguintes atividades dos invasores após o ataque e comprometimento dos *honeypots*:

- construção de *sites Web* usados para *phishing*, isto é, *sites* com páginas semelhantes a páginas de instituições financeiras conhecidas;
- tentativa de envio de *spam*, a partir do *honeypot* recém comprometido, com *links* para as páginas criadas;
- instalação de serviços de redireção de tráfego *Web* para outros *sites* de *phishing*;
- propagação de *spam* e mensagem relacionadas com *phishing* através do uso de *botnets*.

Neste estudo, o envio dos *emails* e outras atividades maliciosas relacionadas com a efetivação do *phishing* foram bloqueadas pelos mecanismos de contenção de tráfego malicioso de saída, empregado pela *honeynet* onde os *honeypots* estavam localizados.

Uma desvantagem da observação desta atividade em um *honeypot* de alta interatividade é o grande risco de algum mecanismo de contenção não funcionar e serem causados prejuízos a terceiros. Outra desvantagem, é o tempo necessário para o processamento da grande quantidade de informações capturadas que necessitam ser analisadas detalhadamente.

2.4.7 Resumo das Limitações dos Trabalhos na Área de Uso de *Honeypots* para o Estudo de *Spam* e *Phishing*

Os trabalhos discutidos neste capítulo, bem como outras iniciativas anteriores de estudo de *spam* com auxílio de *honeypots* (OUDOT, 2003), apresentam diversas limitações com relação ao modo como capturam os dados e a finalidade da sua coleta.

Várias das iniciativas tiveram enfoque em um único aspecto do problema de *spam*, como o estudo apenas de *harvesting* ou apenas do problema de *relays* ou *proxies* abertos. Mas, nenhum destes projetos observa mais de um problema ou correlaciona esses dados. Também não há estudos com dados sobre mensagens indesejadas que não as enviadas por *email*, como por exemplo, *spim* (*spam over instant messaging*) e *pop-up spam*.

A maioria dos estudos foi feita do ponto de vista do destinatário do *spam*, sem investigar o problema sob outro ponto de vista, seja mais próximo da origem ou em pontos de conhecido abuso da infra-estrutura de rede. Essa visão parcial limita a compreensão do problema como um todo. Os poucos projetos que estudaram pontos diferentes do problema o fizeram para tecnologias específicas e/ou por um período muito curto de tempo, o que pode não mostrar tendências de mudança nas técnicas de envio de *spam*, por exemplo.

Finalmente, algumas das iniciativas coletam dados iniciais sobre o problema, mas, como possuem um enfoque no bloqueio dos endereços de origem do *spam*, deixam de coletar dados valiosos para o estudo do problema.

De modo a tratar as limitações descritas hoje nesta área, este trabalho propõe uma infra-estrutura para o estudo de *spam* e *phishing*, baseada em *honeypots*, com o

objetivo de analisar o problema sob diversos pontos de vista, procurando entender como os *spammers* abusam da infra-estrutura da Internet para atingir seus objetivos.

O capítulo 3 descreve a infra-estrutura proposta para o estudo de *spam* e *phishing*, baseada em *honeypots*, e apresenta a implementação de um protótipo desta infra-estrutura, que teve enfoque nos problemas de: abuso de *relays* e *proxies* abertos; abuso de máquinas de usuários finais, potencialmente infectadas por códigos maliciosos e controladas remotamente para o envio de *spam*; e uso de *harvesting* pelos *spammers*, incluindo a efetividade de algumas técnicas de ofuscação de endereços de *email*.

3 USO DE *HONEYPOTS* PARA O ESTUDO DE *SPAM* E *PHISHING*

No capítulo 2 foram discutidas diversas abordagens atuais para o estudo de *spam* e *phishing* com o auxílio de *honeypots*. As principais limitações observadas foram o estudo dos problemas isoladamente e a coleta, na maioria das vezes, feita apenas nas redes de destino das mensagens.

Este trabalho utiliza *honeypots* para estudar o problema do *spam* e do *phishing*, com o objetivo de obter dados sobre os diversos aspectos envolvidos. O enfoque foi analisar o problema de diversos pontos de vista, procurando entender como os *spammers* abusam da infra-estrutura da Internet para atingir seus objetivos.

Este capítulo descreve a infra-estrutura proposta para o estudo de *spam* e *phishing*, baseada em *honeypots*, e apresenta a implementação de um protótipo desta infra-estrutura, que teve enfoque nos problemas de: abuso de *relays* e *proxies* abertos; abuso de máquinas de usuários finais, potencialmente infectadas por códigos maliciosos e controladas remotamente para o envio de *spam*; e uso de *harvesting* pelos *spammers*, incluindo a efetividade de algumas técnicas de ofuscação de endereços de *email*. O protótipo ficou em operação por diversos meses e os resultados obtidos são discutidos no capítulo 4.

3.1 Uma Infra-Estrutura para o Estudo de *Spam* e *Phishing*

Para possibilitar o estudo do problema do envio em massa de mensagens eletrônicas é necessário obter dados sobre as diversas fases e técnicas envolvidas no envio destas mensagens. Como visto nos capítulos anteriores, a maior parte dos trabalhos e das métricas disponíveis avaliam o *spam* apenas no momento em que ele chega ao seu destino, e não em outras fases do seu envio.

Este trabalho propõe uma infra-estrutura extensível, baseada em *honeypots*, para a captura e a análise de *emails* abusivos ou maliciosos em diferentes pontos da rede, com o enfoque em coletar dados sobre o problema do ponto de vista do abuso de serviços pelos *spammers*. Um diagrama da infra-estrutura proposta é exibido na Figura 3.1.

No escopo desta infra-estrutura um *honeypot* que não seja especificamente configurado para capturar *spam* foi denominado *honeypot* genérico. A sua utilização é importante para permitir a comparação de tráfego malicioso observado por estes

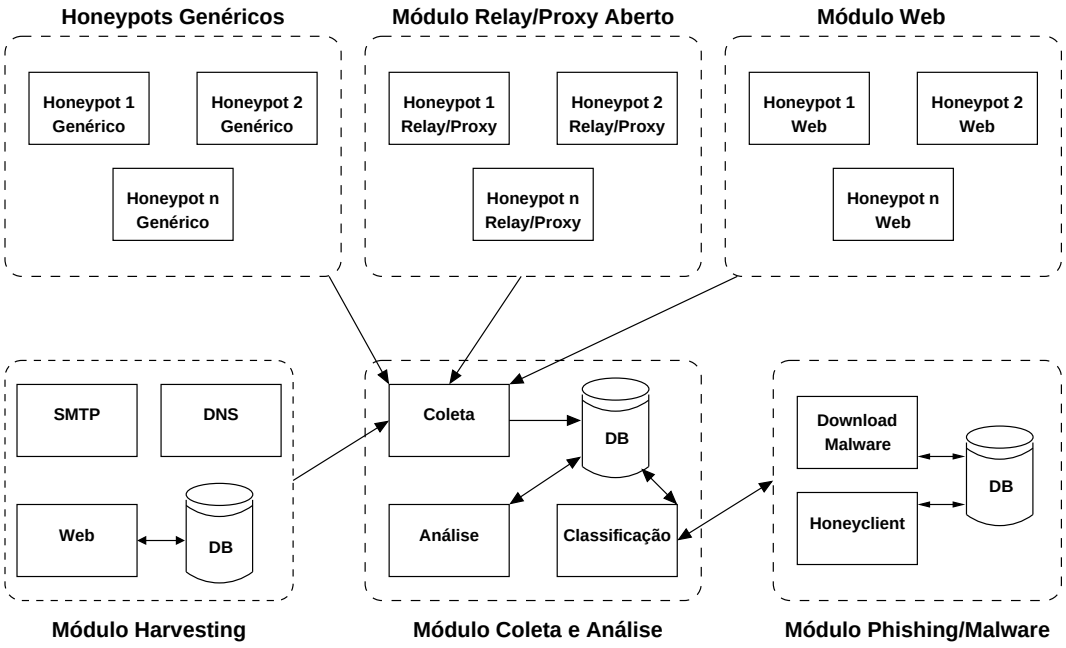


Figura 3.1 - Diagrama da arquitetura proposta.

honeypots com o tráfego observado em outros módulos que sejam específicos para captura de *spam*. Um exemplo seria averiguar se um endereço IP explorando um *proxy* no módulo *Relay/Proxy* também foi visto gerando atividades maliciosas em um ou mais *honeypots* genéricos distribuídos em diversos pontos da Internet. Os demais módulos da infra-estrutura são descritos a seguir.

Módulo *Relay/Proxy* Aberto

O objetivo desse módulo é capturar dados sobre o envio de *spam* através do abuso dos seguintes serviços: servidores SMTP mal configurados, comumente denominados de servidores com *relay* aberto; e servidores de *proxy* mal configurados ou que tenham sido instalados por *bots* e outros programas maliciosos, também chamados de *proxies* abertos. Para a implementação deste módulo são necessários:

- *honeypots* de baixa interatividade;
- emulação de um servidor SMTP (KLENSIN, 2001), capturando o tráfego da porta 25/TCP e simulando as respostas dadas por um servidor de SMTP. Para toda a mensagem recebida, deve comportar-se como um *relay* aberto e aceitar a mensagem. Porém, a mensagem deve ser

armazenada, sem ser entregue ao destinatário, evitando que o sensor envie *spam*;

- emulação de um *proxy* HTTP (FIELDING et al., 1999), configurado para atuar em várias portas normalmente associadas a esse serviço. Ao receber uma solicitação para conectar-se em um servidor SMTP, deve aceitá-la, porém deve conectar-se ao ao próprio emulador de SMTP local;
- emulação de um *proxy* SOCKS (LEECH et al., 1996) versões 4 e 5, que deve possuir a mesma funcionalidade do emulador de *proxy* HTTP descrito acima.

Módulo *Web*

O objetivo desse módulo é capturar dados sobre o envio de *spam* que se dá através do abuso de serviços *Web*, tais como: aplicações com vulnerabilidades e que são conhecidamente exploradas por *spammers*, como por exemplo o script `formmail`; e formulários *Web*, implementados em PHP e que utilizam a função `mail()`. Sem os devidos cuidados de validação de entrada, esses formulários muitas vezes são vulneráveis a ataques de injeção de cabeçalhos de *email* e são abusados por *spammers*. Para a implementação desse módulo são necessários: *honeypots* de baixa interatividade; emulação de servidores *Web*; e emulação de aplicativos vulneráveis.

Módulo *Harvesting*

O objetivo desse módulo é obter dados sobre o processo de captura de endereços de *email* em páginas *Web* (*harvesting*), bem como obter dados sobre o posterior recebimento de *spam* nos endereços capturados. Para atingir esse objetivo, o módulo precisa contar com a seguinte infra-estrutura:

- um conjunto de domínios a serem usados exclusivamente para a captura de *spam*;
- um servidor *Web*, configurado para registrar todos os acessos, incluindo endereço IP de origem, *software* informado pelo cliente ("*user agent*"), URL da última página de onde foi seguido um *link* ("*referer*"), entre outros;
- um servidor SMTP configurado para registrar todas as transações e para receber qualquer *email* para o domínio em questão.

Para possibilitar uma avaliação das técnicas de *harvesting* em uso e quais técnicas de mitigação do problema são efetivas, uma série de endereços de *email* devem estar disponíveis, de diferentes formas, nas páginas e arquivos acessíveis através do servidor *Web*.

Módulo *Phishing/Malware*

O objetivo desse módulo é classificar se um *spam* é malicioso ou não. Uma maneira de se atingir esse objetivo é tentar determinar se uma URL presente no corpo de um *email* é maliciosa. Essa referência pode tanto ser explícita a um código malicioso (um *link* para um arquivo executável, por exemplo), ou uma referência a uma página *Web* que explore vulnerabilidades no *software* do cliente quando visualizada. Para atingir seus objetivos este módulo necessita de dois subsistemas:

- *download* de *malware*, responsável por efetuar o *download* no caso de *links* para executáveis;
- *honeyclient*, um tipo de *honeypot* cujo objetivo é detectar vulnerabilidades ou códigos maliciosos que afetem uma máquina ou aplicativo cliente, por exemplo um navegador. Esse tipo de *honeypot* normalmente direciona um cliente a realizar um acesso a um servidor e monitora os resultados (WANG, 2006).

Módulo Coleta e Análise

Este módulo é o responsável pela coleta, centralização e análise dos dados provenientes dos demais módulos e dos *honeypots* genéricos.

3.2 Implementação de um Protótipo da Infra-Estrutura

Foi implementado um protótipo da arquitetura proposta, composto pelos módulos: *Relay/Proxy* Aberto; Coleta e Análise; *Harvesting*; e pelo Consórcio Brasileiro de *Honeypots*, formado por um conjunto de *honeypots* genéricos instalados em diversas redes brasileiras. Como parte do Consórcio também foi implementado um módulo para captura de *pop-up spam*. Uma visão geral da arquitetura implementada pode ser vista na Figura 3.2.

As seções a seguir descreverão o Consórcio Brasileiro de *Honeypots*, o Módulo *Relay/Proxy* Aberto e o Módulo *Harvesting*. As funcionalidades específicas do módulo

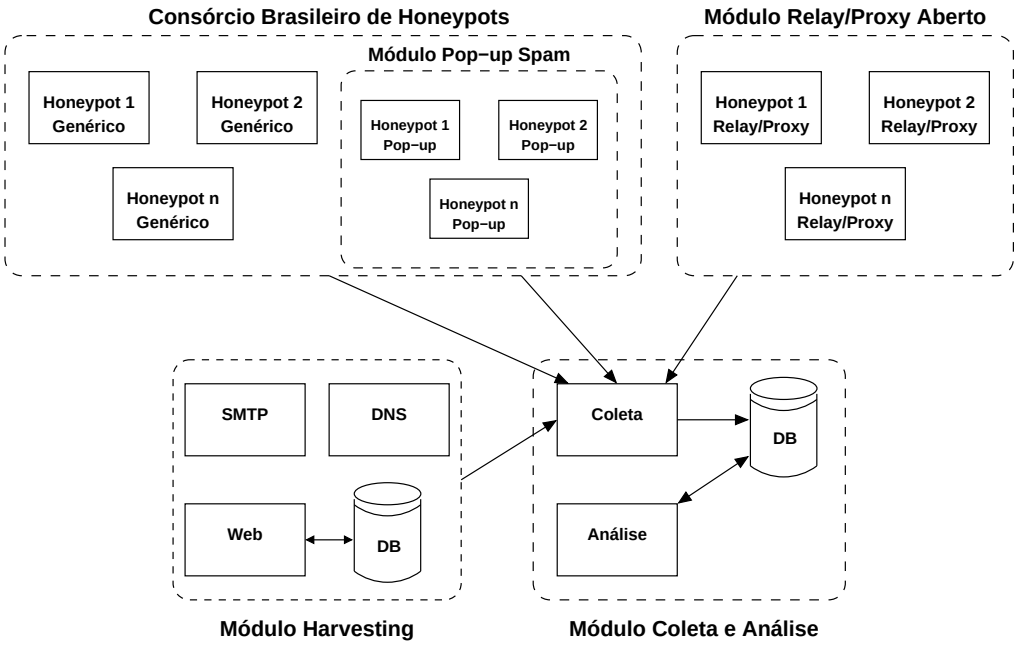


Figura 3.2 - Diagrama da arquitetura do protótipo implementado.

de coleta para tratar os dados de cada um dos outros módulos serão descritas em conjunto com os respectivos módulos. Os resultados obtidos são apresentados no capítulo 4.

3.2.1 Consórcio Brasileiro de *Honeypots*

O Consórcio Brasileiro de *Honeypots* (STEDING-JESSEN et al., 2008), que teve seu início em setembro de 2003, foi criado com os objetivos de explorar o potencial dos *honeypots* de baixa interatividade e de aumentar a capacidade de detecção de incidentes, correlação de eventos e determinação de tendências de ataques no espaço de endereços da Internet brasileira.

O Consórcio consiste em uma rede distribuída de *honeypots* de baixa interatividade, com configurações genéricas, que cobre diversos blocos de endereços IP de diferentes *Autonomous Systems* da Internet no Brasil. Na Tabela 3.1 são apresentados os números gerais do Consórcio.

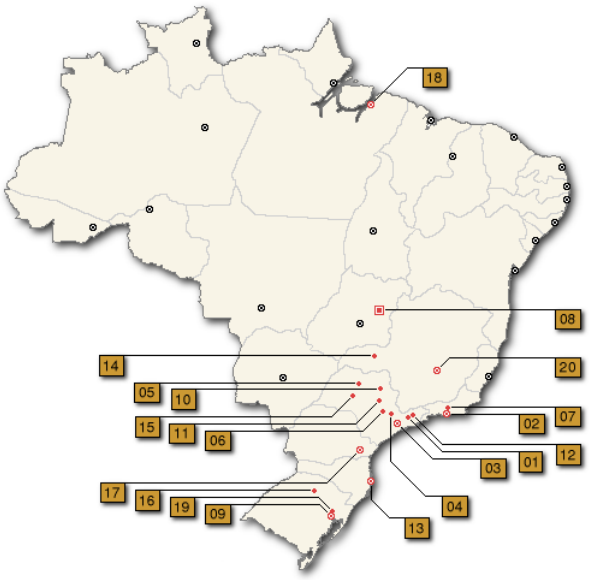
Atualmente[1] o projeto conta com *honeypots* instalados em 20 cidades brasileiras, listadas na Figura 3.3. Dentre as 37 instituições participantes, o projeto conta com

[1]Na presente data, 30/05/2008.

Tabela 3.1 - Alguns números do Consórcio Brasileiro de *Honeypots*.

Instituições Participantes	37
Cidades com *honeypots*	20
Total de *honeypots*	47
Total de IPs emulados	6528
Total de ASNs	23

sensores instalados em redes acadêmicas e de pesquisa, governamentais, militares, de empresas de telecomunicações, comerciais e do setor financeiro.



#	Cidade
01	São José dos Campos
02	Rio de Janeiro
03	São Paulo
04	Campinas
05	São José do Rio Preto
06	Piracicaba
07	Petrópolis
08	Brasília
09	Porto Alegre
10	Ribeirão Preto
11	São Carlos
12	Taubaté
13	Florianópolis
14	Uberlândia
15	Lins
16	Passo Fundo
17	Curitiba
18	Belém
19	São Leopoldo
20	Belo Horizonte

Figura 3.3 - Mapa com as cidades onde estão instalados os *honeypots*.

Fonte: Adaptada de Steding-Jessen et al. (2008).

A seguir serão descritas a arquitetura implementada no Consórcio, a utilização dos dados coletados no contexto do estudo de *spam* e a utilização dos dados para outros fins relacionados à segurança.

3.2.1.1 Arquitetura do Consórcio

A arquitetura do Consórcio é composta de *honeypots* de baixa interatividade, instalados nas redes das instituições parceiras, e de um servidor central para coleta de dados, verificação de *status* dos *honeypots*, geração de relatórios, notificações e

doação de dados. Também fazem parte da arquitetura um servidor para replicação dos dados e um servidor para publicação de estatísticas diárias. Uma visão geral da arquitetura é apresentada na Figura 3.4. A seguir são descritos os componentes principais da arquitetura. A doação de dados, as notificações e os sumários serão descritos na seção 3.2.1.4.

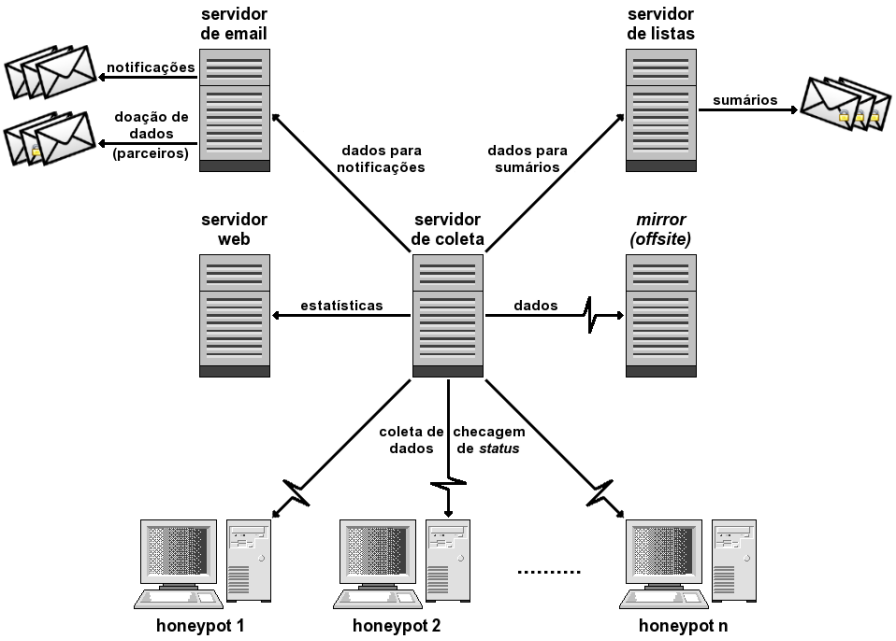


Figura 3.4 - Arquitetura do Consórcio Brasileiro de *Honeypots*.

Fonte: Adaptada de Steding-Jessen et al. (2008).

Cada instituição parceira conta com pelo menos um *honeypot* utilizando uma faixa de endereços IP que não esteja sendo utilizada para redes de produção[2]. Esta faixa de endereçamento deve ser roteável e não deve possuir filtros que impeçam o tráfego da Internet. A instituição também fornece um computador onde será instalado o *honeypot*, ficando responsável pela instalação e gerência deste equipamento.

Cada *honeypot* é instalado segundo os padrões de configuração definidos no projeto, com as seguintes características: OpenBSD como sistema operacional base; `Honeyd` para emulação de serviços; `arpd` para responder requisições ARP para a faixa de endereçamento utilizada; e fuso horário GMT com sincronização de relógio via o

[2]Geralmente as instituições alocam para o projeto blocos CIDR que variam entre `/24` e `/28`.

protocolo NTP. Este padrão de configuração permite que as instituições possam utilizar equipamentos de diferentes arquiteturas e com configurações modestas do ponto de vista de processamento, disco e memória.

Um endereço IP da faixa destinada ao *honeypot* é utilizado exclusivamente para a sua gerência. O *honeypot* utiliza o filtro de pacotes nativo do sistema OpenBSD, o `pf` (HARTMEIER, 2002), para restringir o acesso a este IP de gerência apenas para o IP do servidor de coleta e outros IPs autorizados pela instituição que o mantém. O tráfego para os demais IPs da faixa utilizada é autorizado e seu conteúdo integral armazenado pelo próprio `pf`, em formato `libpcap`. Também são armazenados os *logs* do `Honeyd`. Todos os *logs* são rotacionados diariamente e coletados a partir de um servidor central.

O servidor central conecta-se nos *honeypots* a cada 24 horas, usando um túnel cifrado SSH, para coletar os dados armazenados no período. Após a coleta ser completada em todos os *honeypots* os dados são replicados em outro servidor. O fato de cada *honeypot* não precisar reter os dados por muito tempo diminui consideravelmente os requerimentos de espaço em disco de cada sensor.

Além da coleta, o servidor central também é responsável por realizar uma série de testes para garantir que cada *honeypot* do Consórcio esteja operando corretamente. Em intervalos regulares, diversos testes são realizados, como por exemplo: conectividade com o *honeypot*; tempo desde a última reinicialização (*uptime*); processos em execução; quantidade de espaço disponível em disco; e situação da sincronização do relógio. Caso sejam encontrados problemas, os resultados dos testes são então repassados para os responsáveis de cada *honeypot*, para que possam ser resolvidos.

3.2.1.2 Módulo para Coleta de *Pop-Up Spam*

Pop-Up Spam, também conhecido como *Windows Pop-Up Spam* ou *Windows Messenger Spam*, é uma técnica de envio de *spam* que entrega mensagens de texto através do serviço *Windows Messenger*[3], disponível nos sistemas operacionais Windows®. Este serviço foi originalmente desenvolvido para permitir que administradores de rede pudessem enviar mensagens administrativas para um ou mais usuários de uma rede local (LAN). Estas mensagens são exibidas no formato de janelas *pop-up* do sistema operacional. Este serviço, contudo, pode ser abusado quando uma máquina

[3]Este serviço é distinto da aplicação MSN *Messenger*.

Windows®, com este serviço habilitado, se conecta na Internet. *Spammers* têm abusado desse serviço para o envio de *spam* na forma de janelas *pop-up* (STEWART, 2003; PANG et al., 2004).

O serviço *Windows Messenger* utiliza protocolo UDP e, em geral, é associado à porta 1026/UDP, 1027/UDP ou 1028/UDP. Na utilização legítima do protocolo é feita uma consulta ao serviço RPC do Windows®, que então indica em qual porta o serviço *Messenger* está. Os *spammers*, entretanto, enviam as mensagens diretamente para as portas 1026/UDP, 1027/UDP ou 1028/UDP.

O objetivo desse subsistema é capturar esse tipo de *spam*, com ênfase na extração de URLs que normalmente estão contidas nas mensagens. Ele faz parte do Consórcio Brasileiro de *Honeypots* e foi implementado através da coleta de todo o tráfego destinado aos *honeypots* do Consórcio nas portas 1026/UDP, 1027/UDP e 1028/UDP. Após a coleta este tráfego era analisado em busca de padrões que identificassem mensagens de *pop-up spam* que contivessem URLs.

3.2.1.3 Uso dos Dados no Contexto Deste Estudo

Os dados do Consórcio Brasileiro de *Honeypots* são usados para permitir a comparação de tráfego malicioso observado no Consórcio com o tráfego observado em outros módulos que sejam específicos para captura de *spam*.

O objetivo principal é poder correlacionar os endereços IP que fazem constantes varreduras por portas associadas a *proxies* abertos e servidores de SMTP, nos *honeypots* do Consórcio, com os endereços IP que exploram esse serviço no Módulo *Relay/Proxy* Aberto.

3.2.1.4 Outros Usos dos Dados

Os dados do Consórcio Brasileiro de *Honeypots* têm sido utilizados também para auxiliar o processo de resposta a incidentes e para gerar estatísticas e tendências de ataques (HOEPERS et al., 2005). Como essa rede cobre uma quantidade significativa de endereços IP, em várias redes brasileiras, tem se mostrado uma ferramenta muito útil para se observar varreduras, propagação de códigos maliciosos e novas tendências.

Todos os dados coletados dos *honeypots* são analisados à procura de assinaturas de atividades maliciosas, tais como atividades associadas à propagação de *bots*, varre-

duras por serviços vulneráveis, tentativas de autenticação usando métodos de força bruta, entre outros. Uma vez identificadas, essas tentativas são separadas em função dos endereços IP de origem.

As atividades maliciosas originadas de endereços IPs brasileiros são notificadas pelo CERT.br para os responsáveis por estas redes. São enviados *logs* sanitizados, uma descrição do ataque e dicas de recuperação.

Já as atividades maliciosas oriundas de IPs não brasileiros é doada para outros CSIRTs com responsabilidade nacional, para que estes possam atuar na origem do problema. Na presente data, CSIRTs dos seguintes países recebem do Consórcio dados oriundos de seus blocos de endereçamento IP: Argentina, Austrália, Colômbia, Japão, Polônia, Qatar e Uruguai.

Outra utilização dos dados do Consórcio é a geração de estatísticas e tendências de ataques detalhadas para os membros e dados consolidados e sanitizados para o público em geral (HOEPERS et al., 2005).

Diariamente, cada membro do projeto recebe, via *emails* cifrados, um sumário das atividades capturadas pelos *honeypots* e um conjunto de correlações. O sumário contém informações como: as portas TCP e UDP que receberam o maior número de tentativas de conexão; IPs, *Country Codes* e sistemas operacionais de origem das atividades maliciosas; e o volume de dados capturados nas 24 horas anteriores. As correlações contém informações sobre atividades que não necessariamente produziram um grande volume ou número de pacotes, mas que se sobressaíram por terem sido observadas simultaneamente em um conjunto de *honeypots*, como: portas TCP e UDP; e endereços IP, acompanhados das portas que tentaram acessar em cada sensor.

Além dos sumários mais detalhados para os membros, também são geradas estatísticas públicas, disponíveis na página *Web* do Consórcio. Estas estatísticas são em formato de *flows* de dados direcionados ao conjunto dos *honeypots* (HOEPERS et al., 2005). Os dados publicados não identificam quais os endereços IP que realizaram as atividades e não identificam unicamente nenhum sensor. São apresentados apenas os sumários sobre as portas TCP e UDP que receberam o maior número de tentativas de conexão e os *Country Codes* e sistemas operacionais de origem das atividades maliciosas.

3.2.2 Módulo *Relay/Proxy* Aberto

Esta seção apresenta o módulo criado para capturar dados sobre o envio de *spam* através do abuso de *relays* e *proxies* abertos. São apresentadas a arquitetura e a implementação de um conjunto de sensores, baseado em *honeypots*, para a captura de *spam*. Os resultados da análise dos dados coletados por este módulo serão apresentados no capítulo 4.

3.2.2.1 Arquitetura do Módulo *Relay/Proxy* Aberto

A arquitetura implementada, exibida na Fig. 3.5, contou com 10 *honeypots* de baixa interatividade, responsáveis pela captura de *spams*, instalados em redes de banda larga de 5 operadoras diferentes (cabo e ADSL – *Asymmetric Digital Subscriber Line*). Os *honeypots* foram instalados nas residências de voluntários para refletir as condições a que estão submetidos computadores típicos de usuários conectados via banda larga.

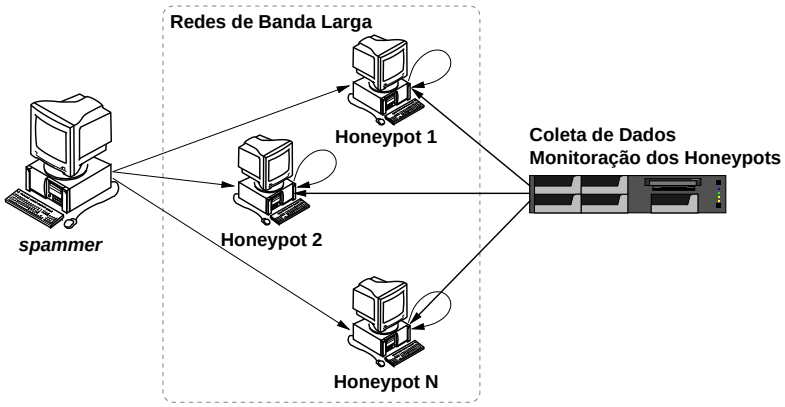


Figura 3.5 - Diagrama da arquitetura implementada.

Em cada operadora foi instalada uma conexão na modalidade doméstica, tipicamente com IP dinâmico e menor largura de banda, e uma na modalidade empresarial, com IP fixo e geralmente com maior largura de banda. Os *honeypots* permaneceram instalados em 4 cidades brasileiras por um período de 15 meses. A compra dos equipamentos e o custeio das conexões de banda larga pelo período de duração da coleta de dados, foi financiada pelo Núcleo de Informação e Coordenação do Ponto BR (NIC.br), com apoio do Comitê Gestor da Internet no Brasil (CGI.br).

Estes *honeypots* foram configurados de modo a simular computadores com *relays* e *proxies* abertos. Desse modo, um *spammer* que tentasse abusar de um destes *honeypots* para o envio de *spam*, seria levado a acreditar que teve sucesso em enviar seus *emails*.

Também fez parte da arquitetura um servidor central, configurado para realizar a coleta dos *spams* capturados, bem como a monitoração periódica dos *honeypots*.

Nas seções seguintes estes componentes serão descritos em mais detalhes.

3.2.2.2 Configurações Básicas dos *Honeypots*

Cada *honeypot* foi instalado em um computador da arquitetura `i386`, com o sistema operacional OpenBSD e a seguinte configuração de *hardware*: Intel Celeron 2.66 GHz, 512 MB de RAM, 80 GB de disco e interface de rede 10/100.

O *honeypot* utilizava o filtro de pacotes `pf` (HARTMEIER, 2002), bloqueando todo o tráfego de entrada, com excessão daquele associado às portas sendo emuladas pelo `Honeyd`, que serão discutidas na próxima seção, e das portas de gerência do *honeypot*.

O relógio interno de cada *honeypot* foi mantido sincronizado através do uso de NTP (*Network Time Protocol*) (MILLS, 2006). O *timezone* escolhido foi o GMT (*Greenwich Mean Time*), por questões de padronização e independência de ajustes locais de horário de verão.

3.2.2.3 Emulação dos Serviços e Captura de *Spam*

A captura de *emails* nos *honeypots* utilizou o *software* `Honeyd` em conjunto com subsistemas de emulação de *relays* e *proxies* abertos.

Embora o `Honeyd` já possua módulos para emulação de SMTP e *proxy* HTTP, ambos usados neste módulo, ele não possui suporte para emulação do protocolo SOCKS. Assim, um emulador de SOCKS versões 4 e 5 foi desenvolvido como parte deste trabalho, com o objetivo de complementar os demais emuladores. Adicionalmente, os módulos já existentes no `Honeyd` também sofreram algumas modificações, visando a compressão de dados e o armazenamento de informações adicionais.

Segue uma descrição do funcionamento dos subsistemas de emulação do `Honeyd` e do emulador SOCKS desenvolvido, seguida da descrição das modificações incorporadas.

Emulação do serviço SMTP: este emulador é um módulo do `Honeyd` que recebe o tráfego destinado à porta 25/TCP e simula as respostas dadas por um servidor de SMTP. Para toda a mensagem recebida, simula o comportamento de um *relay* aberto e aceita a mensagem. Porém, a mensagem é apenas armazenada e nunca entregue ao destinatário.

Emulação de *proxy* HTTP: este emulador também é um módulo do `Honeyd`, que pode ser configurado para atuar em várias portas normalmente associadas a serviços de *proxy* HTTP: 80/TCP, 8080/TCP, 3128/TCP, entre outras portas listadas na Tabela 3.2. Ao conectar-se, o *spammer* tipicamente solicita ao *proxy* que o conecte ao endereço IP do servidor SMTP da vítima, na porta de destino 25/TCP. O emulador de *proxy*, então, ao invés de conectar no endereço solicitado, conecta-se ao próprio emulador de SMTP local que, ao verificar que a conexão veio do emulador de *proxy*, provê uma resposta SMTP especial. Esta resposta tenta fornecer um *banner* SMTP similar ao que seria enviado pelo servidor do endereço de destino solicitado. Julgando estar conectado ao servidor SMTP originalmente solicitado, o *spammer* inicia o envio de suas mensagens. Caso o emulador de *proxy* receba uma solicitação cuja porta de destino seja diferente de 25/TCP (SMTP), o emulador retorna um erro ao cliente.

Emulação de *proxy* SOCKS: este emulador foi desenvolvido, como parte deste trabalho, para comportar-se como um *proxy* SOCKS (LEECH et al., 1996) versões 4 e 5, recebendo tráfego nas portas TCP geralmente associadas a esse serviço, como mostrado na Tabela 3.2. Este emulador comporta-se como um *proxy* SOCKS que não solicita nenhum tipo de autenticação e permite conexões oriundas de qualquer endereço IP. Após conectar-se, o *spammer* solicita que o *proxy* o conecte a um endereço e porta TCP de destino. Caso a porta de destino seja diferente de 25/TCP (SMTP), o emulador retorna um erro ao cliente. Caso contrário, conecta-se ao próprio emulador de SMTP local, que responde de maneira análoga à descrita no emulador de *proxy* HTTP.

A interação entre os módulos de emulação de SMTP e *proxy* descritos pode ser vista na Figura 3.6.

Todas as transações efetuadas pelos módulos do `Honeyd` são armazenadas em arqui-

Tabela 3.2 - Portas emuladas pelos *honeypots*.

protocolo	portas TCP
SMTP	25
HTTP	80, 81, 2282, 3128, 3332, 3382, 3802, 4480, 5490, 6588, 8000, 8080, 8090, 11120, 57123, 63809, 65506
SOCKS	559, 1029, 1080, 1202, 1813, 1978, 1979, 2280, 2425, 3127, 3380, 3800, 4471, 4777, 4894, 5748, 6042, 7531, 9938, 10000, 10001, 10232, 11117, 15859, 19086, 24971, 24972, 24973, 30021, 30022, 35612, 38994, 40934, 41457, 57123, 63808

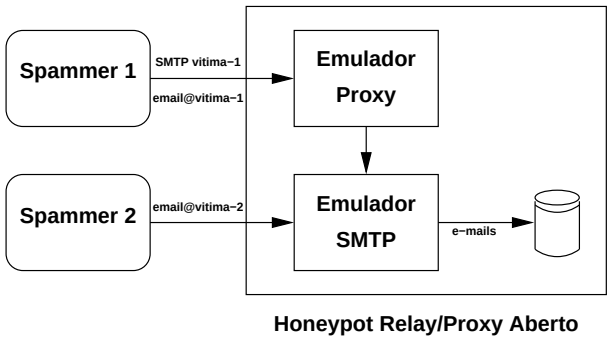


Figura 3.6 - Interação entre os módulos de emulação de SMTP e *proxy*.

vos de eventos (*log*) com informações como data e hora, IP de origem, IP e porta pretendidos de destino, assim como versão do protocolo utilizado. Para permitir a compressão de dados e o armazenamento de informações adicionais, a serem usadas na análise dos *spams*, foram efetuadas algumas modificações nos módulos do `Honeyd`. Estas modificações são descritas a seguir.

Hierarquia de armazenamento: o módulo de SMTP é responsável por armazenar todos os *emails* em disco, numa estrutura de diretórios que leva em conta o endereço IP de origem e a mensagem propriamente dita. Modificou-se essa estrutura para que também fossem registradas a data e a porta TCP usada na entrega do *email*, facilitando o processamento das informações.

***Logs* mais completos:** o módulo de *proxy* HTTP foi modificado para gerar uma quantidade maior de informações no seu respectivo arquivo de *log*, como método de conexão pretendido, IP de origem, IP e porta pretendidos de destino, *status*, entre outros. Também foi modificada a geração de *logs* por parte do módulo SMTP para refletir as mudanças na estrutura de

armazenamento.

Compressão: adicionou-se suporte à compressão de dados no módulo de SMTP, de modo que todo o *email* armazenado fosse comprimido usando-se a biblioteca `zlib`.

3.2.2.4 Identificação e Resposta a *Emails* de Teste

Diversas ferramentas utilizadas por *spammers* procuram identificar se um *proxy* encontrado está de fato permitindo o seu uso para envio de *emails* (KRAWETZ, 2004). Um dos métodos utilizados é o envio de uma mensagem de teste destinada a um endereço de *email* sob controle do *spammer*.

Neste trabalho foram identificados alguns padrões de mensagens de teste. Estas mensagens, além de verificarem que o *proxy* está aberto, também remetem ao *spammer* informações adicionais, tais como: endereço IP do *proxy*, porta e protocolo utilizados.

Na Figura 3.7 é mostrado um exemplo de mensagem de teste. No seu campo `Subject:` é possível observar as informações adicionais sendo enviadas para o *spammer*: o IP do sensor (aqui sanitizado), a porta abusada (6588/TCP) e o protocolo (HTTP).

```
Return-Path: <jacky19790729_2@yahoo.com.tw>
Received: from xxx.xxx.xxx.xxx (218-171-111-176.dynamic.hinet.net [218.171.111.176])
        by 220-134-150-183.HINET-IP.hinet.net (Postfix) with ESMTP id 3356ED318D
        for <service3@ecs168.com.tw>; Tue, 18 Sep 2007 05:34:57 +0000 (GMT)
From: "Jacky" <jacky19790729_2@yahoo.com.tw>
To: service3@ecs168.com.tw
Date: Tue, 18 Sep 2007 13:34:22 +0800
Subject: SM:xxx.xxx.xxx.xxx:6588:HTTP
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="--=====494518713825451=_"

[...]
```

Figura 3.7 - Exemplo de uma mensagem de teste.

Todas as mensagens identificadas com padrões de teste foram enviadas para o endereço sob o controle do *spammer*. Analisando as mensagens de teste enviadas, foi possível notar que nem toda primeira mensagem era um teste. Também observou-se que, em alguns casos, vários testes eram enviados ao longo de uma campanha, não somente no seu início.

3.2.2.5 Coleta de Dados e Monitoração dos *Honeypots*

Todos os *spams* capturados pelos *honeypots* foram coletados em intervalos regulares por um servidor central. Esse mecanismo de coleta foi implementado utilizando-se o programa de cópia e sincronização remota de dados `rsync`, através de um túnel criptografado implementado com o programa de *login* remoto SSH (*Secure Shell*).

Cada *honeypot* possuía a capacidade de armazenar por vários dias os *spams* capturados. Porém, após realizada a sincronização bem sucedida com o servidor central esses dados eram removidos de cada *honeypot* para liberar espaço em disco.

De modo a garantir que todos os *honeypots* estivessem operando corretamente, um mecanismo de monitoração foi desenvolvido. Executando-se esse mecanismo periodicamente, a partir do servidor central, era possível rapidamente determinar se todos os *honeypots* do projeto estavam operando dentro dos padrões normais de funcionamento.

Esse procedimento de monitoração foi especialmente importante devido à natureza da conectividade e do local de instalação dos *honeypots*, pois a qualidade de serviço provido no serviço de banda larga é tipicamente inferior quando comparado ao das conexões dedicadas. Além disso, como os *honeypots* foram instalados em ambientes residenciais de voluntários do projeto, a qualidade do fornecimento de energia elétrica também era inferior àquela de ambientes mais controlados, como o de *data centers*.

As verificações efetuadas em cada *honeypot* são descritas a seguir:

Conectividade do *honeypot*: Testa se o *honeypot* está acessível a partir do servidor central.

Tempo de operação do sistema (*uptime*): Essa medida é particularmente importante para determinar reinicializações inesperadas do computador, causadas, por exemplo, por problemas na rede elétrica.

Carga do sistema (*load*): Obtém o número de processos na fila de execução do sistema operacional nos últimos 1, 5 e 15 minutos. A carga de cada um dos *honeypots* está relacionada com a quantidade de processos executando a um dado instante.

Ocupação do disco do *honeypot*: Verificação importante para assegurar que o

honeypot tem capacidade para armazenar os *spams* capturados.

Sincronização de relógio: Essa medida avalia a diferença do relógio interno do *honeypot* em relação a uma referência externa, via NTP. A correta sincronização de relógio dos sensores é fundamental para a geração de registros confiáveis de eventos.

Presença de processos críticos do *honeypot*: Esta verificação testa pela presença de processos críticos para o correto funcionamento do *honeypot* e captura de *spams*.

Consumo de banda e taxa de recebimento de *emails*: Acompanhamento do consumo de banda (entrada e saída, em pacotes/s e *bytes*/s) e a taxa de recebimento de *emails* dos últimos 15 minutos.

Sincronização de dados com o servidor de coleta: Com essa medida é possível determinar a última data que a sincronização de dados com o servidor central foi efetuada com sucesso.

Caso qualquer teste apresentasse resultados fora dos limiares esperados, um alerta era gerado. Deste modo, era possível recuperar-se rapidamente de erros não graves ou contatar os voluntários para verificar o estado do sensor.

3.2.3 Módulo *Harvesting*

O objetivo desse módulo foi obter dados sobre o processo de captura de endereços de *email* em *sites* na Internet (*harvesting*), seja em páginas HTML ou em arquivos publicados nos *sites*, e sobre o recebimento de *spam* nos endereços capturados. Também foram obtidos dados sobre a efetividade de mecanismos de ofuscação de endereços e o comportamento de *crawlers* legítimos.

Para atingir esse objetivo foram criados três domínios, sem usuários legítimos, para utilização exclusiva neste módulo: um internacional (sob o gTLD "`.org`"); e dois nacionais, um comercial (sob o DPN "`.com.br`") e um acadêmico ("`lss.inpe.br`"). Cada domínio contou com a seguinte infra-estrutura:

Um servidor *Web*: configurado para registrar todos os acessos, incluindo endereço IP de origem, *software* informado pelo cliente ("*user agent*"), URL da última página de onde foi seguido um *link* ("*referer*"), entre outros.

Um servidor SMTP: configurado para registrar todas as transações e para receber qualquer *email* para o domínio em questão.

Um servidor DNS: com autoridade sobre o domínio e configurado para registrar todas as consultas recebidas. Este servidor de nomes publica um registro MX ("*Mail Exchange*") apontando para o servidor SMTP.

Nas próximas seções serão descritos em detalhes os componentes responsáveis pela geração dos endereços de *email* e pelo acompanhamento da sua coleta e utilização.

3.2.4 Geração de Endereços de *Email* e do Conteúdo dos *Sites*

Neste trabalho, além de coletar dados sobre o IP que realizou o *harvesting* e sobre aqueles que enviaram o *email*, foram também coletadas informações sobre a diferença de volume entre tipos diferentes de *email*, sobre a efetividade de métodos de ofuscação utilizados e sobre os tipos de arquivos onde os endereços são coletados. Os arquivos utilizados para publicar endereços de *email* foram dos seguintes tipos: `.txt`, `.pdf`, `.ps`, `.doc`, `.ppt` e `.tar`.

Para obter estes dados, uma série de endereços de *email* foram disponibilizados de formas diferentes nas páginas e arquivos acessíveis através do servidor *Web*. Eles podem ser divididos em três categorias:

Dinâmicos: endereços únicos, gerados dinamicamente, colocados em páginas HTML e no interior de arquivos.

Estáticos: endereços que não mudaram ao longo da coleta e estavam sempre dentro das mesmas páginas HTML e arquivos.

Estáticos com ofuscação: endereços que não mudaram ao longo da coleta, estavam sempre dentro das mesmas páginas HTML e arquivos, mas utilizavam técnicas de ofuscação para dificultar a identificação por programas de *harvesting*. As técnicas usadas foram: troca do caracter, "`@`" por "`at`" e "`em`"; uso de JavaScript; e uso de imagens para apresentar o endereço de *email*.

A geração dos *emails* dinâmicos ocorria a cada vez que uma determinada página era acessada. No momento do acesso, o servidor *Web* identificava que a página continha referência a um programa externo e o executava. Este programa, chamado `genmail`,

era responsável por criar um endereço de *email* único e armazenar as seguintes informações: *email* criado; data/hora do acesso à página; endereço IP de origem; "*User agent*" informado; e a URI requisitada. Caso um dado endereço IP voltasse a acessar uma mesma página, ele receberia o mesmo *email* criado no momento do primeiro acesso.

Caso fosse solicitado o acesso a um arquivo que contivesse um *email* dinâmico, o servidor fazia uma chamada ao programa `genfile`. Este programa era responsável por criar um endereço de *email* dinâmico, inserí-lo no arquivo correspondente e armazenar as mesmas informações que o programa `genmail`.

Para comparar o comportamento de *crawlers* legítimos, como por exemplo aqueles empregados por sites de busca como o Google, com *crawlers* utilizados por *spammers*, os *sites* criados foram divididos em duas áreas: uma área sem restrições e uma área definida como restrita, via o arquivo `robots.txt`. *Crawlers* legítimos, geralmente, respeitam o arquivo `robots.txt`, que sinaliza porções de um *site* que não devem ser percorridas e indexadas.

