Ministério da
Ciência e Tecnologia

INPE-15347-TDI/1383

PROJETO E IMPLEMENTAÇÃO DE UMA INFRA-ESTRUTURA PARA TROCA E ANÁLISE DE INFORMAÇÕES DE HONEYPOTS E HONEYNETS

Cristine Hoepers

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Antonio Montes Filho e Nandamudi Lankapalli Vijaykumar, aprovada em 7 de julho de 2008.

Registro do documento original:
<http://urlib.net/sid.inpe.br/mtc-m18@80/2008/08.21.12.34>

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-15347-TDI/1383

PROJETO E IMPLEMENTAÇÃO DE UMA INFRA-ESTRUTURA PARA TROCA E ANÁLISE DE INFORMAÇÕES DE HONEYPOTS E HONEYNETS

Cristine Hoepers

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Antonio Montes Filho e Nandamudi Lankapalli Vijaykumar, aprovada em 7 de julho de 2008.

Registro do documento original:
<http://urlib.net/sid.inpe.br/mtc-m18@80/2008/08.21.12.34>

INPE
São José dos Campos
2008

Dados Internacionais de Catalogação na Publicação (CIP)

H671p Hoepers, Cristine.
Projeto e implementação de uma infra-estrutura para troca e análise de informações de honeypots e honeynets/ Cristine Hoepers. – São José dos Campos: INPE, 2008.
199p. ; (INPE-15347-TDI/1383)

1. Honeypots. 2. Honeynets. 3. Formato para troca de dados. 4. XML. 5. Correlação de dados. I. Título.

CDU 004.056:004.057.3

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. Edson dos Santos Moreira

Convidado(a) / USP-São Carlos / São Carlos - SP

Dr. Paulo Sergio da Motta Pires

Convidado(a) / UFRN / Natal - RN

Aluno (a): Cristine Hoepers

São José dos Campos, 07 de julho de 2008

AGRADECIMENTOS

Agradeço ao Klaus Steding-Jessen por todo o apoio, idéias e sugestões, não só no desenvolvimento desta tese, mas como em todos os anos de doutorado. Sem seus questionamentos e sugestões, além de sua companhia nos finais de semana dedicados à tese e às disciplinas e em todas as horas de estrada, este trabalho não teria chegado ao fim.

Ao Professor Nandamudi L. Vijaykumar pela paciência, apoio, orientação e amizade, sem os quais este trabalho não teria se concretizado.

A meus pais e a todos os meus familiares, que me deram todo o suporte para que eu chegasse até aqui e compreenderam todos esses anos sem férias, a ausência em datas especiais e que me aguentaram falando sobre este trabalho, mesmo sem ter idéia do que eu estava dizendo.

A meus colegas do CERT.br pelo apoio e entusiasmo por todos os projetos que fizemos em conjunto e pela compreensão por todas as vezes em que minhas preocupações estiveram na tese.

A meu empregador, o NIC.br, por ter me permitido dedicar o tempo e os recursos necessários para completar o doutoramento. A compreensão ao flexibilizar os horários de trabalho e o incentivo para a sua conclusão foram essenciais.

Ao Professor Antonio Montes, que iniciou minha orientação no INPE e que "comprou" as idéias de implantar uma *honeynet* e de mudar o foco do trabalho para *honeypots*.

Aos membros da banca examinadora por terem aceito participar e por todas as valiosas contribuições para tornar este trabalho melhor.

Agradeço, também, a todos os professores, colegas de pós-graduação e demais colegas de trabalho pelo convívio e por todas as contribuições que, direta ou indiretamente, deram ao meu trabalho.

RESUMO

Para caracterizar e monitorar atividades maliciosas na Internet, um dos métodos utilizados tem sido a colocação de sensores em espaços de endereçamento não utilizados. Dentre os tipos de sensores empregados para este fim, destacam-se os *honeypots*, que são recursos de segurança especialmente configurados para coletar informações sobre ataques e cujo valor reside em serem sondados, atacados ou comprometidos. Eles são capazes de coletar informações valiosas sobre os tipos de ataques que ocorrem na Internet e de auxiliar o tratamento de incidentes de segurança. Os trabalhos existentes na área de coleta e análise de dados de *honeypots* concentram-se na visualização e/ou correlação de dados em uma *honeynet* específica ou em um conjunto de *honeypots* que utilizem tecnologias similares. Contudo, é importante realizar uma análise mais completa do tráfego observado entre *honeypots* e *honeynets* que utilizem diferentes tecnologias e estejam implantados em diferentes ambientes espalhados ao redor do mundo. De modo a tratar as limitações na área de análise de dados de *honeypots* e propiciar a interoperabilidade entre diversas tecnologias, este trabalho define dois elementos para formar uma infra-estrutura que permita a análise e correlação desses dados: o *Honeypots Information and Data Exchange Format* (HIDEF), um formato para troca de dados coletados e de informações sobre a arquitetura e tecnologias usadas por *honeypots*; e o *Honeypots Information and Data Exchange and Analysis System* (HIDEAS), sistema que habilita o envio e o recebimento de informações e dados em formatos como o HIDEF. Para validar o formato e o sistema propostos, foi implementado um protótipo e realizado um estudo de caso, que coletou e correlacionou dados de *honeypots* com tecnologias diferentes, bem como de notificações de incidentes de segurança.

DESIGN AND IMPLEMENTATION OF AN INFRASTRUCTURE FOR HONEYPOTS AND HONEYNETS' INFORMATION EXCHANGE AND ANALYSIS

ABSTRACT

Placing sensors in unused Internet address space is one of the techniques used to characterize and monitor malicious activities. Among the diverse types of sensors, honeypots stand out. They are security resources specially configured to collect data about attacks, and whose value lies in being probed, attacked or compromised. Honeypots are able to capture valuable information about Internet attacks and to help computer security incident handling. The related work in the area of honeypots' data collection and analysis is focused on visualization or correlation of data from a unique honeynet or from a set of honeypots that use similar technologies. However, it is very important to make a more complete analysis of the traffic observed among honeypots and honeynets which use different technologies and are deployed in different parts of the world. To address the limitations in the honeypots data analysis area, and to provide interoperability among different technologies, this work presents two elements that comprise an infrastructure that allows the analysis and correlation of honeypots' data: the *Honeypots Information and Data Exchange Format* (HIDEF), a data format to exchange data collected by honeypots and information about the architecture and technologies used by them; and the *Honeypots Information and Data Exchange and Analysis System* (HIDEAS), a system that enables sending and receiving information and data represented in formats like HIDEF. To validate the data format and the system proposed a prototype was implemented. This prototype was used in a case study that correlated data from honeypots deployed with different technologies, as well as from security incident reports.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

AC	–	Autoridade Certificadora
AfriNIC	–	*Regional Registry for Internet Number Resources for Africa*
APNIC	–	*Asia Pacific Network Information Centre*
AR	–	Argentina
ARIN	–	*American Registry for Internet Numbers*
ARK	–	*Archipelago Measurement Infrastructure*
AS	–	*Autonomous System*
ASN	–	*Autonomous System Number*
ATM	–	*Asynchronous Transfer Mode*
BR	–	Brasil
CAIDA	–	*Cooperative Association for Internet Data Analysis*
CAPES	–	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CC	–	*Country Code*
CenPRA	–	Centro de Pesquisas Renato Archer
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*
CN	–	China
CSIRT	–	*Computer Security Incident Response Team*
DDoS	–	*Distributed Denial of Service*
DE	–	Alemanha
DNS	–	*Domain Name System*
DTD	–	*Document Type Definition*
E-R	–	Entidade-Relacionamento
ES	–	Espanha
FR	–	França
GB	–	Grã Bretanha – no contexto de *Country Codes* é associado a
	–	domínios da Inglaterra
GMT	–	*Greenwich Mean Time*
HIDEAS	–	*Honeypots Information and Data Exchange and Analysis System*
HIDEF	–	*Honeypots Information and Data Exchange Format*
HK	–	Hong Kong
HSC	–	*Honeynet Security Console*
HTML	–	*HyperText Markup Language*

HTTP – *HyperText Transfer Protocol*
HTTPS – HTTP sobre SSL/TLS
IANA – *Internet Assigned Numbers Authority*
ICMP – *Internet Control Message Protocol*
IDMEF – *The Intrusion Detection Message Exchange Format*
IDS – *Intrusion Detection System*
IDWG – *Intrusion Detection Exchange Format Working Group*
IESG – *IETF Internet Engineering Steering Group*
IETF – *The Internet Engineering Task Force*
INCH – *Extended Incident Handling Working Group*
INPE – Instituto Nacional de Pesquisas Espaciais
IODEF – *Incident Object Description and Exchange Format*
IP – *Internet Protocol*
IPv4 – *Internet Protocol version 4*
IPv6 – *Internet Protocol version 6*
IRC – *Internet Relay Chat*
IRT – *Incident Response Team*
ISO – *International Organization for Standardization*
ISSN – *International Standard Serial Number*
IT – Itália
JP – Japão
JPCERT/CC – JPCERT *Coordination Center*
KR – Coréia do Sul
LACNIC – *Latin American and Caribbean Internet Addresses Registry*
LAN – *Local Area Network*
LBL – *Lawrence Berkeley Laboratory*
MAC – *Media Access Control*
MD5 – *Message-Digest Algorithm* 5
NIC.br – Núcleo de Informação e Coordenação do Ponto br
NIST – *National Institute of Standards and Technology*
OS – *Operating System*
P2P – *Peer to Peer*
PGP – *Pretty Good Privacy*
PL – Polônia
RFC – *Request for Comments*
RIPE NCC – *RIPE Network Coordination Centre*
RIR – *Regional Internet Registry*
RRDtool – *Round Robin Database Tool*
SHA – *Secure Hash Algorithms*
SMTP – *Simple Mail Transfer Protocol*

SO – Sistema Operacional
SOCKS – *SOCKS Protocol*
SQL – *Structured Query Language*
SSH – *Secure Shell*
SSL – *Secure Sockets Layer*
TCP – *Internet Transmission Control Protocol*
TERENA – *Trans-European Research and Education Networking Association*
TLS – *Transport Layer Security*
TW – *Taiwan*
UDP – *Internet User Datagram Protocol*
UFRN – Universidade Federal do Rio Grande do Norte
UML – *Unified Modeling Language*
URL – *Universal Resource Locator*
US – Estados Unidos da América
USP – Universidade de São Paulo
VLAN – *Virtual LAN*
VNC – *Virtual Network Computing*
W3C – *World Wide Web Consortium*
WWW – *World Wide Web*
XML – *Extensible Markup Language*
XPath – XML *Path Language*

1 INTRODUÇÃO

A segurança de computadores e redes é fundamentada na manutenção da confidencialidade, da integridade e da disponibilidade de dados e sistemas, protegendo-os das possíveis ameaças à sua segurança (BISHOP, 2002; BACE, 2000). Porém, mesmo os sistemas mais bem protegidos devem ser monitorados para que seja possível a detecção de tentativas de ataques ou ataques bem sucedidos (BISHOP, 2002). Tradicionalmente, são os Sistemas de Detecção de Intrusão (IDS) os elementos responsáveis por monitorar sistemas e redes de computadores para detectar incidentes de segurança (BACE, 2000; WYK; FORNO, 2001). Uma vez que um incidente é detectado ele deve ser tratado, de modo a reduzir o dano e retornar o sistema ao seu estado original (BISHOP, 2002; WYK; FORNO, 2001).

Na última década, o número de atividades maliciosas e incidentes de segurança em redes conectadas à Internet têm crescido continuamente (CERT/CC, 2008; CERT.br, 2008a; ALLMAN et al., 2007), trazendo novos desafios no tratamento dos ataques, os quais têm utilizado ferramentas automatizadas e cada vez mais sofisticadas (HOUSEHOLDER et al., 2002; BAILEY et al., 2005). Esta evolução nas técnicas de ataque, que incluem a utilização de criptografia e de técnicas de evasão de detecção, tem causado a diminuição na eficácia dos sistemas IDS e um aumento no número de falsos positivos registrados (PROVOS; HOLZ, 2007). Estudos também mostram que há grandes diferenças entre as atividades maliciosas vistas em redes distintas (PANG et al., 2004; COOKE et al., 2004) e que, para obter uma noção melhor do estado atual da segurança na Internet, é necessário correlacionar eventos de diversas fontes, sejam elas sensores coletando dados sobre ataques ou informações sobre incidentes ocorridos (ALLMAN et al., 2006; YEGNESWARAN et al., 2005).

Como resultado deste cenário, tem surgido, também, a necessidade de novas fontes de informações sobre ataques ocorridos, e de maior troca de informações e correlação entre atividades observadas por pequisadores, operadores de redes e grupos que tratam incidentes de segurança (COOKE et al., 2004; ALLMAN et al., 2006; YEGNESWARAN et al., 2005; ARVIDSSON et al., 2001; DANYLIW et al., 2007). Para caracterizar e monitorar atividades maliciosas na Internet, um dos métodos utilizados tem sido a colocação de sensores em espaços de endereçamento não utilizados (COOKE et al., 2004; BAILEY et al., 2005). Dentre os tipos de sensores empregados para este fim destacam-se os *honeypots* (YEGNESWARAN et al., 2005; PANG et al., 2004; PROVOS; HOLZ, 2007; HOEPERS et al., 2005).