3.2.5 Recebimento e Armazenamento dos *Emails*

A implementação do recebimento e armazenamento de *emails* enviados para os domínios criados para este estudo foi feita através do *software Mail Avenger* (MAZIÈRES, 2008; RAMACHANDRAN; FEAMSTER, 2006). *Mail Avenger* é um servidor SMTP altamente configurável, integrável com filtros anti-*spam* e que possibilita a coleta das seguintes informações adicionais sobre os endereços IP de origem de cada *email* recebido:

***Network route recording*:** era traçada a rota (`traceroute`) seguida até o endereço IP enviando a mensagem.

***Fingerprint* passivo:** com base na conexão TCP para o servidor SMTP eram obtidos dados sobre qual o provável sistema operacional do IP que originou a conexão.

Presença em listas de bloqueio: era realizada uma consulta sobre a presença do endereço IP nas seguintes listas de bloqueio baseadas em DNS (DNSBL): `bl.spamcop.net` e `sbl-xbl.spamhaus.org`.

O *Mail Avenger* foi configurado para inserir estas informações adicionais como um campo de cabeçalho no *email* sendo recebido e para receber mensagens para qualquer endereço de *email* dos domínios criados para o estudo e repassá-los para um *Mail Delivery Agent* (MDA). O *software* também mantinha *logs* detalhados de todas as sessões SMTP realizadas.

Uma outra característica importante do *Mail Avenger* é a facilidade para integrar programas externos no processamento das mensagens recebidas. Neste trabalho foi desenvolvido o programa `fake.mail.local`, para atuar como um substituto do MDA padrão do *Mail Avenger*. Isso foi necessário porque o MDA do *Mail Avenger*, como a maioria dos MDAs, não aceita salvar mensagens em arquivos *mailbox* de usuários inexistentes. Como neste estudo eram criados endereços de *email*, mas não usuários no sistema, o programa `fake.mail.local` armazenava as mensagens recebidas, em um *mailbox* por endereço criado, por domínio. Essa estrutura se mostrou eficiente por permitir o uso de programas convencionais de manipulação de mail durante o processo de análise.

4 RESULTADOS OBTIDOS

Neste capítulo são apresentados os resultados da análise dos *spams* coletados pelos módulos implementados no protótipo da infra-estrutura, descritos em detalhes no capítulo 3.

4.1 Resultados do Módulo *Relay/Proxy* Aberto

Nesta seção são mostrados os resultados da análise dos *spams* coletados pelo módulo *Relay/Proxy* Aberto, tendo como base a sua origem e as portas utilizadas para o seu envio.

Como mostrado na Tabela 4.1, o projeto coletou *spams* por cerca de 15 meses, tendo capturado, no período, aproximadamente 525 milhões de *spams* que seriam entregues a 4,8 bilhões de destinatários. As mensagens capturadas se originaram de 216.888 diferentes endereços IP, alocados para 165 *Country Codes* de origem. Os *Country Codes* existentes são definidos pela ISO 3166 (ISO, 2006). O mapeamento entre um endereço IP e seu *Country Code* foi obtido através das informações de alocação de endereços IP, mantidas nos arquivos de estatísticas dos 5 *Regional Internet Registries* (RIR): AfriNIC, APNIC, ARIN, LACNIC e RIPE NCC.

Tabela 4.1 - Estatísticas gerais referentes aos *emails* capturados.

Início da coleta dos dados	10/06/2006
Fim da coleta dos dados	18/09/2007
Dias de dados coletados	466
Total de *emails* coletados	524.585.779
Total de destinatários	4.805.521.964
Média de destinatários por *spam*	9,16
Média de *emails* por dia	1.125.720
IPs únicos que enviaram *spam*	216.888
ASs únicos que enviaram *spam*	3.006
Países (*Country Codes*) de origem	165

Somando-se a coleta realizada em cada um dos 10 *honeypots*, a média diária de *spams* recebidos foi de aproximadamente 1,1 milhão. Na Figura 4.1 pode-se ver a oscilação no número de *spams* recebidos diariamente, com picos de recebimento de aproximadamente 2 milhões. Também é possível ver uma tendência de crescimento dos *spams* recebidos durante o período.

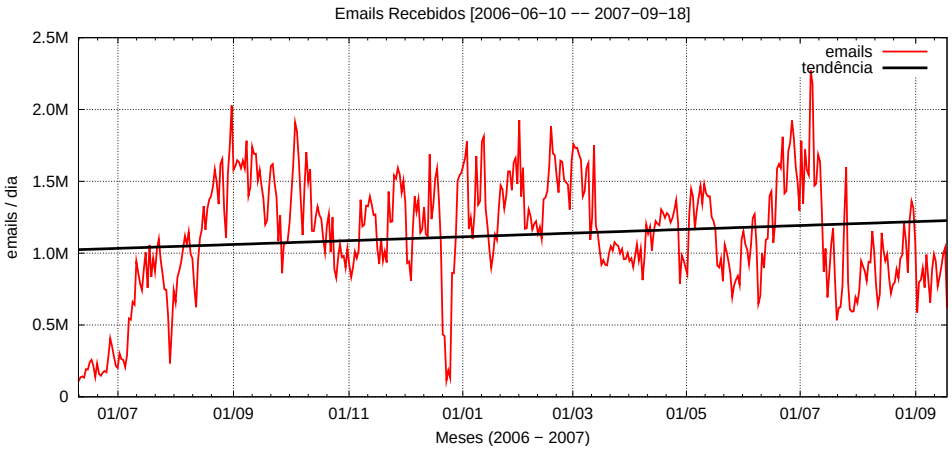


Figura 4.1 - *Emails* recebidos ao longo do período.

Fazendo uma análise de todos os países aos quais estavam alocados os endereços IP de origem dos *spams*, foi possível notar uma grande concentração em alguns países. De todo o *spam* que foi capturado pelo módulo *Relay/Proxy* Aberto, 97,2% foi originado de apenas 5 países: Taiwan, China, Estados Unidos, Canadá e Japão, como pode ser observado na Tabela 4.2. Também é importante notar que 73,43% do total das mensagens enviadas foram originadas de Taiwan. Se levarmos em conta os 10 primeiros países que mais enviaram *spam* para os *honeypots* do projeto, esse número sobe para 99,5% do total de mensagens.

Tabela 4.2 - *Country Codes* (CC) mais freqüentes de origem dos *spams*.

#	CC	*Emails*	%
01	TW	385.189.756	73,43
02	CN	82.884.642	15,80
03	US	29.764.293	5,67
04	CA	6.684.667	1,27
05	JP	5.381.192	1,03
06	HK	4.383.999	0,84
07	KR	4.093.365	0,78
08	UA	1.806.210	0,34
09	DE	934.417	0,18
10	BR	863.657	0,16

A Figura 4.2 apresenta a contribuição dos 5 *Country Codes* que mais originaram *spam* e da soma de todos os outros *Country Codes* identificados, distribuídos ao longo do período de coleta de dados. É possível ver que Taiwan não só originou

o maior número de *spams* no total, mas também foi quem mais originou ao longo de todo o período. Já a China foi o segundo país que mais originou *spam* em todo o período, com exceção do mês de julho de 2007, quando os Estados Unidos se destacaram, originando cerca de 10 milhões de mensagens.

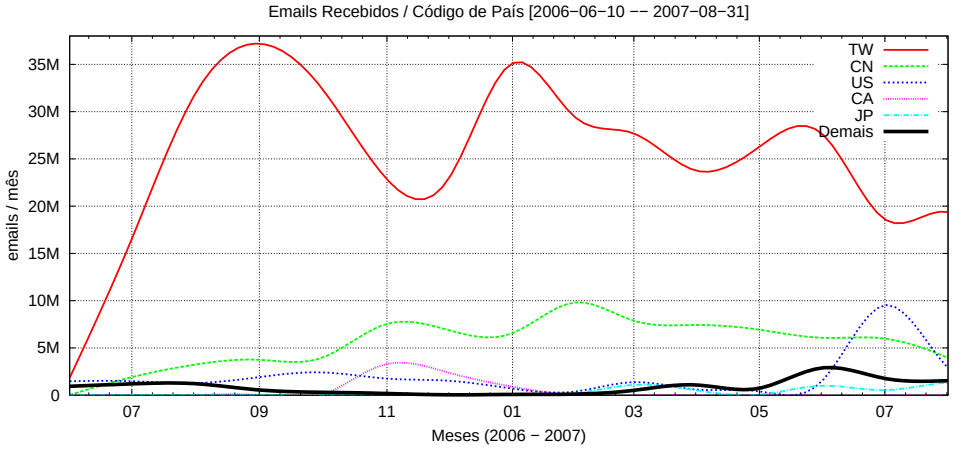


Figura 4.2 - *Emails* recebidos em função do país de origem, totais mensais.

Na Figura 4.3 os mesmos dados são mostrados, mas agora do ponto de vista do percentual de contribuição de cada *Country Code* no total de *spams* recebidos.

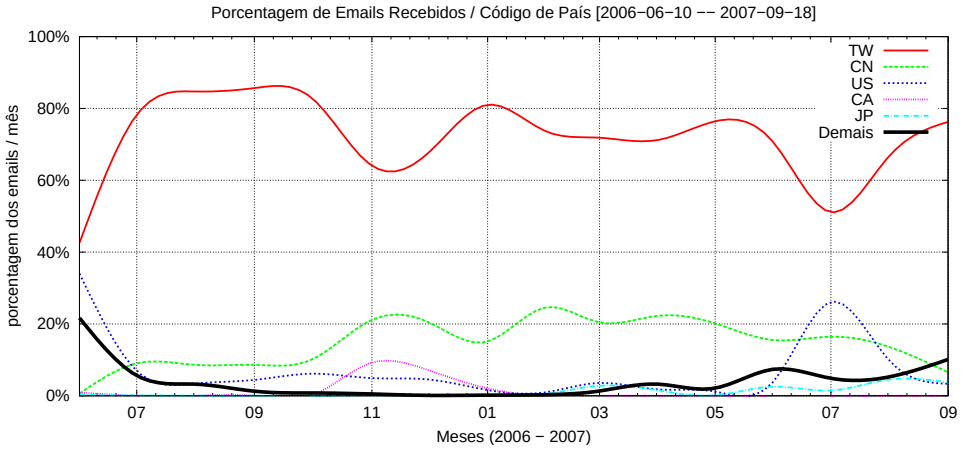


Figura 4.3 - *Emails* recebidos em função do país de origem, percentualmente.

É possível notar que a contribuição de Taiwan e China foi mais regular ao longo

do período, se comparada com os demais. É possível notar, também, que em alguns meses Taiwan foi responsável por mais de 80% dos *spams* e que seu percentual nunca foi menor que 50%.

A Tabela 4.3 mostra a distribuição dos *spams* em função dos *Autonomous Systems* (AS) de origem. Um AS é um grupo interconectado de uma ou mais redes com uma única e bem definida política de roteamento (HAWKINSON; BATES, 1996).

Tabela 4.3 - ASs mais freqüentes de origem dos *spams*.

#	ASN	Nome do AS	*Emails*	%
01	9924	TFN-TW (TW)	170.998.167	32,60
02	3462	HINET (TW)	131.381.486	25,04
03	17623	CNCGROUP (CN)	65.214.192	12,43
04	4780	SEEDNET (TW)	54.430.806	10,38
05	9919	NCIC-TW (TW)	9.186.802	1,75
06	4837	CHINA169 (CN)	9.025.142	1,72
07	33322	NDCHOST (US)	8.359.583	1,59
08	4134	CHINANET (CN)	7.287.251	1,39
09	18429	EXTRALAN (TW)	6.746.124	1,29
10	7271	LOOKAS (CA)	5.599.442	1,07

É possível ver uma clara concentração de atividade: os 4 ASs que mais originaram *spams* foram responsáveis por aproximadamente 80% de toda a atividade registrada no período. Se levarmos em conta os países associados a esses ASs, novamente Taiwan e China se sobressaem. Entre os 10 ASs de maior atividade, 8 são desses dois países. Também chama a atenção que apenas os dois primeiros ASs, TFN-TW e HINET, ambos de Taiwan, originaram aproximadamente 58% do total de *spams* capturados.

A Figura 4.4 apresenta a distribuição, ao longo do período, do total de *spams* recebidos dos 7 ASs que mais originaram mensagens e da soma de todos os outros ASs, identificados como “Demais” no gráfico.

Nesta Figura é possível observar que no período de junho a novembro de 2006 os ASs que mais originaram *spam* foram a SEEDNET (ASN 4780) e a HINET (ASN 3462), ambas de Taiwan. Já de novembro de 2006 até julho de 2007 há uma concentração da origem no AS da TFN-TW (ASN 9924), também de Taiwan. Considerando que estas três redes oferecem serviços de hospedagem e *data center*, uma possível explicação para essa mudança seria que *spammers* estariam utilizando estes serviços para hospedar máquinas para envio de *spam*. A mudança poderia ser uma migração dos

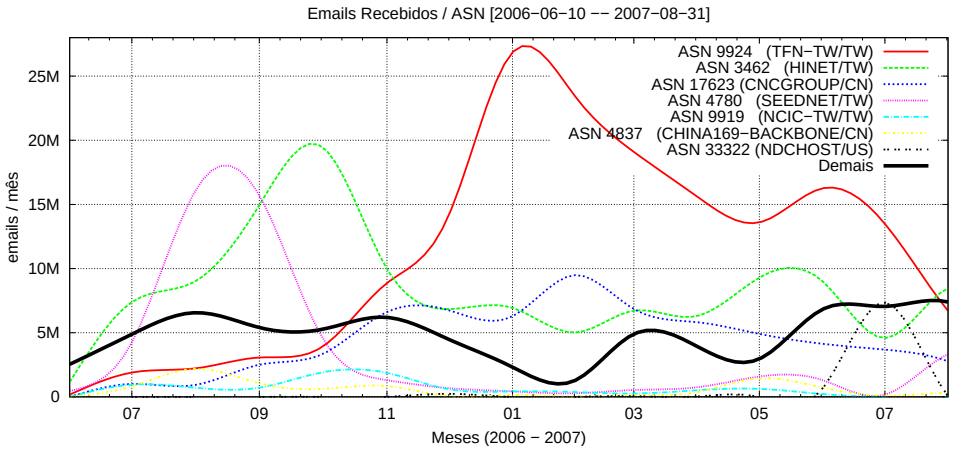


Figura 4.4 - *Emails* recebidos em função do AS, ao longo do período.

dois primeiros ASs para o terceiro. Outra possibilidade, considerando que as redes também oferecem serviços de conectividade banda larga para usuários finais, seria essa mudança refletir o uso de *botnets* nas primeiras redes por um período, seguido do uso de *botnets* na terceira rede em outro período. É interessante, também, notar que a participação dos ASs agrupados em "Demais" é significativa, algo que não acontecia quando feita a análise apenas com base no *Country Code*.

Na Figura 4.5 pode-se ver o percentual da participação dos 7 ASs que mais originaram mensagens e da soma de todos os outros ASs identificados.

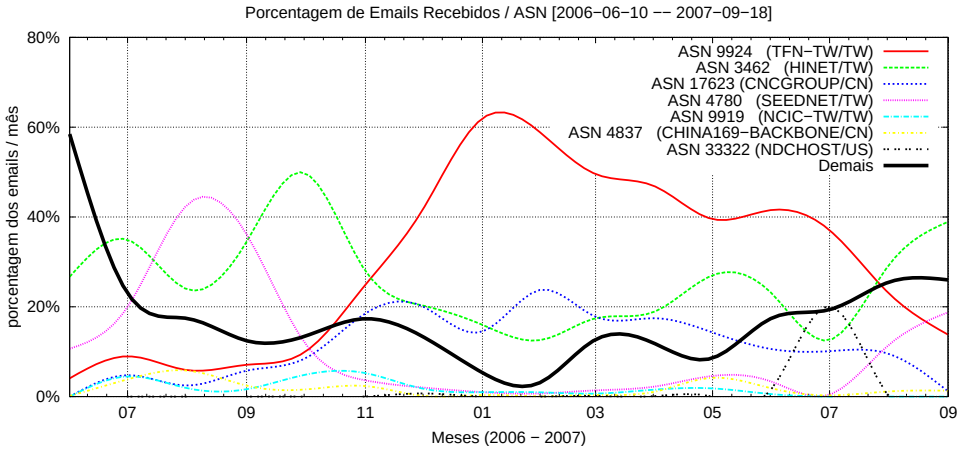


Figura 4.5 - *Emails* recebidos em função do AS, percentualmente, ao longo do período.

Com a visão do ponto de vista de porcentagem de participação dos ASs fica mais evidente que, no mês em que os *honeypots* entraram em operação, a participação por AS não era tão concentrada, com os ASs agrupados em "Demais" representando cerca de 60% de todos as mensagens. A partir de julho de 2006 a participação da SEEDNET e da HINET se sobressai. A partir de novembro de 2006 a maior porcentagem dos *spams* passa a vir da TFN-TW, com uma mudança ocorrendo no final de julho de 2007, quando a HINET e o conjunto de outros ASs passam a originar a maioria das mensagens.

Tanto na Figura 4.4 quanto na Figura 4.5 é interessante notar um pico do AS NDCHOST (ASN 33322), dos Estados Unidos, que provê serviços de hospedagem. A atividade vinda deste AS teve um pico em julho de 2007. Nesse mesmo período, a entidade anti-*spam* SpamHaus identificou que a rede NDCHOST estava provendo hospedagem para um *spammer* conhecido por abusar de *proxies* abertos para envio de *spam*[1].

Uma outra análise realizada foi com relação à distribuição dos IPs de origem no espaço de endereçamento IPv4. A Figura 4.6 apresenta a distribuição das origens de todas as conexões destinadas às portas de *proxy* HTTP ou SOCKS, agrupadas pelo volume de mensagens acumuladas para cada bloco CIDR `/8`.

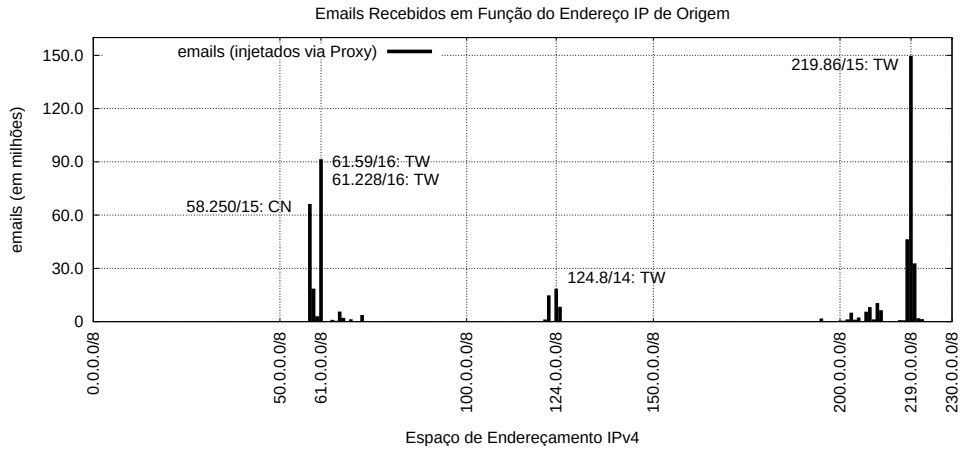


Figura 4.6 - Distribuição dos *emails* recebidos via *Proxy* no espaço de endereçamento IPv4.

[1] http://www.spamhaus.org/rokso/evidence.lasso?rokso_id=ROK7808

Para cada pico na Figura, estão indicados os blocos CIDR que mais contribuíram para um determinado pico de conexões, acompanhados do país para o qual o bloco está alocado.

Uma análise mais detalhada dos picos próximos aos blocos `58.0.0.0/8`, `61.0.0.0/8`, `124.0.0.0/8` e `219.0.0.0/8` revelou que os maiores responsáveis pela atividade observada nestes picos foram o bloco `58.250/15`, alocado para China, e os blocos `61.59/16`, `61.228/16`, `124.8/14` e `219.86/15` de Taiwan. Esta concentração de mensagens em blocos alocados para Taiwan é consistente com a concentração observada na análise dos países e dos ASs de origem.

A Figura 4.7 apresenta a distribuição das origens de todas as conexões destinadas à porta 25/TCP (SMTP) dos *honeypots*.

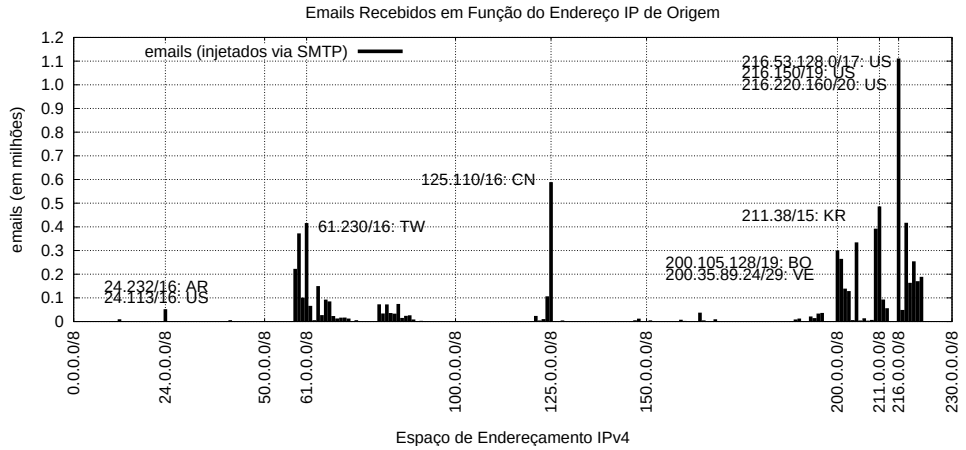


Figura 4.7 - Distribuição dos *emails* recebidos via SMTP no espaço de endereçamento IPv4.

Nota-se que o volume de mensagens geradas com destino à porta 25/TCP é muito menor que aquele gerado com destino às portas de *proxy*. É possível, também, observar que no caso de conexões tentando explorar *relays* abertos, há mais conexões partindo de faixas de endereçamento diferentes. Não havendo uma concentração em apenas 2 países como no caso de *proxy*, mas sim uma distribuição entre um maior número de países.

Outro elemento analisado foram as portas TCP usadas pelos *spammers* para injetar *emails*. Embora cada *honeypot* tivesse sido configurado para emular um grande

número de portas TCP, como visto na Tabela 3.2, apenas 11 portas foram efetivamente utilizadas para o envio de *spam*. A Tabela 4.4 mostra a relação de portas TCP que foram utilizadas para o envio de *spam*, em conjunto com o protocolo utilizado (HTTP, SOCKS ou SMTP), bem como o serviço normalmente associado a cada porta.

Tabela 4.4 - Portas TCP abusadas.

#	Porta	Protocolo	Serviço Usual	%
01	1080	SOCKS	*proxy* SOCKS	37,31
02	8080	HTTP	*alternate* `http`	34,79
03	80	HTTP	`http`	10,92
04	3128	HTTP	`Squid`	6,17
05	8000	HTTP	*alternate* `http`	2,76
06	6588	HTTP	`AnalogX`	2,29
07	25	SMTP	serviço SMTP	1,46
08	4480	HTTP	`Proxy+`	1,38
09	3127	SOCKS	`MyDoom` *Backdoor*	1,00
10	3382	HTTP	`Sobig.f` *Backdoor*	0,96
11	81	HTTP	*alternate* `http`	0,96

Entre as portas exploradas pelos *spammers*, algumas são relacionadas a serviços populares, como por exemplo as portas 80/TCP, 81/TCP, 8000/TCP e 8080/TCP, normalmente associadas com o serviço `http`, e a porta 25/TCP, associada ao serviço SMTP. As portas 6588/TCP e 4480/TCP são utilizadas pelos *softwares* `AnalogX` e `Proxy+`, que são programas legítimos de *proxy*, normalmente usados por usuários residenciais. Abusos nessas portas indicam tentativas de explorar problemas de configuração de *proxies* residenciais. De maneira similar, o *software* `Squid` é muito utilizado para implementar *proxies* em redes corporativas, e o abuso na porta associada a esse serviço (3128/TCP) pode indicar tentativas de explorar servidores mal configurados.

Chamaram atenção, ainda, as atividades em portas associadas a *proxies* instalados por códigos maliciosos, como o `MyDoom` e o `Sobig.f`. Isto indica que *spammers* estão ativamente utilizando, para o envio de *spam*, máquinas previamente infectadas.

Embora a porta mais utilizada para injetar *spams* tenha sido a porta 1080/TCP, associada ao *proxy* SOCKS, na Figura 4.8 pode-se ver que o maior pico de abuso foi registrado pela porta 8080/TCP, que recebeu mais de 25 milhões de *spams* no mês de agosto de 2006.

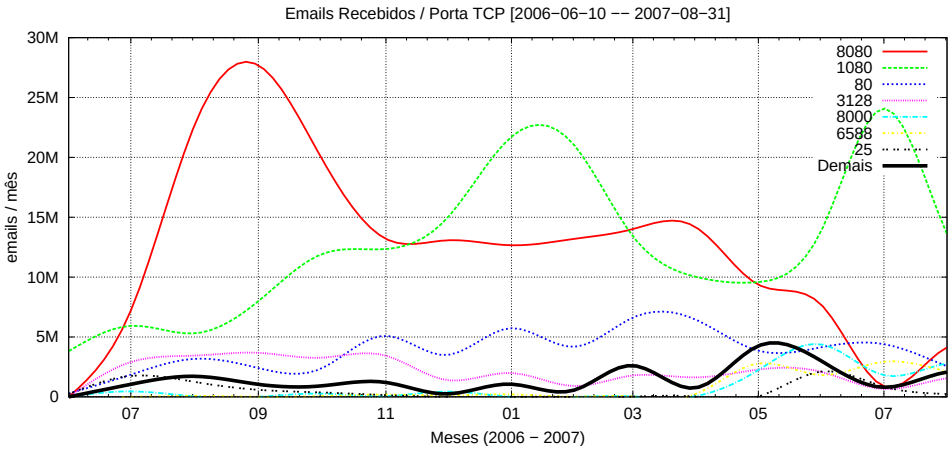


Figura 4.8 - *Emails* recebidos em função da porta TCP, ao longo do período.

Também é interessante notar que o abuso da porta 8080/TCP, que está associada ao *proxy* HTTP, foi mais acentuado no ano de 2006. Já no ano de 2007 foi possível observar uma tendência de aumento do abuso da porta 1080/TCP, associada ao *proxy* SOCKS. Interessante, também, é o baixo volume de mensagens recebidas via porta 25/TCP, quando comparado com o abuso das portas associadas aos *proxies* HTTP e SOCKS.

A Figura 4.9 mostra os *emails* recebidos em função da porta TCP, percentualmente, ao longo do período de coleta.

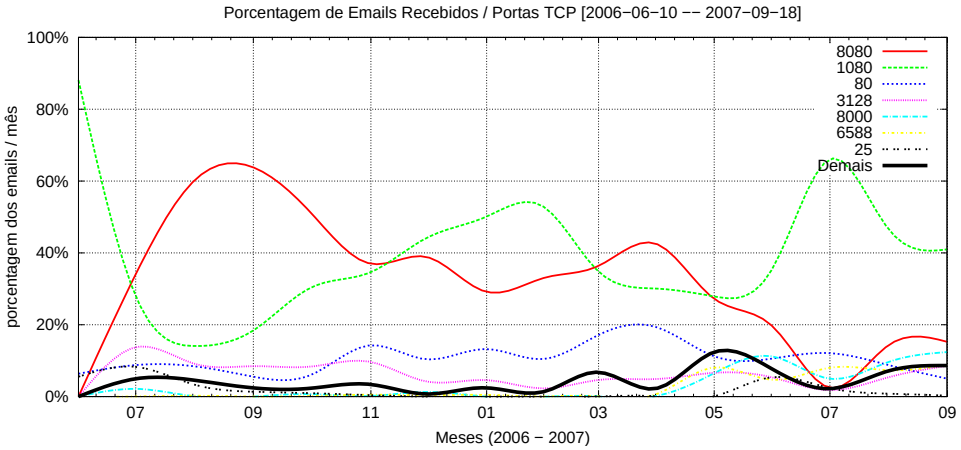


Figura 4.9 - *Emails* recebidos em função da porta TCP, percentualmente, ao longo do período.

Olhando a participação percentual no volume de mensagens recebidas, é interessante notar que a porta 80/TCP, porta padrão do serviço `http`, não só foi a terceira mais abusada, como manteve-se relativamente estável ao longo do período.

Os resultados da análise do abuso de portas TCP evidenciaram que foi muito importante a implementação, neste trabalho, do suporte ao protocolo SOCKS, pois a porta 1080/TCP, que é a porta padrão associada a este protocolo, foi a porta que mais recebeu *spams* em todo o período de coleta.

A Tabela 4.5 mostra, para cada um dos 10 países de onde mais se originaram *spams*, o número de IPs, ASNs e blocos CIDR `/24` únicos que geraram as atividades vistas. Esses valores foram calculados para cada categoria de abuso vista: *relay* SMTP, *proxy* SOCKS, *proxy* HTTP e *proxies* SOCKS e HTTP somados.

Tabela 4.5 - Distribuição de IPs, ASNs e blocos CIDR `/24` únicos em função do *Country Code* (CC) e tipo de abuso, ordenada pelo número de IPs.

Relay SMTP				
CC	IPs	/24	ASNs	*emails*
CN	20.949	14.274	85	1.559.919
KR	14.980	11.458	162	1.633.743
BR	11.614	5.549	71	349.282
US	8.158	7.372	715	2.008.960
TW	2.700	1.966	33	532.584
JP	1.797	1.709	96	29.989
DE	1.333	1.246	84	48.564
CA	743	675	75	13.852
HK	509	391	26	17.722
UA	266	235	70	14.415

Proxies HTTP e SOCKS				
CC	IPs	/24	ASNs	*emails*
TW	90.569	4.798	22	384.657.172
CN	5.400	1.196	31	81.324.723
BR	1.567	481	17	514.375
JP	1.455	228	11	5.322.630
US	1.162	579	113	25.703.729
DE	209	154	18	886.998
HK	205	50	9	4.366.258
KR	184	35	7	2.459.622
CA	61	24	11	6.670.760
UA	20	10	7	1.792.714

Proxy SOCKS				
CC	IPs	/24	ASNs	*emails*
TW	58.081	3.738	19	126.880.528
CN	3.036	882	30	33.610.059
JP	1.455	228	11	5.322.630
BR	1.316	433	17	198.116
US	1.056	542	109	22.030.234
DE	203	149	17	884.906
KR	179	32	6	2.458.838
HK	157	44	9	3.384.258
CA	61	24	11	1.860.900
UA	20	10	7	1.792.714

Proxy HTTP				
CC	IPs	/24	ASNs	*emails*
TW	75.279	3.803	20	257.776.644
CN	4.064	473	13	47.714.664
BR	326	125	1	316.259
US	253	111	41	3.673.495
HK	122	35	5	982.000
KR	18	10	4	784
DE	15	11	3	2.092
CA	13	3	3	4.809.860
UA	0	0	0	0
JP	0	0	0	0

Uma informação interessante que pode ser extraída da análise da Tabela 4.5 é a diferença no número de endereços IP únicos envolvidos no abuso de *relays* e de

proxies. Com excessão de Taiwan, o número de IPs únicos, de cada país, originadores de *spam* via o abuso de *relays* abertos, é muito maior do que o número de IPs abusando *proxies*. Mesmo assim, o volume de *emails* vindos destes países via abuso de *proxy* é maior do que via abuso de *relay*. Esse comportamento pode ser uma evidência de que o abuso de *relays* está partindo de computadores comprometidos, provavelmente fazendo parte de uma *botnet* de envio de *spam* (ou *spambot*). Esse comportamento é consistente com os resultados do trabalho discutido na seção 2.4.5.

Ao analisar a razão entre o número de *emails* recebidos e o número de IPs únicos por país, pode-se ver uma concentração muito maior de mensagens por IP nos casos envolvendo abuso de *proxy*. Se calcularmos esta razão para Taiwan, China e Estados Unidos, que foram os três países que mais originaram *spam*, pode-se ver um volume muito maior de *emails* por IP nos casos de abuso de *proxy*, do que nos casos de *relay*. No caso do abuso de *proxy* a razão *emails*/IP obtida foi: TW 4.247, CN 15.060 e US 22.120. Já no caso do abuso de *relay* a razão obtida foi: TW 197, CN 74 e US 246.

Isso mostra que Taiwan, embora seja o único país com mais endereços IP abusando *proxies* do que *relays*, mantém a proporção entre *emails* e IPs consistente com os demais países. A diferença da razão entre mensagens e IPs únicos por país, dependendo do tipo de abuso, reforça a percepção de que máquinas comprometidas, participando de *botnets* que enviam *spam*, mandam um volume baixo de *emails*. Esses dados também são consistentes com os resultados do Módulo *Harvesting*, que serão apresentados na próxima seção. Esta análise da Tabela 4.5 também aponta que as máquinas abusando de *proxies*, por enviarem um volume maior de mensagens, possam ser, em parte, máquinas hospedadas em *data centers* e controladas por *spammers* para enviar mensagens via *proxies* abertos.

Foram analisadas, também, as requisições recebidas pelos módulos de *proxy* HTTP e SOCKS. Para o módulo de HTTP, as requisições foram divididas em: conexão para porta 25/TCP de destino, conexão para demais portas de destino, requisição "*GET*" (tentativa de usar o *proxy* tipicamente para acessar páginas *Web*) e erros (comandos inválidos). Para o módulo SOCKS efetuou-se divisão similar: conexão para porta 25/TCP de destino, conexão para demais portas de destino e erros. Os resultados são mostrados na Tabela 4.6.

É importante destacar que a grande maioria das requisições foram tentativas de conexão para máquinas externas, na porta de SMTP (25/TCP). No caso do *proxy*

Tabela 4.6 - Requisições efetuadas aos módulos HTTP e SOCKS.

Módulo	Tipo	Requisições	%
HTTP	conexão para 25/TCP	89.496.969	97,62
	conexão para demais	106.615	0,12
	get	225.802	0,25
	erros	1.847.869	2,01
	total	91.677.255	100,00
SOCKS	conexão para 25/TCP	46.776.884	87,31
	conexão para demais	1.055.081	1,97
	erros	5.741.908	10,72
	total	53.573.873	100,00

HTTP as solicitações de conexão na porta 25/TCP, de máquinas externas, representaram 97,62% das requisições. Já no caso do *proxy* SOCKS estas solicitações representaram 87,31% do total. Esses resultados evidenciam que o objetivo da maioria das conexões para esses *honeypots* era a entrega de *spam*, e não outro tipo de abuso associado a *proxies* abertos, como por exemplo acesso a *Web*.

Também observou-se que, no caso das requisições recebidas pelo *proxy* SOCKS, foi grande o número daquelas que geraram erro. Os erros eram gerados, em sua maioria, por requisições que utilizavam o protocolo HTTP e não o SOCKS.

Outra análise realizada, foi com base no sistema operacional de origem dos IPs que injetaram *emails*.

Para esta análise levou-se em conta o início de conexão (TCP SYN), das sessões iniciadas para o envio de *spam*, para cada uma das portas emuladas. Utilizou-se a técnica de *fingerprint* passivo, que tenta fazer uma estimativa do sistema operacional que está originando uma conexão em função de características da pilha TCP/IP de cada sistema operacional, tais como TCP *window size*, *Time to live* (TTL), presença do *bit Don't Fragment* (DF) e tamanho do pacote.

Na Figura 4.10 é possível ver a distribuição das famílias de sistemas operacionais de origem ao longo do período e sua contribuição em termos de *emails* por mês.

Na categoria Windows estão agrupadas todas as versões de sistemas operacionais Windows® identificadas, como XP, 2000, 98, NT, entre outros. Já na categoria Unix foram incluidos os diversos sistemas Unix, tais como Linux, a família BSD, Solaris, Mac OS X, entre outros.

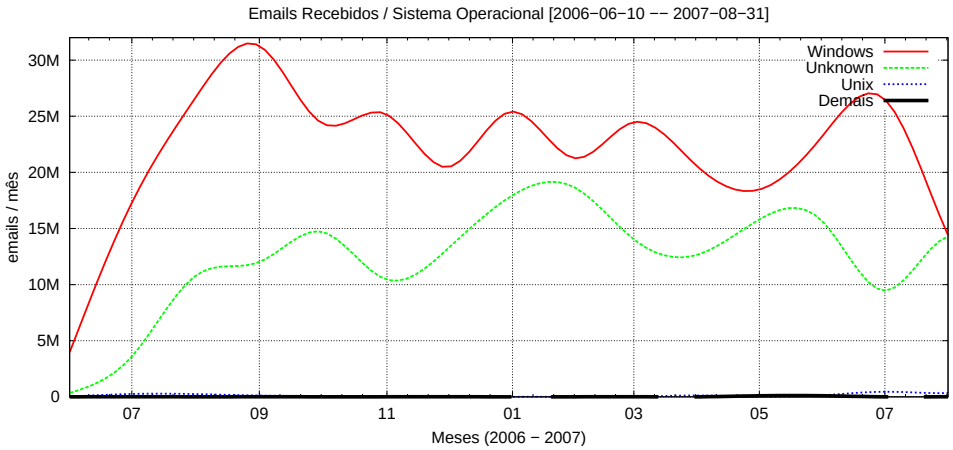


Figura 4.10 - *Emails* recebidos em função de sistema operacional de origem, ao longo do período.

A Figura 4.11 mostra a contribuição percentual de cada família de sistemas operacionais sobre o volume de *emails* capturados, ao longo do período.

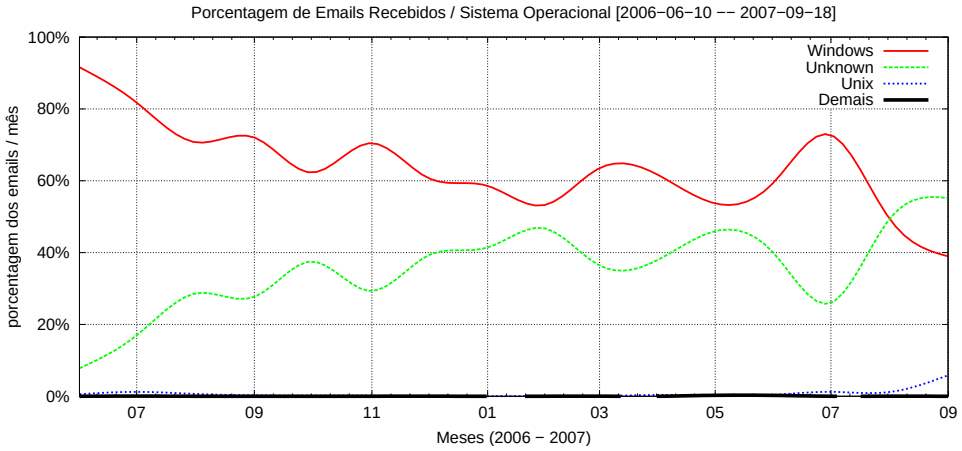


Figura 4.11 - *Emails* recebidos em função de sistema operacional de origem, percentualmente, ao longo do período.

Nota-se, em ambas as Figuras, que são predominantes conexões originando *spam* vindas de sistemas operacionais da família Windows®. Chama a atenção o baixo número de máquinas com sistemas Unix como fontes de *spam*, sendo que sua participação percentual só aumenta levemente no final do período de coleta.

Estes números são compatíveis com a hipótese de que grande número dessas conexões provêm de máquinas Windows® comprometidas, como por exemplo *spambots*, além do fato do sistema Windows possuir uma grande base instalada.

Na categoria "*Unkown*" estão as conexões que não puderam ter seus sistemas operacionais estimados. Uma explicação para o grande número de conexões nesta categoria poderia ser que, na época da coleta, as assinaturas utilizadas ainda não estavam atualizadas o suficiente para detectar sistemas mais novos, como por exemplo Windows® Vista. Outra possibilidade é que *spammers* estão usando de artifícios para mascarar características da pilha TCP/IP de máquinas sob o seu controle, com o objetivo de enganar mecanismos de *fingerprint* passivo.

4.2 Resultados do Módulo *Harvesting*

Nesta seção são mostrados os resultados da análise dos *spams* coletados pelo módulo *Harvesting*. Foram levadas em conta as características do processo de *harvesting* observado, as sessões SMTP estabelecidas com o servidor SMTP implementado e as mensagens recebidas nos domínios estudados.

Na Tabela 4.7 são apresentados os períodos em que os domínios criados para o estudo estiveram ativos e algumas estatísticas gerais dos dados coletados.

Tabela 4.7 - Estatísticas gerais dos dados coletados no módulo *harvesting*.

	`lss.inpe.br`	`.com.br`	`.org`
Entrada em operação	24/01/2008	08/02/2008	21/01/2008
Primeiro *email*	20/02/2008	22/03/2008	22/03/2008
Último *email*	01/05/2008	01/05/2008	01/05/2008
Total de *emails*	221	385	372
Destinatários únicos	35	46	45
IPs de entrega únicos	218	377	368
IPs de *harvesting* únicos	4	4	3

Chama a atenção o longo período decorrido entre a entrada em operação de um domínio e a data em que o primeiro *email* foi recebido: 29 dias no `lss.inpe.br`, 44 dias no `.com.br` e 62 dias no `.org`. Mesmo tendo demorado mais para começar a receber *spams*, os domínios `.com.br` e `.org` receberam mais mensagens no total do que o domínio `lss.inpe.br`. Também é interessante notar que poucos endereços IP foram responsáveis pelo *harvesting* dos *emails*, e nenhum IP esteve envolvido no

harvesting de mais de um domínio. Destes 11 IPs únicos, 8 pertenciam a redes de provedores de banda larga internacionais. Já com relação aos IPs que entregaram mensagens para os domínios, 954 IPs foram únicos, sendo que apenas 9 entregaram mensagens para mais de um domínio.

Com relação aos arquivos que tiveram seus endereços de *email* coletados, foi observado que os únicos de onde não foi coletado nenhum endereço foram os arquivos `.pdf` e `.tar`. Receberam *spam* endereços que constavam em comentários das páginas HTML, em elementos de metadados, em áreas restritas dos *sites*, assim como foram coletados tanto endereços que constavam em campos `mailto` quanto aqueles que eram apenas parte do texto das páginas. Já com relação ao tipo de endereço de *email*, tanto os estáticos quanto os dinâmicos receberam *spam*.

Quanto ao métodos de ofuscação empregados, todos tiveram efetividade. Desde os mais simples, como trocar o caracter "`@`" por "`at`" e "`em`", até os mais complexos como o uso de JavaScript e o uso de imagens para apresentar o endereço de *email*.

A análise do comportamento de *crawlers* legítimos, como por exemplo aqueles empregados por *sites* como o Google, mostrou que todos eles mantiveram-se na área do *site* sem restrições, respeitando as definições do arquivo `robots.txt`. Por outro lado, como esperado, todos IPs envolvido no *harvesting* realizaram acessos nas áreas restritas dos *sites*.

Outra análise realizada foi com base nas sessões SMTP estabelecidas com o servidor. As sessões foram divididas nas seguintes categorias:

Entrega: a sessão SMTP, iniciada por uma máquina remota, resultou na entrega de uma ou mais mensagens para um usuário do domínio atendido.

***Relay*:** a máquina de origem envolvida nesse tipo de sessão tinha por objetivo explorar um *relay* aberto, isto é, enviar um *email* para um endereço não atendido pelo servidor SMTP. Nestes casos era retornado um erro fatal de "*relaying denied*" ao cliente.

***Probe*:** sessão SMTP vazia, composta apenas de uma conexão na porta 25/TCP ou somente do início da conversação SMTP, seguido de um "`QUIT`" logo nos primeiros comandos. O objetivo desse tipo de sessão era provavelmente testar a presença de um servidor SMTP.

HTTP: sessões onde a máquina de origem não seguiu o protocolo SMTP, assumindo tratar-se de um servidor ou *proxy* HTTP, e enviou comandos como GET e CONNECT. Nos casos dos comandos CONNECT, todas as tentativas eram de conectar na porta 25/TCP de máquinas externas. Nesses casos o servidor SMTP retornava um erro para o cliente.

Demais: Nessa categoria foram registrados erros diversos, tais como tentativa de enviar *email* para endereços mal formados, *timeout* de DNS, entre outros.

Como os domínios `.com.br` e `.org`, criados para este módulo, eram atendidos pelo mesmo servidor SMTP, seus dados foram agrupados para esta análise.

A Tabela 4.8 sumariza as sessões SMTP que foram registradas, ordenadas pelo total de sessões dos 10 países mais ativos. Também são apresentados, para cada país, o total de endereços IP e ASs únicos.

Tabela 4.8 - Tipos de sessões SMTP por *Country Code* (CC).

domínio `lss.inpe.br`								
CC	IPs	ASs	Entrega	*Relay*	*Probe*	HTTP	Demais	Total
TW	62	4	8	51	49	1	0	109
ES	54	13	53	0	1	0	1	55
CN	17	7	5	12	1	0	9	27
FR	20	8	20	0	1	0	0	21
IT	20	5	20	0	0	0	0	20
KR	18	6	18	0	0	0	0	18
BR	17	9	18	0	0	0	0	18
DE	17	8	16	0	0	0	1	17
RU	5	3	2	0	1	0	9	12
US	10	8	10	1	1	0	0	12
domínios `.com.br` e `.org`								
CC	IPs	ASs	Entrega	*Relay*	*Probe*	HTTP	Demais	Total
TW	182	5	8	217	115	66	0	406
ES	162	18	162	0	2	0	2	166
IT	98	12	92	0	6	0	0	98
FR	56	10	56	0	0	0	0	56
BR	51	9	48	0	4	0	0	52
KR	48	11	47	0	1	0	1	49
PL	40	20	39	0	3	0	1	43
DE	40	9	41	0	0	0	0	41
CN	35	7	16	5	9	0	7	37
IL	28	5	26	0	2	0	0	28

Observando-se os dados é possível notar que Taiwan e China são praticamente

os únicos a tentar explorar *relays* abertos. Isso foi consistente tanto no domínio `lss.inpe.br` como nos domínios `.com.br` e `.org`.

Também chama a atenção que Taiwan está, se comparado com os demais países, muito mais ativo na condução de *probes* com a intenção de identificar servidores SMTP. O mesmo vale, ainda, para testes com comandos HTTP, como por exemplo CONNECT, indicando a intenção de localizar *proxies* HTTP abertos.

Nas sessões que resultaram em entrega de *email*, contudo, chamou a atenção a presença de endereços IP designados para países que até então não apareciam de modo significativo nos outros módulos implementados, tais como Espanha, França e Itália. Chama a atenção também que há, praticamente, um IP único por sessão de entrega, isto é, cada IP entrega para um endereço de *email* apenas. Essa pequena quantidade de *emails* por IP de origem, grande quantidade de endereços por país e aparente coordenação no envio, é consistente com a hipótese de uso de *bots* para o envio de *spam*.