Honeypots são recursos de segurança, monitorados extensivamente, cujo valor reside em seu uso ilícito ou não autorizado. Nos casos em que o *honeypot* possa ser comprometido e controlado pelo atacante são implantadas contramedidas que impeçam seu uso indiscriminado e que monitorem as atividades (PROVOS; HOLZ, 2007). *Honeypots* e *honeynets* são hoje recursos de ponta utilizadas tanto para compreender as atividades dos atacantes quanto para auxiliar em atividades como detecção de intrusão (PROVOS; HOLZ, 2007). É consenso que são capazes de coletar informações valiosas sobre os tipos de ataques ocorrendo na Internet e de auxiliar o tratamento de incidentes de segurança, além de prover detalhes sobre o modo de atuação e as motivações dos atacantes (SPITZNER, 2002; HONEYNET, 2004; FILHO et al., ; HOEPERS et al., ; YEGNESWARAN et al., 2005). Apesar do grande interesse da comunidade de segurança por *honeypots*, eles ainda não atingiram o auge de sua maturidade, tendo um vasto campo a ser explorado (HONEYNET, 2004).

A maior parte dos trabalhos na área tem se concentrado em validar e desenvolver ferramentas e metodologias para criação e manutenção de *honeypots* (HONEYNET, 2004; PROVOS; HOLZ, 2007). Apesar de existirem trabalhos na área de análise de dados, eles concentram-se na obtenção de estatísticas, visualização ou correlação de dados em um conjunto de *honeypots* que utilizem tecnologias similares, ou em dados de uma implementação particular de *honeynet*, como será discutido em mais detalhes no Capítulo 2.

É importante, contudo, realizar uma análise mais completa e robusta do tráfego observado entre *honeypots* e *honeynets* utilizando diferentes tecnologias e implantados em diferentes ambientes espalhados ao redor do mundo. Além disso, é igualmente importante correlacionar esses dados com dados capturados em redes de produção ou por outras tecnologias de monitoração de redes. Tal análise forneceria uma compreensão mais profunda da distribuição dos ataques e como eles se relacionam (HONEYNET, 2004; YEGNESWARAN et al., 2005). Porém, para permitir a correlação de informações de diferentes implementações de *honeypots* e *honeynets* é necessária não somente a existência de um sistema de coleta e análise de informações, mas também de um formato padrão para a representação destas informações. A existência de um formato padrão facilitaria a troca dos dados, pois possibilitaria a construção de ferramentas únicas para automatizar a geração e o recebimento de informações de múltiplas fontes. Outra questão importante é que as tecnologias utilizadas para a implementação de *honeypots* evoluem de maneira rápida. O desenvolvimento de

ferramentas específicas para determinadas tecnologias, implica no desenvolvimento de novas versões ou na reescrita dessas ferramentas a cada evolução da tecnologia de *honeypots*.

Há também um ponto que inibe a troca de informações: o potencial de dados sensíveis sobre uma rede ou instituição serem compartilhados com terceiros. Qualquer sistema de troca ou correlação de dados precisa levar estas questões em conta e habilitar a representação de informações em diferentes níveis de detalhamento. Alguns dos sistemas de análise discutidos no Capítulo 2 necessitam ter acesso aos dados completos para produzir resultados. Porém, a maior parte das instituições possuem políticas de privacidade e divulgação de informações que impedem que dados referentes ao ambiente de produção sejam compartilhados. Dados normalmente julgados sensíveis são os endereços de rede onde estejam instalados *honeypots*, a arquitetura de sua rede interna, a proximidade de *honeypots* a redes de produção de alto valor e dados derivados de incidentes de segurança ocorridos na instituição. Desta forma, uma instituição deve possuir a liberdade para implantar seus *honeypots* e *honeynets* de maneira independente, utilizando as tecnologias e padrões que julgar apropriados, sem que isso a impossibilite de utilizar uma ferramenta de análise, ou de compartilhar dados de forma sanitizada.

De forma a reduzir os problemas discutidos, este trabalho apresenta as seguintes contribuições, descritas em detalhes nos Capítulos 3 e 4:

- Foi definido o HIDEF (*Honeypots Information and Data Exchange Format*), um formato para representação e troca de dados e informações produzidas por instituições que estejam utilizando *honeypots* e *honeynets*. Este formato procura resolver o problema da falta de interoperabilidade entre diferentes tecnologias de *honeypots*, mantendo ao mesmo tempo compatibilidade com formatos como o IODEF (*Incident Object Description and Exchange Format*) e o IDMEF (*The Intrusion Detection Message Exchange Format*), descritos no Capítulo 2.

- Foi definido um modelo de sistema, o HIDEAS (*Honeypots Information and Data Exchange and Analysis System*), para troca das informações representadas no formato HIDEF.

- Foi implementado um protótipo do sistema HIDEAS, para testar e validar

a viabilidade de utilização do sistema e do formato HIDEF em um ambiente de produção.

- Foi feito um estudo de caso, coletando informações em formato HIDEF provindas de *honeypots* utilizando tecnologias diferentes e informações em formato IODEF, a partir de notificações de incidentes de segurança.

Os demais Capítulos desta tese estão organizados como segue. O Capítulo 2 apresentará conceitos sobre tratamento de incidentes e sobre *honeypots*, importantes para a melhor compreensão dos outros tópicos discutidos. Ainda no Capítulo 2 serão apresentados trabalhos relacionados nas áreas de análise de dados de *honeypots* e uso de formatos padrão para troca de dados sobre ataques em redes de computadores. No Capítulo 3 será descrito o formato de dados HIDEF e o sistema HIDEAS, contribuição desta tese para a área de correlação de dados sobre *honeypots* e sobre ataques em redes de computadores. Um estudo de caso, com os resultados da análise dos dados coletados, será visto no Capítulo 4. O Capítulo 5 contém as conclusões do trabalho desenvolvido e discussões sobre futuros desenvolvimentos dessa linha de pesquisa. O Apêndice A apresenta o detalhamento do modelo de dados do IODEF. O *Schema* do modelo proposto, HIDEF, é apresentado no Apêndice B. No Apêndice C é discutida a modelagem do bando de dados do protótipo implementado. Os artigos publicados em revista indexada e em congressos estão no Apêndice D.

2 CONCEITOS, FUNDAMENTOS E ESTADO DA ARTE

Neste Capítulo são introduzidos os conceitos de incidente de segurança, tratamento de incidentes, grupos de tratamento de incidentes e sua importância para o funcionamento da Internet nos dias de hoje. Também neste Capítulo serão apresentados em mais detalhes os conceitos de *honeypots* e *honeynets*, seus tipos, aplicações e características dos dados por eles coletados. A seguir, este Capítulo apresenta uma discussão sobre os trabalhos atuais relacionados com análise e correlação de dados de *honeypots* e uma visão geral da utilização de XML (*Extensible Markup Language*) para definir formatos de dados para a descrição de ataques em redes de computadores. Ao final, serão discutidas deficiências nos modelos vistos, principalmente no que diz respeito à representação e correlação de dados de *honeypots* que utilizem tecnologias diferentes.