Na Tabela 4.9 é mostrado o número de *emails* recebidos, por país de origem, nos domínios `lss.inpe.br`, `.com.br` e `.org`, implementados neste estudo. Também é exibido o número de IPs distintos que enviaram *spam*, além do número de ASs únicos ao qual pertenciam esses IPs.

Tabela 4.9 - *Emails* recebidos por CC.

domínio `lss.inpe.br`					domínio `.com.br`					domínio `.org`				
CC	IPs	ASs	*emails*	%	CC	IPs	ASs	*emails*	%	CC	IPs	ASs	*emails*	%
ES	52	12	53	24.0	ES	73	15	76	19.7	ES	84	14	85	22.8
IT	20	5	20	9.0	IT	48	9	48	12.5	IT	44	9	44	11.8
FR	20	8	20	9.0	KR	32	10	32	8.3	FR	28	9	28	7.5
KR	18	6	18	8.1	FR	28	7	28	7.3	DE	24	6	24	6.5
BR	17	9	18	8.1	BR	26	8	26	6.8	BR	21	6	22	5.9
DE	16	7	16	7.2	PL	19	11	20	5.2	PL	19	10	19	5.1
US	9	7	10	4.5	DE	17	7	17	4.4	KR	15	4	15	4.0
TW	8	3	8	3.6	IL	16	5	16	4.2	AR	13	6	13	3.5
TR	8	1	8	3.6	US	12	10	12	3.1	IL	10	5	10	2.7
PL	8	6	8	3.6	AR	11	6	12	3.1	UK	7	6	9	2.4

Nessa Tabela, chama a atenção o fato de que a Espanha aparece como o maior originador de *spam* destinado aos 3 domínios de estudo, tendo sido responsável por mais de de 20% do total de mensagens entregues. É importante lembrar que a

Espanha não aparece listada dentre os 10 países que mais enviam *spam* via abuso de *proxies* abertos, como visto na seção anterior. O mesmo é verdade para países como Itália e França. Essa diferença nos resultados deixa clara a multiplicidade de técnicas sendo empregadas atualmente pelos *spammers*, e a importância de se observar o tráfego de *spam* em múltiplos pontos da sua distribuição e não apenas no momento final da sua entrega.

Assim como nas sessões SMTP, a análise dos IPs envolvidos na entrega das mensagens também mostrou que praticamente cada mensagem foi recebida de um IP diferente, o que reforça a evidência de coordenação no processo de entrega, bem como uma evidência de máquinas comprometidas sendo controladas para esse fim. Outra hipótese, da utilização de apenas um IP por mensagem, é que *spammers* estejam utilizando essa técnica para minimizar a eficácia de filtros de bloqueio baseados em DNSBLs.

Na Tabela 4.10 é exibido o número de *emails* recebidos, em função do AS de origem, para os domínios estudados. Também é exibido o número de IPs distintos de cada um desses ASs.

Tabela 4.10 - Número de *emails* recebidos por AS.

domínio `lss.inpe.br`				
AS	Nome do AS	IPs	num	%
3352	Telefonica (ES)	21	22	10.0
3269	Telecom Italia (IT)	13	13	5.9
3320	Deutsche Tel (DE)	9	9	4.1
9121	Turk Tel (TR)	8	8	3.6
4766	Korea Tel (KR)	8	8	3.6
20838	YIF (ES)	6	6	2.7
27699	Telefonica (BR)	5	6	2.7
16338	AUNA Tel (ES)	5	5	2.3
12322	PROXAD (FR)	5	5	2.3
6739	Cableuropa (ES)	5	5	2.3

domínio `.com.br`				
AS	Nome do AS	IPs	num	%
3352	Telefonica (ES)	31	33	8.6
3269	Telecom Italia (IT)	29	29	7.5
4766	Korea Tel (KR)	15	15	3.9
16338	AUNA Tel (ES)	9	9	2.3
5617	TPNET Tel (PL)	8	9	2.3
7738	Telemar (BR)	8	8	2.1
8612	Tiscali (IT)	7	7	1.8
8551	Bezeqint (IL)	7	7	1.8
12322	PROXAD (FR)	6	6	1.6
15557	LDCOM (FR)	6	6	1.6

domínio `.org`				
AS	Nome do AS	IPs	num	%
3352	Telefonica (ES)	31	32	8.6
3269	Telecom Italia (IT)	27	27	7.3
15557	LDCOM (FR)	13	13	3.5
3320	Deutsche Tel (DE)	13	13	3.5
20838	YIF (ES)	11	11	3.0
27699	Telefonica (BR)	9	10	2.7
4766	Korea Tel (KR)	9	9	2.4
5617	TPNET Tel (PL)	8	8	2.2
12357	Comunitel (ES)	8	8	2.2
16338	Cableuropa (ES)	7	7	1.9

De maior interesse nessa Tabela é o fato de que os 10 ASs que mais originaram *spam*, para cada um dos domínios estudados, são quase todos associados a redes de banda larga, quer seja ADSL ou cabo. Isso reforça a hipótese de que as características do tráfego observado podem indicar o comportamento de máquinas de usuários domésticos infectadas por *bots*, sendo usadas para o envio de *spam*.

4.3 Resultados do Módulo *Pop-Up Spam*

Esta seção descreve os resultados do Módulo *Pop-Up Spam*, parte do Consórcio Brasileiro de *Honeypots*. Ele foi implementado para avaliar o volume de mensagens enviadas para as portas 1026/UDP, 1027/UDP e 1028/UDP, com enfoque nas mensagens que continham URLs.

A Figura 4.12 mostra um exemplo de pacote UDP contendo uma mensagem de *pop-up spam*, com destino a porta 1026/UDP.

```
U 2008/01/13 03:51:02.026909 135.115.19.138:30836 -> xxx.xxx.xxx.xxx:1026
  [...]
  c2 00 00 00 20 20 20 20    20 53 54 4f 50 21 20 49    Â...     STOP! I
  4d 4d 45 44 49 41 54 45    20 41 54 54 45 4e 54 49    MMEDIATE ATTENTI
  4f 4e 20 52 45 51 55 49    52 45 44 0a 0a 20 20 20    ON REQUIRED..
  57 69 6e 64 6f 77 73 20    68 61 73 20 66 6f 75 6e    Windows has foun
  64 20 43 52 49 54 49 43    41 4c 20 53 59 53 54 45    d CRITICAL SYSTE
  4d 20 45 52 52 4f 52 53    2e 0a 0a 20 44 6f 77 6e    M ERRORS... Down
  6c 6f 61 64 20 52 65 67    69 73 74 72 79 20 43 6c    load Registry Cl
  65 61 6e 65 72 20 66 72    6f 6d 3a 20 63 6c 65 61    eaner from: clea
  6e 65 72 36 34 2e 63 6f    6d 0a 0a 46 41 49 4c 55    ner64.com..FAILU
  52 45 20 54 4f 20 41 43    54 20 4e 4f 57 20 4d 41    RE TO ACT NOW MA
  59 20 4c 45 41 44 20 54    4f 20 44 41 54 41 20 4c    Y LEAD TO DATA L
  4f 53 53 20 41 4e 44 20    43 4f 52 52 55 50 54 49    OSS AND CORRUPTI
  4f 4e 21 0a 0a 00 00 00    00 00 00                   ON!........
```

Figura 4.12 - Exemplo de um pacote UDP contendo uma mensagem de *pop-up spam*.

Na Figura, chama a atenção que a mensagem induz o usuário a fazer um *download* de um programa do site `cleaner64.com`. Como esta mensagem seria exibida via uma janela do sistema operacional, e ela afirma que teria sido encontrado um erro crítico no sistema, um usuário leigo poderia ser levado a acreditar na mensagem e fazer o *download* do programa.

Caso a página seja acessada, uma mensagem é mostrada informando um número arbitrário de supostos erros de sistema encontrados. A página também apresenta

informações sobre supostas melhorias que seriam feitas no sistema, incluindo a remoção de problemas no *Registry* do Windows, caso fosse realizado o *download* e instalação do *software*. A Figura 4.13 mostra o resultado da execução de alguns antivírus[2] no arquivo executável obtido deste *site*.

```
File setup.exe received on 05.19.2008 21:47:38 (CET)

Antivirus       Version         Last Update     Result
-------------------------------------------------------------------------------
AntiVir         7.8.0.19        2008.05.19      DR/FraudTool.RegSort.A
Avast           4.8.1195.0      2008.05.19      Win32:Adware-gen
Fortinet        3.14.0.0        2008.05.19      Misc/RegSort
GData           2.0.7306.1023   2008.05.19      Win32:Adware-gen
Ikarus          T3.1.1.26.0     2008.05.19      not-a-virus:.FraudTool.Win32.RegSort.a
Kaspersky       7.0.0.125       2008.05.19      not-a-virus:FraudTool.Win32.RegSort.a
Prevx1          V2              2008.05.19      Malicious Software
Sophos          4.29.0          2008.05.19      Sus/ComPack
Symantec        10              2008.05.19      RegSort
Webwasher       6.6.2           2008.05.19      Trojan.Dropper.FraudTool.RegSort.A
```

Figura 4.13 - Análise de um executável obtido via *pop-up spam*.

Pelo resultado da execução dos antivírus, pode-se ver que o executável é apontado como malicioso. Embora não exista unanimidade nestes resultados, o maior número de análises similares aponta como sendo relacionado a fraude (`FraudTool`).

Em 15 meses de coleta de dados, foram capturadas 13.083.540 mensagens de *pop-up spam*. A Figura 4.14 mostra a distribuição, ao longo do período, da presença das 10 URLs mais freqüentes, de um total de 55 URLs, e da soma das demais URLs.

A maior parte destes *sites* tem objetivos similares ao do *site* `cleaner64.com`, ou seja, tentam convencer o visitante que ele tem um problema e que deve realizar o *download* de um arquivo executável para resolvê-lo. Alguns programas prometem, inclusive, impedir que novas mensagens de *pop-up spam* sejam recebidas pelo visitante.

Observar e entender o abuso deste tipo de serviço é importante, pelo seu potencial de uso para outros ataques. Uma mensagem destas, completa, pode ser enviada em um único pacote UDP que, por não ser orientado a conexão, permite que estes pacotes sejam enviados com endereços de origem forjados. Desta forma, é mais difícil localizar o endereço IP do remetente da mensagem, tornando o envio praticamente anônimo.

[2] Arquivo submetido ao serviço *on-line* http://www.virustotal.com/ em 19/05/2008.

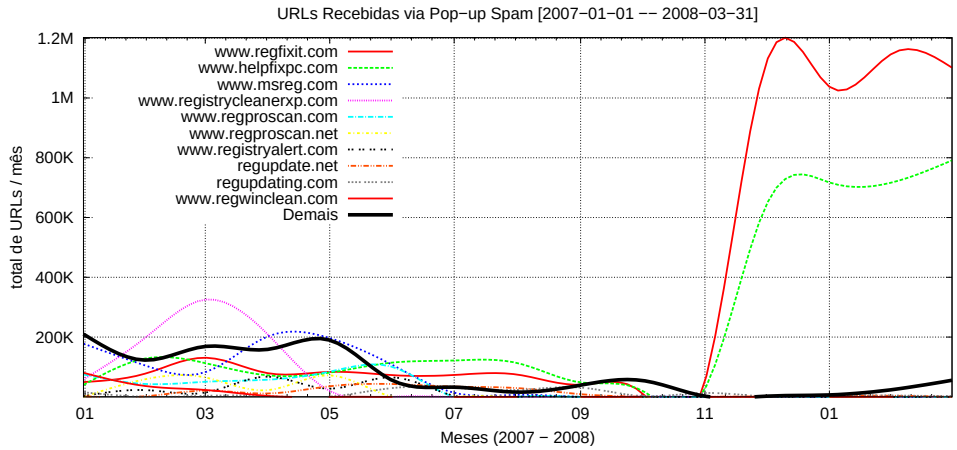


Figura 4.14 - Distribuição de URLs presentes em pacotes de *pop-up spam* ao longo do período de coleta.

Outra característica importante deste tipo de abuso é a relativa credibilidade de uma mensagem recebida via o serviço *Windows Messenger*. Como esta mensagem aparece em uma janela do sistema operacional, um usuário leigo tende a dar maior credibilidade ao seu conteúdo. Estas características tornam este tipo de mensagem um potencial vetor para disseminação de cavalos de tróia relacionados com fraudes financeiras e páginas de *phishing*.

4.4 Resultados de Correlações com o Consórcio Brasileiro de *Honeypots*

Nesta seção são mostrados os resultados da análise de dois anos de tráfego do Consórcio Brasileiro de *Honeypots*, com destino às portas relacionadas com envio de *spam*, observadas nos outros módulos, conforme apresentado nas seções anteriores. O objetivo dessa análise foi tentar correlacionar as varreduras por estas portas com as atividades que as abusaram para envio de *spam*.

Como mostrado na Tabela 4.11, foram analisadas varreduras observadas no Consórcio Brasileiro de *Honeypots* por um período de 2 anos. As varreduras capturadas se originaram de 591.045 diferentes endereços IP, alocados para 183 *Country Codes* de origem. O mapeamento entre um endereço IP e seu *Country Code* foi obtido através das informações de alocação de endereços IP, mantidas nos arquivos de estatísticas dos 5 *Regional Internet Registries* (RIR): AfriNIC, APNIC, ARIN, LACNIC e RIPE NCC.

Tabela 4.11 - Estatísticas gerais referentes às varreduras contra o Consórcio Brasileiro de *Honeypots*.

Início do período de análise	01/01/2006
Fim do período de análise	31/12/2007
Dias analisados	730
Total de inícios de conexão	57.379.718
IPs únicos	591.045
ASs únicos	7.959
Países (*Country Codes*) de origem	183

Para esta análise foram consideradas apenas as conexões com destino às portas associadas aos serviços SMTP (porta 25/TCP), *proxy* HTTP (portas 8000/TCP, 3382/TCP, 6588/TCP, 80/TCP, 8080/TCP, 81/TCP, 3128/TCP e 4480/TCP) e *proxy* SOCKS (1080/TCP e 3127/TCP). Foi desconsiderado o tráfego para IPs que estivessem emulando algum dos serviços acima, de modo a capturar apenas tráfego de varreduras e não tráfego relacionado à interação com *softwares* de emulação.

Uma análise realizada procurou identificar diferenças de comportamento entre IPs de origem com relação ao número de máquinas de destino varridas. Para tanto, todos os IPs de origem de varreduras contra o consórcio foram agrupados em função do número diário de máquinas distintas de destino que estavam sendo afetadas. Foram definidos os limites inferiores de 1, 10 e 100 endereços IP de destino diários.

Na Tabela 4.12 são mostrados para cada limite inferior: o número de IPs de origem que realizaram varreduras em número igual ou superior ao limite inferior dado e a porcentagem com relação ao total de IPs de origem; e o número de inícios de conexão (SYN) gerados pelos IPs que se enquadraram no limite estipulado e sua porcentagem com relação ao total de inícios de conexão.

Tabela 4.12 - Número de IPs de origem e inícios de conexão, em função do número diário de máquinas de destino.

	IPs destino ≥ 1		IPs destino ≥ 10		IPs destino ≥ 100	
IPs origem	591.045	100,00%	48.054	8,13%	15.611	2,64%
SYNs	57.379.718	100,00%	49.243.623	85,82%	43.042.452	75,01%

É interessante observar que mais de 90% de todos os IPs observados varreram, individualmente, menos de 10 máquinas de destino, por dia. Esse número relativamente

pequeno de máquinas de destino, em cada varredura, chama a atenção e sugere que pode haver um grau de coordenação nas varreduras. Do ponto de vista de volume de tráfego gerado, entretanto, é possível ver uma grande concentração: apenas 8,13% dos IPs, observados varrendo 10 ou mais máquinas de destino por dia, geraram 85,82% do tráfego. Quando analisados os IPs com 100 ou mais máquinas de destino por dia, o total de IPs cai para apenas 2,64% do total de IPs, representando 75,01% do total de tráfego.

A Tabela 4.13 mostra os IPs envolvidos em varreduras para as portas de SMTP, *proxy* HTTP e *proxy* SOCKS, em função do número de dias que em que eles foram observados nos *honeypots* do Consórcio. Nesta Tabela, os IPs de origem também foram agrupados com relação ao número de IPs de destino varridos por dia, porém, são apresentados números e porcentagens com relação ao total de IPs que se enquadraram em cada limite estipulado.

Tabela 4.13 - Freqüência, em dias, com que os IPs realizando varreduras foram observados, em função do número diário de máquinas de destino.

	IPs destino ≥ 1			IPs destino ≥ 10			IPs destino ≥ 100		
Dias	IP	% IP	% SYN	IP	% IP	% SYN	IP	% IP	% SYN
1	483.359	81,78	21,71	37.064	77,13	21,02	12.093	77,46	22,19
2-5	86.779	14,68	21,55	9.349	19,46	20,01	2.993	19,17	18,71
6-10	11.204	1,90	7,24	972	2,02	6,96	281	1,80	6,19
11-50	8.609	1,46	15,23	591	1,23	15,32	207	1,33	14,68
51-100	742	0,13	6,94	46	0,10	6,55	24	0,15	7,91
101-200	280	0,05	9,39	23	0,05	10,96	10	0,06	10,07
201-300	44	0,01	2,03	4	0,01	1,51	1	0,01	0,97
301-400	14	0,00	0,26	0	0,00	0,00	1	0,01	1,36
401-500	8	0,00	13,74	3	0,01	17,07	1	0,01	17,92
>500	6	0,00	1,90	2	0,00	0,61	0	0,00	0,00

Esta Tabela mostra um dado interessante com relação à frequência com que um mesmo IP foi observado realizando varreduras: considerando a totalidade dos IPs (IPs destino ≥ 1), 81,78% deles foram vistos apenas um dia durante todo o período analisado. Uma hipótese para esse fato seria a existência de uma coordenação na varredura: um número muito grande de máquinas, realizando varreduras por curto período de tempo com destino a poucos endereços IP, sob coordenação de um mesmo atacante, que combina os resultados posteriormente. Outra hipótese seria que essa grande quantidade de IPs de origem fosse de computadores infectados, conectados

via redes de banda larga residenciais com IP dinâmico. Por trocarem de endereço IP com freqüência, mesmo que realizem varreduras por um período maior, o IP registrado nos sensores seria diferente após cada troca.

A Tabela 4.14 apresenta o percentual de IPs de origem e o percentual de inícios de conexão (SYN) para as portas TCP de destino das varreduras. Os IPs de origem também foram agrupados com relação ao número de IPs destino varridos por dia e os números e porcentagens são calculados com relação ao total de IPs que se enquadraram em cada limite estipulado.

Tabela 4.14 - Portas TCP de destino das varreduras, ordenadas pelo número de inícios de conexão (SYN), em função do número diário de IPs de destino.

IPs destino ≥ 1			IPs destino ≥ 10			IPs destino ≥ 100		
Porta	% IP	% SYN	Porta	% IP	% SYN	Porta	% IP	% SYN
1080	2,46	33,19	1080	21,66	38,24	1080	42,28	40,68
80	73,77	24,62	6588	3,91	20,71	6588	7,96	23,33
6588	0,47	17,92	80	60,85	17,51	80	27,52	11,84
25	18,80	11,99	25	11,73	9,83	25	20,07	10,53
8080	2,11	6,23	8080	10,09	7,10	8080	15,93	7,21
3128	0,98	3,69	3128	6,44	4,11	3128	8,76	4,06
3127	4,18	0,96	8000	1,72	1,03	8000	2,64	0,99
8000	0,23	0,90	3127	2,30	0,90	3127	3,13	0,84
4480	0,08	0,22	4480	0,51	0,25	4480	0,89	0,25
81	0,12	0,18	81	0,66	0,20	81	0,85	0,18
3382	0,04	0,09	3382	0,30	0,10	3382	0,38	0,09

Assim como ocorreu no módulo *Proxy/Relay* Aberto, a porta com maior atividade foi a 1080/TCP, associada ao *proxy* SOCKS. Nota-se, também, que o percentual de conexões é maior conforme o número de IPs de destino levados em consideração aumenta. Já a porta 80/TCP, associada ao serviço HTTP, teve uma maior atividade no Consórcio do que no módulo *Proxy/Relay* Aberto. Essa diferença pode ser pelo fato de que os *honeypots* do Consórcio recebem varreduras não só à procura de *proxies* abertos, mas também à procura de servidores *Web* vulneráveis, por exemplo. No caso do módulo *Proxy/Relay* Aberto foram contabilizadas somente as mensagens efetivamente injetadas via porta 80/TCP. Também é interessante notar que a porta 6588/TCP sofreu varreduras intensas nos *honeypots* do consórcio, mas não figurou entre as portas mais abusadas para o envio de *spam* no módulo *Proxy/Relay* Aberto.

A Tabela 4.15 contém os *Country Codes* (CC) que mais realizaram varreduras, contra

as portas de SMTP e *proxies* SOCKS e HTTP. São apresentados, para cada CC, o percentual de IPs e o percentual de inícios de conexão (SYN).

Tabela 4.15 - *Country Codes* realizando varreduras, ordenados por inícios de conexão, com IPs diários de destino ≥ 10, em função do tipo de varredura.

Porta SMTP			Portas SOCKS			Portas HTTP		
CC	% IP	% SYN	CC	% IP	% SYN	CC	% IP	% SYN
TW	53,74	47,63	MC	0,01	16,80	CN	13,80	39,94
KR	13,15	21,76	US	23,29	16,10	BR	26,96	12,75
CN	8,91	11,85	IR	1,30	13,36	TW	8,41	12,05
US	7,33	5,36	TW	26,84	10,71	US	12,26	9,59
BR	6,49	3,94	KR	8,99	8,95	RU	1,66	6,33
AR	1,33	3,92	EE	0,18	8,42	GB	1,37	2,32
IT	0,64	1,26	CN	7,25	4,85	KR	3,54	2,07
RU	0,85	0,59	UA	0,20	3,67	DE	1,96	1,66
DE	0,98	0,29	DE	3,00	2,84	EE	0,26	1,40
MX	0,90	0,26	CY	0,06	1,96	CA	1,67	0,99

É expressiva a participação de Taiwan nas varreduras por porta SMTP. As varreduras vindas deste país representaram 47,63% de todas as conexões e mais da metade dos endereços IP de origem observados. Este resultado é consistente com o apresentado no módulo *Harvesting*, onde Taiwan representou a grande maioria das tentativas de abusar *relays* abertos. Também é interessante a participação de Mônaco (MC), que foi o país que mais originou tráfego para portas associadas a *proxy* SOCKS. Porém, mesmo fazendo varreduras de forma expressiva por estas portas, seus endereços IP de origem representaram apenas 0,01% do total. Na Figura 4.15 pode-se ver que esta atividade foi concentrada no ano de 2006, mas ainda assim levou Mônaco a figurar entre os países que mais originaram tráfego contra as portas estudadas, em todo o período.

Também chama a atenção que, quando somadas as varreduras para todas as portas, a China passa a ser a maior originadora, com uma concentração maior no ano de 2007. Também é interessante que, ao contrário do que é visto nos resultados sobre o abuso efetivo de *proxies*, o Brasil foi o quarto país que mais originou varreduras contra as portas de SMTP, HTTP e SOCKS. Não são claros os motivos para esta diferença, mas uma hipótese seria que as varreduras por estas portas podem ter outros objetivos que não somente a identificação de *proxies* e *relays* abertos para o envio de *spam*.

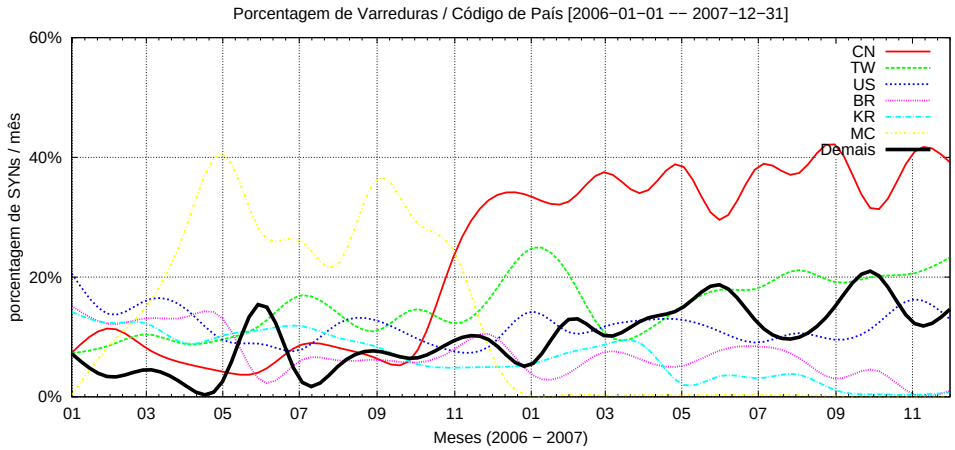


Figura 4.15 - Varreduras com IPs diários de destino ≥ 10, para todas as portas de SMTP, HTTP e SOCKS, por CC de origem.

A Figura 4.16 apresenta o percentual de varreduras recebidas pelas portas TCP estudadas, ao longo do período analisado. Foram consideradas, para esta Figura, somente as varreduras com mais de 10 IPs diários de destino.

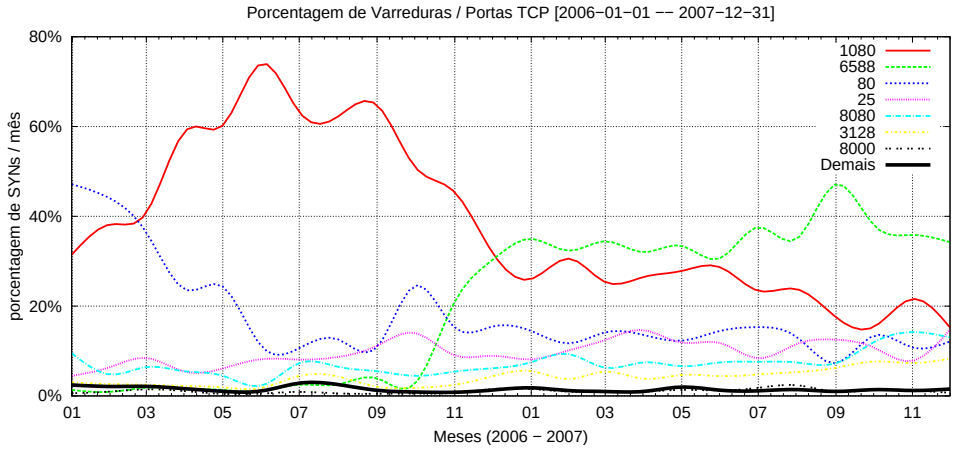


Figura 4.16 - Varreduras recebidas, com IPs diários de destino ≥ 10, em função das portas de SMTP, HTTP e SOCKS.

A porta 1080/TCP, associada ao protocolo SOCKS, foi a que mais recebeu varreduras no total. É interessante notar que o período em que representou o maior percentual de varreduras coincide com o período em que os IPs de Mônaco fizeram varreduras por *proxies* SOCKS. Também chamou a atenção que as varreduras pela

porta 6588/TCP predominaram no ano de 2007, período em que o maior percentual de varreduras veio da China. Do mesmo modo, chamou a atenção que as varreduras pela porta 80/TCP, que é a terceira de maior volume, representavam inicialmente uma grande parcela do total, mas a partir do segundo semestre de 2006 decresceram e se mantiveram abaixo dos 20% em todo o ano de 2007.

A Tabela 4.16 apresenta, para varreduras por SMTP, SOCKS e HTTP com mais 10 IPs diários de destino, quais os *Autonomous Systems* (AS) que mais originaram tráfego.

Tabela 4.16 - ASs realizando varreduras para as portas de SMTP, SOCKS e HTTP, com IPs diários de destino ≥ 10.

Porta SMTP				Portas SOCKS e HTTP			
ASN	Nome do AS	% IP	% SYN	ASN	Nome do AS	% IP	% SYN
3462	HINET (TW)	25,64	22,61	4134	CHINANET (CN)	5,45	20,32
4766	Korea Tel (KR)	6,90	18,18	3462	HINET (TW)	8,60	9,57
7482	APOL-AS (TW)	12,08	10,29	6758	Monaco Tel (MC)	0,00	7,29
4812	CHINANET (CN)	0,44	7,11	24435	SUPERNET (PK)	0,01	5,58
4780	SEEDNET (TW)	6,49	6,27	3249	ESTPAK (EE)	0,11	3,79
18182	SONET-TW (TW)	6,76	4,49	4766	Korea Tel (KR)	2,47	2,94
9924	TFN-TW (TW)	1,85	3,30	4837	CHINA169 (CN)	2,88	2,17
16814	NSS (AR)	0,09	2,73	8167	BRT (BR)	8,15	1,49
4808	CHINA169 (CN)	4,33	2,27	34320	MNW-AS (UA)	0,02	1,46
27699	Telefonica (BR)	3,21	1,83	1251	ANSP (BR)	0,21	1,36

A diferença de origem entre varreduras por *proxies* e *relays* é consistente com a observada quando analisados os *Country Codes* de origem. Vemos que ASs de Taiwan originaram a maior parte do tráfego de varreduras por *relays* SMTP, enquanto ASs de China e Mônaco originaram a maior parte das varreduras por *proxies* abertos.

Após analisar estes dados do Consórcio e correlacioná-los com os módulos *Harvesting* e *Relay/Proxy* aberto, notou-se uma diferença no comportamento de IPs originados de Taiwan. Embora tenha participado nas varreduras por *proxy* no mesmo patamar de outros países, foi muito mais agressivo no seu abuso que outros países. Também foi interessante que vários países que se destacaram no processo de varredura, não tiveram participação expressiva no abuso de *proxies* ou *relays*. Duas hipóteses possíveis são: que as varreduras são coordenadas e feitas propositalmente a partir de vários países; ou que esses países estão à procura de *proxies* abertos para realização de outras atividades, e não para o envio de *spam*.

4.5 Proposta de Mitigação de Alguns Problemas Observados

Os números observados no módulo *Relay/Proxy* aberto evidenciam que, apesar da diversidade de portas de *proxy* sendo abusadas, o objetivo em comum da maioria das conexões para aqueles *honeypots* era a entrega direta de *spam*, através de conexões de saída com destino à porta 25/TCP de servidores de *email*, como discutido na Seção 4.1.

Já os resultados do módulo *harvesting* mostraram que *spammers* varrem não só páginas HTML à busca de endereços de *email*, mas também arquivos em diversos formatos. Mostraram, também, que algumas técnicas de ofuscação de *emails* demonstraram eficácia.

Esta seção discute algumas medidas que podem ser tomadas para reduzir o abuso de *proxies* abertos e a possibilidade de um endereço de *email*, publicado em um *site*, ser obtido por *spammers*.

4.5.1 Políticas e Padrões Existentes para Mitigação do Problema de Entrega Direta de *Emails*

Nos resultados do módulo *Relay/Proxy* aberto ficou claro que, por conta da diversidade de portas associadas a *proxies* e da possibilidade de utilização de novas portas, conter o abuso através da filtragem do tráfego de entrada para as portas associadas a *proxies* não parece ser efetivo. Já o bloqueio de conexões de saída com destino à porta 25/TCP dificultaria bastante o abuso. Nesta seção será discutido porque é tão fácil a entrega direta de mensagens e quais medidas podem ser tomadas.

O protocolo SMTP, definido originalmente pela RFC 821, de 1982, com revisão em 2001, na RFC 2821 (KLENSIN, 2001), foi concebido como um protocolo de transferência de mensagens entre servidores de *email*. Porém, também era usado como protocolo de submissão de mensagens entre cliente e servidor. Este uso para implementar ambas as funções era transparente quando cliente e servidor eram executados em uma mesma máquina. Atualmente, com a separação destas funcionalidades ficando mais clara, tornou-se necessária uma melhor definição dos papéis de submissão e transporte de *emails*.

A RFC 2476, de 1998, apresentou pela primeira vez o protocolo para *Message Submission*, que foi atualizado em 2006, na RFC 4409 (GELLENS; KLENSIN, 2006). Este

protocolo fornece um meio para distinguir a submissão do transporte de mensagens, permitindo assim:

- a aplicação de políticas diferentes para cada tipo de conexão, impedindo *relays* não autorizados ou introdução de *emails* não solicitados;
- a implementação de autenticação na submissão, incluindo aquela realizada remotamente por usuários autorizados;
- a possibilidade de implementar, futuramente, melhorias no serviço de submissão.

A adoção do protocolo de *Message Submission* é uma boa prática reforçada na RFC 5068 (BCP 134) (HUTZLER et al., 2007) e que tem sido recomendada por diversos fóruns de combate ao *spam*, como o *Messaging Anti-Abuse Working Group* (MAAWG), o *London Action Plan* e a Comissão de Trabalho Anti-Spam (CT-Spam) do Comitê Gestor da Internet no Brasil (CGI.br).

No documento *Managing Port 25 for Residential or Dynamic IP Space – Benefits of Adoption and Risks of Inaction* (MAAWG, 2005) o MAAWG recomenda para redes de caráter residencial, além da adoção de *Message Submission*, as seguintes medidas:

- requerer autenticação para a submissão de mensagens, como recomendado na RFC 4954 (SIEMBORSKI et al., 2007);
- não interferir no tráfego para a porta 587/TCP;
- configurar o *software* cliente de *email* para usar porta 587/TCP e autenticação;
- bloquear acesso de saída para porta 25/TCP a partir de todas as máquinas que não sejam MTAs ou explicitamente autorizadas.

Estas recomendações de gerência de porta 25/TCP já são adotadas por alguns membros do MAAWG, como Earthlink, AOL e Comcast. No caso da Comcast, após a adoção da medida, o número de *spams* barrados diariamente na saída da sua rede chegava a cerca de 700 milhões (KOLSTAD, 2005).

A Figura 4.17 ilustra a mudança de cenário em uma rede de caráter residencial, após a adoção de *Message Submission* pelos provedores de *email* dos usuários e a implementação de gerência de porta 25.

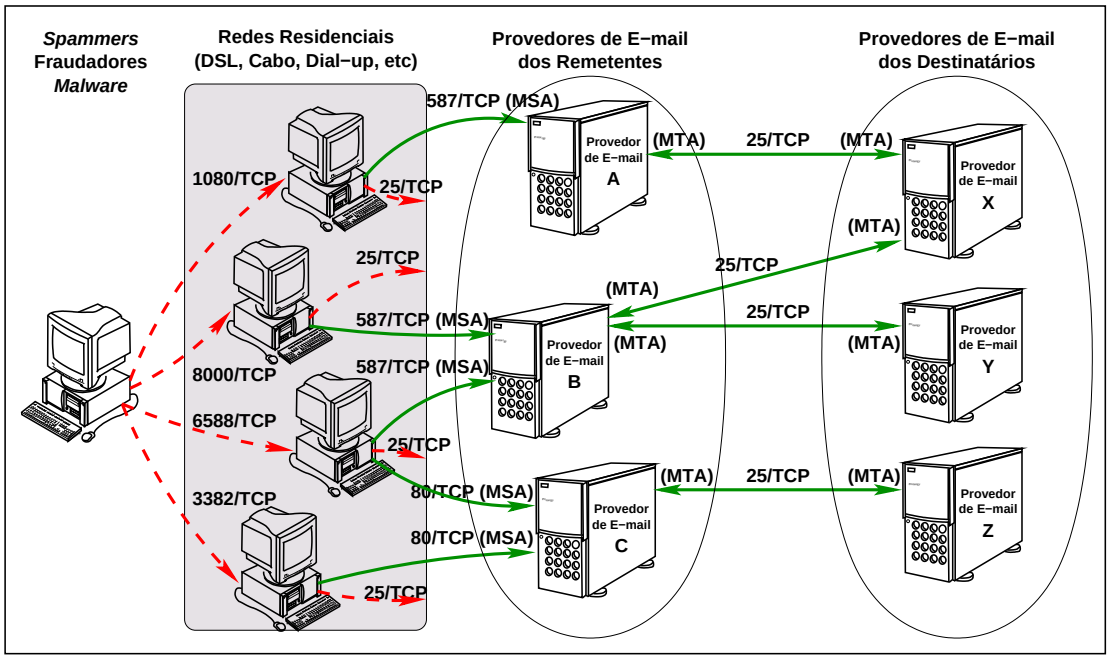


Figura 4.17 - Novo cenário, usando *Message Submission for Mail*, e seu impacto na entrega direta de *emails*.

Na Figura 4.17 nota-se que o usuário, conectado via uma rede residencial, envia *emails* normalmente via *Mail Submission Port* (587/TCP) ou via *Webmail* (80/TCP). Mas, os *spammers*, fraudadores e códigos maliciosos não conseguem mais fazer a entrega direta dos *emails* para os provedores de destino, pois a saída de conexões para porta 25/TCP é impedida.

Com este novo cenário provavelmente os *spammers* e fraudadores adaptarão suas ferramentas para furtar as credenciais do usuário e se autenticar em seu MSAs (*Mail Submission Agents*) para o envio de *spam*, como é ilustrado na Figura 4.18.

Mesmo ocorrendo esta mudança para uso das credenciais do usuário, outro reflexo da implementação de *Message Submission* e Gerência de Porta 25 em redes de perfil residencial é que as mensagens abusivas são mais rastreáveis para os provedores de acesso (MAAWG, 2005). Pois, além do provedor poder implementar políticas de con-

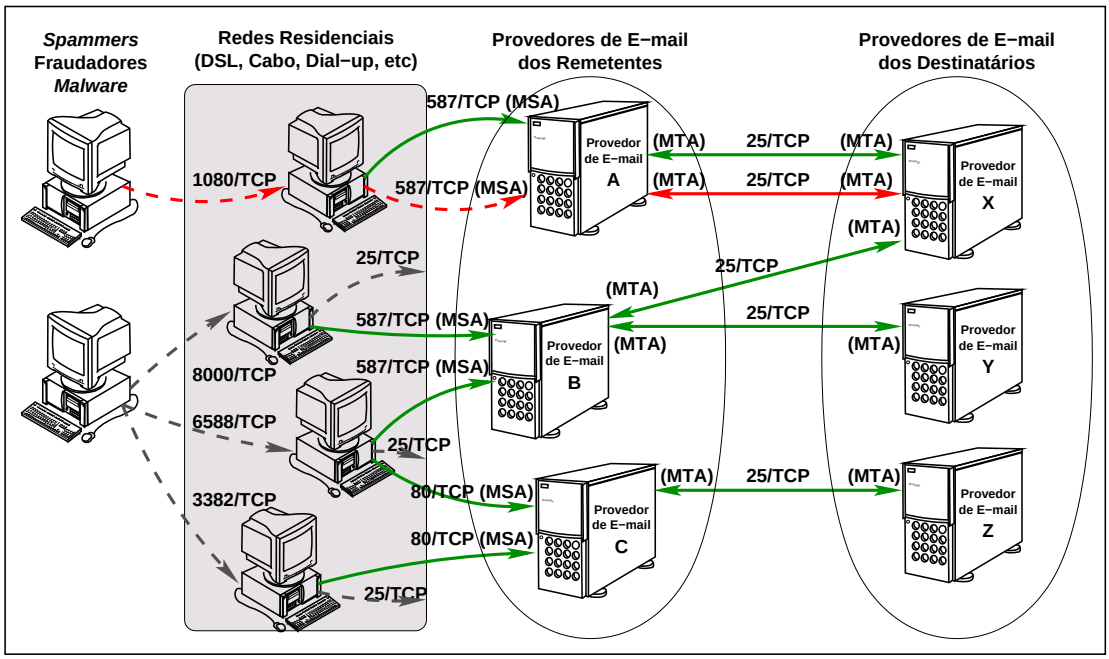


Figura 4.18 - Possível abuso do novo cenário.

trole de vazão de *emails*, ele poderá identificar mais facilmente máquinas infectadas e usuários abusivos.

Apesar dos claros benefícios da adoção de *Message Submission* e Gerência de Porta 25 em redes de perfil residencial, as grandes redes de banda larga do Brasil não adotam, até o presente momento, estas medidas.

Considerando-se os números obtidos no módulo *Relay/Proxy* aberto, que mostram mais de meio bilhão de mensagens injetadas e que a maioria absoluta das conexões visava a entrega direta do *spam*, acredita-se que a adoção de Gerência de Porta 25 dificultaria o abuso por parte dos *spammers*.

4.5.2 Uso de Técnicas de Ofuscação de Endereços de *Email*

Os resultados do módulo *harvesting* mostraram que algumas técnicas simples de ofuscação de endereços demonstraram eficácia para evitar que sejam capturados. As técnicas usadas foram:

- troca do caracter, "`@`" por "`at`" e "`em`";
- uso de JavaScript;

- uso de imagens para apresentar o endereço de *email*.

Embora muitos *sites* recomendem algumas destas técnicas, a ênfase é sempre dada na ofuscação de endereços presentes em páginas HTML. Porém, os resultados mostraram que os *spammers* possuem ferramentas que permitem encontrar endereços de *email* em arquivos de diversos formatos, incluindo os formatos do Microsoft Office (PowerPoint e Word) e PostScript.

Desse modo, fica claro que apenas ofuscar os endereços de *email* nas páginas não os protege de *harvesting*. Também é interessante notar que, por enquanto, mesmo as técnicas mais simples foram efetivas.

5 CONCLUSÕES

Esta tese descreveu o projeto e implementação de uma infra-estrutura, com base em *honeypots*, para o estudo do problema do *spam* e do *phishing*. Em particular, o trabalho teve enfoque em obter dados sobre: o abuso de *relays* e *proxies* abertos; a obtenção de endereços de *email* em documentos disponíveis em *sites* Internet; coleta de URLs enviadas através de mensagens de *pop-up*; e a correlação de todos estes dados com atividades relacionadas com *spam*, vistas no Consórcio Brasileiro de *Honeypots*.

A análise dos *spams* recebidos via o módulo de *Relay/Proxy* Aberto mostrou que a tecnologia implantada foi extremamente eficiente para capturar *spams*, coletando mais de 500 milhões de mensagens. Também ficou evidente a importância de ter sido desenvolvido neste trabalho o suporte para emulação de *proxy* SOCKS, pois a porta associada a este protocolo foi a mais abusada durante todo o período. Outra contribuição importante foi na identificação de mensagens de teste enviadas por *spammers*. Através da identificação de padrões nestas mensagens, foi possível responder somente às mensagens de teste, e não a todas as primeiras mensagens enviadas, como era a abordagem popular nos outros trabalhos na área. Outra vantagem da abordagem definida neste trabalho foi a possibilidade de responder a mensagens de teste enviadas ao longo de uma campanha de *spam*, e não apenas as mensagens enviadas no seu início.

Também foi possível observar que os endereços IP que exploraram *proxies* e *relays* abertos para o envio de *spam* estavam extremamente concentrados. Aproximadamente 97% de todos os *emails* vieram de apenas 5 países, e apenas 4 ASs foram responsáveis por quase 80% de todos os *emails* observados. Os resultados também mostram que a maioria das requisições recebidas foi de tentativas de conexão destinadas à porta SMTP (25/TCP) de máquinas externas, o que evidencia que o objetivo da maioria destas requisições era de fato a entrega de *spam*. Para mitigar o envio destes *spams*, a partir de redes de caráter residencial e de endereçamento IP dinâmico, um método que poderia ter grande eficácia seria a implementação de medidas de gerenciamento de porta 25/TCP, que também tornariam o problema mais rastreável para os provedores de acesso.

Devido à grande quantidade de mensagens capturadas pelo módulo de *Relay/Proxy* Aberto, tornou-se inviável a análise individual de todas as mensagens capturadas.

Este desafio levou a uma cooperação, fomentada pelo NIC.br, com o grupo de *data mining* do DCC/UFMG.

Uma contribuição do módulo de *harvesting*, que concentrou-se na obtenção de dados sobre a captura de endereços de email em sites na Internet, foi expandir a coleta de dados para incluir endereços de *email* não apenas em páginas *Web*, mas também em arquivos de diversos formatos. Durante a análise ficou claro que os *spammers* procuram por endereços de *email* presentes em arquivos, fato que não era estudado por outros trabalhos relacionados. Outra contribuição foi a realização de testes com relação à efetividade de alguns mecanismos de ofuscação de endereços, que confirmaram que simples mudanças na forma de representação podem impedir que o endereço seja coletado por *spammers*, com as técnicas atuais.

Quando cruzam-se os resultados do módulo de *Relay/Proxy* com os resultados obtidos no módulo de *Harvesting* pode-se ver que há uma grande diferença entre quais países abusaram de *proxies* e quais fizeram a entrega direta de *spams*. Também, no módulo de *Harvesting*, foi observado que, na grande maioria dos casos, cada IP entregava apenas uma mensagem.

O módulo de *pop-up spam* mostrou-se efetivo para a coleta de mensagens de *pop-up* enviadas para sensores do Consórcio, que por cobrirem uma grande área de endereçamento, coletaram uma grande quantidade de pacotes. A maior parte deste tráfego possuía mensagens com URLs que apontavam para *sites* onde poderiam ser obtidos programas, supostamente, para evitar mensagens de *pop-up* ou para remover códigos maliciosos dos computadores. Observar e entender este tipo de abuso foi importante, pois este pode ser um potencial vetor de disseminação de *phishing*.

O Consórcio Brasileiro de *Honeypots*, por ter uma grande área monitorada por um longo período de tempo, foi muito importante para a obtenção de dados representativos sobre varreduras por portas associadas a *proxies* e *relays* abertos. A análise destas varreduras reforçou a importância da implementação do suporte para SOCKS, uma vez que esta foi a porta mais varrida, dentre aquelas relacionadas com *spam*. Também foi interessante notar que não há uma relação direta entre os países que mais realizaram varreduras e aqueles que efetivamente abusavam *proxies* e *relays* abertos.

Conforme comentado anteriormente, devido ao grande volume de *spams* capturados

foi inviável a análise individual de cada mensagem. Deste modo, mesmo a infra-estrutura tendo o potencial de capturar mensagens de *phishing*, nenhuma mensagem desse tipo foi identificada nas amostras de *emails* analisadas. Porém, devido à eficiência da arquitetura para a captura de mensagens, tanto no abuso de *proxies* e *relays* quanto no envio direto ou por *pop-up spam*, acredita-se que ela seja apropriada para a captura de mensagens e URLs relacionadas com ataques de *phishing*.

De modo geral, ao serem agrupadas as análises do tráfego capturado por todos os módulos, pode-se ter uma visão mais global do problema do *spam*, incluindo o abuso de serviços e a coleta de endereços de *email*. Como a infra-estrutura ficou diversos meses em operação em redes brasileiras, foi possível obter uma grande quantidade de dados sobre o abuso dessas redes para o envio de *spam*. Deste modo, pôde-se propor estratégias de mitigação para redes aqui instaladas.