2.1 Tratamento de Incidentes de Segurança em Computadores e Redes

Nesta seção, serão discutidos o conceito de Incidente de Segurança e os conceitos de Resposta, Tratamento e Gerenciamento de Incidentes. Além disso, será discorrido sobre a nomenclatura utilizada atualmente nesta área para a definição dos elementos de um ataque e de um evento de segurança. Esta nomenclatura é utilizada pelos padrões discutidos na Seção 2.4 e no modelo proposto no Capítulo 3.

2.1.1 Incidente de Segurança

De maneira geral, um incidente de segurança é definido como qualquer evento adverso, confirmado ou sob suspeita, relacionado à segurança dos sistemas de computação ou das redes de computadores, no qual se tem a percepção de risco ou no qual as informações de uma entidade estão realmente em risco (BROWNLEE; GUTTMAN, 1998; BROWN et al., 2003; CERT/CC, 2006; WYK; FORNO, 2001). De maneira complementar, alguns autores afirmam que é também uma violação, ou uma ameaça iminente de violação, de uma política de segurança, explícita ou implícita (SHIREY, 2007; GRANCE et al., 2004; CERT/CC, 2006; ARVIDSSON et al., 2001).

Um incidente normalmente engloba um grupo de ataques que podem ser distinguidos de outros por causa das peculiaridades dos atacantes, ataques, objetivos, *sites* e período dos ataques (HOWARD; LONGSTAFF, 1998; HOWARD, 1997; ARVIDSSON et al., 2001). Exemplos de incidentes incluem atividades como: tentativas de ganhar acesso não autorizado a sistemas ou a seus dados; interrupção indesejada ou negação de

serviço; uso não autorizado de um sistema para processamento ou armazenamento de dados; modificações nas características de *hardware*, *firmware* ou *software* de um sistema, sem o conhecimento, instruções ou consentimento prévio do dono do sistema; disseminação de códigos maliciosos; uso não apropriado; invasões de sistemas de computadores via rede; varreduras, via rede, à procura de vulnerabilidades em um grupo de sistemas computacionais (*scans*); a página de uma empresa ser desfigurada (CERT/CC, 2006; GRANCE et al., 2004; BROWN et al., 2003; WYK; FORNO, 2001).

Vale ressaltar que é consenso entre os autores que a definição de incidente de segurança depende de cada instituição e, normalmente, isso deverá ser definido em sua política de segurança.

Neste texto, todas as vezes que o termo incidente, ou incidente de segurança, for utilizado ele englobará tanto tentativas de ataques (como varreduras), quanto ataques bem sucedidos (como invasões, negações de serviço, comprometimentos via códigos maliciosos, entre outros).

2.1.1.1 Elementos de um Incidente

Os incidentes de segurança são compostos por diversos elementos, que podem diferir entre redes e instituições. Em 1998 um projeto conjunto do CERT/CC e do *Sandia National Laboratories* procurou definir uma linguagem comum para incidentes de segurança, apresentada no relatório técnico "*A Common Language for Computer Security Incidents*" (HOWARD; LONGSTAFF, 1998). Este relatório não é um dicionário de termos, mas sim uma compilação de um conjunto mínimo de termos, juntamente com uma estrutura indicando o relacionamento entre estes termos.

Desde então este relatório técnico tem servido como base para a definição de termos relacionados com incidentes de segurança, destacando-se a utilização dos termos deste relatório pela RFC 3067 (ARVIDSSON et al., 2001), que descreve os principais requerimentos para um padrão como o IODEF.

De relevância especial para o presente trabalho são os termos apresentados na Figura 2.1. Uma definição destes termos, de acordo com (HOWARD; LONGSTAFF, 1998) é a que segue:

evento – uma **ação** direcionada a um **alvo** com a intenção de causar uma mudança

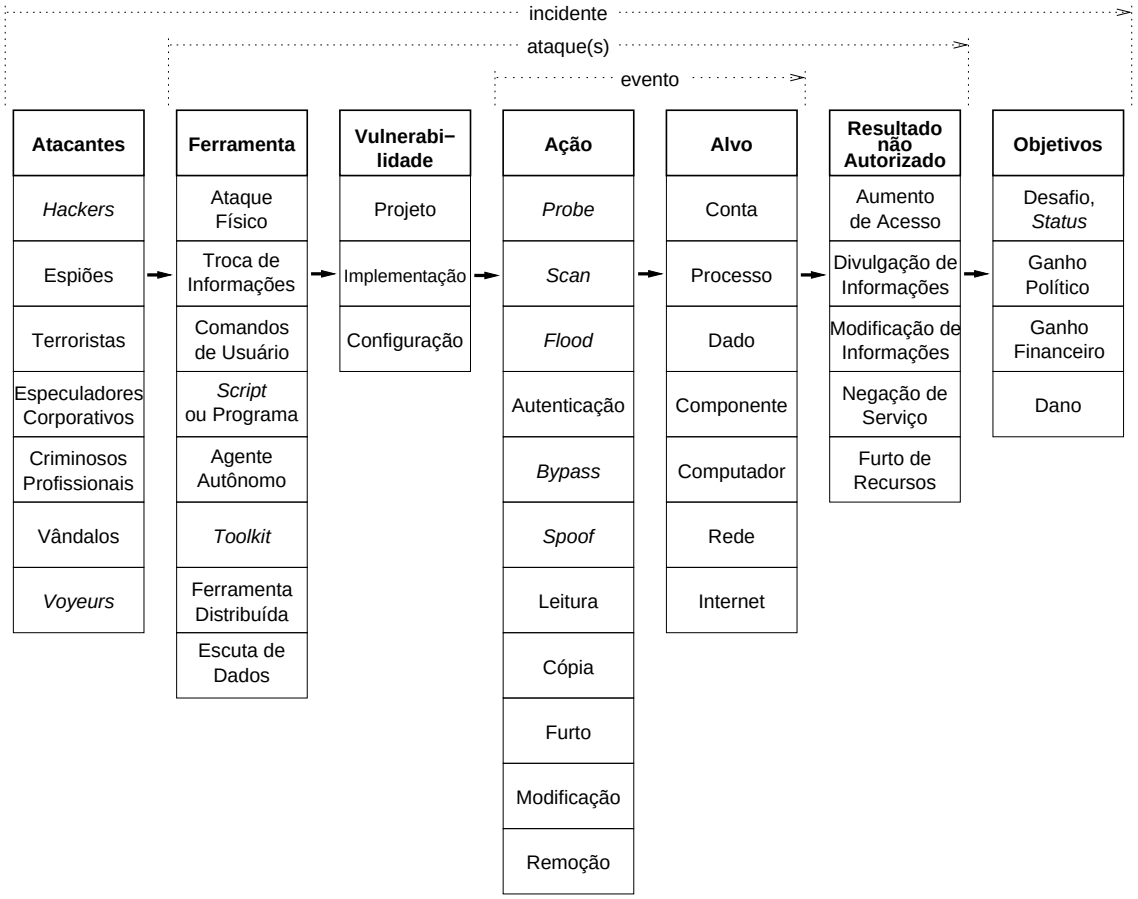

Figura 2.1 - Componentes e interações em um incidente de segurança.

Fonte: Howard e Longstaff (1998) (tradução do autor).

no estado do alvo. Vale enfatizar que nem sempre a ação tem sucesso em modificar o estado do alvo.

ação – um passo tomado, por um usuário ou processo, para alcançar um resultado.

alvo – uma entidade lógica de uma rede ou de um computador (conta, processo ou dados) ou uma entidade física (componente, computador ou rede).

ataque – uma série de passos tomados por um **atacante** para alcançar um **resultado não autorizado**.

ferramenta – um meio de explorar uma **vulnerabilidade** em um computador ou rede.

vulnerabilidade – uma fraqueza em um sistema, permitindo **ação** não autorizada.

resultado não autorizado – uma conseqüência não autorizada de um **evento**.

incidente – um grupo de **ataques** que podem ser distinguidos de outros por causa das peculiaridades dos **atacantes**, **ataques**, **objetivos**, *sites* e período dos ataques.

atacante – um indivíduo que tenta realizar um ou mais **ataques**, a fim de atingir um **objetivo**.

objetivo – o propósito atingido pelo atacante como resultado das ações realizadas.

2.1.2 Gerenciamento, Tratamento e Resposta a Incidentes

Uma vez que um incidente tenha sido detectado é necessário que ele seja devidamente tratado, para que seja possível compreender e erradicar o problema e evitar que ocorra novamente.

Os termos "tratamento de incidentes" (*incident handling*) e "resposta a incidentes" (*incident response*) são normalmente usados de forma intercambiável e, em geral, em um sentido amplo, englobando um grande processo que envolve mais do que apenas prover uma resposta técnica a um incidente ocorrido.

De maneira mais aprofundada, este processo, que pode ser visto na Figura 2.2, está subdividido em três grandes áreas (ALBERTS et al., 2004):

Gerenciamento de Incidentes: processo de controle e administração das tarefas associadas com incidentes de segurança. Inclui o processo de tratamento de incidentes, assim como processos para proteção e preparação, incluindo a definição ou revisão de políticas de segurança e de tratamento de incidentes. Pode incluir, também, ações como treinamento, análise de vulnerabilidades e análise de riscos.

Tratamento de Incidentes: um serviço que envolve todos os processos ou tarefas associados com o tratamento dos incidentes. Este serviço inclui múltiplas funções:

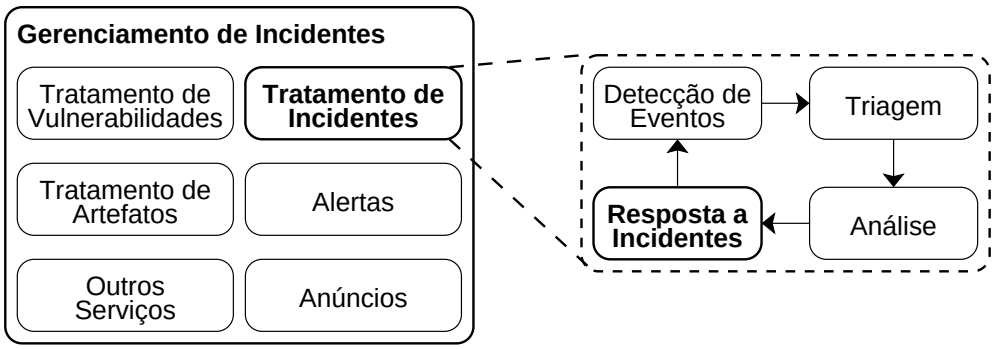

Figura 2.2 - Relacionamento entre gerenciamento, tratamento e resposta a incidentes.

Fonte: Alberts et al. (2004) (Tradução do autor).

- deteção e notificação de eventos: a habilidade de receber e avaliar as informações sobre eventos, notificações de incidentes e alertas gerados por ferramentas;
- triagem: as ações tomadas para categorizar, priorizar e delegar a resposta a eventos e incidentes;
- análise: a tentativa de determinar o que ocorreu, qual o impacto, a ameaça ou o dano resultante; também procura determinar quais passos devem ser seguidos para a recuperação ou mitigação da ocorrência;
- resposta ao incidente: as ações que devem ser tomadas para se recuperar ou mitigar um incidente e para coordenar e disseminar informações; muitas vezes também envolve definir estratégias para evitar que outros incidentes similares ocorram.

Resposta a Incidentes: como discutido no ítem anterior, é o último passo do processo de tratamento de incidentes. Neste processo estão englobados o planejamento, coordenação e execução de qualquer ação apropriada para mitigação e recuperação de um incidente.

2.1.3 Grupos de Resposta a Incidentes de Segurança em Computadores

Para tratar efetivamente incidentes de segurança, é necessário que cada instituição, além de definir políticas e procedimentos, defina quem atuará no tratamento dos

incidentes identificados. Uma das opções é a formalização desse processo através da criação de um Grupo de Resposta a Incidentes de Segurança em Computadores (CSIRT, do inglês *Computer Security Incident Response Team*) para trabalhar de forma dedicada (ALLEN, 2001).

Um CSIRT é uma organização ou grupo que provê serviços e suporte, a uma comunidade bem definida, para prevenção, tratamento e resposta a incidentes de segurança em computadores e redes (ALBERTS et al., 2004). A comunidade atendida (público alvo) geralmente é a entidade que o mantém, como uma empresa, um órgão governamental ou uma organização acadêmica. Mas um CSIRT também pode atender a uma comunidade maior, como um país, uma rede de pesquisas ou clientes externos (CERT/CC, 2006).

Para ser considerado um CSIRT um grupo deve: prover mecanismos seguros para o recebimento de notificações de incidentes; prover assistência sobre tratamento de incidentes ao seu público alvo; e disseminar informações relevantes àqueles a quem atende e aos envolvidos em incidentes de segurança (ALBERTS et al., 2004).

Durante o processo de tratamento de incidentes é extremamente comum um CSIRT trocar informações com outros CSIRTs, de modo a melhor compreender os ataques e prover informações para outros que estejam envolvidos no incidente. Este processo é conhecido como coordenação da resposta ao incidente (BROWN et al., 2003). Esta coordenação também envolve coletar e disseminar informações sobre o ataque, que incluem traços de rede, *logs* de servidores, endereços IP e *sites* envolvidos, fluxos de rede (*flows*), dados de contato de administradores de redes e de outros CSIRTs, informações sobre a vulnerabilidade explorada, características do sistema afetado, entre outras.

Para facilitar este processo de coordenação, diversos CSIRTs desenvolveram um formato comum para os dados normalmente trocados entre os grupos, facilitando a automação do processo e o cruzamento de dados entre diversos incidentes diferentes. Este trabalho deu origem ao padrão descrito na RFC 5070: "*Incident Object Description and Exchange Format*", discutido em mais detalhes na Seção 2.4.3.