5.1 Perspectivas Futuras

Este trabalho gerou diversas métricas sobre o problema do *spam*, e dele podem se originar outros trabalhos, como nas áreas de geração de filtros anti-*spam* e na redução do abuso das redes. Alguns dos possíveis trabalhos futuros incluem:

- escrita dos módulos que não foram implementados no protótipo da infra-estrutura proposta neste trabalho: o Módulo *Web* e *Phishing/Malware*.
- uso da infra-estrutura proposta especificamente para o combate ao *spam*, através da doação das mensagens capturadas para grupos responsáveis pelo desenvolvimento de filtros anti-*spam*;
- modificação do módulo de *proxy* aberto para gerar erros fatais, em intervalos periódicos, para alguns *emails* de destino dos *spams*. Em decorrência deste erro, é possível que alguns programas de envio de *spam* retirem o *email* da vítima das suas bases de dados. Deste modo, poderia ser estudado o impacto desta medida na redução de *spams* recebidos pelos endereços para os quais seriam gerados erros;
- análise mais detalhada do conteúdo dos *emails* coletados, observando-se o corpo das mensagens, endereços de destinatários, entre outras informações;
- estudar a possibilidade de usar a infra-estrutura atual como um método de localização de *bots* que enviem *spam*. Isto poderia ser possível através da

identificação de características de tráfego de rede originado por estes *bots*;

- estudar melhorias e otimizações na implementação da infra-estrutura, de modo a melhorar o seu desempenho e aumentar o número de mensagens capturadas por sensor. Como por exemplo utilizar a linguagem C no lugar de Perl e otimizar a estrutura do sistema de arquivos utilizado pelos sensores;

- um protótipo de um módulo de captura de *spam* enviado através de *Instant Messaging* (*spim*) foi iniciado, mas não concluído. Uma possibilidade de trabalho futuro seria o término desse módulo com o objetivo de estudar a natureza do *spam* enviado através de *Instant Messaging*;

- colocação de sensores para a coleta de *spam* em outros países, permitindo uma visão mais geral do problema e fomentando a cooperação para reduzir o abuso das redes do ponto de vista global.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLMAN, E.; CALLAS, J.; DELANY, M.; LIBBEY, M.; FENTON, J.; THOMAS, M. **RFC 4871**: domainkeys identified mail (DKIM) signatures. May 2007. Disponível em: <http://www.ietf.org/rfc/rfc4871.txt>. Acesso em: 08 maio 2008. 32

ANDREOLINI, M.; BULGARELLI, A.; COLAJANNI, M.; MAZZONI, F. Honeyspam: honeypots fighting spam at the source. In: USENIX WORKSHOP ON STEPS TO REDUCING UNWANTED TRAFFIC ON THE INTERNET (SRUTI '05), 1., 2005, Cambridge, MA, USA. **Proceedings...** Cambridge: USENIX Association, 2005. p. 77–83. 35

BRADNER, S. **RFC 2026**: the internet standards process – revision 3. October 1996. Disponível em: <http://www.ietf.org/rfc/rfc2026.txt>. Acesso em: 30 abr. 2008. 109

BUETI, M. C. **ITU Survey on anti-spam legislation worldwide**. June 2005. Disponível em: <http://www.itu.int/osg/csd/spam/legislation/Background_Paper_ITU_Bueti_Survey.pdf>. Acesso em: 15 maio 2008. 33

CERF, V. G. Spam, spim, and spit. **Commun. ACM**, ACM Press, New York, NY, USA, v. 48, n. 4, p. 39–43, 2005. ISSN 0001-0782. 23

CHUVAKIN, A. Honeynets: high value security data – analysis of real attacks launched at a honeypot. **Elsevier Network Security**, v. 2003, n. 8, p. 11–15, August 2003. ISSN 1353-4858. 29

CRANOR, L. F.; LAMACCHIA, B. A. Spam! **Commun. ACM**, ACM Press, New York, NY, USA, v. 41, n. 8, p. 74–83, 1998. ISSN 0001-0782. 23

FIELDING, R.; GETTYS, J.; MOGUL, J.; FRYSTYK, H.; MASINTER, L.; LEACH, P.; BERNERS-LEE, T. **RFC 2616**: hypertext transfer protocol – HTTP/1.1. June 1999. Disponível em: <http://www.ietf.org/rfc/rfc2616.txt>. Acesso em: 09 mar. 2008. 29, 34, 36, 45, 109

GELLENS, R.; KLENSIN, J. **RFC 4409**: message submission for mail. April 2006. Disponível em: <http://www.ietf.org/rfc/rfc4409.txt>. Acesso em: 09 mar. 2008. 33, 90

HAMBRIDGE, S.; LUNDE, A. **RFC 2635**: don't spew a set of guidelines for mass unsolicited mailings and postings (spam*). June 1999. Disponível em: <http://www.ietf.org/rfc/rfc2635.txt>. Acesso em: 09 mar. 2008. 23

HARTMEIER, D. Design and performance of the OpenBSD stateful packet filter (pf). In: FREENIX TRACK: 2002 USENIX ANNUAL TECHNICAL CONFERENCE, 2002., 2002, Monterey, CA, USA. **Proceedings...** Monterey, CA, USA: USENIX Association, 2002. p. 171–180. ISBN 1-880446-01-4. 50, 54

HAWKINSON, J.; BATES, T. **RFC 1930**: guidelines for creation, selection, and registration of an autonomous system (AS). March 1996. Disponível em: <http://www.ietf.org/rfc/rfc1930.txt>. Acesso em: 09 mar. 2008. 66

HAYES, B. Spam, spam, spam, lovely spam. **American Scientist**, v. 91, n. 3, p. 200–204, May–June 2003. 23, 24

HOEPERS, C.; STEDING-JESSEN, K.; CHAVES, M. H. P. C. Projeto e desenvolvimento de um sistema de controle e acompanhamento de notificações de spam. In: SIMPÓSIO SOBRE SEGURANÇA EM INFORMÁTICA, 5., 2003, São José dos Campos, SP. **Anais...** São José dos Campos, SP: Instituto Tecnológico de Aeronáutica (ITA), 2003. 24

HOEPERS, C.; STEDING-JESSEN, K.; CORDEIRO, L. E. R.; CHAVES, M. H. P. C. A national early warning capability based on a network of distributed honeypots. In: ANNUAL FIRST CONFERENCE ON COMPUTER SECURITY INCIDENT HANDLING, 17., 2005, Singapore. **Proceedings...** Singapore: [s.n.], 2005. 51, 52

HOLZ, T. A short visit to the bot zoo. **IEEE Security & Privacy**, v. 3, n. 3, p. 76–79, May-June 2005. Disponível em: <http://pi1.informatik.uni-mannheim.de/filepool/publications/a-short-visit-to-the-bot-zoo.pdf>. Acesso em: 09 mar. 2008. 24

HUTZLER, C.; CROCKER, D.; RESNICK, P.; ALLMAN, E.; FINCH, T. **RFC 5068**: email submission operations: access and accountability requirements. November 2007. Disponível em: <http://www.ietf.org/rfc/rfc5068.txt>. Acesso em: 09 mar. 2008. 91

IANELLI, N.; HACKWORTH, A. **Botnets as a vehicle for online crime**. December 2005. CERT Coordination Center, Carnegie Mellon University.

Disponível em: <http://www.cert.org/archive/pdf/Botnets.pdf>. Acesso em: 09 mar. 2008. 24

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 3166**: codes for the representation of names of countries and their subdivisions – part 1: country codes. November 2006. Disponível em: <http://www.iso.org/iso/country_codes.htm>. Acesso em: 09 mar. 2008. 63

KLENSIN, J. **RFC 2821**: simple mail transfer protocol. April 2001. Disponível em: <http://www.ietf.org/rfc/rfc2821.txt>. Acesso em: 09 mar. 2008. 29, 44, 90

KOLSTAD, R. The only good spam comes from Hormel. **;login:**, v. 30, n. 1, p. 2–3, February 2005. Disponível em: <http://www.usenix.org/publications/login/2005-02/openpdfs/motd.pdf>. Acesso em: 09 mar. 2008. 23, 31, 33, 91

KRAWETZ, N. Anti-honeypot technology. **IEEE Security & Privacy**, v. 2, n. 1, p. 76–79, January–February 2004. 29, 57

KREIBICH, C.; KANICH, C.; LEVCHENKO, K.; ENRIGHT, B.; VOELKER, G. M.; PAXSON, V.; SAVAGE, S. On the spam campaign trail. In: USENIX WORKSHOP ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS (LEET '08), 1., 2008, San Francisco, CA. **Proceedings...** San Francisco, CA: USENIX Association, 2008. 107

LANGHEINRICH, E. http:bl: taking DNSBL beyond SMTP. **;login:**, v. 33, n. 1, p. 19–24, February 2008. 37

LEECH, M.; GANIS, M.; LEE, Y.; KURIS, R.; KOBLAS, D.; JONES, L. **RFC 1928**: SOCKS protocol version 5. March 1996. Disponível em: <http://www.ietf.org/rfc/rfc1928.txt>. Acesso em: 09 mar. 2008. 29, 35, 45, 55

LINDBERG, G. **RFC 2505**: anti-spam recommendations for SMTP MTAs. February 1999. Disponível em: <http://www.ietf.org/rfc/rfc2505.txt>. Acesso em: 09 mar. 2008. 29

LYON, J.; WONG, M. **RFC 4406**: sender id: authenticating e-mail. April 2006. Acesso em: 08 maio 2008. 32

MAZIÈRES, D. **The Mail Avenger home page**. May 2008. Disponível em: <http://www.mailavenger.org/>. Acesso em: 18 maio 2008. 61

MESSAGING ANTI-ABUSE WORKING GROUP (MAAWG). **Managing port 25 for residential or dynamic ip space**. Dec 2005. Disponível em: <http://www.maawg.org/port25/MAAWG_Port25rec0511.pdf>. Acesso em: 09 mar. 2008. 30, 91, 92

_____. **Email metrics reports: the network operators perspective**. April 2008. Disponível em: <http://www.maawg.org/about/EMR/>. Acesso em: 21 abr. 2008. 23

MILLETARY, J. **Technical trends in phishing attacks**. October 2005. CERT Coordination Center, Carnegie Mellon University. Disponível em: <http://www.cert.org/archive/pdf/Phishing_trends.pdf>. Acesso em: 09 mar. 2008. 23, 24

MILLS, D. **RFC 4330**: simple network time protocol (SNTP) version 4 for ipv4, ipv6 and OSI. January 2006. Disponível em: <http://www.ietf.org/rfc/rfc4330.txt>. Acesso em: 09 mar. 2008. 54

OUDOT, L. **Fighting spammers with honeypots**. November 2003. Disponível em: <http://www.securityfocus.com/infocus/1747>. Acesso em: 09 mar. 2008. 40

PANG, R.; YEGNESWARAN, V.; BARFORD, P.; PAXSON, V.; PETERSON, L. Characteristics of internet background radiation. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT, 4., Taormina, Sicily, Italy. **Proceedings...** New York, NY, USA: ACM, 2004. p. 27–40. ISBN 1-58113-821-0. 51

PATHAK, A.; HU, Y. C.; MAO, Z. M. Peeking into spammer behavior from a unique vantage point. In: USENIX WORKSHOP ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS (LEET '08), 1., 2008, San Francisco, CA. **Proceedings...** San Francisco, CA: USENIX Association, 2008. 24, 29, 38

POSTEL, J. **RFC 706**: on the junk mail problem. November 1975. Disponível em: <http://www.ietf.org/rfc/rfc0706.txt>. Acesso em: 09 mar. 2008. 23

PRINCE, M.; DAHL, B.; HOLLOWAY, L.; KELLER, A.; LANGHEINRICH, E. Understanding how spammers steal your e-mail address: an analysis of the first six months of data from project honey pot. In: CONFERENCE ON EMAIL AND ANTI-SPAM, 2., 2005, Palo Alto, CA, USA. **Proceedings...** Palo Alto: [s.n.], 2005. 36

PROJECT HONEY POT. **Project honey pot website**. May 2008. Disponível em: <http://www.projecthoneypot.org/>. Acesso em: 16 maio 2008. 36

PROVOS, N. A virtual honeypot framework. In: USENIX SECURITY SYMPOSIUM, 13., 2004, San Diego, CA, USA. **Proceedings...** San Diego: USENIX Association, 2004. p. 1–14. Disponível em: <http://www.usenix.org/publications/library/proceedings/sec04/tech/provos.html>. Acesso em: 09 mar. 2008. 34, 35

PROVOS, N.; HOLZ, T. **Virtual honeypots**: from botnet tracking to intrusion detection. 1st. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2007. 480 p. ISBN 978-0-321-33632-3. 27, 28

RAMACHANDRAN, A.; FEAMSTER, N. Understanding the network-level behavior of spammers. In: ACM SIGCOMM, 21., 2006, Pisa, Italy. **Proceedings...** New York, NY: ACM, 2006. 37, 61

SAUVER, J. S. Spam zombies and inbound flows to compromised customer systems. In: MAAWG GENERAL MEETING, 5., 2005, Montreal, Quebec, CA. **Procedings...** MAWWG, 2005. Disponível em: <http://darkwing.uoregon.edu/~joe/zombies.pdf>. Acesso em: 09 mar. 2008. 24

SHIREY, R. **RFC 4949**: internet security glossary, version 2. August 2007. Disponível em: <http://www.ietf.org/rfc/rfc4949.txt>. Acesso em: 30 abr. 2008. 23

SHOSTACK, A.; STEWART, A. **The new school of information security**. 1st. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2008. 288 p. ISBN 978-0-321-50278-0. 23

SIEMBORSKI, R.; ED.; MELNIKOV, E. A. **RFC 4954**: SMTP service extension for authentication. July 2007. Disponível em: <http://www.ietf.org/rfc/rfc4954.txt>. Acesso em: 09 mar. 2008. 91

SPITZNER, L. **Honeypots**: tracking hackers. 1st. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2002. 480 p. ISBN 978-0-321-10895-1. 27

STEDING-JESSEN, K.; HOEPERS, C.; MONTES, A. Mecanismos para contenção de tráfego malicioso de saída em honeynets. In: SIMPÓSIO SOBRE SEGURANÇA EM INFORMÁTICA, 5., 2003, São José dos Campos, SP. **Anais...** São José dos Campos, SP: Instituto Tecnológico de Aeronáutica (ITA), 2003. ISBN 85-87978-05-5. 28

_____. **Consórcio brasileiro de honeypots**: projeto honeypots distribuídos. 2008. Disponível em: <http://www.honeypots-alliance.org.br/index-po.html>. Acesso em: 08 mar. 2008. 47, 48, 49

STEWART, J. **Windows Messenger Popup Spam on UDP Port 1026**. June 2003. Disponível em: <http://www.secureworks.com/research/threats/popup-spam/>. Acesso em: 13 maio 2008. 51

THE HONEYNET PROJECT. **Know your enemy**: learning about security threats. 2nd. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2004. 800 p. ISBN 978-0-321-16646-3. 27, 28

_____. **Know your enemy**: honeynets. May 2006. Disponível em: <http://www.honeynet.org/papers/honeynet/>. Acesso em: 04 abr. 2008. 27, 28

THE HONEYNET PROJECT & RESEARCH ALLIANCE. **Know your enemy**: phishing – behind the scenes of phishing attacks. 2005. Disponível em: <http://www.honeynet.org/papers/phishing/>. Acesso em: 09 mar. 2008. 39

WANG, K. **Honeyclient development project**. 2006. Disponível em: <http://www.honeyclient.org/>. Acesso em: 09 mar. 2008. 46

WONG, M.; SCHLITT, W. **RFC 4408**: sender policy framework (SPF) for authorizing use of domains in e-mail, version 1. April 2006. Disponível em: <http://www.ietf.org/rfc/rfc4408.txt>. Acesso em: 08 maio 2008. 32

XIE, M.; YIN, H.; WANG, H. An effective defense against email spam laundering. In: ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS

SECURITY, 13., 2006, Alexandria, Virginia, USA. **Proceedings...** New York, NY, USA: ACM, 2006. p. 179–190. ISBN 1-59593-518-5. 23, 31

ZAKON, R. **RFC 2235**: Hobbes' internet timeline. November 1997. Disponível em: <http://www.ietf.org/rfc/rfc2235.txt>. Acesso em: 09 mar. 2008. 23

ZHUANG, L.; DUNAGAN, J.; SIMON, D. R.; WANG, H. J.; OSIPKOV, I.; HULTEN, G.; TYGAR, J. D. Characterizing botnets from email spam records. In: USENIX WORKSHOP ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS (LEET '08), 1., 2008, San Francisco, CA. **Proceedings...** San Francisco, CA: USENIX Association, 2008. 107

GLOSSÁRIO

Autonomous System Grupo interconectado de uma ou mais redes com uma única e bem definida política de roteamento.

BCP *Best Current Practice* – série de documentos do IETF em que são publicadas as boas práticas em uso na Internet.

Bot Programa que, além de incluir funcionalidades de *worms*, sendo capaz de se propagar automaticamente através da exploração de vulnerabilidades existentes ou falhas na configuração de *softwares* instalados em um computador, dispõe de mecanismos de comunicação com o invasor, permitindo que o programa seja controlado remotamente. O invasor, ao se comunicar com o *bot*, pode orientá-lo a desferir ataques contra outros computadores, furtar dados, enviar *spam*, etc.

Botnets Redes formadas por diversos computadores infectados com *bots*. Podem ser usadas em atividades de negação de serviço, esquemas de fraude, envio de *spam*, etc.

Campanha de Spam conjunto de *emails* com o mesmo, ou aproximadamente o mesmo conteúdo, ou com conteúdo similar, por exemplo contendo uma mesma URL (ZHUANG et al., 2008). Cada campanha tem um objetivo em particular, quer seja a venda de um produto, fraude financeira ou distribuição de *malware* (KREIBICH et al., 2008).

Cavalo de Tróia Programa que além de executar funções para as quais foi aparentemente projetado, também executa outras funções normalmente maliciosas e sem o conhecimento do usuário.

CERT.br O CERT.br é o Grupo de Resposta a Incidentes de Segurança para a Internet brasileira, mantido pelo Comitê Gestor da Internet no Brasil (CGI.br). É o grupo responsável por receber, analisar e responder a incidentes de segurança em computadores, envolvendo redes conectadas à Internet brasileira.

CIDR (*Classless Inter-Domain Routing*) É um padrão, baseado em prefixos, para a interpretação de endereços IP. Este padrão facilita o roteamento por permitir que grupos de endereços sejam agrupados em uma única entrada, em uma tabela de roteamento.

Código Malicioso Termo genérico que se refere a todos os tipos de programa que executam ações maliciosas em um computador. Exemplos de códigos maliciosos são os vírus, *worms*, *bots*, cavalos de tróia, etc.

Crawler Programa que automaticamente percorre páginas *Web*, acessando o seu conteúdo e recursivamente seguindo todos os seus *links*.

Daemon Programa que executa sem intervenção e de modo contínuo e que provê serviços para um sistema ou rede.

***Fingerprint* Passivo** Técnica que tenta estimar o sistema operacional que está originando uma conexão TCP em função de características da pilha TCP/IP de cada sistema operacional, tais como TCP *window size*, *Time to live* (TTL), presença do bit *Don't Fragment* (DF) e tamanho do pacote.

Harvesting Técnica utilizada por *spammers*, que consiste em varrer páginas *Web*, arquivos de listas de discussão, entre outros, em busca de endereços de *email*.

Honeyclient *Honeypot* cujo objetivo é detectar vulnerabilidades ou códigos maliciosos que afetem uma máquina ou aplicativo cliente, por exemplo um navegador. Esse tipo de *honeypot* tipicamente direciona um cliente a realizar um acesso a um servidor e monitora os resultados: modificações que ocorram no lado do cliente, em função desse acesso, são maliciosas.

Honeypot Recurso computacional de segurança dedicado a ser sondado, atacado ou comprometido.

Libpcap (*Packet Capture library*) é uma interface de captura de pacotes de rede. Os arquivos gerados por esta interface são chamados de arquivos `pcap`, ou ainda, arquivos em formato `libpcap`.

Lista de Bloqueio No que diz respeito a *spam*, listas de bloqueio podem ser listas de *spammers* conhecidos, os endereços IPs usados por eles ou ainda seu provedores ou redes de origem. Usando informação de uma lista de bloqueio, filtros de *spam* podem barrar mensagens vindo dessas origens.

Malware Do inglês *malicious software*. Veja código malicioso.

Phishing Também conhecido como *phishing scam* ou *phishing/scam*. Mensagem não solicitada que se passa por comunicação de uma instituição conhecida, como um banco, empresa ou site popular, e que procura induzir usuários ao fornecimento de dados pessoais e financeiros. Inicialmente, este tipo de mensagem induzia o usuário ao acesso a páginas fraudulentas na Internet. Atualmente, o termo também se refere à mensagem que induz o usuário à instalação de códigos maliciosos, além da mensagem que, no próprio conteúdo, apresenta formulários para o preenchimento e envio de dados pessoais e financeiros.

Proxy Servidor que atua como intermediário entre um cliente e outro servidor. Normalmente é utilizado para aumentar a performance de acesso a determinados serviços ou permitir que mais de uma máquina se conecte à Internet.

Proxy Aberto *Proxy* mal configurado que pode ser abusado por atacantes e utilizado como uma forma de tornar anônimas algumas ações na Internet, como atacar outras redes ou enviar *spam*.

Referer A URL que um navegador estava visitando antes de efetuar uma requisição. A palavra "*referer*" é o termo utilizado no padrão HTTP (FIELDING et al., 1999), embora seja um erro de ortografia da palavra em Inglês "*referrer*".

Relay Aberto Servidores SMTP mal configurados que possibilitam a entrega de mensagens a partir de qualquer origem para quaisquer destinatários.

RFC *Request for Comments* – série de documentos que constituem o canal oficial de publicação de padrões para a Internet. As RFCs são publicadas pelo *Internet Engineering Task Force* (IETF), em particular pelo *Internet Engineering Steering Group* (IESG) e *Internet Architecture Board* (IAB).

A série de RFCs começou a ser publicada em 1969 e esses documentos cobrem diversos tópicos, além de padrões Internet, como discussões sobre novos tópicos de pesquisa e memorandos de *status* sobre a Internet. A publicação de RFCs é responsabilidade do *RFC Editor*, sob a direção geral do IAB (BRADNER, 1996).

Sanitização de *Logs* processo em que uma porção dos *logs* é alterada com o objetivo de omitir certas informações, em geral endereços IP ou nome de máquinas de uma determinada rede. Essa técnica é muito utilizada na notificação de incidentes e no uso de *logs* na literatura.

SOCKS Protocolo de rede que implementa um tipo específico de *proxy*.

Spam Termo usado para se referir aos *emails* não solicitados, que geralmente são enviados para um grande número de pessoas.

Spambot Outro termo para *Spam Zombie*, isto é, computador infectado por código malicioso e usado para o envio de *spam* sem o conhecimento do usuário.

Spam Trap Endereço de *email* criado intencionalmente com o objetivo de capturar mensagens não solicitadas.

Spam Zombie Computador infectado por código malicioso, capaz de transformar o sistema do usuário em um servidor de email para envio de *spam*. Em muitos casos, o usuário do computador infectado demora a perceber que seu computador está sendo usado por um invasor para este fim.

Spammer Pessoa que envia *spam*.

Spim Modalidade de *spam*, enviada através de *Instant Messaging* (IM).

Spit Modalidade de *spam*, enviada através de telefonia IP.

User Agent O identificador que é associado a um navegador ou agente acessando uma página *Web*.

Vírus Programa ou parte de um programa de computador, normalmente malicioso, que se propaga infectando, isto é, inserindo cópias de si mesmo e se tornando parte de outros programas e arquivos de um computador. O vírus depende da execução do programa ou arquivo hospedeiro para que possa se tornar ativo e dar continuidade ao processo de infecção.

Worm Programa capaz de se propagar automaticamente através de redes, enviando cópias de si mesmo de computador para computador. Diferente do vírus, o worm não embute cópias de si mesmo em outros programas ou arquivos e não necessita ser explicitamente executado para se propagar. Sua propagação se dá através da exploração de vulnerabilidades existentes ou falhas na configuração de softwares instalados em computadores.

A APÊNDICE A - ARTIGOS EM REVISTAS INDEXADAS

A.1 INFOCOMP

Em cumprimento a requisito exigido para a defesa da Tese, foi escrito e submetido um artigo para a revista indexada INFOCOMP *Journal of Computer Science*[1] (Qualis B nacional, área multidisciplinar), submetido em 01/11/2007 e o aceite recebido em 04/12/2007. O artigo foi publicado na edição de março de 2008 da revista, volume 7, número 1, páginas 45–53.

Este artigo, intitulado "*Using Low-Interaction Honeypots to Study the Abuse of Open Proxies to Send Spam*" segue em anexo neste apêndice.

[1] http://www.dcc.ufla.br/infocomp/

Using Low-Interaction Honeypots to Study the Abuse of Open Proxies to Send Spam

KLAUS STEDING-JESSEN[1]
NANDAMUDI L. VIJAYKUMAR[2]
ANTONIO MONTES[3]

[1]Brazilian Network Information Center - NIC.br
Computer Emergency Response Team Brazil - CERT.br, São Paulo (SP)
jessen@cert.br
[2]National Institute for Space Research - INPE
Computing and Applied Mathematics Associated Laboratory - LAC, São José dos Campos (SP)
vijay@lac.inpe.br
[3]Ministry of Science and Technology - MCT
Renato Archer Research Center - CenPRA, Campinas (SP)
antonio.montes@cenpra.gov.br

Abstract. One of the main problems in creating effective mechanisms to mitigate the spam problem is the lack of more precise data on it. This paper describes the design and implementation of a honeypot-based architecture to study the abuse of open proxies to send spam. The sensors were installed in Brazilian broadband networks and they captured more than 500 million emails over a period of 15 months. In this paper we present the analysis of these data and describe some contributions to the spam capture field that were implemented.

Keywords: network security, honeypot, spam, open proxy.

(Received November 01, 2007 / Accepted December 04, 2007)

1 Introduction

Spam is one of the Internet abuses that has increased the most, being responsible for a significant amount of the total number of emails in circulation today [10, 21]. Spam has also been used for sending phishing-related messages (an email trying to induce an user to give personal of financial data) and for disseminating malicious code [22].

The word spam is used to describe the act of sending unsolicited or inappropriate messages, specially in bulk and with commercial content [10, 28, 7]. The term became popular to describe sending email in bulk quantities after an incident in April 1994, when an unsolicited message was sent to six thousand Usenet Newsgroups [31, 10]. It is important to note, however, that the concern about unsolicited emails is not new and dates back to 1975 [24].

With the growth of the Internet popularity, the access to email, initially restricted to smaller groups like the academic community, grew considerably among the general public. And, because the cost of sending emails is very low, compared to the cost of regular mail, this has been a major incentive for sending commercial emails in huge quantities [2].

The spam software efficiency has also increased, making existing spam blocking techniques even less effective and making the spammer (person who sends spam) tracking more difficult [4, 22]. As an example, the usage of machines infected by malicious code like bots (program that is able to propagate itself by infecting vulnerable computers and can be controlled remotely by an attacker) for sending spam and phishing [22] is increasing, allowing spammers to remain anonymous.

Currently, one of the main problems in developing ef-

fective mechanisms to mitigate spam is the lack of more precise data about the dimension of the problem and about the methods in use for the dissemination of spam and malicious code.

Although there is a notion that spam corresponds to a significant amount of the total number of emails in circulation and plays an important role in the dissemination of malicious code and phishing, most public numbers come from informal research or from companies selling anti-spam products or services [10].

This paper describes the design and implementation of an architecture to study the spam problem, in particular the abuse of open proxies. Open proxies have traditionally been used for sending spam and perpetrating other malicious activities [11, 17]. This kind of abuse happens mainly in end-user machines, connected via broadband, and with the potential of being infected by malicious code and controlled remotely to send spam [14, 27, 12].

The architecture implemented is comprised of a set of sensors based on honeypots. The data captured was then collected by a central server. This paper also describes the extensions implemented in Honeyd [25] in order to expand the protocols supported and the recorded information.

The sensors were installed in Brazilian broadband networks and captured more than 500 million spams over 15 months. The data captured allowed a better analysis of the spam problem and also a better understanding of how spam is being sent. This analysis also helps the creation of better mechanisms to fight spam.

This paper is organized as follows. In section 2 some concepts about honeypots and abuse of services for sending spam are explained. In section 3 we discuss some related work using honeypots to capture and study spam. In section 4 we describe the architecture implemented for spam capture and the services being emulated on the honeypots. We also present in this section the results obtained. Conclusions and future work are discussed in section 5.

2 Definitions

In this section we review some concepts in the honeypot area, as well as abuse of services for sending spam.

2.1 Honeypots

A honeypot is a security device designed to be probed, attacked or exploited [29, 26].

The value of honeypots relies on the fact that everything observed is suspicious and potentially malicious. In this way, they are a security tool with a low number of false-positives and that can provide high value information. They also generate a smaller amount of data to be analyzed, specially if compared to other more traditional security tools like Intrusion Detection Systems (IDS).

Based on the access level to the system and the resources an attacker is given, honeypots can be classified as low-interaction and high-interaction [26, 13].

2.1.1 High-Interaction Honeypots

In high-interaction honeypots attackers interact with real operating systems, services and programs. Once compromised, these honeypots are used to observe the attackers behaviour, their tools, motivations and explored vulnerabilities. This kind of honeypot must have a robust containment mechanism in order to prevent, once compromised, its use to attack other networks [30, 26].

This type of honeypot is justifiable when the main objective is to study the attackers behaviour, their motives and also to study, in detail, the tools being used and the vulnerabilities being exploited. Their use, however, is time consuming and requires good containment techniques, as well as specialized personnel to operate them.

2.1.2 Low-Interaction Honeypots

On low-interaction honeypots, tools are installed in order to emulate operating systems and services and then interact with the attackers and malicious code. This kind of honeypot has a small chance of being compromised and is ideal for production networks, when there are no available personnel to administer a honeynet or when the risk of a high-interaction honeypot is not acceptable.

Typical use of low-interaction honeypots include: port scan identification, generation of attack signatures, trend analysis and malware collection [26].

In this paper we will focus on using low-interaction honeypots emulating open proxy and open relay machines to capture spam.

2.2 Abuse of Services for Sending Spam

The SMTP protocol (Simple Mail Transfer Protocol) is usually associated with port 25/TCP (Transmission Control Protocol), and its main goal is the email transport in a reliable and efficient way [16]. An SMTP relay is a system that receives an email from an SMTP client and retransmits it to another SMTP server for its final delivery or for additional transmission. Normally, this relay service is selective and only retransmits emails for authorized clients. Misconfigured SMTP servers, usually called open relays, allow the delivery of messages from any source to any recipient [19]. This type of server is abused to send spam because it makes the detection of the real origin more difficult. It can also be used to bypass mail blocking lists.

A proxy is a server that works as an intermediary between a client and another server. Normally it is used to access certain services with increased performance or to allow more than one machine to connect to the Internet sharing the same IP (Internet Protocol) address. A proxy acts as a intermediary, making connections on behalf of other clients [5].

A misconfigured proxy allows connections to be made from any origin to any destination IP address or port. This type of proxy is called an open proxy. Open proxies are also intentionally installed by malicious code, bots and trojan horses.

Spammers continuously scan the Internet searching for open proxies [17]. Once located, these computers are then used to make connections to the spam recipients' SMTP servers. These open proxies are used to deliver spam in a more anonymous way [3, 17].

3 Related Work

In this section we discuss related work using honeypots to study or fight spam.

3.1 Honeyd

`Honeyd` [25] is a network daemon which implements a framework for creating virtual honeypots with the purpose of emulating several computers, operating systems and network services.

This software has become very popular to implement low-interaction honeypots. It can also be used, although in a limited way, for the study and prevention of spam. Two `Honeyd` modules can be used in this way:

Open relay emulator: it emulates the characteristics of an open relay SMTP server. The email messages received are all stored and never delivered to its recipients.

Open proxy emulator: it emulates the behaviour of an open HTTP (Hypertext Transfer Protocol) proxy, implementing a subset of the HTTP protocol [5], in particular the GET and CONNECT commands. GET requests are answered with a generic or error page in HTML (hypertext Markup Language). Attempts to use the CONNECT command to connect to the SMTP port of a remote server are redirected to a local SMTP emulator, described above.

By using these two emulators it is possible to capture emails for further studies and also for sending them to collaborative projects creating spam filter signatures [25].

The modules mentioned above, however, are restricted to HTTP proxy emulation. This fact severely impacts the amount of emails that can be captured. A contribution of this work was the development of a SOCKS proxy emulator to be used with `Honeyd` to capture spam. The details of this new module can be seen in section 4.

3.2 HoneySpam

HoneySpam [1] is a framework based on honeypots that has the objective of fighting spam at its origin, instead of doing so in the destination. The system architecture is designed to be used as a spam identification and blocking tool on a corporate network. Its main functionalities are described below.

Harvesting interference: harvesting is a process of collecting valid email addresses in Web pages. This module creates Web pages with a large number of links between them, slowing down the process.

Block list creation: the HTML pages generated in the previous module contain email addresses dynamically created pointing to a local SMTP server. As soon as these addresses start receiving emails, this information can be used to create email block lists based on the source IP addresses sending spam.

Another objective of generating invalid, dynamically generated email addresses is to pollute the spammer's email databases, decreasing its usefulness.

Open proxy and open relay emulation: it also uses the open relay and open HTTP proxy implemented by `Honeyd` [25], described in section 3.1. Connections to these emulated services are registered and used to block traffic from the source IP addresses.

Because this framework main goal is to block spam using filters, it is not the right tool to collect spam for further studies.

4 Honeypot Network to Capture Spam

In this section we present the architecture used to implement a set of sensors, based on honeypots, to capture spam. We also present some preliminary results based on the analysis of the data collected.

4.1 Architecture

The implemented architecture, shown in Figure 1, used 10 low-interaction honeypots for capturing spam. These honeypots were deployed in 5 Brazilian broadband networks (both cable and ADSL – Asymmetric Digital Subscriber Line), in 4 Brazilian cities for 15 months. They were deployed at volunteers' homes, to be exposed to the same conditions a typical home user computer with broadband connectivity would be.

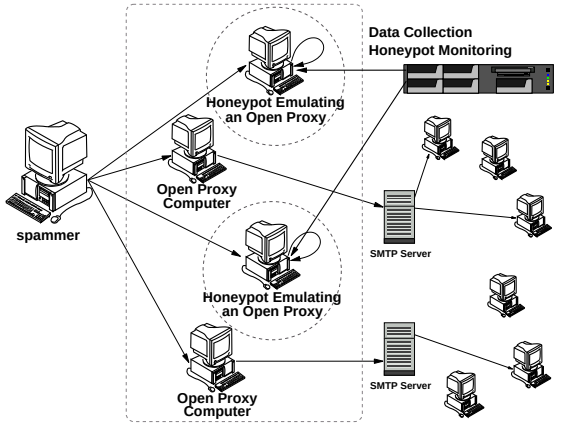


Figure 1: Diagram of the implemented architecture.

Table 1: TCP ports being emulated by the honeypots.

protocol	TCP port
SMTP	25
HTTP	80, 81, 2282, 3128, 3332, 3382, 3802, 4480, 5490, 6588, 8000, 8080, 8090, 11120, 57123, 63809, 65506
SOCKS	559, 1029, 1080, 1202, 1813, 1978, 1979, 2280, 2425, 3127, 3380, 3800, 4471, 4777, 4894, 5748, 6042, 7531, 9938, 10000, 10001, 10232, 11117, 15859, 19086, 24971, 24972, 24973, 30021, 30022, 35612, 38994, 40934, 41457, 57123, 63808

In each ISP we installed a residential connection, with dynamic IP address and less bandwidth, and a commercial one, usually with a static IP address and more bandwidth.

The honeypots were configured to emulate open proxy computers. Any spammer trying to use one of these honeypots to send spam would tend to believe that the emails were delivered successfully.

Another part of the architecture was a central server, configured to collect all the spam captured, as well as to periodically monitor the honeypots.

In the following sections we describe these components in more detail.

4.1.1 Honeypots Basic Configuration

Each honeypot was configured using an i386-based computer, with OpenBSD as its operating system and with the following hardware characteristics: 2.66 GHz Intel Celeron, 512 MB of RAM, 80 GB of disk and a 10/100 network interface.

The honeypot used the `pf` packet filter [8] blocking all incoming traffic, with the exception of traffic used for its management and traffic related to the TCP ports being emulated by `Honeyd`. The list of TCP ports emulated is shown in Table 1.

The honeypot's internal clock was synchronized using NTP (Network Time Protocol) [23]. The timezone used in each honeypot was GMT (Greenwich Mean Time), for standartization purposes and also to remain independent from local daylight saving changes.

4.1.2 Capture

The email capture on the honeypots was implemented using `Honeyd` together with the SMTP, HTTP and SOCKS proxy emulation subsystems.

Although `Honeyd` already had both SMTP and HTTP proxy subsystems, it does not have support for SOCKS protocol versions 4 and 5. In this way, a SOCKS emulator was developed as part of this work and to complement the other emulators. Additionally, the existing `Honeyd` modules were also modified.

The description of the emulation subsystems follows.

SMTP emulation: this `Honeyd` module receives traffic directed to port 25/TCP and emulates the responses of a typical SMTP server. For each message received, it behaves as an open relay server, accepting the message. The message, however, is stored locally and never delivered to its recipients.

HTTP proxy emulation: this `Honeyd` module is configured to run on several TCP ports normally associated with HTTP proxy services: 80/TCP, 8080/TCP, 3128/TCP, among other TCP ports listed in Table 1. When the spammer connects to this module, he typically requests to the proxy a connection to the IP address of the victim's SMTP server, on port 25/TCP. The proxy emulator, instead of granting the request, makes a connection to its own local SMTP server, which sends an SMTP response back to the spammer. This response gives an SMTP banner similar to the one that would be given by the requested SMTP server. The spammer then starts sending his emails, convinced that he is connected to the originally requested SMTP server.

SOCKS proxy emulation: this emulator was developed as part of this work. It emulates a SOCKS [18] (versions 4 and 5) proxy, receiving network traffic to the TCP ports normally associated to this service, as shown in Table 1. This emulator acts like a proxy that does not require authentication and allows con-

nections coming from any IP address. After connecting to this emulator, the spammer requests a connection to an outside TCP address and port. If the requested TCP port is different from 25/TCP (SMTP), the emulator returns an error message to the client. If the requested port is 25/TCP, redirects the spammer to the local SMTP emulator, as described above in the HTTP proxy emulator.

Every transaction made by the `Honeyd` modules is logged, including timestamp, client IP, destination IP and TCP port as well as protocol version requested.

A description of the modifications made to the existing `Honeyd` modules follows.

Storage structure: the SMTP module is responsible for storing the emails received on disk, in a directory structure that takes into account the origin IP address and the message itself. We made some modifications to this structure scheme to include the date and the TCP port used for the message delivery, in order to help process the information.

Additional information logged: the HTTP proxy module has been modified to log more information, like the connection method, origin IP address, destination IP address, TCP port and status. The SMTP module log generation has also been modified in order to use the storage structure changes explained previously.

Data compression: data compression support has been added to the SMTP module, using the `zlib` library. This change allowed us to keep every email received in a compressed form, saving disk space.

4.1.3 Spam Collection

The spam captured on the honeypot was collected, at regular intervals, by a central server. This collection mechanism was implemented using the remote copy and synchronization program `rsync`, through an encrypted SSH (Secure Shell) tunnel.

Each honeypot had the ability to keep the captured spams for several days. However, after a successful synchronization with the central server, the local data was deleted from the honeypot to save disk space.

4.1.4 Honeypot Status Monitoring

In order to guarantee that every honeypot was working correctly, a status monitoring scheme was implemented. By executing this monitoring function several times a day, it was possible to quickly check if every honeypot was operating as expected.

This status monitoring procedure was specially important because of the nature of the connectivity and physical location of the honeypots. Broadband connections typically have less quality of service when compared to dedicated connections. As the honeypots were installed in residential environments, stability of electric power supply was also expected to have more problems than those in controlled environments, like Internet Data Centers.

Each status check performed is described below:

- Honeypot connectivity. Tests if the honeypot is accessible from the central server.
- System uptime. Particularly useful test to check for unexpected reboots, caused, for example, by electrical power problems.
- System load. Measures the number of processes in the operating system execution queue in the last 1, 5 and 15 minutes. The honeypot load is related to the amount of processes running in a given moment.
- Honeypot disk usage. Very important check to make sure that each honeypot has enough disk space to store the captured spams.
- Clock synchronization. This test evaluates the honeypot clock difference in comparison to an external reference, using NTP. The correct clock synchronization on the honeypots is very important to generate reliable log information.
- Critical processes running on the honeypot. This test checks if certain critical processes, important to capture spam, are running on the honeypot.
- Bandwidth usage and email capture rate. Measures the bandwidth usage (inbound and outbound, both in packets/s and bytes/s) and the email capture rate in the last 15 minutes.
- Data synchronization with the central server. Checks the last time the data synchronization with the central server occurred.

4.2 Results

In this section we show some of the results from the analysis made on the captured emails, taking into account their origin and TCP ports used.

As shown in Table 2, this project collected spams for 15 months, capturing approximately 525 million spams that would have been delivered to 4.8 billion recipients. The daily average amount of emails collected, combining all 10 honeypots, was approximately 1.1 million spams. These email messages came from 216,888 different IP

Table 2: General statistics related to the captured emails.

Data collection start	2006-06-10
Data collection end	2007-09-18
Days of collected data	466
Total emails collected	524,585,779
Total recipients	4,805,521,964
Average recipients per spam	9.16
Average emails per day	1,125,720
Unique IPs sending spam	216,888
Unique ASes sending spam	3,006
Unique Country Codes	165

Table 3: Top Country Codes originating spams.

#	CC	Emails	%
01	TW	385,189,756	73.43
02	CN	82,884,642	15.80
03	US	29,764,293	5.67
04	CA	6,684,667	1.27
05	JP	5,381,192	1.03
06	HK	4,383,999	0.84
07	KR	4,093,365	0.78
08	UA	1,806,210	0.34
09	DE	934,417	0.18
10	BR	863,657	0.16

addresses, allocated to 165 different countries (country codes, as defined by ISO 3166 [15]).

It is possible to note a significant concentration on the spam's country of origin. 97% of all spam captured by the project originated from only 5 countries: Taiwan, China, United States, Canada and Japan. This result can be seen in Table 3. Taiwan contribution in terms of the spam origin is also very significant, with 73.43% of all the messages sent. If we analyze the top 10 countries originating spam to the project's honeypots, this number is 99.50% of all messages.

In Figure 2 it is possible to see the percentage of spams sent by country code, during the project duration.

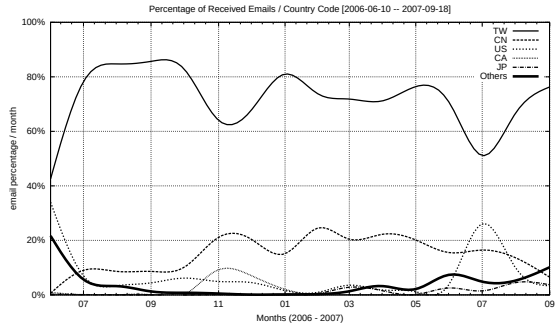


Figure 2: Percentage of received emails by country code.

Table 4: Top ASes originating spam.

#	ASN	AS Name	Emails	%
01	9924	TFN-TW (TW)	170,998,167	32.60
02	3462	HINET (TW)	131,381,486	25.04
03	17623	CNCGROUP (CN)	65,214,192	12.43
04	4780	SEEDNET (TW)	54,430,806	10.38
05	9919	NCIC-TW (TW)	9,186,802	1.75
06	4837	CHINA169 (CN)	9,025,142	1.72
07	33322	NDCHOST (US)	8,359,583	1.59
08	4134	CHINANET (CN)	7,287,251	1.39
09	18429	EXTRALAN (TW)	6,746,124	1.29
10	7271	LOOKAS (CA)	5,599,442	1.07

An AS (Autonomous System) is an connected group of one or more network blocks with a well-defined routing policy [9]. Table 4 shows the spam distribution by originating AS. Again it is possible to see that the activity is concentrated: the top 4 ASes originating spam were responsible for approximately 80% of all activity recorded.

If we take into account the countries that are associated to these ASes, again Taiwan and China play an important role. Also, the top 2 ASes, TFN-TW and HINET, both from Taiwan, originated 58% of all spam captured.

Another aspect analyzed was the TCP port used by the spammers to inject spams. Although each honeypot was configured to emulate several TCP ports, as seen in Table 1, only 11 TCP ports were used. Table 5 shows the set of TCP ports used for sending spam, as well as the protocol (HTTP or SOCKS) and the service normally associated with each port.

The development of the SOCKS support, as part of this work, was very important as port 1080/TCP, which is the default port associated to this protocol, was the port that collected the most spam during the period.

Some of the TCP ports abused by the spammers are associated with popular services — examples are 80/TCP, 8000/TCP and 8080/TCP, normally used by the `http` service. Other TCP ports, like 6588/TCP and 4480/TCP (`AnalogX` and `Proxy+`, proxies normally used by home users) show an attempt by spammers to explore misconfigured home proxies. It is also important to note the activity on TCP ports associated with proxies installed by malicious code, like `MyDoom` and `Sobig.f`, indicating that spammers are actively using previously infected machines to send spam.

Figure 3 shows the percentage of emails received by TCP port, during the collecting period.

The requests made to the HTTP and SOCKS proxy modules were also analyzed. These results are shown in Table 6. For the HTTP module the requests were divided as follows: outgoing connections to port 25/TCP, outgoing connection to other TCP ports, "GET" requests

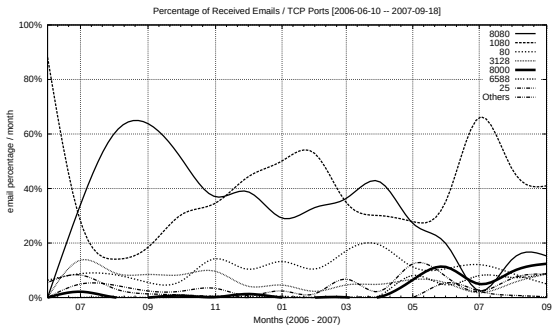


Figure 3: Percentage of emails received by TCP port.

(attempts to use the proxy to access a Web server) and errors (invalid commands). For the SOCKS module we made a similar separation: connections to port 25/TCP, connections to other ports and errors. It is important to note that the majority of the requests (97.62% for HTTP and 87.31% for SOCKS) were attempts to connect to port 25/TCP on third party machines. This makes it clear that the main objective of the majority of the connections to these honeypots was to deliver spam, and not other type of proxy abuse.

Table 5: TCP ports abused.

#	Port	Protocol	Usual Service	%
01	1080	SOCKS	socks	37.31
02	8080	HTTP	alternate http	34.79
03	80	HTTP	http	10.92
04	3128	HTTP	Squid	6.17
05	8000	HTTP	alternate http	2.76
06	6588	HTTP	AnalogX	2.29
07	25	SMTP	smtp	1.46
08	4480	HTTP	Proxy+	1.38
09	3127	SOCKS	MyDoom Backdoor	1.00
10	3382	HTTP	Sobig.f Backdoor	0.96
11	81	HTTP	alternate http	0.96

Table 6: Requests made to the HTTP and SOCKS modules.

Module	Type	Requests	%
HTTP	connection to 25/TCP	89,496,969	97.62
	connection to other	106,615	0.12
	get	225,802	0.25
	errors	1,847,869	2.01
	total	91,677,255	100.00
SOCKS	connection to 25/TCP	46,776,884	87.31
	connection to other	1,055,081	1.97
	errors	5,741,908	10.72
	total	53,573,873	100.00

5 Conclusions

This paper described the design and implementation of an architecture, based on honeypots, for the study of the spam problem, in particular the abuse of open proxies. It also described some extensions to the current spam capture technology and showed that this architecture, in place for 15 months, proved to be very efficient in capturing spam.

The analysis of the received spam made it clear that the origin of the IP addresses exploring open proxies for sending spam was very concentrated: approximately 97% of all email came from only 5 different country codes. Approximately 89% of all messages came from two countries: Taiwan and China. When the origin was analyzed based on source Autonomous System a similar concentration was observed. The top 4 ASes (out of a total of 3,006) were responsible for almost 80% of all the emails observed. This high concentration of activity, both in terms of country codes and Autonomous Systems, could be helpful to fight the problem: if acceptable use and anti-spam policies were enforced in these networks, it could have a significant impact in reducing spam.

The results also show that most of the requests received by the proxy modules were directed to the SMTP port (25/TCP) of third party machines. This shows that the main objective of the majority of these connections was, in fact, spam delivery. To mitigate the delivery of spam via the abuse of residential and dynamic IP networks, a method that could be very effective is port 25 management [20]. The main objective is to differentiate email client submission traffic from email transport between SMTP servers. Traffic between mail servers would continue to use port 25/TCP, but email submission traffic from clients to servers would use an alternate submission mechanism: the authenticated Message Submission for Email, on port 587/TCP [6]. With this mechanism in place it is possible to filter outgoing port 25/TCP traffic in residential and dynamic IP networks, reducing the amount of spam leaving these networks and making the problem more traceable for ISPs [20].

After this initial phase of operation and validation of the technology in use, some directions for future work are possible.

All data was analyzed only based on its origin, not on its content. As a future work it could be possible to conduct a more detailed analysis of the collected emails, taking into account the email body, recipient addresses, among others.

It is also important to propose best practices recommendations to Brazilian ISPs in order to reduce the abuse of their networks for sending spam.

Another possible line of work would be the deploy-

ment of sensors in other countries to have a more global view of the problem.

Acknowledgements

The authors would like to thank the Brazilian Internet Steering Committee (CGI.br), and the Brazilian Network Information Center (NIC.br), for their financial support regarding equipment and connectivity costs. We would also like to thank the CERT.br team for their enthusiasm and dedication.

References

[1] Andreolini, M., Bulgarelli, A., Colajanni, M., and Mazzoni, F. Honeyspam: Honeypots fighting spam at the source. In *Proceedings of the USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI '05)*, pages 77–83, Cambridge, MA, USA, July 2005.

[2] Cerf, V. G. Spam, spim, and spit. *Commun. ACM*, 48(4):39–43, 2005.

[3] Chuvakin, A. Honeynets: High Value Security Data – Analysis of real attacks launched at a honeypot. *Elsevier Network Security*, 2003(8):11–15, August 2003. ISSN: 1353-4858.

[4] Cranor, L. F. and LaMacchia, B. A. Spam! *Commun. ACM*, 41(8):74–83, 1998.

[5] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. `http://www.ietf.org/rfc/rfc2616.txt`, June 1999.

[6] Gellens, R. and Klensin, J. RFC 4409: Message Submission for Mail. `http://www.ietf.org/rfc/rfc4409.txt`, April 2006.

[7] Hambridge, S. and Lunde, A. RFC 2635: DON'T SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam*). `http://www.ietf.org/rfc/rfc2635.txt`, June 1999.

[8] Hartmeier, D. Design and performance of the openbsd stateful packet filter (pf). In *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference (FREENIX '02)*, Monterey, CA, USA, June 2002.

[9] Hawkinson, J. and Bates, T. RFC 1930: Guidelines for creation, selection, and registration of an Autonomous System (AS). `http://www.ietf.org/rfc/rfc1930.txt`, March 1996.

[10] Hayes, B. Spam, spam, spam, lovely spam. *American Scientist*, 91(3):200–204, May–June 2003.

[11] Hoepers, C., Steding-Jessen, K., and Chaves, M. H. P. C. Projeto e Desenvolvimento de um Sistema de Controle e Acompanhamento de Notificações de Spam. In *Anais do V Simpósio sobre Segurança em Informática (SSI'2003)*, São José dos Campos, SP, Novembro 2003.

[12] Holz, T. A short visit to the bot zoo. *IEEE Security & Privacy*, 3(3):76–79, May-June 2005. `http://www-pi1.informatik.uni-mannheim.de/publications/show/13`.

[13] Honeynet Project. Know Your Enemy: Honeynets. `http://www.honeynet.org/papers/honeynet/`.

[14] Ianelli, N. and Hackworth, A. Botnets as a vehicle for online crime. `http://www.cert.org/archive/pdf/Botnets.pdf`, December 2005. CERT Coordination Center, Carnegie Mellon University.

[15] International Organization for Standardization. ISO 3166: Codes for the representation of names of countries and their subdivisions – Part 1: Country codes. `http://www.iso.org/iso/country_codes.htm`, November 2006.

[16] Klensin, J. RFC 2821: Simple Mail Transfer Protocol. `http://www.ietf.org/rfc/rfc2821.txt`, April 2001.

[17] Krawetz, N. Anti-honeypot technology. *IEEE Security & Privacy*, 2(1):76–79, January–February 2004.

[18] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and Jones, L. RFC 1928: SOCKS Protocol Version 5. `http://www.ietf.org/rfc/rfc1928.txt`, March 1996.

[19] Lindberg, G. RFC 2505: Anti-Spam Recommendations for SMTP MTAs. `http://www.ietf.org/rfc/rfc2505.txt`, February 1999.

[20] Messaging Anti-Abuse Working Group (MAAWG). Managing Port 25 for Residential or Dynamic IP Space. `http://www.maawg.org/port25/MAAWG_Port25rec0511.pdf`, Dec 2005.

[21] Messaging Anti-Abuse Working Group (MAAWG). Email Metrics Program: Report #5 – First Quarter 2007. `http://www.maawg.org/about/MAAWG20071Q_Metrics_Report.pdf`, June 2007.

[22] Milletary, J. Technical trends in phishing attacks. http://www.cert.org/archive/pdf/Phishing_trends.pdf, October 2005. CERT Coordination Center, Carnegie Mellon University.

[23] Mills, D. RFC 4330: Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. http://www.ietf.org/rfc/rfc4330.txt, January 2006.

[24] Postel, J. RFC 706: On the junk mail problem. http://www.ietf.org/rfc/rfc0706.txt, November 1975.

[25] Provos, N. A Virtual Honeypot Framework. In *Proceedings of 13th USENIX Security Symposium*, pages 1–14, San Diego, CA, USA, August 2004. http://www.usenix.org/publications/library/proceedings/sec04/tech/provos.html.

[26] Provos, N. and Holz, T. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 1st edition, July 2007. ISBN-13: 978-0321336323.

[27] Sauver, J. S. Spam zombies and inbound flows to compromised customer systems. In *Proceedings of the MAAWG General Meeting*, March 2005. http://darkwing.uoregon.edu/~joe/zombies.pdf.

[28] Shirey, R. RFC 2828: Internet Security Glossary. http://www.ietf.org/rfc/rfc2828.txt, May 2000.

[29] Spitzner, L. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 1st edition, September 2002. ISBN: 0321108957.

[30] The Honeynet Project. *Know Your Enemy: Learning about Security Threats*. Addison-Wesley Professional, 2nd edition, May 2004. ISBN: 0321166469.

[31] Zakon, R. RFC 2235: Hobbes' Internet Timeline. http://www.ietf.org/rfc/rfc2235.txt, November 1997.

B APÊNDICE B - ARTIGOS EM CONGRESSOS

B.1 LACNIC XI

Em cumprimento a requisito exigido para a defesa da Tese, foi escrito e submetido um artigo para o seguinte congresso: "3° Evento de Segurança de Redes para América Latina e Caribe", realizado em conjunto com a décima primeira reunião anual do LACNIC (LACNIC XI), na cidade de Salvador/BA em maio de 2008.

Este artigo, intitulado "Políticas e Padrões para a Redução do Abuso de *Proxies* Abertos para o Envio de Spam" foi aceito no referido congresso e segue em anexo neste apêndice.

Políticas e Padrões para a Redução do Abuso de *Proxies* Abertos para o Envio de *Spam*

Klaus Steding-Jessen[1]
Nandamudi L. Vijaykumar[2]
Antonio Montes[3]
Cristine Hoepers[1]

[1]Núcleo de Informação e Coordenação do Ponto br - NIC.br
Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil - CERT.br
São Paulo, SP, Brasil
{jessen,cristine}@cert.br
[2]Instituto Nacional de Pesquisas Espaciais - INPE
Laboratório Associado de Computação e Matemática Aplicada - LAC
São José dos Campos, SP, Brasil
vijay@lac.inpe.br
[3]Ministério da Ciência e Tecnologia - MCT
Centro de Pesquisas Renato Archer - CenPRA
Campinas, SP, Brasil
antonio.montes@cenpra.gov.br

Resumo. Este artigo discute o problema do abuso de *proxies* abertos para o envio direto de *emails* por *spammers*, incluindo a análise de alguns resultados do projeto SpamPots que evidenciam este problema. Também são descritos um conjunto de políticas e padrões para a mitigação do abuso de *proxies* abertos para o envio de *spam* e para aumentar a rastreabilidade de *spammers* e fraudadores. Estes padrões procuram diferenciar a submissão do transporte de *emails*, já foram avaliados pela comunidade Internet e estão em uso em algumas redes de banda larga de caráter residencial.

Abstract. In this paper we discuss how spammers abuse open proxies to perform direct delivery, including the analysis of some results from the SpamPots Project that emphasize this kind of abuse. We also describe a set of policies and standards to mitigate the abuse of open proxies to send spam, as well as to enhance the traceability of spammers and fraudsters. These standards aim to differentiate email submission from email transport, and have been evaluated by the Internet community and are in use in some residential broadband networks.

1 Introdução

O *spam* é uma das formas de abuso da Internet que mais tem crescido, atualmente sendo responsável por uma parcela significativa dos *emails* que trafegam na rede [6, 13]. Além disso, o *spam* tem sido amplamente utilizado para enviar mensagens relacionadas com *phishing* (mensagem que procura induzir usuários ao fornecimento de dados pessoais e financeiros) e para disseminação de códigos maliciosos [14].

Existe, também, um aumento da sofisticação dos *softwares* de envio de *spam*, o que torna as técnicas existentes de bloqueio ainda menos eficientes e a rastreabilidade do *spammer* (remetente do *spam*) mais difícil [3, 14]. Um exemplo disso é o crescimento na utilização de máquinas infectadas por códigos maliciosos, como os *bots* (programas que, além de serem capazes de se propagar através da exploração de vulnerabilidades em computadores, podem ser controlados remotamente por um invasor), para o envio de *spam* e *phishing* [14], permitindo que o *spammer* permaneça no anonimato. Máquinas com *proxies* abertos são abusadas de maneira similar, como ficou evidente nos resultados do projeto SpamPots [18].

Este artigo descreve um conjunto de políticas e padrões para a mitigação do abuso de *proxies* abertos para o envio de *spam* e para aumentar a rastreabilidade de fraudadores e *spammers*. Estes padrões, que procuram diferenciar a submissão do transporte de *emails*, já foram avaliados pela comunidade Internet e já estão em uso em algumas redes de banda larga de caráter residencial.

Este artigo está organizado como segue. Na seção 2 é discutido o problema do abuso de *proxies* abertos para o envio direto de *emails* por *spammers*, incluindo a análise de alguns resultados do projeto SpamPots que evidenciam este problema. Na seção 3 são descritos os padrões

e políticas que podem ser adotados para reduzir o volume de *spams*, bem como aumentar a sua rastreabilidade. As conclusões e considerações finais estão na seção 4.

2 O problema do Abuso de *Proxies* Abertos

Um *proxy* é um servidor que atua como intermediário entre um cliente e outro servidor, ou seja, um serviço de *proxy* faz conexões em nome de outros clientes [4]. Quando um *proxy* está mal configurado, ele permite o redirecionamento indiscriminado de conexões de terceiros para quaisquer endereços IP e portas, sendo denominado *proxy* aberto. *Proxies* abertos são também intencionalmente instalados por códigos maliciosos, como *bots* e cavalos-de-tróia.

Spammers continuamente realizam varreduras na Internet em busca de computadores que estejam com *proxies* abertos [11], que são explorados para efetuar conexões para os servidores SMTP dos destinatários do *spam*. Os *spammers* utilizam estes *proxies* abertos para obter anonimicidade [2, 11].

Na seção 2.1 será descrita brevemente uma arquitetura utilizada para a captura de *spams* que seriam enviados via abuso de *proxies* abertos. Na seção 2.2 são mostrados os resultados gerais observados em 15 meses de captura de dados. Uma discussão específica sobre as evidências de entrega direta de mensagens via abuso de *proxies* é apresentada na seção 2.3.

2.1 Arquitetura Utilizada para Captura de Dados de *Proxies* Abertos

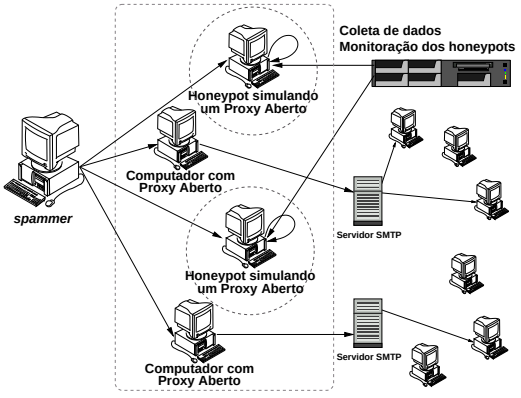


Figura 1: Diagrama da arquitetura do Projeto SpamPots.

A arquitetura implementada, exibida na Fig. 1, contou com 10 *honeypots* de baixa interatividade [16], responsáveis pela captura de *spams*, instalados em redes brasileiras de banda larga de 5 operadoras diferentes (cabo e ADSL – *Asymmetric Digital Subscriber Line*). Os *honeypots* foram instalados nas residências de voluntários para refletir as condições a que estão submetidos computadores típicos de usuários residenciais com conexões banda larga.

Tabela 1: Portas emuladas pelos *honeypots*.

protocolo	portas TCP
SMTP	25
HTTP	80, 81, 2282, 3128, 3332, 3382, 3802, 4480, 5490, 6588, 8000, 8080, 8090, 11120, 57123, 63809, 65506
SOCKS	559, 1029, 1080, 1202, 1813, 1978, 1979, 2280, 2425, 3127, 3380, 3800, 4471, 4777, 4894, 5748, 6042, 7531, 9938, 10000, 10001, 10232, 11117, 15859, 19086, 24971, 24972, 24973, 30021, 30022, 35612, 38994, 40934, 41457, 57123, 63808

Em cada operadora foi instalada uma conexão na modalidade doméstica, tipicamente com IP dinâmico e menor largura de banda, e uma na modalidade empresarial, com IP fixo e geralmente com maior largura de banda. Os *honeypots* permaneceram instalados em 4 cidades brasileiras por um período de 15 meses.

Estes *honeypots* foram configurados de modo a simular computadores com *proxies* abertos. Desse modo, um *spammer* que tentasse abusar de um destes honeypots para o envio de *spam*, seria levado a acreditar que teve sucesso em enviar seus *emails*.

Também fez parte da arquitetura um servidor central, configurado para realizar a coleta dos *spams* capturados, bem como a monitoração periódica dos *honeypots*.

A captura de *emails* nos *honeypots* utilizou o *software* `Honeyd` [15] em conjunto com subsistemas de emulação de SMTP, *proxies* HTTP e SOCKS 4 e 5, este último implementado como parte do projeto. As portas TCP emuladas nos *honeypots* estão listadas na Tab. 1.

Todas as transações efetuadas pelos emuladores são armazenadas em arquivos de eventos (*log*) com informações como data e hora, IP de origem, IP e porta pretendidos de destino, assim como versão do protocolo utilizado.

Todos os *spams* capturados pelos *honeypots* foram coletados em intervalos regulares por um servidor central. Esse mecanismo de coleta foi implementado utilizando-se o programa de cópia e sincronização remota de dados `rsync`, através de um túnel criptografado implementado com o programa de *login* remoto SSH (*Secure Shell*).

2.2 Resultados Gerais

Nesta seção são mostrados alguns resultados da análise dos *spams* coletados, tendo como base a sua origem e as portas utilizadas para o seu envio.

Como mostrado na Tab. 2, o projeto coletou *spams* por cerca de 15 meses, tendo capturado no período aproximadamente 525 milhões de *spams* que seriam entregues a 4,8 bilhões de destinatários. Somando-se a coleta realizada em cada um dos 10 *honeypots*, a média diária foi de aproximadamente 1,1 milhão de *spams* recebidos. As mensa-

gens capturadas se originaram de 216.888 diferentes endereços IPs, alocados para 165 países (*country codes*, definidos pela ISO 3166 [8]) de origem [1].

Tabela 2: Estatísticas gerais referentes aos *emails* capturados.

Início da coleta dos dados	10/06/2006
Fim da coleta dos dados	18/09/2007
Dias de dados coletados	466
Total de emails coletados	524.585.779
Total de destinatários	4.805.521.964
Média de destinatários por spam	9,16
Média de emails por dia	1.125.720
IPs únicos que enviaram spam	216.888
ASs únicos que enviaram spam	3.006
Países (country codes) de origem	165

Foi observada, contudo, uma grande concentração nos países de origem dos *spams*. De todo o *spam* que foi capturado pelo projeto, 97% foi originado em apenas 5 países: Taiwan, China, Estados Unidos, Canadá e Japão. Também é significativa a contribuição de Taiwan, com 73,43% do total das mensagens enviadas.

Na Fig. 2 tem-se a contribuição percentual, por *country code*, ao longo do período de duração do projeto. É possível notar, por exemplo, que a contribuição de Taiwan e China é mais regular ao longo do período, se comparada com os demais países.

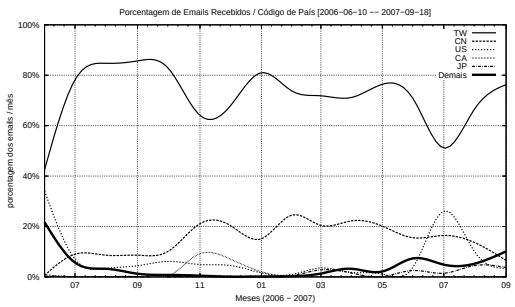


Figura 2: *Emails* recebidos em função do país de origem, percentualmente.

A Tab. 3 mostra a distribuição dos *spams* em função dos ASs de origem. Novamente é possível ver uma clara concentração de atividade: os 4 ASs que mais originaram *spams* foram responsáveis por aproximadamente 80% de toda a atividade registrada no período.

Se levarmos em conta os países associados a esses ASs, novamente Taiwan e China se sobressaem: dos 10 ASs de maior atividade, 8 são desses dois países. Também chama a atenção que apenas os dois primeiros ASs, TFN-TW e HINET, ambos de Taiwan, originaram aproximadamente 58% do total de *spams* capturados.

Outro elemento para análise foi a porta TCP usada pelo *spammer* para injetar *emails*. Embora cada *honeypot* tivesse sido configurado para emular um grande número de portas TCP, como visto na Tab. 1, apenas 11 foram

Tabela 3: ASs mais freqüentes de origem dos *spams*.

#	ASN	Nome do AS	*Emails*	%
01	9924	TFN-TW (TW)	170.998.167	32,60
02	3462	HINET (TW)	131.381.486	25,04
03	17623	CNCGROUP (CN)	65.214.192	12,43
04	4780	SEEDNET (TW)	54.430.806	10,38
05	9919	NCIC-TW (TW)	9.186.802	1,75
06	4837	CHINA169 (CN)	9.025.142	1,72
07	33322	NDCHOST (US)	8.359.583	1,59
08	4134	CHINANET (CN)	7.287.251	1,39
09	18429	EXTRALAN (TW)	6.746.124	1,29
10	7271	LOOKAS (CA)	5.599.442	1,07

efetivamente utilizadas para o envio de *spam*. A Tab. 4 mostra a relação de portas TCP que foram utilizadas para o envio de *spam*, em conjunto com o protocolo utilizado (HTTP ou SOCKS) bem como o serviço normalmente associado a cada porta.

Tabela 4: Portas TCP abusadas.

#	Porta	Protocolo	Serviço Usual	%
01	1080	SOCKS	socks	37,31
02	8080	HTTP	alternate http	34,79
03	80	HTTP	http	10,92
04	3128	HTTP	Squid	6,17
05	8000	HTTP	alternate http	2,76
06	6588	HTTP	AnalogX	2,29
07	25	SMTP	smtp	1,46
08	4480	HTTP	Proxy+	1,38
09	3127	SOCKS	MyDoom Backdoor	1,00
10	3382	HTTP	Sobig.f Backdoor	0,96
11	81	HTTP	alternate http	0,96

A porta que mais recebeu *spams* em todo o período de coleta foi a porta 1080/TCP, que é a porta padrão associada ao protocolo SOCKS. Entre as outras portas exploradas pelos *spammers*, algumas são relacionadas a serviços populares, como por exemplo a 80/TCP, 8000/TCP e 8080/TCP (normalmente associadas com o serviço `http`). Já a busca por outras portas, como 6588/TCP e 4480/TCP (`AnalogX` e `Proxy+`, *proxies* normalmente usados por usuários residenciais) indicam tentativa de explorar problemas de configuração de *proxies* residenciais. Chamou a atenção, ainda, atividade em portas associadas a *proxies* instalados por códigos maliciosos, como `MyDoom` e `Sobig.f`, indicando que *spammers* estão ativamente utilizando máquinas previamente infectadas para o envio de *spam*.

A Fig. 3 mostra os *emails* recebidos em função da porta TCP, percentualmente, ao longo do período de coleta.

2.3 Evidências de Entrega Direta de Mensagens

Foram analizadas, também, as requisições de conexão feitas para os módulos de *proxy* HTTP e SOCKS. Para

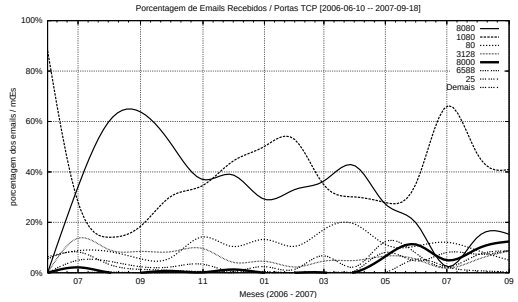


Figura 3: Emails recebidos em função da porta TCP, percentualmente, ao longo do período.

esta análise foi contabilizada apenas a primeira requisição feita por um dado endereço IP, ou seja, considerou-se apenas a primeira intenção de abuso do *proxy* aberto.

Para o módulo de HTTP, as requisições foram divididas em:

- conexão para porta 25/TCP de destino;
- conexão para demais portas de destino;
- requisição "*GET*" (tentativa de usar o *proxy* tipicamente para acessar páginas *Web*); e
- erros (comandos inválidos).

Para o módulo SOCKS efetuou-se divisão similar:

- conexão para porta 25/TCP de destino;
- conexão para demais portas de destino; e
- erros.

Estes resultados são mostrados na Tab. 5.

Tabela 5: Requisições efetuadas aos módulos HTTP e SOCKS.

Módulo	Tipo	Requisições	%
HTTP	conexão para 25/TCP	89.496.969	97,62
	conexão para demais	106.615	0,12
	get	225.802	0,25
	erros	1.847.869	2,01
	total	91.677.255	100,00
SOCKS	conexão para 25/TCP	46.776.884	87,31
	conexão para demais	1.055.081	1,97
	erros	5.741.908	10,72
	total	53.573.873	100,00

É possível observar que, tanto nas requisições HTTP (97,62%) quanto nas SOCKS (87,31%), a grande maioria foi de tentativas de conexão para máquinas externas na porta de SMTP (25/TCP).

O abuso de *proxies* abertos, com objetivo de entrega direta na porta 25/TCP, é ilustrado na Fig. 4. *Spammers*, fraudadores e códigos maliciosos abusam de computadores em redes residenciais para entregar emails diretamente ao MTA (*Mail Transfer Agent*) do destinatário, como indicado em linhas tracejadas. Desta forma, subvertem o caminho legítimo de uma mensagem, que passaria pelo MTA do provedor do usuário, representado na figura pelas linhas contínuas. Ao conectar-se diretamente no MTA de destino, o *spammer* burla mecanismos de controle de vazão, dificulta a filtragem de mensagens com base na origem ou no volume e, principalmente, torna o envio do *spam* anônimo.

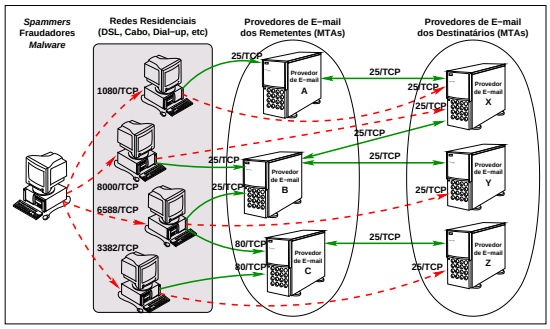


Figura 4: Cenário de abuso de redes residenciais para entrega direta de *email*.

Os números observados evidenciam que, apesar da diversidade de portas de *proxy* sendo abusadas, o objetivo em comum da maioria das conexões para esses *honeypots* era a entrega direta de *spam*, através de conexões de saída com destino à porta 25/TCP de servidores de *email*.

Ficou claro, também, que por conta da diversidade de portas associadas a *proxies* e da possibilidade deles serem colocados em novas portas, conter o abuso através da filtragem do tráfedo de entrada para as portas a eles associadas não é efetivo. Já o bloqueio de conexões de saída com destinho à porta 25/TCP dificulta o abuso.

Na próxima seção serão apresentadas políticas e padrões existentes para gerenciar conexões de saída para a porta 25/TCP, que tem o objetivo de reduzir *spams* partindo de redes com perfil residencial ou que não possuam servidores legítimos de *email*.

3 Políticas e Padrões para Mitigação do Problema

O protocolo SMTP, definido originalmente pela RFC 821, de 1982, com revisão em 2001, na RFC 2821 [9], foi concebido como um protocolo de transferência de mensagens entre servidores de *email*. Porém, também era usado como protocolo de submissão de mensagens entre cliente e servidor. Este uso para implementar ambas as funções era transparente quando cliente e servidor eram executados em uma mesma máquina. Atualmente, com a separação destas funcionalidades ficando mais clara, tornou-se necessária uma melhor definição dos papéis de submissão e transporte de *emails*.

A RFC 2476, de 1998, apresentou pela primeira vez o protocolo para *Message Submission*, que foi atualizado em 2006, na RFC 4409 [5]. Este protocolo fornece um meio para distinguir uma submissão do transporte de mensagens, permitindo assim:

- a aplicação de políticas diferentes para cada tipo de conexão, impedindo *relays* não autorizados ou introdução de *emails* não solicitados;
- a implementação de autenticação na submissão, incluindo aquela realizada remotamente por usuários autorizados;
- a possibilidade de implementar, futuramente, melhorias no serviço de submissão.

A adoção do protocolo de *Message Submission* é uma boa prática reforçada na RFC 5068 (BCP 134) [7] e que tem sido recomendada por diversos fóruns de combate ao *spam*, como o *Messaging Anti-Abuse Working Group* (MAAWG), o *London Action Plan* e a Comissão de Trabalho Anti-Spam (CT-Spam) do Comitê Gestor da Internet no Brasil (CGI.br).

No documento *Managing Port 25 for Residential or Dynamic IP Space – Benefits of Adoption and Risks of Inaction* [12] o MAAWG recomenda para redes de caráter residencial, além da adoção de *Message Submission*, as seguintes medidas:

- requerer autenticação para a submissão de mensagens, como recomendado na RFC 4954 [17];
- não interferir no tráfego para a porta 587/TCP;
- configurar o *software* cliente de *email* para usar porta 587/TCP e autenticação;
- bloquear acesso de saída para porta 25/TCP a partir de todas as máquinas que não sejam MTAs ou explicitamente autorizadas.

Estas recomendações de gerência de porta 25/TCP já são adotadas por diversos membros do MAAWG, incluindo Earthlink, AOL e Comcast. No caso da Comcast, após a adoção da medida, o número de *spams* barrados diariamente na saída chegava a cerca de 700 milhões [10].

A Fig. 5 ilustra a mudança de cenário em uma rede de caráter residencial, após a adoção de *Message Submission* pelos provedores de *email* dos usuários e a implementação de gerência de porta 25.

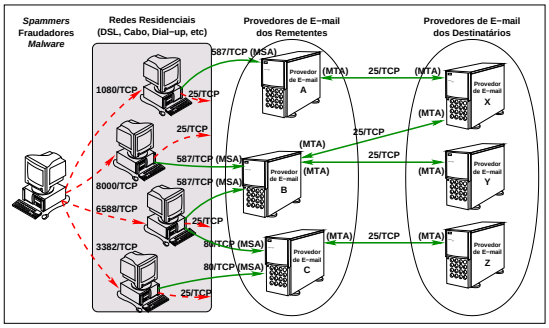


Figura 5: Novo cenário, usando *Message Submission for Mail*, e seu impacto na entrega direta de *emails*.

Na Fig. 5 nota-se que o usuário, conectado via uma rede residencial, envia *emails* normalmente via *Mail Submission Port* (587/TCP) ou via *Webmail* (80/TCP). Mas, os *spammers*, fraudadores e códigos maliciosos não conseguem mais fazer a entrega direta dos *emails* para os provedores de destino, pois a saída de conexões para porta 25/TCP é impedida.

Com este novo cenário provavelmente os *spammers* e fraudadores adaptarão suas ferramentas para furtar as credenciais do usuário e se autenticar em seu MSAs (*Mail Submission Agents*) para o envio de *spam*, como é ilustrado na Fig. 6.

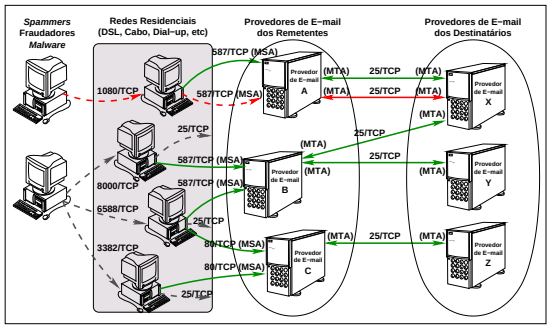


Figura 6: Possível abuso do novo cenário.

Mesmo ocorrendo esta mudança para uso das credenciais do usuário, outro reflexo da implementação de *Message Submission* e Gerência de Porta 25 em redes de perfil residencial é que as mensagens abusivas são mais rastreáveis para os provedores de acesso [12]. Pois, além do provedor poder implementar políticas de controle de vazão de *emails*, ele poderá identificar mais facilmente máquinas infectadas e usuários abusivos.

4 Conclusões

Este artigo discutiu o abuso de *proxies* abertos, em redes brasileiras de banda larga, para o envio de *spam*. Foi levantada, também, a possibilidade de adoção de políticas e padrões para a redução deste tipo de abuso.

Pudemos observar nos resultados do Projeto SpamPots que a maioria das requisições recebidas pelos módulos de *proxy* foram de tentativas de conexão destinadas à porta SMTP (25/TCP) de máquinas externas. Isto evidencia que o objetivo da maioria destas requisições era de fato a entrega de *spam*.

Para mitigar o envio destes *spams*, a partir de redes de caráter residencial e de endereçamento IP dinâmico, um método que poderia ter grande eficácia seria a implementação de *Message Submission* em conjunto com medidas de gerenciamento de porta 25 [12]. A definição de *Message Submission* é padrão já amadurecido na comunidade Internet, tendo passado por revisões junto ao IETF. E, como visto, as medidas de gerência de porta 25 têm demonstrado impacto positivo na coibição de abusos de redes de perfil residencial para envio de *spam*.

Essas medidas têm como objetivo diferenciar o tráfego de transporte de *emails* entre MTAs, do tráfego de

submissão de *emails* por parte de clientes. O tráfego entre servidores continua utilizando a porta 25/TCP, porém, para o tráfego entre clientes e servidores passa a ser utilizado um mecanismo alternativo de submissão de *email*, em conjunto com autenticação [17] e em outra porta de envio (*message submission for email* – 587/TCP) [5, 7]. Com estas medidas implementadas é possível filtrar a saída do tráfego para a porta 25/TCP em redes residenciais e de endereçamento dinâmico, reduzindo o envio de *spam* a partir destas redes e tornando o problema mais rastreável para os provedores [12].

Agradecimentos

Os autores gostariam de agradecer ao Comitê Gestor da Internet no Brasil, CGI.br, e ao Núcleo de Informação e Coordenação do Ponto br, NIC.br, por tornarem o projeto SpamPots possível através do financiamento dos equipamentos e conectividade. Gostariam de agradecer também à equipe do CERT.br, pelo seu entusiasmo e dedicação.

Referências

[1] CERT.br. Resultados Preliminares do Projeto SpamPots: Uso de Honeypots de Baixa Interatividade na Obtenção de Métricas sobre o Abuso de Redes de Banda Larga para o Envio de Spam. http://www.cert.br/docs/whitepapers/spampots/, Setembro 2007.

[2] Chuvakin, A. Honeynets: High Value Security Data – Analysis of real attacks launched at a honeypot. *Elsevier Network Security*, 2003(8):11–15, August 2003. ISSN: 1353-4858.

[3] Cranor, L. F. and LaMacchia, B. A. Spam! *Commun. ACM*, 41(8):74–83, 1998.

[4] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. http://www.ietf.org/rfc/rfc2616.txt, June 1999.

[5] Gellens, R. and Klensin, J. RFC 4409: Message Submission for Mail. http://www.ietf.org/rfc/rfc4409.txt, April 2006.

[6] Hayes, B. Spam, spam, spam, lovely spam. *American Scientist*, 91(3):200–204, May–June 2003.

[7] Hutzler, C., Crocker, D., Resnick, P., Allman, E., and Finch, T. RFC 5068: Email Submission Operations: Access and Accountability Requirements. http://www.ietf.org/rfc/rfc5068.txt, November 2007.

[8] International Organization for Standardization. ISO 3166: Codes for the representation of names of countries and their subdivisions – Part 1: Country codes. http://www.iso.org/iso/country_codes.htm, November 2006.

[9] Klensin, J. RFC 2821: Simple Mail Transfer Protocol. http://www.ietf.org/rfc/rfc2821.txt, April 2001.

[10] Kolstad, R. The Only Good Spam Comes from Hormel. *;login:*, 30(1):2–3, February 2005. http://www.usenix.org/publications/login/2005-02/openpdfs/motd.pdf.

[11] Krawetz, N. Anti-honeypot technology. *IEEE Security & Privacy*, 2(1):76–79, January–February 2004.

[12] Messaging Anti-Abuse Working Group (MAAWG). Managing Port 25 for Residential or Dynamic IP Space – Benefits of Adoption and Risks of Inaction. http://www.maawg.org/port25/MAAWG_Port25rec0511.pdf, Dec 2005.

[13] Messaging Anti-Abuse Working Group (MAAWG). Email Metrics Reports. http://www.maawg.org/about/EMR/, 2007.

[14] Milletary, J. Technical trends in phishing attacks. http://www.cert.org/archive/pdf/Phishing_trends.pdf, October 2005. CERT Coordination Center, Carnegie Mellon University.

[15] Provos, N. A Virtual Honeypot Framework. In *Proceedings of 13th USENIX Security Symposium*, pages 1–14, San Diego, CA, USA, August 2004. http://www.usenix.org/publications/library/proceedings/sec04/tech/provos.html.

[16] Provos, N. and Holz, T. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 1st edition, July 2007. ISBN-13: 978-0321336323.

[17] Siemborski, R., Ed., and A. Melnikov, E. RFC 4954: SMTP Service Extension for Authentication. http://www.ietf.org/rfc/rfc4954.txt, July 2007.

[18] Steding-Jessen, K., Vijaykumar, N. L., and Montes, A. Uso de *Honeypots* de baixa interatividade para o estudo do abuso de *Proxies* abertos para o envio de *Spam*. *To appear in: INFOCOMP Journal of Computer Science*, 2008.

B.2 SBRC 2008

Em cumprimento a requisito exigido para a defesa da Tese, foi escrito e submetido um artigo para o seguinte congresso: "26º Simpósio Brasileiro de Redes de Computadores (SBRC)", na cidade do Rio de Janeiro/RJ em maio de 2008.

Este artigo, intitulado "Caracterização de Estratégias de Disseminação de Spams" foi aceito no referido congresso e segue em anexo neste apêndice.

Caracterização de Estratégias de Disseminação de Spams

Pedro H. Calais Guerra[1], **Dorgival Olavo Guedes**[1], **Wagner Meira Jr.**[1]
Cristine Hoepers[2], **Klaus Steding-Jessen**[2]

[1] Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte, MG.

[2]CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
NIC.br - Núcleo de Informação e Coordenação do Ponto br, São Paulo, SP

{pcalais,dorgival,meira}@dcc.ufmg.br

{cristine,jessen}@cert.br

Abstract. *To subsidize research on ways to identify and possibly block spam in its origin, avoiding network resources being consumed, we characterize some strategies that define spammers' behavior patterns. For that we use data collected from low-interaction honeypots, configured to emulate open relays and open proxies. After collecting data, we identified message groups that differ only due to text obfuscation, which correspond to a same original spam campaign. We then applied data mining techniques on those groups to find out how such groups use the network resources. The results show that it is possible to identify* spammers *with specific patterns on the way they abuse different ports in parallel and how they start* spam *campaigns from different origins at the same time.*

Resumo. *A fim de subsidiar estudos para identificar e possivelmente bloquear o* spam *em sua origem, evitando que recursos da rede sejam consumidos, caracterizamos algumas estratégias que definem padrões de comportamento de* spammers*. Para tal, utilizamos dados coletados em* honeypots *de baixa interatividade, emulando* proxies *e* relay *abertos. Em seguida, detectamos os grupos de mensagens que diferem apenas por ofuscação de texto, que correspondem a uma mesma campanha de* spam*. Aplicamos então técnicas de mineração de dados para verificar as formas com que os grupos exploram os recursos da rede. Os resultados mostram que é possível identificar* spammers *com padrões específicos na forma como abusam diferentes portas em paralelo, e como iniciam campanhas de* spam *de diferentes origens ao mesmo tempo.*

1. Introdução

O desenvolvimento e popularização da Internet, além de diversos benefícios, acentuou também o crescimento de alguns problemas, como por exemplo o *spam* [Hayes 2003, Messaging Anti-Abuse Working Group (MAAWG) 2007]. O fato do custo de envio de *e-mails* ser muito baixo, comparado ao da correspondência convencional, serve como incentivo ao uso do correio eletrônico para o envio de *e-mails* comerciais não-solicitados em grandes quantidades [Cerf 2005]. Além disso, o *spam* tem sido um meio usual para enviar mensagens relacionadas com a obtenção de dados pessoais com objetivos ilícitos (*phishing*) e para disseminação de códigos maliciosos [Milletary 2005]. Devido à proporção

atingida pela questão do *spam*, várias abordagens técnicas têm sido utilizadas para lidar com o problema como, por exemplo, a adoção de recomendações para configuração de sistemas de *e-mail* [Lindberg 1999, Killalea 2000], uso de filtros de conteúdo de mensagens, como o SpamAssassin [SpamAssassin 2007] e listas de bloqueio [Cook et al. 2006]. Ao mesmo tempo, observa-se um aumento da sofisticação dos *softwares* de envio de *spam*, o que torna as técnicas existentes de bloqueio menos eficientes e a rastreabilidade do *spammer* mais difícil [Cranor and LaMacchia 1998, Milletary 2005]. Um exemplo disso é o crescimento na utilização de máquinas infectadas por códigos maliciosos, como os *bots*, para o envio de *spam* e *phishing*, permitindo que o *spammer* permaneça no anonimato [Milletary 2005]. Esses fatores têm motivado pesquisas no sentido de desenvolver mecanismos para combater o *spam* na sua origem, antes que ele chegue aos servidores de *e-mail* dos destinatários [Messaging Anti-Abuse Working Group (MAAWG) 2005, Gellens and Klensin 2006]. Esse tipo de solução teria também a vantagem de evitar o consumo de recursos da rede com a transmissão de mensagens que seriam eventualmente descartadas pelo destinatário.

Para atingir esse objetivo, no entanto, é necessário entender com detalhes como os *spammers* agem para distribuir suas mensagens na rede. Nesse sentido, o objetivo deste artigo é caracterizar diversas estratégias empregadas por emissores de *spam* para enviar suas mensagens. Definimos como uma estratégia qualquer recurso utilizado pelo *spammer* para maximizar o alcance de suas mensagens, reduzindo a probabilidade de que a mensagem seja retida em filtros anti-spam e que ela seja identificada e rastreada.

Em geral, os *spammers* disfarçam e variam o conteúdo que enviam de maneira sistemática, inserindo trechos aleatórios no corpo da mensagem e nos links nela contidos [Sophos.com 2004]. O objetivo nesse caso é evitar ao máximo o envio de mensagens idênticas, pois isso facilitaria a detecção de sua atuação. Sem a capacidade de identificar grupos de mensagens que tiveram origem em um texto base comum (que definem uma campanha de *spam*), não é possível isolar o tráfego gerado por diferentes *spammers*. Sendo assim, para analisar o comportamento de *spammers* na rede é necessário identificar os grupos de mensagens que correspondam a uma mesma campanha, neutralizando o impacto da ofuscação das mensagens e das estratégias de disseminação, para então estudar as características de exploração da rede de cada um desses grupos. Neste artigo propomos uma metodologia de identificação de grupos de mensagens associadas à mesma campanha de *spam*.

A partir da identificação das mensagens de uma campanha, também propomos uma metodologia de caracterização das estratégias de disseminação de *spam*. Essa metodologia é baseada na detecção de invariantes e padrões de co-ocorrência das estratégias de disseminação em cada grupo de mensagens associadas a uma única campanha. Esses invariantes e padrões representam comportamentos dos *spammers* e podem servir para a definição de critérios para a detecção e a identificação de campanhas de *spam*.

Demonstramos a efetividade da nossa estratégia aplicando-a a parte das mais de 500 milhões de mensagens de *spam* capturadas durante quinze meses por *honeypots* de baixa interatividade [Provos and Holz 2007], configurados de modo a simular computadores atuando como *proxies* e *relays* abertos [Steding-Jessen et al. 2008]. O processo de análise se inicia com a sumarização de características essenciais das mensagens coletadas. As mensagens sumarizadas são processadas para se obter agrupamentos contendo as

mensagens derivadas de uma mesma mensagem original por técnicas de ofuscação. As mensagens de cada agrupamento são então avaliadas em busca de correlações invariantes, na forma de características que co-ocorrem freqüentemente. Dados os grandes volumes de dados e a necessidade de automatização do processo de análise, técnicas de mineração de dados foram empregadas em cada etapa do processo.

Neste trabalho identificamos alguns padrões de comportamento de ofuscação de conteúdo e comportamento de rede que, em última análise, podem ser usados para subsidiar o estabelecimento de novas políticas que visem minimizar os efeitos negativos do *spam*. Consideramos ainda como contribuições a escolha de características a serem consideradas de cada mensagem e os processos de agrupamento e de extração de correlações entre essas características.

2. Trabalhos Relacionados

Diversos trabalhos recentes na literatura estudam as estratégias de abuso dos *spammers*. Um trabalho de 2006 [Ramachandran and Feamster 2006] estuda como os *spammers* exploram a infra-estrutura da Internet para enviar suas mensagens, incluindo as faixas de IP mais usadas para se enviar *spam* e tipos de abuso mais comuns (p.ex., *botnets*, BGP *hijacking*). Entre outras conclusões, os autores destacam o fato de que as mensagens de *spam* tendem a ser enviadas de faixas muito restritas de endereços IP. São mostradas também algumas estatísticas em relação à origem das mensagens, como os sistemas operacionais mais comuns e os sistemas autônomos (AS) que enviam mais mensagens.

O trabalho de Pu e Webb [Pu and Webb 2006] apresenta algumas análises acerca da evolução temporal dos *spammers* no que se refere ao uso de técnicas para construir suas mensagens. Essas técnicas foram computadas a partir das características identificadas nas mensagens pelo filtro *SpamAssassin* [SpamAssassin 2007]. Dessa forma, os autores mostraram que, ao longo do tempo, algumas técnicas de ofuscação deixam de ser usadas, muitas vezes em virtude de mudanças no ambiente, como a correção de alguma falha de segurança nos programas clientes de e-mail. Por outro lado, outras estratégias conseguem persistir por mais tempo.

O artigo de Li e Hsieh [Li and Hsieh 2006] é outro trabalho recente que analisa as estratégias de disseminação de *spam*. Os autores agruparam as mensagens pela URL contida no conteúdo e analisaram a estrutura do grafo que representa as relações entre IPs e URLs, ou seja, uma aresta entre um IP e uma URL significa que o IP enviou uma mensagem referenciando a URL. Eles analisam algumas propriedades desse grafo, com destaque para o surgimento de grandes grupos de IPs que enviam mensagens referenciando a mesma URL.

Por fim, o trabalho de Luiz Henrique Gomes [Gomes et al. 2007] apresentou uma extensa caracterização de cargas de trabalho de *spam*, apresentando comparações com cargas de trabalho de *e-mails* legítimos. Os autores derivaram modelos para representar a taxa de chegada de *spams* e o tamanho das mensagens.

Embora técnicas de mineração de dados estejam sendo extensivamente utilizadas para detecção de spam, não estamos a par de nenhum trabalho que explore a unificação das mensagens como estratégia para caracterizar *spammers*. Em geral os trabalhos avaliam as mensagens como um todo ou individualmente, não havendo uma abordagem sistemática de entendimento das estratégias de disseminação de *spam*.

3. Metodologia

Nesta seção apresentamos a metodologia proposta de caracterização das estratégias dos *spammers*. Como mencionado, essa metodologia se divide em três fases: a coleta dos dados, a sua unificação, com o objetivo de obter grupos coesos de mensagens com relação à sua finalidade, e a caracterização propriamente dita, quando os grupos de mensagens são analisados em busca de características invariantes nas estratégias dos *spammers*. Essas três fases são detalhadas nas seções a seguir.