2.2 *Honeypots* e *Honeynets*

Para caracterizar e monitorar atividades maliciosas na Internet, um dos métodos utilizados tem sido a colocação de sensores em espaços de endereçamento não utili-

zados (COOKE et al., 2004; BAILEY et al., 2005). Dentre os tipos de sensores empregados para este fim destacam-se os *honeypots* (YEGNESWARAN et al., 2005; PANG et al., 2004; PROVOS; HOLZ, 2007; HOEPERS et al., 2005).

Honeypots são recursos de segurança especialmente configurados para coletar informações sobre ataques e cujo valor reside em serem sondados, atacados ou comprometidos (SPITZNER, 2002; PROVOS; HOLZ, 2007). Em geral eles são computadores especialmente configurados, de modo a registrar todas as atividades ocorridas no decorrer de sondagens, ataques e invasões a sistemas de informação ou redes de computadores (PROVOS, 2004).

Uma vez que um *honeypot* não é um sistema de produção, todo o tráfego a ele direcionado é, *a priori*, malicioso ou anômalo. Este fato faz com que os dados coletados por *honeypots* sejam valiosos, pois, ao contrário dos dados coletados por sistemas IDS, dificilmente contém falsos positivos e permitem a fácil extração de assinaturas de atividades maliciosas (SPITZNER, 2002).

2.2.1 Histórico

A primeira experiência documentada sobre a implementação de mecanismos de acompanhamento das atividades de invasores data de 1988, quando Clifford Stoll tornou pública a história da invasão ocorrida nos sistemas do *Lawrence Berkeley Laboratory* (LBL). Ao constatar a invasão, ao invés de fechar as portas de acesso para o invasor, ele tomou a decisão de acompanhar e registrar todos os seus passos. Este acompanhamento revelou não só a origem do ataque, mas também os motivos do atacante (STOLL, 1988; STOLL, 1989).

Em 1992 foi publicado por Bill Cheswick um artigo descrevendo o acompanhamento de uma invasão em um sistema da AT&T que havia sido projetado especificamente para ser invadido (CHESWICK, 1992). No mesmo ano Steven Bellovin, que também participou deste projeto, publicou um artigo descrevendo o desenvolvimento das ferramentas que foram utilizadas tanto como armadilhas quanto para capturar as ações do invasor (BELLOVIN, 1992).

Em 1998 surgiu o termo *honeypot* sendo empregado para definir um recurso de segurança preparado especificamente para ser sondado, atacado ou comprometido e para registrar essas atividades (SPITZNER; RANUM, 2002). Na mesma época surgiu a primeira ferramenta de código aberto, o *Deception Toolkit*, cujo objetivo era

explicitamente iludir atacantes, desenvolvida por Fred Cohen (COHEN, 1998).

Em 1999 um grupo de pesquisadores e profissionais da área de segurança criou uma rede especificamente projetada para ser comprometida, chamada de *honeynet*, dando início ao *Honeynet Project*, cujo objetivo é revelar as ferramentas, táticas e motivações dos invasores (HONEYNET, 2004). Em janeiro de 2002, o *Honeynet Project* fundou a *Honeynet Research Alliance*, um fórum de participação restrita a instituições fazendo pesquisa na área de *honeypots* e *honeynets*, onde estas instituições podem trocar informações e envolver-se em projetos de cooperação (HONEYNET, 2004). Desde então, o número de instituições realizando pesquisas nesta área tem crescido, expandindo a tecnologia e suas aplicações.

Em 2007 a *Honeynet Research Alliance* foi reestruturada, de forma que o *Honeynet Project* passou a ser uma organização internacional, sem fins lucrativos, e os membros da aliança passaram a integrar Capítulos do *Honeynet Project* (HONEYNET, 2008).

2.2.2 Tipos de *Honeypots*

Existem basicamente dois tipos de *honeypots*: os de baixa interatividade e os de alta interatividade (SPITZNER, 2002; PROVOS; HOLZ, 2007).

Os *honeypots* de baixa interatividade são configurados de modo a apenas emular determinados sistemas operacionais e serviços. Quando um atacante interage com um destes *honeypots*, ele não está interagindo diretamente com o sistema, mas sim com um programa ou conjunto de programas que emulam suas características, como sistema operacional, versões de aplicativos, vulnerabilidades, etc. Deste modo, este tipo de *honeypot* não é comprometido pelo atacante, mas consegue capturar diversas informações sobre o tipo de ataque sendo perpetrado, tendências em varreduras, novos códigos maliciosos automatizados, entre outros (PROVOS; HOLZ, 2007; HOEPERS et al., 2005; SPITZNER, 2003).

Já os *honeypots* de alta interatividade são sistemas reais, normalmente colocados em uma rede com mecanismos de contenção de tráfego malicioso, conhecida como *honeynet*. Estes sistemas, uma vez atacados, podem ser comprometidos, sendo que o invasor possui total controle sobre a máquina. Neste tipo de *honeypot* é possível observar o invasor instalando ferramentas, tentando conduzir ataques e modificando a configuração do sistema (HONEYNET, 2004; PROVOS; HOLZ, 2007; FILHO et al.,).

2.2.3 Tecnologias de *Honeypots*

Existem inúmeras tecnologias disponíveis para a implantação de *honeypots*, tanto de baixa quanto de alta interatividade. Algumas destas tecnologias, denominadas de *honeyclients*, são destinadas a coletar dados sobre ameaças e ataques contra aplicações clientes de rede, como navegadores e leitores de *e-mails*. Estas tecnologias ainda estão em fase experimental e não têm sido amplamente utilizadas. Deste modo, nesta seção, serão discutidas brevemente as tecnologias mais maduras e difundidas para implantação de *honeypots* que emulem sistemas operacionais e aplicativos utilizados em servidores na Internet.

Para a implementação de *honeypots* de baixa interatividade os *softwares* mais difundidos são o Honeyd e o Nepenthes (PROVOS; HOLZ, 2007). O Honeyd é um *framework* que permite a emulação de centenas de sistemas em diferentes endereços IP (*Internet Protocol*). Para cada endereço IP é possível especificar o Sistema Operacional (SO) a ser emulado e as aplicações e serviços emulados em cada um. Para a emulação das aplicações é possível utilizar subsistemas nativos do Honeyd ou então programas externos (PROVOS; HOLZ, 2007; PROVOS, 2004). O Honeyd pode ser executado em diferentes SOs, sendo que o formato de alguns dos registros de eventos (*logs*) são dependentes de SO. O Nepenthes, por sua vez, é um *honeypot* dedicado a coletar códigos maliciosos que se propagam automaticamente, como *worms* e *bots* (PROVOS; HOLZ, 2007; GÖBEL et al., 2006). Ele emula o comportamento de um *software* vulnerável, decodifica o ataque e obtém uma cópia do código malicioso (GÖBEL et al., 2006).