3.1. Coleta de Dados

A captura das mensagens de *spam* analisadas foi realizada por 10 *honeypots* de baixa interatividade, instalados em redes brasileiras de banda larga de 5 operadoras diferentes (cabo e ADSL). Também fez parte da arquitetura um servidor central, configurado para realizar a coleta do *spam* capturado, bem como a monitoração periódica dos *honeypots* [Steding-Jessen et al. 2008].

Os *honeypots* foram configurados de modo a simular computadores com *proxies* e *mail relays* abertos, tradicionalmente abusados para o envio de *spam* e para a realização de outras atividades maliciosas [Krawetz 2004]. Um *spammer* que tentasse abusar de um desses *honeypots* para o envio de *spam* seria levado a acreditar que teve sucesso em enviar suas mensagens, embora nenhum *spam* fosse efetivamente entregue.

A captura de mensagens utilizou o *software* `Honeyd` [Provos and Holz 2007] em conjunto com subsistemas de emulação de SMTP e *proxies* HTTP e SOCKS. Todas as transações efetuadas pelos subsistemas do `Honeyd` foram armazenadas em *logs* com informações como data e hora, IP de origem, IP e porta pretendidos de destino, assim como versão do protocolo utilizado. Todos os *spams* capturados pelos *honeypots* foram coletados em intervalos regulares por um servidor central através de um túnel criptografado. Ao todo, foram coletados mais de 500 milhões de mensagens, durante um período de 15 meses [Steding-Jessen et al. 2008].

3.2. O Problema da Unificação de Mensagens

A unificação de mensagens tem por objetivo neutralizar o efeito de técnicas de ofuscação e de distribuição utilizadas pelos *spammers*, que alteram sistematicamente o conteúdo das mensagens, seja o seu corpo ou o próprio assunto do e-mail [Sophos.com 2004]. O objetivo é tornar cada mensagem enviada única, e para tal os *spammers* contam com ferramentas desenvolvidas para enviar *spam*, as quais oferecem os mais diversos recursos para que eles possam personalizar (e ofuscar) suas mensagens. Um exemplo típico de estratégia é a inserção de termos aleatórios no texto da mensagem, impedindo que seja gerada uma "assinatura" da mensagem de *spam* que permitiria identificá-la facilmente.

O problema da unificação de mensagens consiste em identificar grupos de mensagens que satisfaçam algum critério de equivalência semântico, isto é, agrupar as mensagens que, por exemplo, embora possuam URLs diferentes e conteúdos com algumas variações, tenham a mesma finalidade, isto é, anunciar o mesmo produto/serviço ou atingir um objetivo comum. Em outra palavras, o problema pode ser entendido como a identificação das campanhas de *spam*.

A Figura 1 ilustra graficamente o processo de unificação. No grafo à esquerda estão as mensagens coletadas, onde os vértices origem das arestas representam emissores

de *spam* e os vértices destino são as mensagens. Sem o processo de unificação, e dado o esforço de ofuscação dos *spammers*, cada mensagem se repete muito pouco (ou não se repete) e qualquer análise sobre o comportamento dos *spammers* é difícil ou inviável.

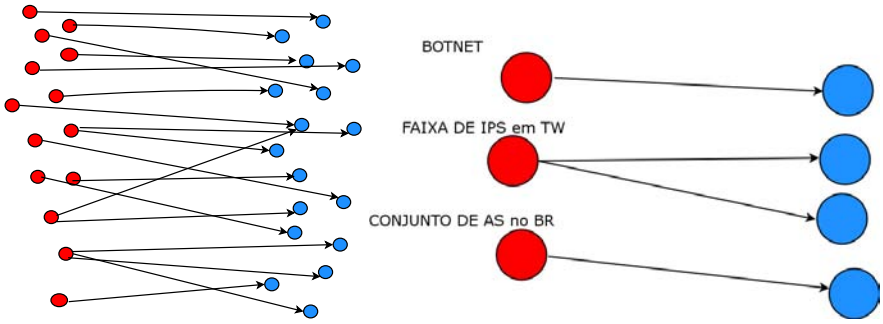


Figura 1. O processo de unificação de mensagens

O grafo à direita, por sua vez, exibe a visão dos dados após a unificação das mensagens que, embora possam ser diferentes, tratem do mesmo assunto e pretendam vender o mesmo produto ou serviço. Essa visão permite identificar grupos de máquinas que enviaram as mesmas mensagens e permite considerar esse grupo de máquinas como uma entidade única geradora de *spam*. Essa entidade pode ser um *botnet* [Cooke et al. 2005], que é um conjunto de máquinas de usuários comuns infectadas por um programa malicioso e programadas para enviar *spam* (máquinas "zumbis"), ou um conjunto de máquinas que situem-se na mesma faixa de IP, ou ainda um conjunto de sistemas autônomos (AS, ou *autonomous systems*).

Consideramos a unificação necessária para analisar a estratégia de disseminação de *spam* e entender como os *spammers* agem pelos seguintes motivos:

- A unificação neutraliza o efeito do volume variável de mensagens associado a cada campanha de *spam*. Para algumas análises, é mais interessante considerar apenas quais campanhas distintas que foram observadas no período, independente do número de mensagens em cada uma.
- A unificação cria novas dimensões que podem ser analisadas e correlacionadas, como o volume de mensagens enviadas durante uma campanha, ou a duração do período durante o qual tais mensagens foram enviadas.
- Como mencionado anteriormente, em pouco mais de um ano de coleta, o sistema armazenou centenas de milhões de mensagens. O processamento dessa quantidade de dados tem alto custo, mesmo com o emprego de técnicas de paralelização de algoritmos. A unificação provê uma sumarização que reduz o volume de dados a ser analisado.

3.3. Identificação de Padrões de Comportamento dos *Spammers*

Após identificarmos os grupos de mensagens componentes de cada campanha de spam, identificamos padrões de comportamento dos *spammers* a elas associados. Para isso procuramos por comportamentos semelhantes que podem identificar estratégias comuns entre os diversos grupos de mensagens. Invariantes identificados nesses grupos sintetizam as diferentes estratégias de *spammers*.

Nesse sentido, aplicamos duas técnicas de mineração de dados: agrupamento e identificação de padrões freqüentes [Tan et al. 2005]. Uma técnica de agrupamento objetiva segmentar um conjunto de entidades em grupos homogêneos que possuam alta semelhança entre seus membros e alta dessemelhança em relação a membros dos outros grupos. Neste trabalho, aplicamos o algoritmo (*k-means*) para identificar grupos de mensagens que exibam comportamento parecido no que se refere ao abuso dos recursos de rede. A mineração de padrões freqüentes procura encontrar atributos que co-ocorrem com freqüência em uma base de dados [Tan et al. 2005]. Aplicamos mineração de padrões freqüentes para encontrar combinações de características das mensagens que ocorrem comumente. Em particular, investigamos as correlações entre país de origem, país de destino e idioma das mensagens.

4. O Processo de Unificação de Mensagens

Nesta seção, apresentaremos a técnica proposta para agrupar as mensagens de *spam* e, em seguida, os principais resultados que foram obtidos.

4.1. Árvore de Padrões Freqüentes

A abordagem proposta para tratar o problema da unificação de mensagens explora o fato de que os *spammers*, em geral, mantêm uma parte da mensagem fixa e variam de forma sistemática e automatizada alguns fragmentos bem definidos. Por exemplo, cada mensagem de uma mesma campanha pode ter um campo *assunto* ligeiramente diferente, embora mantenha algumas palavras-chave padrão; no corpo da mensagem, a saudação pode alternar entre "Hello" e "Hi"; as URLs mencionadas na mensagem podem conter fragmentos aleatórios, que não fazem sentido algum e se prestam apenas a tornar a URL única, a fim de se evitar que ela seja identificada e bloqueada.

O processo de unificação desenvolvido e que explora essas possibilidades é constituído de duas partes. Na primeira, extraímos características relevantes de cada mensagem, tais como idioma, *layout*, tipo da mensagem (HTML, texto, figura), URL e assunto. O idioma de cada mensagem foi extraído através de uma implementação da técnica baseada em cálculo de N-Gramas [Cavnar and Trenkle 1994]. O *layout* corresponde às características de formatação de cada mensagem e baseia-se na proposta de Claudiu Musat [Musat 2006]. Por exemplo, uma mensagem de texto que possua duas linhas em branco, seguida de uma URL e duas linhas de texto será mapeada para o *layout* BBUTT. *Tags* HTML também são consideradas. Como será mostrado, o *layout* é um invariante importante para permitir o agrupamento das mensagens da mesma campanha de *spam*. Isto é, mesmo que os *spammers* insiram textos e caracteres aleatórios, a aparência geral da mensagem não é alterada. A URL de cada mensagem foi quebrada em *tokens* para que cada componente da URL seja uma característica da mensagem a ser considerada.

A partir das características de cada mensagem, monta-se uma árvore de padrões freqüentes (*FP-Tree*) [Tan et al. 2005]. Essa árvore é uma representação em que as características de cada mensagem são inseridas de forma que as mensagens que possuam características em comum compartilhem o mesmo caminho. A árvore é construída de forma que as características mais freqüentes (globalmente) fiquem nos níveis mais altos e as características infreqüentes ou aleatórias fiquem nos níveis mais baixos (próximos às folhas). Dessa forma, duas mensagens que possuam muitas características freqüentes em comum (como o idioma, o tipo e o *layout*) e sejam diferentes apenas por uma

característica infreqüente tendem a ser filhas de um mesmo nodo. Em geral, essa característica infreqüente é um fragmento da URL gerado aleatoriamente pelo *spammer*. A metodologia proposta prevê que as mensagens que sejam filhas de um mesmo nodo sejam agrupadas, pois elas compartilham invariantes (todas as características dos níveis superiores da árvore) e diferem apenas por um padrão infreqüente.

4.2. Resultados da Unificação

A Figura 2 exibe a distribuição do número de mensagens encontradas em cada grupo gerado pela heurística da árvore de padrões freqüentes, em comparação com os agrupamentos que seriam obtidos por um agrupamento baseado exclusivamente em equivalência binária dos conteúdos, que pode ser utilizada como padrão de comparação. A escala é *log-log*. Vemos que a árvore de padrões freqüentes consegue agrupar mais mensagens do que o simples agrupamento binário. Algumas campanhas possuem cerca de 1 milhão de mensagens, enquanto o grupo mais popular do agrupamento binário possui apenas cerca de 100 mil. Um caso clássico de unificação em que mensagens únicas são agrupadas são aquelas utilizadas para validar *e-mails* das vítimas. Nelas o *spammer* insere o *e-mail* do destinatário em uma URL no corpo da mensagem, de forma a poder registrá-lo como válido quando o usuário seleciona o link. Na árvore de padrões freqüentes, cada *e-mail* encontrado em uma URL seria um padrão infreqüente e portanto todas as mensagens estariam no mesmo nível da árvore, compartilhando todas as outras características.

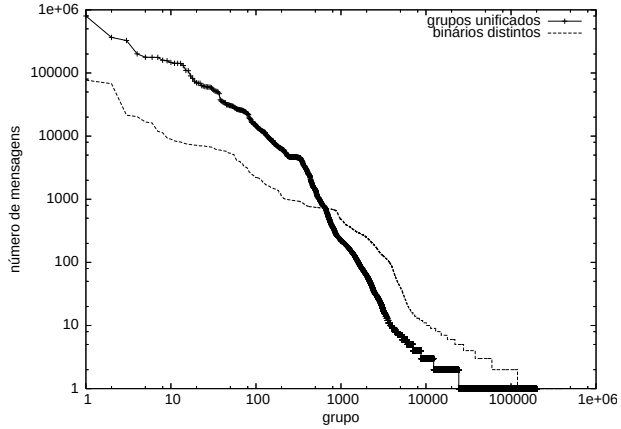


Figura 2. Número de mensagens por campanha de *spam*

A Figura 3 exibe o número de mensagens encontradas em cada nível da árvore de padrões freqüentes. Podemos notar que a árvore é muito desbalanceada e o número de características consideradas por mensagem varia significativamente. Percebe-se que os primeiros níveis contêm poucas mensagens, pois representam características que são muito freqüentes e não são úteis para discriminar as mensagens. Em particular, o primeiro nível contém o fragmento de URL "http://", já que 96,5% das mensagens possuem alguma URL. Características como o idioma da mensagem e o seu tipo também são colocadas na parte superior da árvore. As mensagens mais comuns são do tipo HTML, correspondendo a 86,61% do total. Mensagens de texto puro respondem por 13,35% e os 0,04% restantes são mensagens enviadas como imagens (GIF e JPEG). Após essa divisão de mensagens por idioma e tipo, a próxima característica mais útil para discriminar

mensagens é o *layout*. No período analisado, foram encontrados apenas 2.234 *layouts* de mensagem distintos. Descendo mais na árvore, encontramos os fragmentos de URL que os *spammers* fixam ao enviar suas mensagens. Em geral, os domínios das URL são fixos; entretanto, quando os *spammers* são donos de sub-domínios, eles conseguem inserir aleatoriedade no restante do endereço usado na URL. O pico no nível 6 acontece porque nesse nível concentraram-se as características aleatórias das mensagens, em especial, fragmentos aleatórios das URLs, e há um grande número de caminhos da árvore (e portanto mensagens) que terminam nesse nível. Um outro pico é notado no nível 9, isto é, existem muitas mensagens que compartilham 8 características (idioma, tipo, *layout* e fragmentos da URL) e diferem apenas por uma característica menos representativa. Cerca de 60% das mensagens foram agrupadas nos níveis 6 e 9 da árvore (Figura 3). Por outro lado, algumas mensagens chegam a compartilhar até cerca de 40 características, o que acontece quando a mensagem contém várias URLs e, portanto, o número de fragmentos compartilhados aumenta.

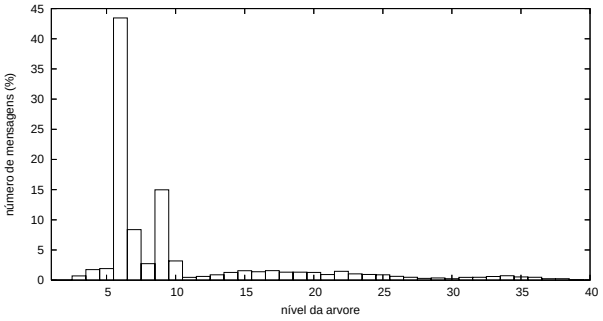


Figura 3. número de mensagens em cada nível da árvore de padrões freqüentes

O trabalho de Yeh e Lin [Yeh and Lin 2006] propõe uma metodologia para agrupamento de mensagens para fins de filtragem de *spam*. Os autores utilizam primordialmente a informação das URLs e comparam a semelhança entre as mesmas para decidir se duas mensagens são duplicatas, caso a similaridade seja superior a um certo limiar. Consideramos nossa abordagem mais robusta no sentido de exigir que as mensagens compartilhem outras características, de não necessitar de parâmetros e de não gerar grupos para mensagens que possuam URLs similares mas as mensagens sejam totalmente distintas, o que é comum em caso de redirecionadores, como *tinyurl.com*. Além disso, os padrões não são pré-definidos; eles surgem naturalmente na montagem da árvore. Em nenhum momento, definimos regras em que as mensagens que serão agrupadas precisem compartlhar certas características e diferir-se por outras. Isso torna a técnica mais resistente ao processo constante de mudança e evolução inerente ao *spamming*. Cabe ressaltar, ainda, que à medida que novas características sejam detectadas e se mostrem relevantes, o resultado da unificação pode ser aperfeiçoado.

5. Estratégias de Disseminação de Spam

Com base na unificação das mensagens discutida, apresentamos a seguir os resultados mais relevantes encontrados para os padrões de comportamento de *spammers*. O período analisado é de um mês, de 01/06/2007 e 01/07/2007. Nesse período, 9.301.182 mensagens foram coletadas e 150.912.121 destinatários distintos foram registrados. Apresen-

tamos três tipos de análise, relativas à forma como os *spammers* utilizaram as portas das máquinas com falhas de segurança, ao relacionamentos entre origem e destino das mensagens e o idioma de cada uma e à caracterização da infra-estrutura usada pelos *spammers* para enviar suas mensagens (a origem de cada campanha).

5.1. Portas Abusadas

Ao se aproveitar de uma máquina como *open proxy* ou *mail relay*, um *spammer* se conecta a uma porta na máquina-alvo referente a algum serviço que usualmente ofereça a opção de repasse de conexões. Dessa forma o *spammer* estabelece uma conexão para uma outra máquina a fim de injetar as mensagens que deseja enviar ocultando sua origem real. A Tabela 1 exibe a proporção de mensagens recebidas por cada porta abusada, obtida a partir da análise dos *logs* dos *honeypots*.

Embora a Tabela 1 mostre que as portas 1080 (SOCKS) e 8080 (HTTP) são as mais abusadas e que o uso de *mail relays* abertos para o envio de mensagens (porta 25) é pouco significativo (menos de 1% das mensagens), não há informações suficientes para se determinar as estratégias que podem estar relacionadas com o abuso daquelas portas. Isso se deve ao fato desse resultado ser apenas um agregado de todas as mensagens e representar um comportamento médio que não é necessariamente representativo.

Tabela 1. número de mensagens recebidas por cada porta abusada

Porta	Porcentagem	Porta	Porcentagem
1080	43,96%	4480	4,23%
8080	17,15%	81	4,10%
8000	10,09%	3382	1,93%
6588	6,63%	25	0,67%
3128	5,74%	3127	0,02%
80	5,48%		

Para melhor identificar as diferentes estratégias de abuso das portas, utilizamos o algoritmo de agrupamento *k-means* [Tan et al. 2005], bastante conhecido e por lidar bem com dados numéricos e poucas dimensões, para encontrar os grupos de campanhas de *spam* que exploram os mesmos conjuntos de portas. Após a execução do algoritmo, encontramos 8 grupos significativos que denotam comportamentos distintos dos *spammers* em relação ao abuso de portas.

Os resultados apresentados na Tabela 2 mostram que cerca de um terço das campanhas de *spam* identificadas (32%) foram enviadas para um mesmo conjunto específico de portas (grupo 1: 8080, 8000, 4480, 3127 e 1080). Por outro lado, muitas campanhas acabam abusando apenas uma porta, caso dos grupos 2 a 5. Uma parcela menor abusa outros conjuntos de portas, em que a porta 6588 é a mais freqüente (grupo 8).

A Figura 4 mostra como as campanhas do grupo 1 distribuíram suas tentativas de abuso das portas do conjunto durante o período amostrado. Pode-se ver que as portas são exploradas em paralelo durante todo o período. Há uma preponderância das portas 8080 e 8000 na maior parte do tempo, mas em em outros momentos todas as portas têm proporções semelhantes.

O conhecimento desses comportamentos distintos pode favorecer políticas de detecção de atividade de *spammers*, já que as proporções de abuso das portas podem ser

Tabela 2. grupos de portas, por frequência relativa de uso das principais portas

No.	**freqüência relativa - uso das portas**	***% de grupos***
1	8080 (30%), 8000 (18%), 4480 (14%), 6588 (14%), 1080 (13%), 3128 (11%)	32%
2	8000 (100%)	20%
3	8080 (100%)	15%
4	1080 (100%)	12%
5	4480 (100%)	7%
6	3128 (74%), 8080 (6%), 8000 (6%), 4480 (5%)	5%
7	80 (80%), 8080 (6%), 8000 (4%)	5%
8	6588 (95%), 8080 (3%), 3128 (2%)	4%

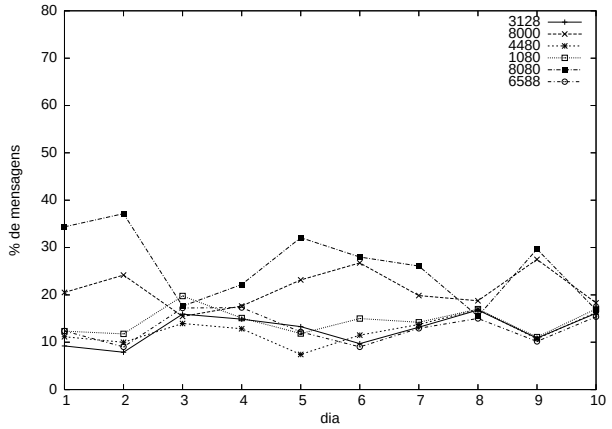


Figura 4. freqüência relativa de mensagens por porta para uma campanha

monitoradas e freqüências similares às descritas podem ser usadas como um forte indício da atividade de *spammers*.

5.2. Correlações entre origem, destino e idioma

A fim de entender melhor as estratégias utilizadas por *spammers* para disseminar suas mensagens, correlacionamos três características: o país de origem da mensagem, o país de destino e o idioma em que a mensagem está redigida. O país de origem foi obtido através da informação de alocação do IP de origem registrado para a mensagem, informação esta obtida através das tabelas de alocação dos cinco *Registries* Regionais [CYMRU 2007]. O país de destino da mensagem foi obtido através do domínio do *e-mail* dos destinatários (o domínio foi convertido para o IP correspondente e então o mesmo processo usado para obter o país de origem foi aplicado). Para essa análise, desconsideramos domínios genéricos como *hotmail.com* e *gmail.com*.

Como o número de mensagens é muito grande e as combinações entre origem, destino e idioma são potencialmente inúmeras, aplicamos mineração de padrões freqüentes [Tan et al. 2005] e analisamos os padrões e as regras de associação que podem ser derivadas deles. A Tabela 3 exibe as combinações entre país de destino, país de origem e idioma mais freqüentemente encontradas nas mensagens. Os resultados mostram que grande parte do *spam* que circulam pela Internet brasileira vêm da Ásia, em especial da China e de Taiwan. No período analisado, mais de 95% de todo o *spam* observado nos emuladores de *open proxy* e *open relay* dos *honeypots*, no Brasil, estavam redigidas em

chinês, o que pode indicar que o Brasil seja abusado por emissores de *spam* Asiáticos para ocultar a origem das mensagens.

Tabela 3. combinações de país de origem, país de destino e idioma

país de origem	país de destino	idioma	% de mensagens
Taiwan	Taiwan	Chinês	81,32%
China	Taiwan	Chinês	10,51%
Taiwan	EUA	Inglês	2,07%
EUA	Taiwan	Chinês	1,56%
Hong Kong	Taiwan	Chinês	1,41%

Aplicando regras de associação sobre esses atributos de cada mensagem, conseguimos revelar padrões e sutilezas que não são imediatamente visíveis. Algumas dessas regras estão listadas na Tabela 4, juntamente com algumas métricas de interesse. O suporte é a freqüência de ocorrência da regra no universo de mensagens.

Tabela 4. regras de associação

regra	antecedente	conseqüente	suporte	confiança	lift
1	Chinês	Taiwan (Origem)	27,8%	88,5%	1,02
2	Taiwan (Origem)	Chinês	26,5%	80,8%	1,02
3	China (Origem)	Chinês	9,1%	83,6%	1,05
4	EUA (Origem)	Chinês	0,7%	74,0%	0,93
5	Brasil (Origem)	Chinês	4,13%	25,1%	0,31
6	Brasil (Origem)	Inglês	2,83%	40,2%	1,12
7	Brasil (Origem)	Português	0,21%	14,8%	3987,94
8	Brasil (Origem)	Alemão	0,001%	0,001%	18,90

A regra 1 indica que 88%[1] das mensagens redigidas em chinês são enviadas a partir de IPs alocados para Taiwan. Olhando por outro ângulo, se a mensagem vem de Taiwan, a probabilidade dela estar redigida em chinês é de mais de 80% (regra 2). Se a origem é a China, a proporção é um pouco maior: 83% (regra 3). As regras de associação estão aqui sugerem a idéia de que o spam na Ásia, apesar de passar pela Internet brasileira, é produzido para "consumo interno".

A regra 4 revela que 74% das mensagens oriundas dos Estados Unidos analisadas durante o período de coleta estão escritas em chinês. Há diferentes explicações para esse fato: pode haver um encadeamento de *proxies*, de forma que a origem das mensagens pode ser simplesmente uma outra máquina abusada, ou pode ser um indício de *IP Hijacking* [Ramachandran and Feamster 2006], ou mesmo que empresas da Ásia usem serviços de *hosting* ou de *spammers* localizados em redes norte-americanas. Mais informações seriam necessárias para uma definição nesse caso.

A regra 5 permite levantar hipóteses semelhantes para o Brasil: o idioma de uma em cada quatro mensagens de *spam* cuja origem é o próprio Brasil é o chinês. O *lift*[2] é baixo (0,31), indicando que o caso mais comum é a origem ser mesmo Taiwan ou China e que o Brasil também encaminha outros tipos de mensagens (em especial, inglês e português). As regras 6 e 7 representam esses casos. Em particular, o *lift* para mensagens

[1] confiança: probabilidade do conseqüente dado o antecedente.

[2] lift: razão entre a probabilidade obtida e a esperada, se os eventos que compõem a regra fossem independentes.

enviadas em português a partir do Brasil é bastante alto, o que indica que o IP de origem brasileiro aumenta muito a chance da mensagem estar em português e ser destinada a brasileiros. Isso é um reflexo do fato de que no período analisado não foram encontradas mensagens de *spam* em português oriundas de IPs na Ásia ou EUA.

Por fim, a regra 8 indica que mensagens em alemão também circulam pela rede brasileira. Embora em quantidade pequena, se comparado, principalmente, com o volume de mensagens em chinês, essas mensagens superam sensivelmente o que seria esperado por uma simples amostragem estatística dos dados (valor alto de *lift*).

Essas regras indicam que técnicas que levam em conta aspectos de idioma podem ser úteis para sinalizar comportamentos inesperados em termos da linguagem detectada nas mensagens observadas na rede.

5.3. Infra-Estrutura

A partir dos grupos de campanhas de *spam*, verificamos de quantos endereços IP de origem e *autonomous systems* (AS) distintos as mensagens em cada campanha foram enviadas. O número de IPs e AS em cada caso são mostrados na Figura 5. Os resultados indicam padrões de comportamento distintos. Alguns *spammers* contam com mais de 200 máquinas para enviar suas mensagens e algumas campanhas se originam de mais de 50 sistemas autônomos, mas a cauda longa indica que uma grande parte das campanhas são enviadas de um grupo limitado de IPs. Esses resultados parecem indicar dois tipos de infra-estrutura: a centralizada, em que um pequeno conjunto de máquinas envia as mensagens (usualmente devido à contratação de serviços de *hosting* em um AS) e a descentralizada, em que máquinas residenciais podem estar sendo abusadas através de exploração de *proxies* abertos ou da implantação de *botnets*.

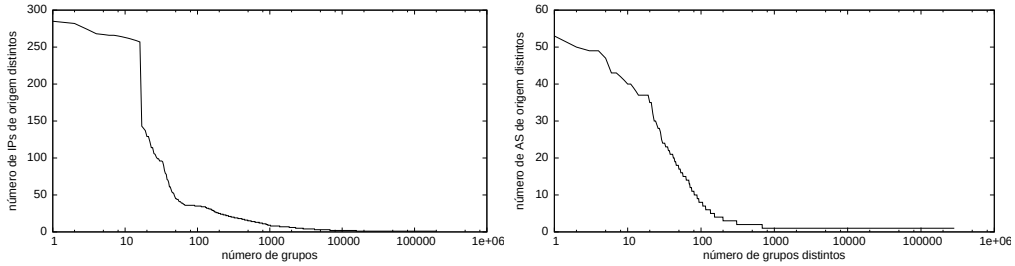


Figura 5. número de IPs e AS distintos para cada campanha de *spam*

6. Conclusão e Trabalhos Futuros

Embora filtros de *spam* consigam bloquear com relativa eficácia os *spams* que chegam à caixa-postal de usuários de correio eletrônico, eles não impedem que os preciosos recursos de banda e armazenamento da infra-estrutura da Internet sejam consumidos. A fim de subsidiar estudos que pretendam identificar e possivelmente bloquear o *spam* enquanto ele ainda trafega pela rede, caracterizamos algumas estratégias que definem padrões de comportamento de *spammers*. Para tal, utilizamos dados coletados em *honeypots* de baixa interatividade na Internet brasileira. Em seguida, detectamos os grupos de mensagens que diferem apenas por ofuscações de texto aleatórias e que correspondem ao mesmo *spam*. Para isso, características relevantes de cada mensagem foram extraídas e inseridas em uma

árvore de padrões freqüentes, que agrupa as mensagens que compartilham características freqüentes e se diferenciam apenas por uma característica infreqüente. A heurística permite detectar as diferentes campanhas de *spam*. Para essas campanhas, aplicamos técnicas de mineração de dados (agrupamento e regras de associação) para verificar as diferentes formas com que os grupos de *spam* exploram os recursos da rede.

Entre alguns dos padrões de comportamento encontrados, destacamos: (i) existem oito perfis distintos de abuso das portas das máquinas alvo, e, em cada um, diferentes portas são exploradas em diferentes freqüências; (ii) os relacionamentos entre país de origem, país de destino e idioma das mensagens indicaram a predominância de *spam* oriundo da China e Taiwan, escrito em chinês, mas ainda, as regras de associação geradas também revelam mensagens provenientes de IPs brasileiros escritas em idiomas como chinês e alemão; (iii) existem diferentes estratégias em relação à quantidade de máquinas utilizadas para disseminar as mensagens.

Como trabalhos futuros, pretendemos aperfeiçoar ainda mais o processo de unificação para agrupar mais mensagens e apresentar uma validação formal. Em seguida, pretendemos aplicar novas minerações de dados sobre os grupos correlacionando os padrões já encontrados, por exemplo, para verificar se o tipo de comportamento em relação às portas abusadas está relacionado com a origem do spam.

7. Agradecimentos

Este trabalho foi parcialmente financiado por CNPq, CAPES, FAPEMIG, FINEP e NIC.BR.

Referências

Cavnar, W. B. and Trenkle, J. M. (1994). N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

Cerf, V. G. (2005). Spam, spim, and spit. *Commun. ACM*, 48(4):39–43.

Cook, D., Hartnett, J., Manderson, K., and Scanlan, J. (2006). Catching spam before it arrives: domain specific dynamic blacklists. In *ACSW Frontiers '06: Proceedings of the 2006 Australasian workshops on Grid computing and e-research*, pages 193–202, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Cooke, E., Jahanian, F., and McPherson, D. (2005). The zombie roundup: understanding, detecting, and disrupting botnets. In *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pages 6–6, Berkeley, CA, USA. USENIX Association.

Cranor, L. F. and LaMacchia, B. A. (1998). Spam! *Commun. ACM*, 41(8):74–83.

CYMRU (2007). `http://www.cymru.com/BGP/asnlookup.html`.

Gellens, R. and Klensin, J. (2006). RFC 4409: Message Submission for Mail. `http://www.ietf.org/rfc/rfc4409.txt`.

Gomes, L. H., Cazita, C., Almeida, J. M., Almeida, V., and Wagner Meira, J. (2007). Workload models of spam and legitimate e-mails. *Perform. Eval.*, 64(7-8):690–714.

Hayes, B. (2003). Spam, spam, spam, lovely spam. *American Scientist*, 91(3):200–204.

Killalea, T. (2000). RFC 3013: Recommended Internet Service Provider Security Services and Procedures. http://www.ietf.org/rfc/rfc3013.txt.

Krawetz, N. (2004). Anti-honeypot technology. *IEEE Security & Privacy*, 2(1):76–79.

Li, F. and Hsieh, M.-H. (2006). An empirical study of clustering behavior of spammers and group-based anti-spam strategies. *Proceedings of the Third Conference on Email and Anti-Spam (CEAS). Mountain View, CA.*

Lindberg, G. (1999). RFC 2505: Anti-Spam Recommendations for SMTP MTAs. http://www.ietf.org/rfc/rfc2505.txt.

Messaging Anti-Abuse Working Group (MAAWG) (2005). Managing Port 25 for Residential or Dynamic IP Space. http://www.maawg.org/port25/MAAWG_Port25rec0511.pdf.

Messaging Anti-Abuse Working Group (MAAWG) (2007). Email Metrics Program: Report #5 – First Quarter 2007. http://www.maawg.org/about/MAAWG20071Q_Metrics_Report.pdf.

Milletary, J. (2005). Technical trends in phishing attacks. Technical report, CERT Coordination Center, Carnegie Mellon University. http://www.cert.org/archive/pdf/Phishing_trends.pdf.

Musat, C. N. (2006). Layout based spam filtering. *Transaction on Engineering, Computing and Technology*.

Provos, N. and Holz, T. (2007). *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 1st edition. ISBN-13: 978-0321336323.

Pu, C. and Webb, S. (2006). Observed trends in spam construction techniques: A case study of spam evolution. *Proceedings of the Third Conference on Email and Anti-Spam (CEAS). Mountain View, CA.*

Ramachandran, A. and Feamster, N. (2006). Understanding the network-level behavior of spammers. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 291–302, New York, NY, USA. ACM.

Sophos.com (2004). The Spam Economy: The Convergent Spam and Virus Threats. August 2004. http://www.sophos.com/whitepapers/Sophos_spam-economy_wpus.pdf.

SpamAssassin (2007). http://spamassassin.apache.org.

Steding-Jessen, K., Vijaykumar, N. L., and Montes, A. (2008). Using low-interaction honeypots to study the abuse of open proxies to send spam. *To appear in: INFOCOMP Journal of Computer Science*.

Tan, P., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co.

Yeh, C.-C. and Lin, C.-H. (2006). Near-duplicate mail detection based on url information for spam filtering. In *Information Networking. Advances in Data Communications and Wireless Networks*, pages 842–851. Springer Berlin / Heidelberg.

B.3 CEAS 2008

Em cumprimento a requisito exigido para a defesa da Tese, foi escrito e submetido um artigo para o seguinte congresso: "*Fifth Conference on Email and Anti-Spam* (CEAS 2008)", na cidade de *Mountain View*/CA, Estados Unidos, em agosto de 2008.

Este artigo, intitulado "*A Campaign-based Characterization of Spamming Strategies*" foi aceito no referido congresso e segue em anexo neste apêndice.

A Campaign-based Characterization of Spamming Strategies

Pedro H. Calais, Douglas E. V. Pires
Dorgival Olavo Guedes, Wagner Meira Jr.
Computer Science Department
Federal University of Minas Gerais
Belo Horizonte, MG - Brazil

Cristine Hoepers,
Klaus Steding-Jessen
Computer Emergency Response Team Brazil
Network Information Center Brazil
São Paulo, SP - Brazil

Abstract

This paper presents a methodology for the characterization of spamming strategies based on the identification of spam campaigns. To deeply understand how spammers abuse network resources and obfuscate their messages, an aggregated analysis of spam messages is not enough. Grouping spam messages into campaigns is important to unveil behaviors that cannot be noticed when looking at the whole set of spams collected. We propose a spam identification technique based on a frequent pattern tree, which naturally captures the invariants on message content and detect campaigns that differ only due to obfuscated fragments. After that, we characterize these campaigns both in terms of content obfuscation and exploitation of network resources. Our methodology includes the use of attribute association analysis: by applying an association rule mining algorithm, we were able to determine co-occurrence of campaign attributes that unveil different spamming strategies. In particular, we found strong relations between the origin of the spam and how it abused the network, and also between operating systems and types of abuse.

1 Introduction

Despite current strategies to minimize the impact of spams, it is necessary a continuous effort to understand in detail how spammers generate, distribute and disseminate their messages in the network, to maintain and even improve the effectiveness of anti-spam mechanisms (Pu & Webb, 2006). The goal of this paper is to characterize different spamming strategies employed by spam senders. We define as an strategy any technique employed by spammers to maximize the effectiveness of their attacks, reducing the probability that the message is blocked by spam filters and preventing their activities from being identified and tracked.

Our approach is to characterize spam campaigns in addition to individual messages. We define a campaign as a set of messages that have the same goal (e.g., advertising a specific product) and employ the same obfuscation strategy, which comprises either content obfuscation and network exploitation strategies. In general, spammers obfuscate and change the content of their messages on a systematic and automated way. They try to avoid sending identical messages, which would make the task of detecting their messages easier. Thus, in order to characterize the strategies and traffic generated by different spammers, it is necessary to identify groups of messages that are generated following the same procedure and are part of the same spam campaign. In this paper, we propose a novel and scalable methodology for identifying spam campaigns. After the campaigns have been identified and messages are associated with those campaigns, we can then characterize how each campaign have exploited the network resources and how their contents have been obfuscated.

We consider the identification of spam campaigns a crucial step for identifying spamming strategies and improving our understanding of how spammers abuse network resources for a number of reasons. First, the identification of campaigns creates new dimensions that can be analyzed and correlated. Aggregate analysis on spam data is limited in determining spamming strategies. By grouping spam messages in their associated campaigns, we can characterize how the spammer disseminated his or her messages. Second, the volume of spam messages is huge, and processing such amount of data is costly and sometimes unfeasible. Grouping messages into campaigns provides a summarization criteria, drastically reducing the amount of data to be treated, while maintaining their key characteris-

tics. Finally, the identification of spam campaigns neutralize the effect of the variable volume of messages associated to each spam campaign, which might hide frequent behaviors happening only on smaller campaigns.

After identifying the messages that were generated from the same spam campaign, we propose a methodology for characterizing spam dissemination strategies. The methodology is based on the detection of invariants and co-occurrence of mechanisms for sending messages adopted by a single campaign. These invariants and patterns represent spamming behaviors and may be used for definition of criteria for detection, identification and minimization of the impact of spam.

We applied our characterization methodology to 97 million spam messages captured during 12 months by low-interaction honeypots (Provos & Holz, 2007), which were configured to emulate computers with open relays and open proxies. We were able to find strong relations between the origin of the spam and how it abused the network and also between operating systems and these abuse types.

2 Related Work

Many recent works have studied spammers' abuse strategies, both considering network behavior and content obfuscation.

In (Ramachandran & Feamster, 2006) the authors analyze how spammers exploit the Internet infrastructure to send their messages, including the most popular IP ranges exploited for sending spam and the more common abuse types, such as bots and BGP hijacking. In particular, the authors show that spam messages tend to be sent from very restricted IP ranges. Some statistics about the origin of the messages show the most common operating systems originating spams and the autonomous systems (AS) that account for the highest volume of spams. Our paper also characterizes spamming network strategies, but instead of looking at the group of messages as a whole, we group messages into campaigns and then analyze how the groups of IPs that disseminated each campaign have abused network resources. This approach provides insights on how spammers act, which would not be possible on an aggregated analysis. Moreover, we also found relations among operating systems, spam origin and abuse types, which extends the analysis presented on (Ramachandran & Feamster, 2006).

A recent work on characterization of strategies of spam dissemination is presented in (Li & Hsieh, 2006). The authors grouped spams according to the messages' URLs and analyzed the graph representing the relationships between IPs and URLs. Some properties of that graph were analyzed, for example, the identification of large groups of IPs that send spam messages with the same URL. The notion of campaign was implicitly used, by grouping messages by their URLs, but, as URL obfuscation is a common practice, the groups of IPs referencing the same URLs could be even bigger. Our work considers not only URLs, but also other features while identifying campaigns.

SpamScatter (Anderson et al., 2007) is a technique that determines spam campaigns by performing *image shingling*, which looks for similarities between images from different spam web pages. The methodology adopted by the authors is similar to ours: campaigns are first identified and then characterized. However, while their work analyzes the scam hosting infrastructure, our focus is on the characterization of the network infrastructure abuse.

In fact, the idea of identifying spam campaigns is not new. In the literature, most research on grouping near-duplicate spam messages aims to detect campaigns as a strategy for blocking them, based on the fact that an inherent characteristic of unsolicited e-mails is that they are sent in high volumes during short periods of time. Different strategies for grouping messages into campaigns can be mentioned, such as techniques that consider URL information (Yeh & Lin, 2006), signature-based approaches such as I-Match (Kolcz & Chowdhury, 2007) and techniques that compute similarities between spam images (Wang et al., 2007). Our goal is different from those works in the sense that we intend to identify spam campaigns to characterize them in terms of content and network obfuscation. Although our findings may support the development and improvement of anti-spam techniques, this is not our main objective.

Regarding content characterization, (Pu & Webb, 2006) presented some analysis of temporal evolution of spammer strategies regarding the techniques they use to construct their messages. These techniques were extracted from the rules identified by the anti-spam filter *SpamAssassin*. The authors showed that some obfuscation techniques are abandoned over time, possibly due to changes in the environment, such as a bug fix on an e-mail client program. On the other hand, some strategies are able to persist for long periods of time. Our work is complementary to theirs and also provides an interesting framework for trend detection, which would be a future work direction.

Another characteristic of our work is the use of data mining techniques to unveil spamming strategies. Although data mining has been extensively applied on a wide range of contexts such as e-commerce, bioinformatics and industrial applications (Tan et al.,

2005), we are not aware of any work that applied data mining techniques for spam characterization purposes.

3 Methodology

In this section, we present our methodology for characterizing spammers' strategies for dissemination of spam messages. The methodology is divided into three distinct phases: data collection, campaign identification, and characterization. These three phases will be detailed in the next subsections.

3.1 Data Collection

The data collection architecture comprises a set of sensors based on low-interaction honeypots (Provos & Holz, 2007) to study the spam problem, in particular the abuse of open proxies and open relays. A proxy is a server that acts as an intermediary, making connections on behalf of other clients. An open proxy allows connections to be made from any origin to any destination IP address or port, and is traditionally abused for sending spam. Examples of common proxy protocols are HTTP and SOCKS. Misconfigured SMTP servers, usually called open relays, allow the delivery of messages from any source to any recipient and are also abused by spammers.

We deployed 10 honeypots in 5 Brazilian broadband networks (both cable and ADSL), that captured approximately 525 million spams over 15 months. These spams came from 216,888 different IP addresses, allocated to 165 different countries (Country Codes (CC), as defined in ISO 3166) and would have been delivered to 4.8 billion recipients (Steding-Jessen et al., 2008).

These honeypots were collecting data not at the final spam destination, like in spamtrap accounts or in mail servers. Instead, we measured the abuse of proxies and relays by spammers, and capture the spam at this stage, before reaching its final destination. Also, an advantage of using honeypots was that all messages collected were spams, with no false positives.

We used `Honeyd` (Provos & Holz, 2007) and its SMTP and HTTP server emulation subsystems to capture spam. A SOCKS proxy emulator was developed as part of this work to complement the existing emulators (Steding-Jessen et al., 2008). Although the honeypots were listening on several TCP ports, only the following were abused: 25, 80, 81 1080, 3127, 3128, 3382, 4480, 6588, 8000 and 8080.

All connections to the `Honeyd` modules were logged, including timestamp, client IP, destination IP and TCP port, as well as protocol requested. `Honeyd` also logs the Operating System of the source IP for each TCP connection, using passive fingerprinting techniques (Provos & Holz, 2007). All logs and data captured were then collected by a central server.

Any spammer trying to abuse one of these honeypots to send spam would tend to believe that the emails were delivered successfully. The message, however, was stored locally and never delivered to its recipients. The only exceptions were emails sent by spammers to test if the proxy/relay was delivering messages. Each test message was specially crafted, and contained information about the proxy/relay being tested. The message included IP address, port and protocol, and was intended to a recipient address under the spammer control. The honeypots were configured to deliver these messages only.

3.2 Campaign Identification

Campaign identification determines groups of messages that have the same goal and employ the same dissemination strategy. In practice, we want to minimize the effects of the obfuscation techniques employed by spammers, who systematically change the content of the messages they send, either the message body or subject (Sophos.com, 2004). The ultimate goal of obfuscation is to make each message unique, and, for that purpose, spammers use *bulk mailers* developed for sending spams. These tools offer many features for customization and obfuscation, such as the insertion of random pieces of text on the message body or in the URL, making the generation of spam signatures more difficult.

It is important to emphasize that the challenge associated with campaign identification comes not only from the variety of obfuscation strategies employed, but also their constant evolution, since mechanisms stop being used and novel mechanisms arise (Pu & Webb, 2006).

The basic premise of our strategy for campaign identification is that spammers, in general, keep some parts of the message static, while other parts are changed systematically and in an automated fashion. This premise is supported by the fact that the spam messages are generated by tools, which employ the same obfuscation mechanisms for a given campaign. Further, since the duration of the campaigns is relatively short, we assume that the obfuscation mechanisms employed for a given campaign do not change significantly across time. For example, each message from a given campaign may have slightly different terms in the *subject* field, although some keywords are always present, which is based on the intuitive relevance of keeping the message's subject readable, since too much obfuscation in this field may reduce the probability of the

message being read. Other examples are, in the message body, the insertion of greetings that may alternate between "Hello" and "Hi"; the inclusion of random fragments in URLs, which have no meaning and are inserted to make the URL unique, preventing its identification and block.

The problem of identifying spam campaigns may be seen as the determination of a hierarchical clustering of the messages, where messages that are similar w.r.t. a given criterion are clustered together. In order to implement such clustering, it is necessary to both determine the criteria and their hierarchical organization. Notice that our strategy also accounts for various strategies and their evolution.

3.3 Characterization of Spammer Behavior

In this phase we analyze both message and campaign information, searching for patterns and invariants that describe spammers' strategies.

Our strategy is based on the premise that we are evaluating machines that are abused for the transmission of spams, and we identify four groups of criteria for performing such characterization: source, destination, type of abuse, and content obfuscation strategy.

Source and destination consider not only the individual IPs/addresses from/to the spams, but also abstractions such as ASes, countries and ISPs. The type of abuse includes basically proxies (HTTP and SOCKS) and relays. The obfuscation strategy already includes several criteria, and we envisage that others may be incorporated as those strategies evolve.

In this context, the characterization may be seen as a two-step procedure. First, we generate profiles for each campaign, determining the characteristics that are shared by the spams in that campaign. We then group the campaigns, obtaining approximations of spammers' behavior. Invariants identified among these groups represent different spamming strategies.

There are several techniques that may be applied in this context, and any correlation analysis techniques are applicable, as we discuss in the next sections.

4 Frequent-pattern approach for campaign identification

In this section we describe the implementation of our strategy for identifying spam campaigns. As discussed previously, our strategy identifies the invariant parts of the spam messages and organize them hierarchically. It is divided into two major steps, which are described next.

In the first step, we extract relevant features from each spam message, such as its language, layout, message type (HTML, text, image), URL and subject. The language of each message is extracted using a technique based on the computation of n-grams (Cavnar & Trenkle, 1994). Message layout is a codification that maps the formatting properties of the message to a sequence of characters, based on the proposal of Claudiu Musat (Musat, 2006). For example, the layout of a plain text message containing two blank lines, followed by one URL and two lines of text would be mapped to the layout BBUTT. Message layout is an important invariant to be considered for the purpose of grouping messages from the same campaign, since the general appearance of the messages remain unchanged, although spammers usually insert random pieces of text on their messages. Besides language, type, layout, and subject, URL information is also crucial for clustering messages into campaigns (Yeh & Lin, 2006). We split each URL into tokens (splitting by "/", "." and "?") and they are considered as independent features.