Os *honeypots* de alta interatividade, como são sistemas reais implantados em ambientes controlados ou em *honeynets*, possuem uma complexidade maior do ponto de vista de possíveis tecnologias adotadas. Como cada *honeypot* pode ter um SO diferente, as informações disponíveis para análise após um comprometimento são as mais diversas, envolvendo registros de eventos do SO e de aplicações, estados de processos, informações do *kernel*, arquivos inseridos pelo invasor, entre outros dados. Com relação aos mecanismos de controle e captura de tráfego gerado por *honeypots* de alta interatividade, eles podem ser implementados com diversas tecnologias. O mais difundido é o *honeywall*, um conjunto de ferramentas integradas que restringe a saída de tráfego malicioso gerado por invasores, impedindo ataques a redes de terceiros, mas permitindo a interação do invasor com o *honeypot* (HONEYNET, 2004; PROVOS; HOLZ, 2007). O *honeywall* também possui mecanismos para captura do

tráfego gerado pelos atacantes e formas de agregar e fazer análises preliminares nos dados coletados em uma *honeynet* específica.

2.2.4 Características dos Dados de *Honeypots*

Independente da tecnologia utilizada na implementação, dois conjuntos de informações estão associados aos *honeypots*: informações administrativas e dados coletados. Para poder fazer uma correta avaliação sobre as tendências de ataques e, principalmente, para poder cruzar informações entre diversos *honeypots* instalados em redes distintas, é necessário que seja possível armazenar estas informações e correlacioná-las com aquelas coletadas por diversos grupos que estejam fazendo pesquisa na área de tecnologias de *honeypots* (HONEYNET, 2004).

As seções a seguir discorrerão um pouco mais sobre cada um destes conjuntos de informações e sua características.

2.2.4.1 Informações Administrativas

As informações administrativas são informações que dizem respeito à tecnologia utilizada para implementar o *honeypot*, à sua localização e particularidades de configuração da rede em que ele se encontra. Exemplos desse tipo de informação são:

- tipo de *honeypot*, se de baixa ou alta interatividade;
- sistema operacional base, onde as soluções e emuladores são executados em *honeypots* de baixa interatividade ou o sistema real de um *honeypot* de alta interatividade;
- em caso de *honeypots* de baixa interatividade, qual a solução adotada para emulação de sistemas operacionais e aplicativos, por exemplo: Honeyd, Nepenthes, programas desenvolvidos localmente ou outras soluções;
- para cada sistema, real ou emulado, também é necessário registrar serviços de rede e aplicativos sendo executados ou emulados; em caso de sistemas reais, é preciso também armazenar a versão do *software* e as correções de segurança aplicadas;
- em *honeypots* de baixa interatividade, mesmo que não exista um aplicativo sendo emulado, o *honeypot* pode ser configurado para deixar determinadas

portas em estado aberto (LISTEN), sendo importante registrar para cada sistema emulado, quais as portas TCP, UDP e ICMP que estejam em estado aberto.

- endereços de rede que estão sendo usados em uma *honeynet*, por um *honeypot* de alta interatividade ou emulados em um *honeypot* de baixa interatividade, como: bloco de rede (CIDR), endereços individuais e *Autonomous System Number*;

- filtros existentes na borda da rede onde está o *honeypot*, que impeçam determinados tipos de tráfego de chegarem ao *honeypot*;

- caso os dados compartilhados já sejam sanitizados, de modo a não revelar a localização do sensor, é importante que um mapeamento entre os dados reais e os valores usados para sanitização sejam registrados; essa informação pode ser provida para algumas instituições com quem se tenha uma relação de confiança ou omitida em casos em que os dados sejam compartilhados mais abertamente;

- pessoa ou organização responsável por determinado *honeypot*, sua localização física e *hardware* utilizado.

Estas informações são importantes para permitir que, no processo de análise e correlação de dados, o analista possa determinar o nível de detalhe da informação coletada por um determinado *honeypot*. Alguns exemplos são: identificar se determinado tráfego não foi visto por um *honeypot* por haver um filtro de rede ou por conta do tráfego realmente não ter ocorrido na rede do *honeypot*; verificar se o fato de um emulador ser executado em um *honeypot*, e não em outro, pode explicar diferenças nos detalhes do tráfego observado.

Estas informações não necessariamente são utilizadas para todas as análises, mas são importantes caso esteja sendo correlacionado um evento de interesse e apareçam discrepâncias entre os dados coletados por diferentes sensores. Nestes casos, é possível contatar o responsável ou retirar das próprias informações registradas uma resposta para o observado.

2.2.4.2 Dados Coletados

O objetivo final da instalação de *honeypots* em redes de computadores é a coleta de dados sobre atividades maliciosas. Estes dados, apesar de serem similares aos dados observados por sistemas IDS (*Intrusion Detection System*) implantados em redes de produção, muitas vezes possuem mais detalhes. Estes detalhes são capturados pelos emuladores ou por soluções de captura de dados em *honeypots* de alta interatividade.

Exemplos dos dados capturados em *honeypots* são: tentativas de ataque recebidas; traços de rede de atividades maliciosas; códigos maliciosos que venham a ser capturados; e possíveis artefatos gerados por atividades maliciosas em *honeypots* de alta interatividade.

Além dos dados brutos, é importante registrar as análises já realizadas sobre as atividades ocorridas, como comprometimentos de *honeypots* de alta interatividade ou novos códigos maliciosos capturados em *honeypots* de baixa interatividade. Uma vez que esta análise tenha sido feita, esse registro pode ser compartilhado com outros analistas.

2.3 Trabalhos Relacionados na Área de Análise e Correlação de Dados de *Honeypots* e *Honeynets*

Nesta seção são descritos projetos e modelos existentes para análise e correlação de dados de *honeypots* e de *honeynets*. A maior parte destes estudos concentra-se em visualização e/ou correlação de dados em uma *honeynet* específica ou em conjunto de *honeypots* que utilizem tecnologias similares (HONEYNET, 2004). Também serão descritas algumas ferramentas públicas disponíveis para análise e correlação destes dados.

2.3.1 Centralização de Dados no *Honeynet Project*

O *Honeynet Project* é uma organização internacional sem fins lucrativos, formada por Capítulos ao redor do mundo (HONEYNET, 2008), reunindo instituições de diversos países, que estão fazendo pesquisa na área de *honeypots* e *honeynets* (HONEYNET, 2004). Estes Capítulos eram anteriormente conhecidos como a *Honeynet Research Alliance* (HONEYNET, 2004).