Using the messages' features extracted in the first step, we build a frequent pattern tree, also called FP-Tree (Tan et al., 2005). In this tree, each node after the root represents a feature extracted from the spam messages which is shared by the sub-trees beneath. Each path in the tree represents sets of features that co-occur in messages, in non-increasing order of frequency of occurrences. Thus, two messages that have several frequent features in common (such as the language, type, and layout) and are different just on infrequent features will share a common path on the tree. The root is the only empty node, separating sub-trees which have nothing in common. From our observations, the most common infrequent feature is an URL fragment randomly generated. These random fragments cause the number of children to increase significantly, and are exactly the points in which the campaigns are delimited, that is, all the messages that are in the sub-trees beyond a significant increase in the number of children are grouped into the same campaign.

We should emphasize that our approach is scalable because messages are not compared pairwise, what would lead to a quadratic complexity. The cost of the algorithm is the cost of inserting messages' features in the FP-Tree, which is linear. Among the near-duplicate detection techniques, the FP-Tree would fit in the category of signature-based approaches, as the sequence of features in the tree defines the campaign unique identifier. Our approach, however, is not sensitive to random text, which is a common drawback of such techniques (Kolcz & Chowdhury, 2007). Another advantage of the FP-Tree is that it not only detects the

spams that are part of the same campaign, but also describes how the messages were constructed and obfuscated, as it will be detailed in Section 5.1.

5 Spam Dissemination Strategies

Table 1 shows details about the data used in our analysis. From the total messages collected, as described in Section 3.1, we decided to consider for analysis a period of 12 months. We also restricted the analysis to messages collected in two honeypots that were deployed on the most stable broadband networks. This reduced the number of messages analysed to approximately 97.5 million. Among those, we identified 6.9 million unique cryptographic hash signatures and 2.1 million unique URLs.

Table 1: Overview of the data analyzed in this paper

Characteristic	Values
Trace Period	2006-07-08 to 2007-06-23
Spam Messages	97,511,104
Unique hashes	6,910,340
Spams with URLs	88,735,105 (91 %)
Unique URLs	2,110,748
Spam campaigns	16,115

In this Section we present our characterization of spam strategies as observed in our honeypots. We divide this analysis in two major categories: first we identify campaigns and the content obfuscation techniques used in them; after that, using the added insight provided by clustering attacks in campaigns, we analyse their behavior in terms of network activity.

5.1 Spam campaigns

After applying our campaign identification technique, 16,115 spam campaigns were identified. Figure 1 shows the cumulative distribution function for the number of messages that are part of each campaign. We can see that while most campaigns are small, there is a significant number of campaigns which comprise more than 100,000 spam messages each.

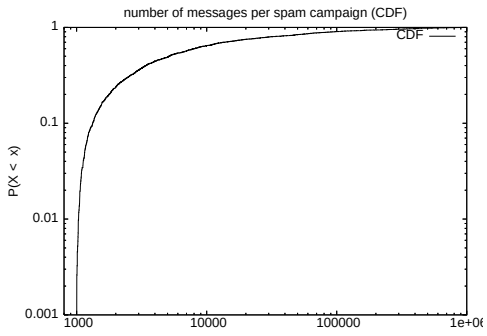


Figure 1: Number of messages in each campaign

Figure 2 shows a small portion of the resulting FP-Tree. When colors are available, each node's color represents a different feature that varied among the messages at that level. The diameter of the node is proportional to the log of the frequency of the characteristic in the campaign. Invariants in the campaigns are detected because they are more frequent than obfuscated features.

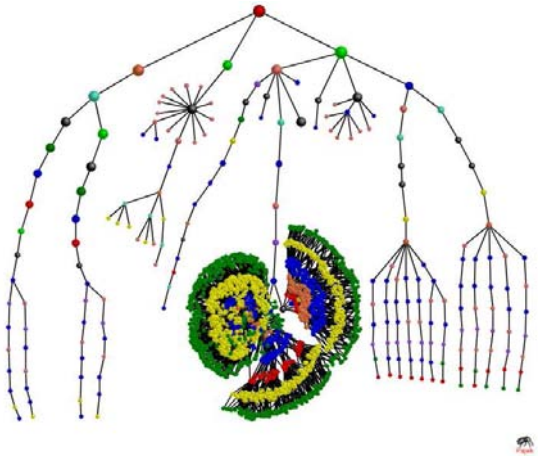


Figure 2: Frequent Pattern Tree showing distinct spam campaigns

For example, if a spammer sends URLs with the format `www.domain.com?parameter=random`, all random fragments would be inserted in the fifth position of the URL, making these messages very distinct from others that obfuscate in one of the remaining positions of the URL, which would characterize a different campaign.

An advantage of our approach is that it is not necessary to specify beforehand where obfuscation should be expected and which exact obfuscation pattern to look for. Spammers that own a web domain can insert obfuscation on virtually any position of the URL and even that would be automatically detected by our technique. In summary, one fundamental aspect of our campaign identification technique is that obfuscation patterns are not defined *a priori*; they are naturally detected. Further, given that the evolution of spamming techniques is an incontestable phenomenon, the FP-Tree is not tied to any currently known pattern. Our technique is extensible in the sense that, as soon as relevant characteristics of new campaigns are detected, they can be readily inserted in the FP-Tree. We also believe that our technique would be useful in other contexts, such as blog spam and spam in online social networks, in which the insertion of random text to avoid fingerprinting is also a common practice.

As an example of this analysis, in Figure 2, the three dense groupings seen at the bottom of the Figure, in

the middle, are three campaigns that shared some attributes (since they had a common path to the root until depth 7). They were then differentiated by the way they instantiated some other features, since the color of the nodes at following levels vary after that. All three campaigns varied some of their attributes randomly over a range of possibilities, yielding the larger number of children at the lower level.

Table 2: Number of Instances per Features

Feature	Number of instances
Message Type	16
Language	21
Message Layout	65,063
URL Fragments	1,967,160

In order to illustrate how the various campaigns exploit the obfuscation of the various features, we determined the number of different instances for each of the four attributes we used in our experiments (language, message type, message layout and URLs) and present a summary in Table 2. Message Type and Language are detected as important invariants, as only 21 types (including combinations, e.g.: spams which contain an HTML part with an image attachment). Message layout is also an important feature for grouping spams, given than only 65,063 distinct message formats where found for the set of 97.5 million messages.

Observing the FP-Tree, we find that language is often the first attribute selected as a classifier, as it would be expected (messages in a campaign are written in the same language). After that, message type and format come next, and URL fragments are usually the last discriminative feature added to the tree. That is expected, since messages in each campaign are likely to have similar URLs. The way those URLs are grouped and how often they are obfuscated in each campaign, however, varies widely among campaigns.

We have identified three types of campaigns in terms of obfuscation of URLs: static campaigns, campaigns with sub-campaigns, and random-obfuscated campaigns. In the FP-Tree sample shown in Figure 2, we can clearly identify those three types. Fixed campaigns are the ones in which the spammer inserts the same URL in all the messages of the campaign. Usually, these URLs correspond to small links with meaningful and readable names, such as *buydvds.com*. Those are the short branches which end at depths 3 or 4, usually. A different strategy frequently observed is the selling of different products in the same campaign, what generates a set of URLs in which each URL correspond to a different product from the same web site. For example, *dvd1.htm, dvd2.htm and dvd3.htm* are different products associated with the same campaign. As long as the spammer keeps other parts of the URL and its layout fixed, these distinct messages will be grouped into the same campaign because the portion of the URL that specifies the product is infrequent compared to the other messages' characteristics. Those are the branches at either side of the tree shown. Finally, the third class of campaign is the one in which spammers constantly obfuscate their URLs inserting random fragments which are different for each message. That refers to the three groupings at the bottom center discussed previously.

5.2 Network patterns

To understand how spammers use the network, we combine the analysis of grand totals from the collected data with information derived from the identification of campaigns and other data mining techniques. In the following discussion we highlight the major findings so far.

Spammers abuse proxies and relays differently

Table 3 shows a comparison for the three different types of abuse logged by the honeypots (HTTP and SOCKS proxies, and open mail relays). For each abuse we show the number of messages delivered and the number of unique sources of abuse with three different granularities: IP addresses, ASes and Country Codes (CC) of origin. (Percentages do not add up to 100 for IPs, CCs and ASes because of machines using more than one form of abuse.)

It is clear that messages abusing the honeypots as open relays are relatively rare, corresponding to only 2.6 % percent of the total. However, ratios are dramatically different when we look at the distribution of the origins of the abuse. Messages abusing HTTP and SOCKS come from relatively fewer ASes (and countries) than those abusing the open relay, which come from all over the world (142 countries). Despite being responsible for a small volume of messages, open relay abuses account for 98.6 % of the unique ASes abusing the honeypots during the period considered. Moreover, machines abusing the open relay send much fewer messages over time: while an AS abusing the open relay sent less than 2,000 messages over the period, ASes exploiting HTTP and SOCKS sent more than 1,000,000 and 380,000 messages on average, respectively.

This preference for abusing proxies may be explained by the need to better cover the origin of the spam. If spammers contacted mail relays directly, it would be simpler to track messages back to their origin, since even open relays would record the IP address of the previous connection, assuming it came from a valid SMTP server.

Table 3: Observed abuses

Metric	Abuse: HTTP	Abuse: SOCKS	Abuse: Open Relay
Messages	67,051,062 (68.8 %)	27,922,938 (28.6 %)	2,537,104 (2.6 %)
Unique IPs	41231 (49.3 %)	16,183 (19.36 %)	38,252 (45.8 %)
Unique ASes	59 (2.7 %)	72 (3,3 %)	2170 (98.6 %)
Unique CCs	14 (9.9 %)	14 (9.9 %)	142 (100 %)
Messages / IP	1626.2	1725.4	66.3
Messages / AS	1,136,459	387,818	1,169

Proxies and relays may be abused in a single campaign

There might seem, at a first glance, that such discrepancies were due to different spamming techniques (and, therefore, different campaigns) using proxies or open relays, but that turned out not to be the case. Indeed, 90 % of the campaigns identified abused only HTTP/SOCKS in the honeypots; however, the other 10 % abused both open relays and proxies. There were no campaigns that abused only open relays.

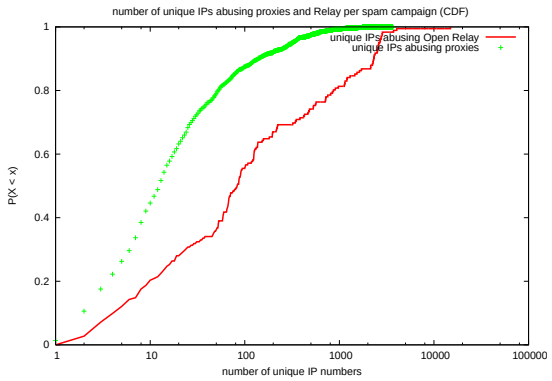


Figure 3: Unique IPs abusing proxies and relay on each campaign

Even on campaigns that included abuse to both proxies and relays, the pattern differed between them. Figure 3 shows, for those campaigns, the cumulative distribution function of the number of IP addresses seen in any campaign abusing proxies or the relay. When abusing proxies, campaigns come from a more concentrated set of addresses. Approximately 50 % of the campaigns come from only 10 sources when abusing HTTP/SOCKS proxies, while 80 % come from more than 10 IPs (and 40 % come from more than 100 addresses) when abuses are directed to the open relay.

Spammers chain proxies to SMTP servers

If we look at the next hop intended for the proxy connections, there were 215,719 different IPs targeted, in 189 unique CCs. An interesting fact is that 94 % of all connections were to port 25 (SMTP) of those machines, which could have been legitimate mail servers or open relays. That indicates that the most common behavior of spammers is to use one hop over a SOCKS/HTTP proxy and then use SMTP. If that was not the case, we should see a larger number of connections through the honeypot proxies aimed at other port numbers in the next hop machines.

By inspecting the addresses most popular among those selected by spammers as the next hop for the proxy connections we see most connections are for well established mail servers. For example, machines `mta-v2.mail.vip.tpe.yahoo.com` and `mta-v1.mail.vip.tpe.yahoo.com` are among the most targeted from proxies in both groups of campaigns (the ones that abuse only proxies and those that abuse proxies and relays).

Spammers chain proxies to open mail relays

On the other hand, campaigns that were seen abusing the honeypots' mail relays targeted not only well known mail servers, but also mail relays the spammer could find as the next hop for the proxy connections.

Proof of that can be found in the honeypot themselves. Among the campaigns that abused proxies and relays, we found in the honeypots records of approximately 2800 requests for a honeypot to connect the port 25 of another honeypot used during the collection. That is in fact a lower bound, since we considered only the addresses of honeypots with fixed IP addresses; there might be other requests targeted at honeypots in networks using dynamic addresses.

In our continuing work we intend to classify addresses more precisely between relays and servers, and also understand how spammers find those open relays: whether from pre-assembled lists, or through active port scanning.

Correlations among message features

In search of evidences that would explain such differences regarding proxy and relay abuses, we correlated four different characteristics present on each campaign: the type of abuse, the message's CC of origin, the intended CC of destination of the spam message and the language used. The Language was derived using the n-grams technique mentioned in Section 4; the CC of origin was obtained from IP allocation tables from the 5 Regional Internet Registries; and the destination country was derived from the domain extracted

Table 4: Association rules - Origin, Destination, Language and Abuse

rule	antecedent (if)	consequent (then)	support	confidence	lift
1	in Chinese, on HTTP	From TW	23.8 %	86.0 %	1.1
2	From BR	in Chinese, on Open Relay, to TW	0.02 %	46.7 %	3.8
3	From AR	in Chinese, on Open Relay, to TW	0.01 %	76.7 %	4.5
4	From GB	in Chinese, on Open Relay, to TW	0.02 %	81.9 %	3.1
5	From PT	in Chinese, on Open Relay, to TW	0.01 %	43 %	2.3
6	From AR, on SOCKS	in Spanish	0.01 %	95 %	4.3
7	From CN, on HTTP	in Chinese, to TW	7.5 %	84 %	1.3
8	From CN, on Open Relay	in Chinese, to TW	6.3 %	78 %	1.1
9	From US	in Chinese, on Open Relay, to TW	0.8 %	59 %	1.0
10	From US, on HTTP/SOCKS	in English	3.1 %	56 %	1.4
11	From US, on HTTP/SOCKS	in Chinese	1.1 %	31 %	0.9

from the victims' e-mail addresses. For this analysis, we did not include address in the `.com` domain (e.g., `yahoo.com` or `gmail.com`), since the user behind such accounts could be virtually on any part of the globe.

Using that information, we applied an association rule mining algorithm (Tan et al., 2005) to each campaign's data. Some of the most interesting association rules found on our analysis are shown in Table 4.

In that table, rule 1 shows the most frequent abuse observed on our dataset: 23.8 % of the abuses are related to messages written in Chinese abusing HTTP. Moreover, 86 % of the messages with those characteristics are sent from an IP address in the TW Country Code. Rules 2 to 5 indicate that spams written in Chinese are also observed being sent from BR, AR, GB and PT abusing open relays, with high confidence. In the case of AR, on the other hand, rule 6 shows that 95 % of the spams sent from that country abusing SOCKS were written in Spanish. Rules 7 and 8 indicate that CN sends spams in Chinese both through proxies and open relays to TW. Rules 9 to 11 show the most common abuses related to messages being sent from US. While US IP addresses are detected sending spams in Chinese through open relays (rule 9) and in English through Proxies (rule 10), US is also seen sending spams in Chinese through Proxies and SOCKS (rule 11), which is different from what is observed for the other country codes.

From these results, we conclude that there is a strong relation between abuse type, origin and destination of spams. While TW primarily sends spams through HTTP and SOCKS aiming TW recipients (writing in Chinese), most of the other country codes (BR, AR, PT and other 139 CCs) abuse open relays sending messages in Chinese. The only exception is US: it also sends Chinese spams through Proxies and SOCKS, what might indicate a different spamming strategy.

Observed behavior

The association rules, analyzed in conjunction with Table 3, suggest that HTTP and SOCKS are exploited directly by spammers, i.e., at the origin of the attack. That is supported by the concentrated IP addresses that originate such abuses, the coincidence between the language used in the message and the language associated to the CC. On the other hand, open relay abuses come from machines all over the world. In Table 4, we have presented only some few rules showing countries sending spams in Chinese through open relays, but, actually, this is seen for all the other countries in our dataset. Those may be HTTP/SOCKS proxies also abused, or may have a more organized structure, kept indirectly under control of spammers, known as *spam bots*, like machines in a botnet (Cooke et al., 2005). Prior work have reported that botnets usually send very low volumes of spams over longs periods (Ramachandran & Feamster, 2006), which is a strong evidence that, in fact, these open relay abuses spread worldwide are compromised infected machines.

The results about abuses from AR illustrate that clearly. 76.7 % of all spam coming from AR carry messages in Chinese, addressed to machines in TW and abuse open relays. On the other hand, 95 % of the spam coming from AR hosts abusing SOCKS are in Spanish.

The use of probe messages

Section 3.1 mentioned that we observed some messages that could be identified as probe messages sent by spammers to assert the correct behavior of the machines abused by them. They often have a nearly empty body with some information about the machine probed and should be recognized and dealt with properly, otherwise spammers would leave and would not try to abuse the honeypot infrastructure (Andreolini et al., 2005). We identified two major types of probe messages based on their format and programmed our honeypots to respond only to those messages. Table 5 summarizes the results in terms of the country codes and IP addresses of origin of those messages.

It seems the two kinds of messages are associated with two different spamming tools. Messages of type 1 came mostly from TW, from a relatively large number of IP addresses. Type 2 probes came from both US and TW machines, with somewhat different behaviors: from US

Table 5: Probe Messages

CC	Messages	Unique IPs
Probe Message Type 1		
TW	251,228 (99.8 %)	979 (97.8 %)
US	23 (0.15 %)	16 (1.6 %)
KR	7 (0.05 %)	6 (0.06 %)
Probe Message Type 2		
US	1836 (51.0 %)	19 (2.5 %)
TW	1473 (40.1 %)	672 (89.4 %)
CN	47 (1.3 %)	31 (4.2 %)
CA	34 (1.2 %)	13 (1.8 %)
KR	34 (1.2 %)	5 (0.9 %)
other	12 (0.07 %)	6 (0.9 %)
unknown	163 (4.5 %)	2 (0.3 %)

only 19 hosts (2.5 % of the type 2 probing machines) were responsible for more than 50 % of the probes, while a similar number of probes originated from 672 hosts (89.4 % of the machines). Probe messages seem to be related to the real origin of the message, as they come from the same CCs identified as those responsible for the abuses on proxies. That suggests a dedicated set of high-throughput machines in the first case, and a more widely distributed infra-structure in the second case.

Behavior varies for different operating systems

Finally, Table 6 unveils some strong patterns correlating the operating systems of the machines which abused the honeypots during the period of this analysis and the type of abuse associated to each one. Association rules 1 to 3 show that machines configured with Linux, FreeBSD and Solaris abuse the honeypots mainly as open relays, in the vast majority of the abuses observed. The high value for lift[1] in all cases (higher than 8) indicates that chances of observing abuses on open relays is much higher when the messages come from Linux, FreeBSD and Solaris computers, although these operating systems account for less than 3 % of the total flows observed. Correlating these results with our prior observations, we believe that these rules are due to the fact that *bulk mailers*, in general, are developed for Windows platforms, and not Linux or Solaris. On the other hand, it is relatively common to find in these operating systems some sort of HTTP server, possibly poorly configured.

On the other hand, rules 4 to 6 show that Windows is usually used to exploit SOCKS (with 31 % of confidence) and proxies (with 62 % of confidence). In our dataset, over half of the operating systems of the machines which abused our honeypots could not be identified (rules 7 and 8). As the proportions of HTTP/SOCKS abuses are similar to those observed for Windows, and very different from machines configured with Linux-based systems, we believe that those connections may be in fact associated with Windows Vista, which was a new OS on 2006 and might not have a proper signature yet.

[1]lift: ratio between the calculated probability and the expected probability, if the events of the association rule were independent.

6 Conclusions and Future Work

In this paper we have presented a methodology for characterizing spamming strategies based on the grouping of spams into spam campaigns and then detecting invariant an co-occurrence patterns among them.

Our technique builds a frequent pattern tree (FP-Tree) using relevant features extracted from spam messages (*e.g.*, layout, language, URL fragments). Based on that, messages that share a common frequent path in the tree and differ only on infrequent features are grouped into campaigns. We have tested our technique on a dataset of approximately 97.5 million spam messages collected on low-interaction honeypots deployed on Brazilian networks acting as HTTP and SOCKS proxies and open mail relays.

After identifying the distinct spam campaigns, we showed that data mining techniques (such as clustering and association rule mining) can be useful to unveil relevant spamming behavior patterns. We found that HTTP and SOCKS abuses originate from few nodes and show strongly correlations between language and CC of origin, suggesting that they are close to the origin of the campaign. On the other hand, Open Relay abuses are more disperse, originating from many different sources and exhibiting no correlation between language and CC of origin, probably being triggered by *spam bots*. We also determined some relations between operating systems and abuse types, indicating that Linux and Solaris systems are rarely used as the origin of abuses to HTTP and SOCKS proxies.

As future work, we will compare our campaign identification technique with other approaches available on the literature and assess if the frequent pattern tree can be applied for spam filtering purposes. We will also study more deeply how spam campaigns abuse the network infrastructure, specially in the case when connections to proxies are relayed to an intermediary open relay machine before being delivered to a official mail server. A temporal analysis of campaigns evolution is also being considered.

We intend to deploy honeypots in other countries. With that we can have a more global view of how spammers abuse network resources around the Internet.

Table 6: Association rules – Operating Systems and Abuse Types

rule	antecedent (if)	consequent (then)	support	confidence	lift
1	OS: Linux	Abuse: Open Relay	1.3 %	97.0 %	8.0
2	OS: FreeBSD	Abuse: Open Relay	0.7 %	100 %	8.2
3	OS: Solaris	Abuse: Open Relay	0.6 %	100 %	8.2
4	OS: Windows	Abuse: Open Relay	4.1 %	7 %	0.6
5	OS: Windows	Abuse: HTTP	7.1 %	62 %	0.9
6	OS: Windows	Abuse: SOCKS	15.3 %	31 %	1.2
7	OS: Unknown	Abuse: HTTP	49.8 %	72 %	1.0
8	OS: Unknown	Abuse: SOCKS	16.1 %	26 %	1.0

Acknowledgements

This work was partially supported by NIC.br, CNPq, CAPES, Finep, Fapemig and by the 5S-VQ project (MCT/CNPq/CTINFO grant number 551013/2005-2).

References

Anderson, D. S., Fleizach, C., Savage, S., & Voelker, G. M. (2007). Spamscatter: Characterizing internet scam hosting infrastructure. *USENIX Security*.

Andreolini, M., Bulgarelli, A., Colajanni, M., & Mazzoni, F. (2005). Honeyspam: honeypots fighting spam at the source. *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop* (pp. 11–11). Berkeley, CA, USA: USENIX Association.

Cavnar, W., & Trenkle, J. (1994). N-gram-based text categorization. *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval* (pp. 161–175). Las Vegas, US.

Cooke, E., Jahanian, F., & McPherson, D. (2005). The zombie roundup: understanding, detecting, and disrupting botnets. *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop* (pp. 6–6). Berkeley, CA, USA: USENIX Association.

Kolcz, A., & Chowdhury, A. (2007). Hardening fingerprinting by context. *Proceedings of the 4th Conference on Email and Anti-Spam (CEAS). Mountain View, CA.*. Mountain View, CA.

Li, F., & Hsieh, M.-H. (2006). An empirical study of clustering behavior of spammers and group-based anti-spam strategies. *Proceedings of the Third Conference on Email and Anti-Spam (CEAS). Mountain View, CA.*

Musat, C. (2006). Layout based spam filtering. *Transactions on Engineering, Computing and Technology.*

Provos, N., & Holz, T. (2007). *Virtual honeypots: From botnet tracking to intrusion detection.* Addison-Wesley Professional. 1st edition, ISBN-13: 978-0321336323.

Pu, C., & Webb, S. (2006). Observed trends in spam construction techniques: A case study of spam evolution. *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS). Mountain View, CA.*

Ramachandran, A., & Feamster, N. (2006). Understanding the network-level behavior of spammers. *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications* (pp. 291–302). New York, NY, USA: ACM.

Sophos.com (2004). The Spam Economy: The Convergent Spam and Virus Threats. August 2004. `http://www.sophos.com/whitepapers/Sophos_spam-economy_wpus.pdf`.

Steding-Jessen, K., Vijaykumar, N. L., & Montes, A. (2008). Using low-interaction honeypots to study the abuse of open proxies to send spam. *To appear in: INFOCOMP Journal of Computer Science.*

Tan, P., Steinbach, M., & Kumar, V. (2005). *Introduction to data mining, (first edition).* Addison-Wesley Longman Publishing Co.

Wang, Z., Josephson, W., Lv, Q., Charikar, M., & Li, K. (2007). Filtering image spam with near-duplicate detection. *Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS). Mountain View, CA.*. Mountain View, CA.

Yeh, C.-C., & Lin, C.-H. (2006). Near-duplicate mail detection based on url information for spam filtering. *Information Networking. Advances in Data Communications and Wireless Networks* (pp. 842–851). Springer Berlin / Heidelberg.

C APÊNDICE C - MINICURSOS E PALESTRAS CONVIDADAS

Este apêndice lista os minicursos e palestras convidadas, nas áreas de *honeypots*, *honeynets* e estudo do problema do *spam*, que foram apresentadas pelo autor decorrentes do trabalho nesta tese.

C.1 Minicursos

- "*Honeypots and Honeynets*", Universidade de Groningen, Groningen, Holanda, 6–10 de outubro de 2003;
- "*Honeypots* e *Honeynets*", V Simpósio Segurança em Informática (SSI'2003), São José dos Campos, SP, 6 de novembro de 2003;
- "*Honeypots Workshop*", *Q-CERT Honeypots Workshop*, Doha, Qatar, 18–19 de abril de 2007;

C.2 Palestras Convidadas

- "Uso de *Honeypots* de Baixa Interatividade na Resposta a Incidentes de Segurança", 3ª Reunião do Grupo de Trabalho em Segurança (GTS-3), São Paulo, SP, 20 de abril de 2004;
- "*Incident Response Initiatives in Brazil*", ITU WSIS *Thematic Meeting on Cybersecurity*, Genebra, Suíça, 8 de junho de 2005;
- "Tecnologias e Políticas para Combate ao *Spam*", 5ª Reunião do Grupo de Trabalho em Segurança (GTS-5) , São Paulo, SP, 5 de julho de 2005;
- "*Distributed Honeypot Deployment in Brazil*", *Annual DoD Honeynet Workshop*, Pasco, WA, Estados Unidos, 8 de março de 2006;
- "Boas Práticas para Mitigação de Spams e de Fraudes via *E-mail*", *Security Week* 2006, Painel SecGov, São Paulo, SP, 27 de março de 2006;
- "*Distributed Honeypots Project: How It's Being Useful for CERT.br*", *2006 Collaboration Meeting for CSIRTs with National Responsibility*, Pittsburgh, PA, Estados Unidos, 2 de julho de 2006;
- "*Spam* e Fraudes por *E-mail*: Iniciativas de Combate no Brasil e no Mundo", Minitrilha CERT.br, 8° Simpósio Segurança em Informática (SSI'2006), São José dos Campos, SP, 9 de novembro de 2006;

- "*The Brazilian Honeypots Alliance*", *FIRST Technical Colloquium*, Doha, Qatar, 15 de abril de 2007;

- "*SpamPots Project: Using Honeypots to Measure the Abuse of End-User Machines to Send Spam*", 2° Evento de Segurança de Redes para América Latina e Caribe (em conjunto com LACNIC X), Isla Margarita, Venezuela, 23 de maio de 2007;

- "*SpamPots Project: Using Honeypots to Measure the Abuse of End-User Machines to Send Spam*", *2007 Collaboration Meeting for CSIRTs with National Responsibility*, Madrid, Espanha, 24 de junho de 2007;

- "*Data Donation Initiatives at the Brazilian Honeypots Alliance*", *4th FIRST Network Monitoring Special Interest Group (NM-SIG) Meeting*, Noordwijk, Holanda, 17 de outubro de 2007;

- "*Monitoring the Abuse of Open Proxies for Sending Spam*", *6th International GOVCERT.NL Symposium*, Noordwijk, Holanda, 19 de outubro de 2007;

- "Implantação de Honeypots de Baixa Interatividade com Honeyd e Nepenthes", Campus Party Brasil, São Paulo, SP, 13 de fevereiro de 2008;

- "Gerência de Porta 25/TCP em Conexões Residenciais: Controle de *Direct Delivery* e Adoção de *Mail Submission Port*", 13ª Reunião Ordinária da CBC-1 da Anatel, CPqD, Campinas, SP, 22 de fevereiro de 2008;

D APÊNDICE D - CÓDIGO FONTE

Este apêndice contém os códigos fonte de vários módulos desenvolvidos pelo autor ao longo deste trabalho.

D.1 socks.pl

```
1  #! /usr/bin/perl -T
2  #-*-Perl-*-
3  #
4  # socks.pl -- SOCKS version 4 and 5 module for Honeyd
5  #
6  # Copyright (c) 2006 Klaus Steding-Jessen
7  #
8  # Permission to use, copy, modify, and distribute this software for any
9  # purpose with or without fee is hereby granted, provided that the above
10 # copyright notice and this permission notice appear in all copies.
11 #
12 # THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
13 # WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
14 # MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
15 # ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
16 # WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
17 # ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
18 # OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
19
20 (my $program_name = $0) =~ s@.*/@@;
21
22 use strict;
23 use warnings;
24 use Getopt::Std;
25 use Fcntl qw(:flock);
26 use POSIX qw(strftime);
27
28 ###
29 ### constants
30 ###
31
32 my $BUFFER_SIZE = 1492;
33
34 my %option = ();
35 getopts('Vhf:', \%option);
36
37 my $version='0.1';
38
```

```
39  # unbuffered output.
40  $| = 1;
41
42  # set PATH.
43  $ENV{'PATH'} = '/bin:/usr/bin:/usr/sbin';
44
45  # used by the SIGALRM handler.
46  my $timeout = 0;
47
48  ###
49  ### configuration defaults -- can be changed via configuration file.
50  ### parameter, regex value and default value.
51  ###
52  my %conf = (
53              "logfile" => {
54                  regex => '[\w\.-\/]+',
55                  value => '/var/tmp/logfile' },
56              "verbose" => {
57                  regex => 'yes|no',
58                  value => 'yes' },
59              "timeout" => {
60                  regex => '\d+',
61                  value => 30 },
62              "action_list" => {
63                  regex => 'port:\s*(?:default|\d+),\s*action:\s*'.
64                      '(?:deny|allow),\s*prog:\s*(?:NONE|[\w\-\/\.]+)'.
65                      ',\s*prog_args:(?:NONE|[\w\/\.\s\-\%]+)',
66                      value => '' },
67              );
68
69  # display usage if requested.
70  show_usage() if ($option{'h'});
71
72  # display version if requested.
73  show_version() if ($option{'V'});
74
75  # parse config options from the configuration file, if requested.
76  if (defined($option{'f'})) {
77      my ($confref) = process_config_file($option{'f'}, \%conf);
78      %conf = %{$confref};
79  }
80
81  # logfile.
82  my $logfile = $conf{'logfile'}{'value'};
83  my $date = strftime("%F", localtime);
84  $logfile = sprintf("%s.%s", $logfile, $date);
85
```

```
 86  # get values from the environment variables set by honeyd.
 87  my ($srchost, $dsthost, $srcport, $dstport) = get_honeyd_env();
 88
 89  ###
 90  ### main
 91  ###
 92
 93  # install SIGALRM handler.
 94  $SIG{ALRM} = sub {
 95      $timeout = 1;
 96  };
 97
 98  alarm $conf{'timeout'}{'value'};
 99  logentry($logfile, sprintf("%s:%d: connection to %s:%d",
100                             $srchost, $srcport, $dsthost, $dstport));
101  my $nread = 0;
102  my $total_read = 0;
103  my $state = "initial";
104  my $socks_dsthost = "";
105  my $socks_dstport = "";
106
107  # read stdin from honeyd -- loop reading data from the client.
108  while (($nread = sysread(STDIN, my $buffer, $BUFFER_SIZE)) &&
109         (!$timeout)) {
110
111      $total_read += $nread if (defined($nread));
112
113      if ($state eq "initial") {
114          my $socks_version = get_socks_version(\$buffer, $nread);
115
116          # SOCKS version 5
117          if ($socks_version == 5) {
118              ($state, $socks_dsthost, $socks_dstport) =
119                  socks5_handler(\$buffer);
120
121          # SOCKS version 4
122          } elsif ($socks_version == 4) {
123              ($state, $socks_dsthost, $socks_dstport) =
124                  socks4_handler(\$buffer);
125
126          # unknown data
127          } else {
128              $state = "unknown";
129          }
130      } # state == initial
131
132      if ($state eq "connected") {
```

```
133         # use default action/prog/prog_args if no specific rules exists
134         # for this port
135         if (!defined($conf{'action_list'}{$socks_dstport}{'action'})) {
136             $conf{'action_list'}{$socks_dstport}{'action'} =
137                 $conf{'action_list'}{'default'}{'action'};
138             $conf{'action_list'}{$socks_dstport}{'prog'} =
139                 $conf{'action_list'}{'default'}{'prog'};
140             $conf{'action_list'}{$socks_dstport}{'prog_args'} =
141                 $conf{'action_list'}{'default'}{'prog_args'};
142         }
143
144         if ($conf{'action_list'}{$socks_dstport}{'prog'} eq "NONE") {
145             logentry($logfile,
146                     sprintf("%s: data to %s:%s: %d byte(s) read: %s",
147                             $srchost, $socks_dsthost, $socks_dstport,
148                             $nread, hexdump(\$buffer, 16)));
149         } else {
150             exec_prog($conf{'action_list'}{$socks_dstport}{'prog'},
151                       $conf{'action_list'}{$socks_dstport}{'prog_args'});
152         }
153     } # state == connected
154
155     if ($state eq "deny") {
156     }
157
158     if ($state eq "unknown") {
159         logentry($logfile,
160                  sprintf("%s: non SOCKS data: %d byte(s) read: %s",
161                          $srchost, $nread,
162                          hexdump(\$buffer, 16)));
163     }
164
165     alarm $timeout;
166 }
167
168 if ($timeout) {
169     my $msg = sprintf("timed out after %d seconds",
170                       $conf{'timeout'}{'value'});
171     logentry($logfile, $msg);
172     die("$program_name: $msg\n");
173 }
174
175 logentry($logfile, "ERROR: sysread: $!") if (!defined($nread));
176 alarm 0;
177
178 logentry($logfile, sprintf("%s: %d byte(s) read, exiting",
179                            $srchost, $total_read));
```

```
180 exit 0;
181
182 ###
183 ### get SOCKS version from packet.
184 ###
185 sub get_socks_version {
186     my ($buffer_ref, $size) = @_;
187     my $buffer = ${$buffer_ref};
188     my $version;
189
190     # test for SOCKS version 5
191     if ($size == 3) {
192         my ($ver, $nmethods, $methods) = unpack("CCC", $buffer);
193         $version = 5 if ($ver == 5);
194     # and for SOCKS version 4
195     } elsif ($size == 9) {
196         my ($vn, $cd, $dstport, $dstaddr) = unpack("CCnA4x", $buffer);
197         $version = 4 if ($vn == 4);
198     } else {
199         $version = 0;
200     }
201     return $version;
202 }
203
204 ###
205 ### handles socks4 connection attempts.
206 ###
207 sub socks4_handler {
208
209     my ($buffer_ref) = @_;
210     my $buffer = ${$buffer_ref};
211
212     # SOCKS: A protocol for TCP proxy across firewalls, by Ying-Da Lee
213     #
214     # CONNECT request:
215     # +----+----+----+----+----+----+----+----+----+----+....+----+
216     # | VN | CD | DSTPORT |      DSTIP        | USERID       |NULL|
217     # +----+----+----+----+----+----+----+----+----+----+....+----+
218     #    1    1      2              4           variable       1
219     #
220     #   VN      socks version (4)
221     #   CD      command code (1 == connect)
222     #   NULL1   byte of all zero bits
223
224     my ($vn, $cd, $dstport, $dstaddr, $null) = unpack("CCnA4x", $buffer);
225     my $dsthost = "";
226
```

```
227     if ($cd == 1) {
228         $dsthost = join('.', unpack('C4', $dstaddr));
229 
230         logentry($logfile,
231                  sprintf("%s: SOCKS 4 connect request: ".
232                          "dst host: %s, dst port: %d",
233                          $srchost, $dsthost, $dstport));
234 
235     # reply:
236     #
237     # +----+----+----+----+----+----+----+----+
238     # | VN | CD | DSTPORT |      DSTIP        |
239     # +----+----+----+----+----+----+----+----+
240     #     1    1      2              4
241     #
242     # VN is the version of the reply code and should be 0. CD is the result
243     # code with one of the following values:
244     #
245     # 90: request granted
246     # 91: request rejected or failed
247     # 92: request rejected becasue SOCKS server cannot connect to
248     #     identd on the client
249     # 93: request rejected because the client program and identd
250     #     report different user-ids
251 
252         # request granted
253         my $nwrite = syswrite(STDOUT, pack("CCnA4",
254                                            0, 90, $dstport, $dstaddr));
255         if (defined($nwrite)) {
256             logentry($logfile, sprintf("%s: request granted sent back ".
257                                        "to client: %d byte(s) written",
258                                        $srchost, $nwrite));
259         } else {
260             logentry($logfile, "ERROR: syswrite: $!");
261         }
262     } else {
263         logentry($logfile, sprintf("socks4 command code: %02X", $cd));
264     }
265 
266     # ok, we're connected.
267     return ("connected", $dsthost, $dstport);
268 }
269 
270 ###
271 ### handles socks5 connection attempts.
272 ###
273 sub socks5_handler
```

```
274 {
275     my ($buffer_ref) = @_;
276     my $buffer = ${$buffer_ref};
277 
278     my ($version, $nmethods, $methods) = unpack("CCC", $buffer);
279 
280     logentry($logfile, sprintf("%s: SOCKS 5 method select: ".
281                                "VER: %s, NMETHODS: %s, METHODS: %s\n",
282                                $srchost, $version, $nmethods, $methods));
283 
284     # socks5 method select reply
285     #
286     # METHOD SELECT reply:
287     #  VER      1 byte  socks version (5)
288     #  METHOD   1 byte  method to use (0 == "no authentication required")
289     #
290     syswrite(STDOUT, pack("CC", 5, 0));
291 
292     ### method is selected, wait for connect request.
293     my $nread = sysread(STDIN, $buffer, $BUFFER_SIZE);
294 
295     # verbose only
296     logentry($logfile, sprintf("%s: %d byte(s) read: %s", $srchost, $nread,
297                                hexdump(\$buffer, 16)));
298 
299     my ($ver, $cmd, $rsv, $atyp, $dstaddr, $dstport) =
300         unpack("CCCCa4n", $buffer);
301 
302     my $dstip;
303     if ($atyp == 1) {
304         $dstip = join('.', unpack('C4', $dstaddr));
305 
306         # connect reply
307         syswrite(STDOUT,
308                  pack("CCCCa4n", $ver, 0, $rsv, $atyp,
309                       $dstaddr, $dstport));
310 
311         logentry($logfile, sprintf("%s: socks5 connect request: ".
312                                    "VER: %s, CMD: %s, RSV: %s, ".
313                                    "ATYP: %s, DST.ADDR: %s, ".
314                                    "DST.PORT: %s\n", $srchost, $ver,
315                                    $cmd, $rsv, $atyp, $dstip, $dstport));
316 
317     # ok, we're connected.
318     return ("connected", $dstip, $dstport);
319 
320     } elsif ($atyp == 3) {
```

```
321         my $hostname_size = $nread - 6;
322         my $template = sprintf("CCCCa%dn", $hostname_size);
323
324         my ($ver, $cmd, $rsv, $atyp, $dstaddr, $dstport) =
325             unpack($template, $buffer);
326
327         my ($size, $dstaddr_hostname) = unpack("CA*", $dstaddr);
328
329         # connect reply
330         syswrite(STDOUT,
331                  pack($template, $ver, 0, $rsv, $atyp,
332                       $dstaddr, $dstport));
333
334         $dstip = $dstaddr_hostname;
335
336         logentry($logfile, sprintf("%s: SOCKS 5 connect request: ".
337                                    "VER: %s, CMD: %s, RSV: %s, ATYP: ".
338                                    "%s, DST.ADDR: %s, DST.PORT: %s\n",
339                                    $srchost, $ver, $cmd, $rsv, $atyp,
340                                    $dstip, $dstport));
341
342         # ok, we're connected.
343         return ("connected", $dstip, $dstport);
344     }
345 }
346
347 ###
348 ### exec a program given as argument.  this function never returns.
349 ###
350 sub exec_prog {
351     my ($prog, $prog_args) = @_;
352
353     logentry($logfile, sprintf("%s: execing %s %s", $srchost,
354                                $prog, $prog_args));
355
356     eval "exec \"$prog $prog_args\"";
357     exit 0;
358 }
359
360 ###
361 ### process the configuration file.  Exits in case of error.
362 ###
363 sub process_config_file {
364     my ($filename, $confref) = @_;
365     my %conf = %{$confref};
366
367     # filename: words, numbers, '-', '/' and '.' are ok.
```

```
368     if ($filename =~ /^([\w\.\-\/]+)$/) {
369         $filename = $1;
370     } else {
371         $filename = quotemeta($filename);
372         die("$program_name: invalid filename: $filename\n");
373     }
374 
375     open(CONF, "$filename") ||
376         die ("$program_name: can't open $filename: $!\n");
377 
378     while (my $line = <CONF>) {
379         chomp($line);
380 
381         # skip comments and empty lines.
382         next if ($line =~ /^$|^\s*$|^\s*#/);
383 
384         # remove leading and trailing blanks
385         $line =~ s/(^\s+|\s+$)//g;
386 
387         # continuation lines
388         while ($line =~ /^(.*)\\$/) {
389           my $content = $1;
390           my $cont_line = <CONF>;
391           die("$program_name: $line: no more lines to continue\n")
392               if (!defined($cont_line));
393           $cont_line =~ s/(^\s+|\s+$)//g;
394           $line = $content . $cont_line;
395           chomp($line);
396         }
397 
398         foreach my $key (keys %conf) {
399             if ($line =~ /^$key\s*[=:]\s*($conf{$key}{'regex'})\s*$/) {
400                 my $value = $1;
401 
402                 # test if is list type
403                 if ($key =~ /_list$/) {
404                     my @components = split(/,/, $value);
405                     my $list_key = "";
406                     foreach my $item (@components) {
407 
408                         if ($item =~ /^\s*([^:]+):\s*(.*)$/) {
409                             # first entry is a key
410                             if ($list_key eq "") {
411                                 $list_key = $2;
412                             } else {
413                                 $conf{$key}{$list_key}{$1} = $2;
414                             }
```

```
415                         } else {
416                             die("$program_name: invalid parameter: ".
417                                 "$item\n");
418                         }
419                     }
420                 } else {
421                     $conf{$key}{'value'} = $value;
422                 }
423 
424                 last;
425             } elsif ($line =~ /^$key\s*/) {
426                 die("$program_name: invalid parameter: $line\n");
427             }
428         }
429     }
430     close(CONF);
431     return \%conf;
432 }
433 
434 ###
435 ### return a hexdump representation of data
436 ###
437 sub hexdump {
438     my ($data_ref, $width) = @_;
439     my $data = ${$data_ref};
440     my @hex_row = ();
441     my @ascii_row = ();
442     my $result = "";
443 
444     foreach my $byte (unpack("C*", $data)) {
445         push @hex_row, sprintf("%02X", $byte);
446         push @ascii_row, (($byte < 32) || ($byte > 126))
447             ? '.' : chr($byte);
448 
449         # print row, if there is enough elements
450         if ($#ascii_row == ($width - 1)) {
451             $result .= sprintf("\n%s |%s|",
452                                join(' ', @hex_row), join('', @ascii_row));
453             @hex_row = ();
454             @ascii_row = ();
455         }
456     }
457     # any elements left?
458     if ($#hex_row != -1) {
459         $result .= sprintf("\n%s%s |%s|",
460                            join(' ', @hex_row),
461                            '   ' x ($width - ($#hex_row+1)),
```

```
462                             join('', @ascii_row));
463     }
464     return $result;
465 }
466 
467 ###
468 ### create a log entry
469 ###
470 sub logentry {
471     my ($logfile, $msg) = @_;
472 
473     my $datum = strftime "%F %T %z", localtime;
474     my $pid = $$;
475 
476     open(LOG, ">>$logfile") ||
477         die "$program_name: $logfile: $!\n";
478     flock(LOG, LOCK_EX);
479     seek(LOG, 0, 2);
480 
481     printf LOG ("%s: %s[%d]: %s\n", $datum, $program_name, $pid, $msg);
482 
483     flock(LOG, LOCK_UN);
484     close(LOG);
485 
486 }
487 
488 ###
489 ### get values from the environment variables set by honeyd and
490 ### return srchost, dsthost, srcport and dstport
491 ###
492 sub get_honeyd_env {
493 
494     # init vars
495     my ($srchost, $dsthost, $srcport, $dstport) =
496         ('127.0.0.1', '127.0.0.1', 0, 0);
497 
498     if (defined($ENV{'HONEYD_IP_SRC'}) &&
499         $ENV{'HONEYD_IP_SRC'} =~ /^(\d+\.\d+\.\d+\.\d+)$/) {
500         $srchost = $1;
501     }
502 
503     if (defined($ENV{'HONEYD_IP_DST'}) &&
504         $ENV{'HONEYD_IP_DST'} =~ /^(\d+\.\d+\.\d+\.\d+)$/) {
505         $dsthost = $1;
506     }
507 
508     if (defined($ENV{'HONEYD_SRC_PORT'}) &&
```

```
509         $ENV{'HONEYD_SRC_PORT'} =~ /^(\d+)$/) {
510         $srcport = $1;
511     }
512
513     if (defined($ENV{'HONEYD_DST_PORT'}) &&
514         $ENV{'HONEYD_DST_PORT'} =~ /^(\d+)$/) {
515         $dstport = $1;
516     }
517
518     return ($srchost, $dsthost, $srcport, $dstport);
519 }
520
521 ###
522 ### print program usage and exit.
523 ###
524 sub show_usage {
525     print <<EOF;
526 Usage: $program_name [-Vh] -f file
527        -h             display this help and exit.
528        -V             display version number and exit.
529        -f file        configuration file.
530 EOF
531
532     exit 0;
533 }
534
535 ###
536 ### print program version and exit.
537 ###
538 sub show_version {
539     printf("%s %s\n", $program_name, $version);
540     exit 0;
541 }
542
543 # EOF
```

D.2 fake.mail.local

```
#! /usr/bin/perl -T
#-*-Perl-*-
#
# fake.mail.local -- store mail in a mailbox (for fake users)
#
# Copyright (c) 2008 Klaus Steding-Jessen
#
# Permission to use, copy, modify, and distribute this software for any
# purpose with or without fee is hereby granted, provided that the above
# copyright notice and this permission notice appear in all copies.
#
# THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
# WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
# ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
# WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
# ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
# OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

(my $program_name = $0) =~ s@.*/@@;

my $version = '0.3';

# the uid we drop privileges to (uid=32767(nobody))
my $non_privileged_uid = 32767;

use strict;
use warnings;
use Getopt::Std;
use Fcntl qw(:flock);
use POSIX qw(strftime setuid);

my %option = ();
getopts('Vhf:l:m:', \%option);

# display usage if requested
show_usage() if ($option{'h'});

# display version if requested
show_version() if ($option{'V'});

###
### main
###
```

```
45
46  # logfile
47  my $logfile = "/var/log/fake.mail.local.log";
48  if (defined($option{'l'})) {
49      if ($option{'l'} =~ /^([\w\/\.\-]+)$/) {
50          $logfile = $1;
51      }
52  }
53
54  # mailbox directory
55  my $mailboxdir = "/var/avenger/mail";
56  if (defined($option{'m'})) {
57      if ($option{'m'} =~ /^([\w\/\.\-]+)$/) {
58          $mailboxdir = $1;
59          $mailboxdir =~ s/\/$//g; # remove trailing "/", just in case
60      }
61  }
62
63  # from user
64  my $from = 'foo\@bar';
65  if (defined($option{'f'})) {
66      if ($option{'f'} =~ /^([\w\.\-\@\_]+)$/) {
67          $from = $1;
68      }
69  }
70
71  # drop privileges
72  if (!setuid($non_privileged_uid)) {
73      logentry($logfile, sprintf("setuid(): %s", $!));
74      die("$program_name: setuid(): $!\n");
75  }
76
77  # WARNING: this program loads each message entirely in memory.
78  # No temporary files are created and no external programs are executed.
79  #
80  # Please make sure your MTA limits the size of each message in a
81  # reasonable size.
82  my $msg_bytes = 0;
83  my @msg = ();
84  while (my $line = <STDIN>) {
85      $line =~ s/^From />From /g;
86      $msg_bytes += length($line);
87      push @msg, $line;
88  }
89
90  # process each user on the cmdline
91  foreach my $user (@ARGV) {
```

```
92          my $mailbox = "unknown";
93          my $domain = "unknowndomain.org";
94          if ($user =~ /^([\w\.\-]+)\@(.*)$/) {
95              $mailbox = $1;
96              $domain = $2;
97          }
98
99          my $from_line  = gen_from_line($from);
100         # prepend our 'From ' line
101         unshift @msg, $from_line;
102         $msg_bytes += length($from_line);
103
104         # append a extra "\n" at the end
105         push @msg, "\n";
106         $msg_bytes++;
107
108         # append the destination domain to mailboxdir
109         $mailboxdir = sprintf("%s/%s", $mailboxdir, $domain);
110
111         # if mailboxdir doesn't exist, try to create it
112         if (! -d $mailboxdir) {
113             if (!mkdir($mailboxdir, 0755)) {
114                 logentry($logfile, sprintf("mkdir(): %s: %s",
115                 $mailboxdir, $!));
116                 die("$program_name: mkdir(): $!\n");
117             }
118         }
119
120         if (!store_msg_in_mailbox($mailboxdir, $mailbox, \@msg)) {
121             logentry($logfile,
122                     sprintf("from: <%s>, to: <%s>, mailbox: ".
123                             "%s/%s, size: %d byte(s)",
124                             $from, $user, $mailboxdir,
125                             $mailbox, $msg_bytes));
126         } else {
127             logentry($logfile,
128             sprintf("ERROR: from: <%s>, to: <%s>, mailbox: %s/%s: %s",
129             $from, $user, $mailboxdir, $mailbox, $!));
130             die("$program_name: store_msg_in_mailbox(): $!\n");
131         }
132 }
133
134 exit 0;
135
136 ##
137 ## return a 'From ' line.
138 ##
```

```
139 sub gen_from_line {
140     my ($from) = @_;
141     my $from_line = "";
142     my $now_string = localtime;
143 
144     $from_line = sprintf("From %s  %s\n", $from, $now_string);
145     return $from_line;
146 }
147 
148 ##
149 ## store the message in a mailbox.
150 ##
151 sub store_msg_in_mailbox {
152     my ($mailboxdir, $mailbox, $msg_ref) = @_;
153     my @msg = @{$msg_ref};
154 
155     my $mailboxfile = sprintf("%s/%s", $mailboxdir, $mailbox);
156 
157     if (!defined(open(MAILBOX, ">>$mailboxfile"))) {
158         logentry($logfile, sprintf("%s: %s", $mailboxfile, $!));
159         return 1; # return failure
160     }
161 
162     flock(MAILBOX, LOCK_EX);
163     seek(MAILBOX, 0, 2);
164 
165     foreach my $line (@msg) {
166         print MAILBOX $line;
167     }
168 
169     flock(MAILBOX, LOCK_UN);
170 
171     if (!close(MAILBOX)) {
172         logentry($logfile, sprintf("%s: %s", $mailboxfile, $!));
173         return 1; # return failure
174     }
175     return 0; # return success
176 }
177 
178 sub rmnonprint {
179     my ($input) = @_;
180     my $output = "";
181 
182     $input =~ s/[^[:print:]]/./g;
183     return $input;
184 }
185 
```

```
186 ##
187 ## create a log entry, removing non printables from the input
188 ##
189 sub logentry {
190     my ($logfile, $msg) = @_;
191 
192     my $datum = strftime "%F %T %z", localtime;
193     my $pid = $$;
194 
195     open(LOG, ">>$logfile") ||
196         die "$program_name: $logfile: $!\n";
197     flock(LOG, LOCK_EX);
198     seek(LOG, 0, 2);
199 
200     printf LOG ("%s: %s[%d]: %s\n", $datum, $program_name, $pid,
201                 rmnonprint($msg));
202 
203     flock(LOG, LOCK_UN);
204     close(LOG);
205 }
206 
207 ##
208 ## print program usage and exit.
209 ##
210 sub show_usage {
211     print <<EOF;
212 Usage: $program_name [-Vh] -l logfile -m mailboxdir -f from -- user ...
213        -h             display this help and exit.
214        -V             display version number and exit.
215        -l             logfile.
216        -m             mailbox dir
217        -f             sender's email.
218 EOF
219 
220     exit 0;
221 }
222 
223 ##
224 ## print program version and exit.
225 ##
226 sub show_version {
227     printf("%s %s\n", $program_name, $version);
228     exit 0;
229 }
230 
231 # EOF
```

D.3 genmail

```
#! /usr/bin/perl -T
#-*-Perl-*-
#
# genmail -- generates a email address on demand
#
# Copyright (c) 2008 Klaus Steding-Jessen
#
# Permission to use, copy, modify, and distribute this software for any
# purpose with or without fee is hereby granted, provided that the above
# copyright notice and this permission notice appear in all copies.
#
# THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
# WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
# ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
# WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
# ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
# OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

use strict;
use POSIX qw(strftime);
use Fcntl qw(:flock);
use Time::Local;
use DB_File;

# unbuffered output
$|=1;

# all times are in UTC
$ENV{'TZ'} = 0;

# program name
(my $program_name = $0) =~ s@.*/@@;

##############################################################################

# REQUEST_URI
my $uri = "UNKNOWN-URI";
if ($ENV{'REQUEST_URI'} =~ /^([\.\/\w\-\:]+)$/) {
    $uri=$1;
}

# HTTP_USER_AGENT
my $user_agent = "UNKNOWN-UA";
```

```
45  if ($ENV{'HTTP_USER_AGENT'} =~ /^(.*)$/) {
46      # remove any non printable character from input
47      $user_agent = rmnonprint($1);
48  }
49
50  # REMOTE_ADDR
51  my $srchost = "UNKNOWN-IP";
52  if ($ENV{'REMOTE_ADDR'} =~ /^([\d\.]+)$/) {
53      $srchost=$1;
54  }
55
56  # QUERY_STRING: field1=value1&field2=value2&field3=value3
57  my $object = "email_1";
58  my $domain = "unknowndomain.org";
59  if ($ENV{'QUERY_STRING'} =~ /^([\w\.\=\&]+)$/) {
60      my $string = $1;
61      foreach my $pair (split(/\&/, $string)) {
62          if ($pair =~ /^domain=([\w\-\.]+)$/) {
63              $domain = $1;
64          } elsif ($pair =~ /^object=([\w]+)$/) {
65              $object = $1;
66          }
67      }
68  }
69
70  ##########################################################################
71  # define logfile, email dbfile and request dbfile based on domain
72
73  # logfile
74  my $logfile = "/var/www/logs/genmail.log-$domain";
75
76  # email dbfile: email -> timestamp
77  my $dbfile = "/var/www/db/emails.db-$domain";
78
79  # request dbfile: (uri, object, srchost, user_agent) -> email
80  my $dbfilerequest = "/var/www/db/requests.db-$domain";
81
82  ##########################################################################
83
84  # lock the requestdb
85  open(DBREQUEST, ">>$dbfilerequest")
86      || die "$program_name: $dbfilerequest: $!\n";
87  flock(DBREQUEST, LOCK_EX);
88
89  my %requests = ();
90  tie(%requests, 'DB_File', $dbfilerequest, O_RDWR|O_CREAT, 0600, $DB_BTREE)
91      || die("$program_name: $!\n");
```

```
 92
 93  my $email = "";
 94  my $status = "";
 95  my $key = sprintf("%s\|%s\|%s\|%s", $uri, $object, $srchost, $user_agent);
 96
 97  # test if we need to generate a new mail to this client
 98  if (defined($requests{$key})) {
 99
100      $email = $requests{$key};
101      $status = sprintf("same email: %s", $email);
102
103  } else {
104
105      # ok, we need to create a new email to this request
106
107      # lock the emaildb
108      open(DB, ">>$dbfile") || die "$program_name: $dbfile: $!\n";
109      flock(DB, LOCK_EX);
110
111      my %used_emails = ();
112      tie(%used_emails, 'DB_File', $dbfile, O_RDWR|O_CREAT, 0600, $DB_BTREE)
113          || die("$program_name: $!\n");
114
115      # create a new email
116      my $tries = 3;
117      while ($tries--) {
118
119          $email = gen_email(0, $domain);
120
121          # already used?
122          if (defined($used_emails{$email})) {
123              $email = sprintf("%s\@%s", "error", $domain);
124          } else {
125              # never seen before, good, record it
126              $used_emails{$email}  = strftime("%s", gmtime);
127              $requests{$key} = $email;
128
129              $status = sprintf("NEW email: %s", $email);
130              last;
131          }
132      }
133
134      untie(%used_emails);
135
136      # unlock the emaildb
137      flock(DB, LOCK_UN);
138      close(DB);
```

```
139 }
140
141 untie(%requests);
142
143 # unlock the requestdb
144 flock(DBREQUEST, LOCK_UN);
145 close(DBREQUEST);
146
147 # log the activity
148 logentry($logfile, sprintf("URI: \"%s\", object: %s, IP: %s, ".
149                            "UA: \"%s\": %s",
150                            $uri, $object, $srchost,
151                            $user_agent, $status));
152
153 ###
154 ### output the email
155 ###
156
157 print "Content-type: text/plain\n\n";
158 printf("%s", $email);
159
160 exit 0;
161
162 ###
163 ### gen_email()
164 ###
165
166 sub gen_email {
167     my ($size, $domain) = @_;
168
169     my @names = ( "aaron", "abby", "abe", "abel", "abigail", "abraham",
170                   "adam", "adeline", "adrian", "adriana", "alan", "albert",
171                   "alec", "alex", "alexander", "alexandra", "alexis",
172                   "alfred", "alice", "alicia", "alison", "allan",
173                   "allison", "amalia", "amanda", "amber", "amelia", "amy",
174                   "ana", "anastacia", "anderson", "andrea", "andrew",
175                   "angela", "angelina", "ann", "anna", "anne", "annie",
176                   "anthony", "antony", "arnold", "arthur", "ashley",
177                   "ashton", "astrid", "augusta", "aurora", "austin",
178                   "axel", "bailey", "baldwin", "barbara", "barney",
179                   "barry", "beatrice", "ben", "benjamin", "benny",
180                   "bernadette", "bernard", "beth", "betty", "bill",
181                   "billie", "black", "bob", "bobby", "brad", "bradley",
182                   "brandon", "brenda", "bret", "brian", "bridget",
183                   "brigitte", "brooke", "brown", "bruce", "bryan", "burt",
184                   "calvin", "camelia", "cameron", "camille", "carl",
185                   "carla", "carlos", "carlton", "carmen", "carol",
```

```
                "carolina", "caroline", "carter", "casey", "cassandra",
                "catherine", "cecil", "cecile", "cecilia", "cezar",
                "charles", "charlie", "charlotte", "chloe", "chris",
                "christiane", "christine", "christopher", "cindy",
                "claire", "clara", "clare", "claude", "claudine",
                "clinton", "cole", "collin", "connor", "constance",
                "cora", "corine", "corinne", "cornelia", "courtney",
                "craig", "craigh", "cruz", "crystal", "curt", "curtis",
                "cyndi", "cynthia", "damian", "dan", "dana", "daniel",
                "daniela", "daniele", "danielle", "danny", "darcy",
                "daren", "dario", "dave", "david", "debbie", "debby",
                "deborah", "denis", "denise", "derek", "dexter", "diana",
                "diane", "diego", "dolores", "dominique", "don", "dona",
                "donald", "donna", "donny", "dora", "doreen", "dorothea",
                "dorotheee", "dorothy", "douglas", "duane", "duke",
                "duncan", "dustin", "dwight", "dylan", "earl", "ed",
                "eddie", "eddy", "edgar", "edison", "edith", "edmund",
                "edward", "elaine", "eleanor", "eleonore", "eliot",
                "elisa", "elisabeth", "elizabeth", "ellen", "ellie",
                "elliott", "elly", "elmer", "elton", "emanuela",
                "emerson", "emilie", "emily", "emma", "emmanuelle",
                "enrique", "eric", "erica", "erika", "ernest", "ernie",
                "esther", "ethan", "eugene", "evan", "eve", "evelina",
                "evelyn", "evelyne", "everett", "fabian", "farrell",
                "felicia", "felix", "ferdinand", "filip", "filippo",
                "florence", "florentina", "florian", "florin", "flower",
                "forest", "forrest", "foster", "fox", "francis",
                "francoise", "frank", "frankie", "franklin", "fraser",
                "fred", "freddie", "freddy", "frederique", "fredrick",
                "freeman", "freeze", "fritz", "fulton", "gabby", "gabe",
                "gabriel", "gabriella", "gabrielle", "gallagher",
                "gareth", "garey", "garret", "garry", "geena", "geffrey",
                "geoffrey", "george", "georgia", "gerald", "geraldine",
                "gerrard", "gerry", "gertrude", "gibson", "gil",
                "gilbert", "gillian", "gina", "giovanni", "giselle",
                "gladys", "glen", "glenn", "gloria", "godfrey", "gordon",
                "grace", "gracie", "graham", "grant", "gray", "green",
                "greg", "gregg", "gregory", "greta", "gretta", "grey",
                "griffin", "gus", "guy", "gwenyth", "hailey", "hamilton",
                "hank", "hannah", "harley", "harold", "harris",
                "harrison", "harry", "harvey", "hayley", "heather",
                "hector", "helen", "helene", "hellen", "hellena",
                "henriette", "henry", "herbert", "herman", "hillary",
                "horace", "howard", "hugo", "humphrey", "ian", "ileen",
                "ines", "inez", "irine", "iris", "irvin", "irving",
                "isaac", "isabel", "isabell", "isabelle", "isaiah",
                "ivan", "jack", "jackie", "jackson", "jade", "james",
```

```
                "jane", "janet", "jannine", "jaqueline", "jared",
                "jarod", "jasmine", "jason", "jay", "jayme", "jazmin",
                "jazmine", "jean", "jeanette", "jeanna", "jeanne", "jed",
                "jeff", "jeffrey", "jena", "jenna", "jennifer", "jenny",
                "jerald", "jeremiah", "jeremy", "jericho", "jerome",
                "jerrard", "jerry", "jesse", "jessica", "jhonathan",
                "jill", "jillian", "jim", "jimmy", "joan", "joanna",
                "joanne", "joby", "jocelyn", "jodi", "jodie", "jody",
                "joe", "joey", "john", "johnathan", "johnny", "jonathan",
                "jordan", "joseph", "josh", "joshua", "joyce", "jude",
                "judith", "judy", "julia", "julian", "juliana", "julie",
                "juliet", "julius", "junior", "justine", "kacey",
                "kailey", "kaitlin", "kaitlyn", "karen", "kate",
                "katherine", "katie", "keaton", "keegan", "keith",
                "kelly", "kelvin", "ken", "kendall", "kennedy", "kenny",
                "kent", "kevin", "kim", "kimberly", "king", "kirk",
                "kris", "kristal", "kurt", "kurtis", "lance", "larry",
                "laura", "lauren", "laurence", "lawrence", "lenox",
                "leo", "leon", "leona", "leonard", "leopold", "lesley",
                "leslie", "liane", "lilian", "liliana", "lillian",
                "lillie", "lilly", "lily", "lincoln", "linda", "lindsey",
                "lisa", "livia", "liz", "liza", "lizabeth", "lloyd",
                "logan", "lois", "loraine", "loreen", "lorena", "lorene",
                "lorraine", "lou", "louie", "louis", "louisa", "louise",
                "loyd", "lucas", "luce", "lucia", "lucian", "lucie",
                "lucius", "lucy", "ludovic", "luigi", "luis", "luke",
                "luna", "luther", "lydia", "lyle", "lyn", "lynda",
                "lynn", "mack", "mackenzie", "madeline", "madison",
                "magdalena", "maggie", "malcom", "mandy", "manuel",
                "marc", "marcel", "marcelle", "marco", "marcus", "marcy",
                "margaret", "margaud", "margaux", "maria", "mariah",
                "marian", "mariana", "marie", "marilena", "marine",
                "mario", "marissa", "marius", "mark", "marla", "marlin",
                "marlon", "marquerite", "marshal", "martha", "marthe",
                "martin", "martine", "marvin", "mary", "marylyn",
                "maryse", "mason", "mat", "mateo", "mathilda",
                "mathilde", "matt", "matteo", "matthew", "maura",
                "maureen", "maurice", "mauro", "maurta", "max",
                "maximilian", "maxwell", "mckenna", "meg", "megan",
                "mel", "melanie", "melisa", "melissa", "merlin", "merry",
                "mia", "michael", "miguel", "mike", "miriam", "mitch",
                "molly", "monica", "monique", "monroe", "morgan",
                "morris", "murphy", "nadia", "nancy", "naomi", "natalie",
                "natasha", "nate", "nath", "nathalie", "nathan", "neal",
                "ned", "neil", "nellie", "nicholas", "nichole", "nicole",
                "nigel", "noah", "noel", "norman", "norton", "octavia",
                "octavius", "odette", "oli", "oliver", "olivia",
```

```
280                 "osborn", "osborne", "osbourne", "oscar", "osvaldo",
281                 "oswald", "otto", "ozzie", "page", "paige", "pam",
282                 "pamela", "panda", "paola", "parker", "pat", "patric",
283                 "patricia", "patrick", "paul", "paula", "paulette",
284                 "pauline", "peg", "peggie", "penelope", "percy", "perry",
285                 "pete", "peter", "peyton", "phil", "philip",
286                 "philippine", "phillip", "phoebe", "phoenix", "quentin",
287                 "quin", "quincy", "quinton", "rachel", "ralf", "ralph",
288                 "randall", "raphael", "raphaela", "ray", "raymond",
289                 "rebecca", "reed", "regina", "renee", "reynold", "rian",
290                 "rich", "richard", "richie", "rick", "rickey", "ricki",
291                 "ricky", "ridley", "riley", "ripley", "rita", "ritchie",
292                 "rob", "robbie", "robby", "robert", "roberta", "robin",
293                 "robyn", "rod", "rodney", "rodrigo", "roger", "rolph",
294                 "ron", "ronald", "ronda", "ronny", "rosaleen", "rosalin",
295                 "rosaline", "rose", "roseanne", "ross", "rowland", "rox",
296                 "roxana", "roxane", "roxie", "roxy", "roy", "royce",
297                 "ruben", "ruby", "rudolf", "rudolph", "rudy", "rupert",
298                 "russ", "russel", "ruth", "ruthie", "ryan", "ryana",
299                 "ryley", "sabine", "sabrina", "sally", "sam", "samantha",
300                 "sammy", "samuel", "sandie", "sandra", "sandy", "sara",
301                 "sarah", "sasha", "scarlet", "scarlett", "scot", "scott",
302                 "scotty", "sean", "sebastian", "selma", "seth",
303                 "seymour", "sharleen", "sharon", "shaun", "sheila",
304                 "shelby", "shelly", "sherman", "sidney", "silvester",
305                 "silvia", "sonia", "sonny", "sophia", "sophie", "stacee",
306                 "stacey", "stacie", "stacy", "stan", "stanford",
307                 "stanley", "stefan", "stella", "steph", "stephen",
308                 "steve", "steven", "stewart", "stu", "stuart", "sue",
309                 "sullivan", "susan", "susana", "susane", "susanna",
310                 "susanne", "suse", "susie", "suzan", "sydney",
311                 "sylvester", "sylvia", "tania", "tasha", "tatiana",
312                 "taylor", "ted", "teddy", "terance", "terence", "teresa",
313                 "tess", "thelma", "theo", "theodora", "theodore",
314                 "theresa", "thomas", "tiffany", "tim", "timmy",
315                 "timothy", "tina", "tisha", "toby", "tod", "todd", "tom",
316                 "tommie", "tommy", "toni", "tony", "tracy", "travis",
317                 "trent", "trevor", "trish", "tristan", "troy", "trudy",
318                 "truman", "tucker", "tyler", "tylor", "tyrell", "tyson",
319                 "valentin", "valentine", "valerie", "vanessa", "velma",
320                 "vera", "vergil", "vernon", "veronica", "vic", "vickie",
321                 "vicky", "victor", "victoria", "vikky", "viktor",
322                 "vince", "vincent", "vinnie", "virgil", "virginia",
323                 "vivian", "vivianne", "vlad", "vladimir", "wade",
324                 "walker", "wallace", "wally", "walt", "walter", "wanda",
325                 "warner", "warren", "wendy", "wesley", "whitney", "wil",
326                 "will", "william", "williams", "willy", "wilma",
```

```
                    "wilson", "winnie", "winona", "winston", "wolf",
                    "wolfgang", "woody", "wright", "wyatt", "wynonna",
                    "xaviar", "xavier", "xaviera", "yasmin", "yasmina",
                    "yasmine", "yolanda", "zach", "zachary", "zack",
                    "zander", "zed", "zeke", "zena", "zeph", "zephyr", "zoe",
                    "zoey"
                    );

    my $email = "";
    my $username = "";

    my $type = int(rand(4));

    ###
    ### prefix + name, e.g. asmith@
    ###
    if ($type == 0) {

        my $prefix = "";
        for(my $i = 1 + int(rand(3)); $i ; $i--) {
            $prefix .= chr(ord('a') + int(rand(26)));
        }
        $email = sprintf("%s%s\@%s", $prefix,
                        $names[int(rand($#names + 1))],
                        $domain);

    ###
    ### name + prefix, e.g. smithf@
    ###
    } elsif ($type == 1) {

        my $prefix = "";
        for(my $i = 1 + int(rand(3)); $i ; $i--) {
            $prefix .= chr(ord('a') + int(rand(26)));
        }
        $email = sprintf("%s%s\@%s", $names[int(rand($#names + 1))],
                        $prefix,
                        $domain);

    ###
    ### prefix + prefix + prefix, e.g. rms@
    ###
    } elsif ($type == 2) {

        $email = sprintf("%s%s%s\@%s",
                        chr(ord('a') + int(rand(26))),
                        chr(ord('a') + int(rand(26))),
```

```
374                         chr(ord('a') + int(rand(26))),
375                         $domain);
376
377     ###
378     ### name + number, e.g. john1234@
379     ###
380     } elsif ($type == 3) {
381
382         my $number = 1 + int(rand(9999));
383         $email = sprintf("%s%d\@%s", $names[int(rand($#names + 1))],
384                         $number, $domain);
385     }
386
387     return $email;
388 }
389
390 ###
391 ### remove non printables from input
392 ###
393 sub rmnonprint {
394     my ($input) = @_;
395     my $output = "";
396
397     $input =~ s/[^[:print:]]/./g;
398     return $input;
399 }
400
401 ###
402 ### create a log entry, removing non printables from the input
403 ###
404 sub logentry {
405
406     my ($logfile, $msg) = @_;
407
408     my $datum = (strftime "%F %T %z", localtime);
409     my $pid = $$;
410
411     open(LOG, ">>$logfile") || die "$program_name: $logfile: $!\n";
412
413     flock(LOG, LOCK_EX);
414     seek(LOG, 0, 2);
415
416     printf LOG ("%s: %s[%d]: %s\n", $datum, $program_name, $pid,
417                 rmnonprint($msg));
418
419     flock(LOG, LOCK_UN);
420     close(LOG);
```

```
421  }
422
423  # EOF
```

D.4 genfile

```
1  #! /usr/bin/perl -T
2  #-*-Perl-*-
3  #
4  #  genfile -- generates a file on demand
5  #
6  # Copyright (c) 2008 Klaus Steding-Jessen
7  #
8  # Permission to use, copy, modify, and distribute this software for any
9  # purpose with or without fee is hereby granted, provided that the above
10 # copyright notice and this permission notice appear in all copies.
11 #
12 # THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
13 # WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
14 # MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
15 # ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
16 # WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
17 # ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
18 # OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
19
20 use strict;
21 use POSIX qw(strftime);
22 use Fcntl qw(:flock);
23 use Time::Local;
24 use DB_File;
25
26 # unbuffered output
27 $|=1;
28
29 # all times are in UTC
30 $ENV{'TZ'} = 0;
31
32 # program name
33 (my $program_name = $0) =~ s@.*/@@;
34
35 ##########################################################################
36 # extension -> MIME type
37
38 my %mime_type = (
39                  # extension   # content-type
40                  'txt', 'text/plain',
41                  'tar', 'application/x-tar',
42                  'ps',  'application/postscript',
43                  'pdf', 'application/pdf',
44                  'ppt', 'application/vnd.ms-powerpoint',
```

```
45                  'doc', 'application/msword',
46                  );
47
48  ##########################################################################
49  # redirect values
50
51  my $status_code = 301; # Moved Permanently
52
53  my $status_text = "Moved Permanently";
54
55  my $redirect_page = "/error.html";
56
57  ##########################################################################
58
59  # REQUEST_URI
60  my $uri = "UNKNOWN-URI";
61  if ($ENV{'REQUEST_URI'} =~ /^([\.\/\w\-\:]+)$/) {
62      $uri = $1;
63  }
64
65  # HTTP_USER_AGENT
66  my $user_agent = "UNKNOWN-UA";
67  if ($ENV{'HTTP_USER_AGENT'} =~ /^(.*)$/) {
68      # remove any non printable character from input
69      $user_agent = rmnonprint($1);
70  }
71
72  # REMOTE_ADDR
73  my $srchost = "UNKNOWN-IP";
74  if ($ENV{'REMOTE_ADDR'} =~ /^([\d\.]+)$/) {
75      $srchost = $1;
76  }
77
78  # SERVER_NAME
79  my $server_name = "UNKNOWN-SERVER";
80  if ($ENV{'SERVER_NAME'} =~ /^([\w\.\-]+)$/) {
81      $server_name = $1;
82  }
83
84  # we can only generate one email embedded per file
85  my $object = "email_1";
86
87  ##########################################################################
88  # define logfile, email dbfile, request dbfile and file-temples
89  # based on domain.  domain is defined based on SERVER_NAME.
90
91  my $domain = "unknowndomain.org";
```

```
92  if ($server_name =~ /^[^\.]+\.(.*)$/) {
93      $domain = $1;
94  }
95
96  # logfile
97  my $logfile = "/var/www/logs/genfile.log-$domain";
98
99  # email dbfile: email -> timestamp
100 my $dbfile = "/var/www/db/emails.db-$domain";
101
102 # request dbfile: (uri, object, srchost, user_agent) -> email
103 my $dbfilerequest = "/var/www/db/requests.db-$domain";
104
105 # filetemplates dir
106 my $filetemplatesdir = "/var/www/file-templates-$domain";
107
108 ##########################################################################
109
110 # check if we know how to produce this file
111 my $template_file = sprintf("%s/%s", $filetemplatesdir, $uri);
112 if (! -f $template_file) {
113
114     logentry($logfile, sprintf("URI: \"%s\", object: %s, IP: %s, ".
115                                "UA: \"%s\": %s",
116                                $uri, $object, $srchost, $user_agent,
117                                "not a known file, redirecting to ".
118                                "$redirect_page..."));
119
120     # redirect the browser to our error page
121     printf("Status: %d %s\n", $status_code, $status_text);
122     printf("Location: %s\n\n", $redirect_page);
123
124     exit 0;
125 }
126
127 # lock the requestdb
128 open(DBREQUEST, ">>$dbfilerequest")
129     || die "$program_name: $dbfilerequest: $!\n";
130 flock(DBREQUEST, LOCK_EX);
131
132 my %requests = ();
133 tie(%requests, 'DB_File', $dbfilerequest, O_RDWR|O_CREAT, 0600, $DB_BTREE)
134     || die("$program_name: $!\n");
135
136 my $email = "";
137 my $status = "";
138 my $key = sprintf("%s\|%s\|%s\|%s", $uri, $object, $srchost, $user_agent);
```

```
139
140 # test if we need to generate a new mail to this client
141 if (defined($requests{$key})) {
142
143     $email = $requests{$key};
144     $status = sprintf("same email: %s", $email);
145
146 } else {
147
148     # ok, we need to create a new email to this request
149
150     # lock the emaildb
151     open(DB, ">>$dbfile") || die "$program_name: $dbfile: $!\n";
152     flock(DB, LOCK_EX);
153
154     my %used_emails = ();
155     tie(%used_emails, 'DB_File', $dbfile, O_RDWR|O_CREAT, 0600, $DB_BTREE)
156         || die("$program_name: $!\n");
157
158     # create a new email
159     my $tries = 3;
160     while ($tries--) {
161
162         $email = gen_email(0, $domain);
163
164         # already used?
165         if (defined($used_emails{$email})) {
166             $email = sprintf("%s\@%s", "error", $domain);
167         } else {
168             # never seen before, good, record it
169             $used_emails{$email}  = strftime("%s", gmtime);
170             $requests{$key} = $email;
171
172             $status = sprintf("NEW email: %s", $email);
173             last;
174         }
175     }
176
177     untie(%used_emails);
178
179     # unlock the emaildb
180     flock(DB, LOCK_UN);
181     close(DB);
182 }
183
184 untie(%requests);
185
```

```
186 # unlock the requestdb
187 flock(DBREQUEST, LOCK_UN);
188 close(DBREQUEST);
189
190 ###
191 ### output the file
192 ###
193
194 my $ext = "";
195 my $template_file_basename = "";
196
197 if ($template_file =~ /\/([^\/]+)$/) {
198     $template_file_basename = $1;
199 }
200
201 if ($template_file_basename =~ /\.([^\.]+)$/) {
202     $ext = $1;
203 }
204
205 # content_type: return 'text/plain' if nothing better was found
206 my $content_type = 'text/plain';
207 if (defined($mime_type{$ext})) {
208     $content_type = $mime_type{$ext};
209 }
210
211 open(FILE, $template_file) ||
212     die("genfile: can't open $template_file: $!\n");
213
214 printf("Content-type: %s\n\n", $content_type);
215
216 # output template file, replacing email placeholder
217 my $file_bytes = 0;
218 while(my $line = <FILE>) {
219     if ($line =~ /XXXMAILADDRESSXXX/) {
220         $line =~ s/XXXMAILADDRESSXXX/$email/g;
221     } elsif ($line =~ /Y+MAILADDRESSY+/) {
222         $line =~ s/(Y+MAILADDRESSY+)/exactreplace($1,$email)/ge;
223     }
224     $file_bytes += length($line);
225     print $line;
226 }
227
228 close(FILE);
229
230 # log the activity
231 logentry($logfile,
232          sprintf("URI: \"%s\", object: %s, IP: %s, UA: \"%s\", ".
```

```
233                   "size: %d byte(s): %s",
234                   $uri, $object, $srchost, $user_agent,
235                   $file_bytes, $status));
236
237 exit 0;
238
239 ###
240 ### gen_email()
241 ###
242
243 sub gen_email {
244     my ($size, $domain) = @_;
245
246     my @names = ( "aaron", "abby", "abe", "abel", "abigail", "abraham",
247                   "adam", "adeline", "adrian", "adriana", "alan", "albert",
248                   "alec", "alex", "alexander", "alexandra", "alexis",
249                   "alfred", "alice", "alicia", "alison", "allan",
250                   "allison", "amalia", "amanda", "amber", "amelia", "amy",
251                   "ana", "anastacia", "anderson", "andrea", "andrew",
252                   "angela", "angelina", "ann", "anna", "anne", "annie",
253                   "anthony", "antony", "arnold", "arthur", "ashley",
254                   "ashton", "astrid", "augusta", "aurora", "austin",
255                   "axel", "bailey", "baldwin", "barbara", "barney",
256                   "barry", "beatrice", "ben", "benjamin", "benny",
257                   "bernadette", "bernard", "beth", "betty", "bill",
258                   "billie", "black", "bob", "bobby", "brad", "bradley",
259                   "brandon", "brenda", "bret", "brian", "bridget",
260                   "brigitte", "brooke", "brown", "bruce", "bryan", "burt",
261                   "calvin", "camelia", "cameron", "camille", "carl",
262                   "carla", "carlos", "carlton", "carmen", "carol",
263                   "carolina", "caroline", "carter", "casey", "cassandra",
264                   "catherine", "cecil", "cecile", "cecilia", "cezar",
265                   "charles", "charlie", "charlotte", "chloe", "chris",
266                   "christiane", "christine", "christopher", "cindy",
267                   "claire", "clara", "clare", "claude", "claudine",
268                   "clinton", "cole", "collin", "connor", "constance",
269                   "cora", "corine", "corinne", "cornelia", "courtney",
270                   "craig", "craigh", "cruz", "crystal", "curt", "curtis",
271                   "cyndi", "cynthia", "damian", "dan", "dana", "daniel",
272                   "daniela", "daniele", "danielle", "danny", "darcy",
273                   "daren", "dario", "dave", "david", "debbie", "debby",
274                   "deborah", "denis", "denise", "derek", "dexter", "diana",
275                   "diane", "diego", "dolores", "dominique", "don", "dona",
276                   "donald", "donna", "donny", "dora", "doreen", "dorothea",
277                   "dorotheee", "dorothy", "douglas", "duane", "duke",
278                   "duncan", "dustin", "dwight", "dylan", "earl", "ed",
279                   "eddie", "eddy", "edgar", "edison", "edith", "edmund",
```

```
280                 "edward", "elaine", "eleanor", "eleonore", "eliot",
281                 "elisa", "elisabeth", "elizabeth", "ellen", "ellie",
282                 "elliott", "elly", "elmer", "elton", "emanuela",
283                 "emerson", "emilie", "emily", "emma", "emmanuelle",
284                 "enrique", "eric", "erica", "erika", "ernest", "ernie",
285                 "esther", "ethan", "eugene", "evan", "eve", "evelina",
286                 "evelyn", "evelyne", "everett", "fabian", "farrell",
287                 "felicia", "felix", "ferdinand", "filip", "filippo",
288                 "florence", "florentina", "florian", "florin", "flower",
289                 "forest", "forrest", "foster", "fox", "francis",
290                 "francoise", "frank", "frankie", "franklin", "fraser",
291                 "fred", "freddie", "freddy", "frederique", "fredrick",
292                 "freeman", "freeze", "fritz", "fulton", "gabby", "gabe",
293                 "gabriel", "gabriella", "gabrielle", "gallagher",
294                 "gareth", "garey", "garret", "garry", "geena", "geffrey",
295                 "geoffrey", "george", "georgia", "gerald", "geraldine",
296                 "gerrard", "gerry", "gertrude", "gibson", "gil",
297                 "gilbert", "gillian", "gina", "giovanni", "giselle",
298                 "gladys", "glen", "glenn", "gloria", "godfrey", "gordon",
299                 "grace", "gracie", "graham", "grant", "gray", "green",
300                 "greg", "gregg", "gregory", "greta", "gretta", "grey",
301                 "griffin", "gus", "guy", "gwenyth", "hailey", "hamilton",
302                 "hank", "hannah", "harley", "harold", "harris",
303                 "harrison", "harry", "harvey", "hayley", "heather",
304                 "hector", "helen", "helene", "hellen", "hellena",
305                 "henriette", "henry", "herbert", "herman", "hillary",
306                 "horace", "howard", "hugo", "humphrey", "ian", "ileen",
307                 "ines", "inez", "irine", "iris", "irvin", "irving",
308                 "isaac", "isabel", "isabell", "isabelle", "isaiah",
309                 "ivan", "jack", "jackie", "jackson", "jade", "james",
310                 "jane", "janet", "jannine", "jaqueline", "jared",
311                 "jarod", "jasmine", "jason", "jay", "jayme", "jazmin",
312                 "jazmine", "jean", "jeanette", "jeanna", "jeanne", "jed",
313                 "jeff", "jeffrey", "jena", "jenna", "jennifer", "jenny",
314                 "jerald", "jeremiah", "jeremy", "jericho", "jerome",
315                 "jerrard", "jerry", "jesse", "jessica", "jhonathan",
316                 "jill", "jillian", "jim", "jimmy", "joan", "joanna",
317                 "joanne", "joby", "jocelyn", "jodi", "jodie", "jody",
318                 "joe", "joey", "john", "johnathan", "johnny", "jonathan",
319                 "jordan", "joseph", "josh", "joshua", "joyce", "jude",
320                 "judith", "judy", "julia", "julian", "juliana", "julie",
321                 "juliet", "julius", "junior", "justine", "kacey",
322                 "kailey", "kaitlin", "kaitlyn", "karen", "kate",
323                 "katherine", "katie", "keaton", "keegan", "keith",
324                 "kelly", "kelvin", "ken", "kendall", "kennedy", "kenny",
325                 "kent", "kevin", "kim", "kimberly", "king", "kirk",
326                 "kris", "kristal", "kurt", "kurtis", "lance", "larry",
```

```
                "laura", "lauren", "laurence", "lawrence", "lenox",
                "leo", "leon", "leona", "leonard", "leopold", "lesley",
                "leslie", "liane", "lilian", "liliana", "lillian",
                "lillie", "lilly", "lily", "lincoln", "linda", "lindsey",
                "lisa", "livia", "liz", "liza", "lizabeth", "lloyd",
                "logan", "lois", "loraine", "loreen", "lorena", "lorene",
                "lorraine", "lou", "louie", "louis", "louisa", "louise",
                "loyd", "lucas", "luce", "lucia", "lucian", "lucie",
                "lucius", "lucy", "ludovic", "luigi", "luis", "luke",
                "luna", "luther", "lydia", "lyle", "lyn", "lynda",
                "lynn", "mack", "mackenzie", "madeline", "madison",
                "magdalena", "maggie", "malcom", "mandy", "manuel",
                "marc", "marcel", "marcelle", "marco", "marcus", "marcy",
                "margaret", "margaud", "margaux", "maria", "mariah",
                "marian", "mariana", "marie", "marilena", "marine",
                "mario", "marissa", "marius", "mark", "marla", "marlin",
                "marlon", "marquerite", "marshal", "martha", "marthe",
                "martin", "martine", "marvin", "mary", "marylyn",
                "maryse", "mason", "mat", "mateo", "mathilda",
                "mathilde", "matt", "matteo", "matthew", "maura",
                "maureen", "maurice", "mauro", "maurta", "max",
                "maximilian", "maxwell", "mckenna", "meg", "megan",
                "mel", "melanie", "melisa", "melissa", "merlin", "merry",
                "mia", "michael", "miguel", "mike", "miriam", "mitch",
                "molly", "monica", "monique", "monroe", "morgan",
                "morris", "murphy", "nadia", "nancy", "naomi", "natalie",
                "natasha", "nate", "nath", "nathalie", "nathan", "neal",
                "ned", "neil", "nellie", "nicholas", "nichole", "nicole",
                "nigel", "noah", "noel", "norman", "norton", "octavia",
                "octavius", "odette", "oli", "oliver", "olivia",
                "osborn", "osborne", "osbourne", "oscar", "osvaldo",
                "oswald", "otto", "ozzie", "page", "paige", "pam",
                "pamela", "panda", "paola", "parker", "pat", "patric",
                "patricia", "patrick", "paul", "paula", "paulette",
                "pauline", "peg", "peggie", "penelope", "percy", "perry",
                "pete", "peter", "peyton", "phil", "philip",
                "philippine", "phillip", "phoebe", "phoenix", "quentin",
                "quin", "quincy", "quinton", "rachel", "ralf", "ralph",
                "randall", "raphael", "raphaela", "ray", "raymond",
                "rebecca", "reed", "regina", "renee", "reynold", "rian",
                "rich", "richard", "richie", "rick", "rickey", "ricki",
                "ricky", "ridley", "riley", "ripley", "rita", "ritchie",
                "rob", "robbie", "robby", "robert", "roberta", "robin",
                "robyn", "rod", "rodney", "rodrigo", "roger", "rolph",
                "ron", "ronald", "ronda", "ronny", "rosaleen", "rosalin",
                "rosaline", "rose", "roseanne", "ross", "rowland", "rox",
                "roxana", "roxane", "roxie", "roxy", "roy", "royce",
```

```
                    "ruben", "ruby", "rudolf", "rudolph", "rudy", "rupert",
                    "russ", "russel", "ruth", "ruthie", "ryan", "ryana",
                    "ryley", "sabine", "sabrina", "sally", "sam", "samantha",
                    "sammy", "samuel", "sandie", "sandra", "sandy", "sara",
                    "sarah", "sasha", "scarlet", "scarlett", "scot", "scott",
                    "scotty", "sean", "sebastian", "selma", "seth",
                    "seymour", "sharleen", "sharon", "shaun", "sheila",
                    "shelby", "shelly", "sherman", "sidney", "silvester",
                    "silvia", "sonia", "sonny", "sophia", "sophie", "stacee",
                    "stacey", "stacie", "stacy", "stan", "stanford",
                    "stanley", "stefan", "stella", "steph", "stephen",
                    "steve", "steven", "stewart", "stu", "stuart", "sue",
                    "sullivan", "susan", "susana", "susane", "susanna",
                    "susanne", "suse", "susie", "suzan", "sydney",
                    "sylvester", "sylvia", "tania", "tasha", "tatiana",
                    "taylor", "ted", "teddy", "terance", "terence", "teresa",
                    "tess", "thelma", "theo", "theodora", "theodore",
                    "theresa", "thomas", "tiffany", "tim", "timmy",
                    "timothy", "tina", "tisha", "toby", "tod", "todd", "tom",
                    "tommie", "tommy", "toni", "tony", "tracy", "travis",
                    "trent", "trevor", "trish", "tristan", "troy", "trudy",
                    "truman", "tucker", "tyler", "tylor", "tyrell", "tyson",
                    "valentin", "valentine", "valerie", "vanessa", "velma",
                    "vera", "vergil", "vernon", "veronica", "vic", "vickie",
                    "vicky", "victor", "victoria", "vikky", "viktor",
                    "vince", "vincent", "vinnie", "virgil", "virginia",
                    "vivian", "vivianne", "vlad", "vladimir", "wade",
                    "walker", "wallace", "wally", "walt", "walter", "wanda",
                    "warner", "warren", "wendy", "wesley", "whitney", "wil",
                    "will", "william", "williams", "willy", "wilma",
                    "wilson", "winnie", "winona", "winston", "wolf",
                    "wolfgang", "woody", "wright", "wyatt", "wynonna",
                    "xaviar", "xavier", "xaviera", "yasmin", "yasmina",
                    "yasmine", "yolanda", "zach", "zachary", "zack",
                    "zander", "zed", "zeke", "zena", "zeph", "zephyr", "zoe",
                    "zoey"
                    );

    my $email = "";
    my $username = "";

    my $type = int(rand(4));

    ###
    ### prefix + name, e.g. asmith@
    ###
    if ($type == 0) {
```

```
421
422         my $prefix = "";
423         for(my $i = 1 + int(rand(3)); $i ; $i--) {
424             $prefix .= chr(ord('a') + int(rand(26)));
425         }
426         $email = sprintf("%s%s\@%s", $prefix,
427                         $names[int(rand($#names + 1))],
428                         $domain);
429
430     ###
431     ### name + prefix, e.g. smithf@
432     ###
433     } elsif ($type == 1) {
434
435         my $prefix = "";
436         for(my $i = 1 + int(rand(3)); $i ; $i--) {
437             $prefix .= chr(ord('a') + int(rand(26)));
438         }
439         $email = sprintf("%s%s\@%s", $names[int(rand($#names + 1))],
440                         $prefix,
441                         $domain);
442
443     ###
444     ### prefix + prefix + prefix, e.g. rms@
445     ###
446     } elsif ($type == 2) {
447
448         $email = sprintf("%s%s%s\@%s",
449                         chr(ord('a') + int(rand(26))),
450                         chr(ord('a') + int(rand(26))),
451                         chr(ord('a') + int(rand(26))),
452                         $domain);
453
454     ###
455     ### name + number, e.g. john1234@
456     ###
457     } elsif ($type == 3) {
458
459         my $number = 1 + int(rand(9999));
460         $email = sprintf("%s%d\@%s", $names[int(rand($#names + 1))],
461                         $number, $domain);
462     }
463
464     return $email;
465 }
466
467 ###
```

```
468 ### replace str by email, but maintain the original length
469 ###
470 sub exactreplace {
471     my ($str, $email) = @_;
472     my $output = "";
473 
474     $output = sprintf("%s%s", $email,
475                       " " x (length($str) - length($email)));
476     return $output;
477 }
478 
479 ###
480 ### remove non printables from input
481 ###
482 sub rmnonprint {
483     my ($input) = @_;
484     my $output = "";
485 
486     $input =~ s/[^[:print:]]/./g;
487     return $input;
488 }
489 
490 ###
491 ### create a log entry, removing non printables from the input
492 ###
493 sub logentry {
494 
495     my ($logfile, $msg) = @_;
496 
497     my $datum = (strftime "%F %T %z", localtime);
498     my $pid = $$;
499 
500     open(LOG, ">>$logfile") || die "$program_name: $logfile: $!\n";
501 
502     flock(LOG, LOCK_EX);
503     seek(LOG, 0, 2);
504 
505     printf LOG ("%s: %s[%d]: %s\n", $datum, $program_name, $pid,
506                 rmnonprint($msg));
507 
508     flock(LOG, LOCK_UN);
509     close(LOG);
510 }
511 
512 # EOF
```

ÍNDICE

T

U

V

W

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programas de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. Aceitam-se tanto programas fonte quanto os executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.