O Capítulo Global, que era conhecido anteriormente apenas como *The Honeynet*

Project (HONEYNET, 2004), mantém diversas *honeynets*, cujos dados são centralizados para posterior análise. Para que seja possível armazenar estes dados em um servidor central o Projeto definiu um conjunto de diretrizes que devem ser cumpridas na captura dos dados em todas as *honeynets*, para permitir posterior centralização dos dados (HONEYNET, 2004). Estas diretrizes são:

- nenhum dado capturado deve ser armazenado localmente nos *honeypots*;
- não deve haver poluição de dados, ou seja, não devem ser feitos testes ou haver nenhuma atividade na rede que não seja de atacantes;
- deve ser mantido um relatório com a análise de cada *honeypot* comprometido;
- todas as atividades da *honeynet* devem ser arquivadas pelo período de um ano;
- deve haver um registro padronizado com a configuração de cada *honeypot* que estiver ativo na *honeynet*;
- os dados capturados pelo *firewall*, roteador ou IDS devem ser todos armazenados em fuso horário GMT (*Greenwich Mean Time*);
- os *honeypots* podem ter horário configurado para o fuso horário local, mas devem ter os horários das informações capturadas convertidos para GMT posteriormente;
- os sistemas utilizados para capturar os dados devem possuir medidas de segurança que os protejam contra ataques, de modo a manter a integridade dos dados.

O Capítulo Global também possui um conjunto de boas práticas que são seguidas para coleta, em um servidor centralizado, dos dados capturados nas *honeynets* (HONEYNET, 2004):

- cada *honeynet* deve possuir um identificador único, com uma convenção de nome e um mapeamento que permita identificar sua localização e configuração;
- deve existir um modo seguro para transmitir os dados das *honeynets* para o servidor central, de modo a garantir a confidencialidade, a integridade e a autenticidade dos dados;

- as organizações podem sanitizar dados como: endereços IP utilizados pela *honeynet* e localização da *honeynet* em sua infra-estrutura de rede. Não devem ser sanitizados dados sobre os ataques ou sua origem;
- todos os dados vindos de uma *honeynet* devem estar em fuso horário GMT.

Uma vez que as *honeynets* estejam todas seguindo os padrões acima definidos, elas transmitem os dados para um servidor central via um túnel SSH (*Secure Shell*) (HONEYNET, 2004).

Embora seja descrito como é feita a coleta dos dados de suas *honeynets* para um servidor central, não há informações sobre como está estruturado este servidor ou qual tipo de correlação de dados é feita. A única referência à análise de dados por parte do Capítulo Global está na página *Web* do *Honeynet Project*, onde são listadas algumas ferramentas utilizadas (HONEYNET, 2008):

- `Privmsg` – *script* em Perl utilizado para extrair, de arquivos binários em formato `libpcap`[1], os diálogos dos invasores em canais IRC;
- `Sleuthkit` – ferramenta de código aberto para realização de análise forense em sistemas comprometidos;
- `WinInterrogate` – ferramenta de código aberto, para sistemas Windows, utilizada para analisar processos e sistemas de arquivos.

O *Honeynet Project* não possui um método formal para que todos os Capítulos possam compartilhar dados ou análises, mas algumas instituições enviam seus dados para um segundo servidor central, também mantido pelo *Honeynet Project*. Este servidor aceita conexões via SSH e pode manipular os seguintes dados: *logs* padrão gerados pelo IDS Snort, arquivos binários em formato `libpcap` e *logs* de *firewall* (HONEYNET, 2004).

Estas restrições permitem que somente membros que estiverem utilizando as tecnologias suportadas pelo servidor possam enviar dados. Vale notar também que, apesar de possuir maneiras de centralizar estes dados no servidor mantido pelo *Honeynet Project*, esta arquitetura não contempla a troca de informações diretamente entre

[1] `pcap` (*Packet Capture library*) é uma interface para captura de pacotes de rede, disponível em diversos sistemas operacionais; os arquivos gerados por esta interface são chamados de arquivos `pcap` ou, como são mais conhecidos em sistemas Unix, arquivos em formato `libpcap`.

as instituições. Também não há registros sobre qual tipo de análise ou correlação esteja sendo feita nestes dados.

2.3.2 *Honeynet Security Console*

O *Honeynet Security Console* (HSC) é uma ferramenta para correlacionar eventos ocorridos em uma rede ou *honeynet* local, permitindo ver e armazenar: *logs* do IDS Snort, arquivos em formato `libpcap` gerados pelo Tcpdump, *logs* de `syslog`, *logs* de *firewall* arquivados pelo `syslog` e *logs* da ferramenta Sebek (HONEYNET, 2003), utilizada para captura dos comandos executados por um invasor em um *honeypot* de alta interatividade.

O HSC foi desenvolvido por Jeff Bell, do *Florida Honeynet Project*, como um meio para correlacionar dados nos diversos formatos gerados pela *honeynet* por eles mantida. A abordagem do HSC consiste em armazenar, da forma mais simples possível, em um banco de dados SQL (MySQL) os dados capturados na *honeynet* do projeto. Após ter estes dados armazenados no banco de dados, a ferramenta HSC é responsável por entender os diferentes formatos dos dados e correlacioná-los.

A escolha desta abordagem deveu-se ao fato do projeto utilizar diversos programas já existentes para converter dados de cada um dos aplicativos para um banco de dados relacional. Com o uso destas ferramentas são criadas tabelas individuais, com tipos de dados e mapeamentos diferentes, para os dados de cada aplicação. Um exemplo disto é a definição do tipo de um endereço IP. Na tabela de *logs* do *firewall* um endereço IP é armazenado como `VARCHAR`, enquanto na tabela das informações geradas pelo Sebek um IP é do tipo `INT`. Com esta inconsistência entre os tipos nas duas tabelas a ferramenta HSC necessita não só fazer conversões entre tipos de dados, mas também fazer correlações posteriores às respostas recebidas do Banco de Dados.

Esta ferramenta não está mais sendo mantida por seu autor, mas seu histórico e última versão ainda estão disponíveis em seu *site* (ACTIVEWORX, 2007).

2.3.3 Estatísticas do Consórcio Brasileiro de *Honeypots*

O Consórcio Brasileiro de *Honeypots*, descrito em mais detalhes na Seção 4.1.1, mantém em sua página (`http://www.honeypots-alliance.org.br/stats/`) estatísticas diárias das atividades observadas nos 45 *honeypots* que fazem parte de sua

infra-estrutura (STEDING-JESSEN et al., 2008; HOEPERS et al., 2005). Os dados utilizados para gerar as estatísticas são os *logs* em formato `libpcap` gerados pelo *firewall* do Sistema OpenBSD (`pf`) (HARTMEIER, 2002), que é utilizado em todos os *honeypots* para capturar o tráfego completo.

A cada 24 horas um servidor central para coleta de dados transfere, através de conexões SSH, todos os *logs* de cada um dos *honeypots*. Uma vez que os dados de todos os *honeypots* estejam disponíveis no servidor de coleta de dados, eles são processados para que sejam transformados do formato `libpcap` para o formato de *flows*. Estes *flows* são então processados pelas ferramentas ORCA (ORCAWARE, 2006) e RRDtool (OETIKER, 2008) para geração das estatísticas e gráficos disponibilizados diariamente pelo projeto em sua página *Web*. Atualmente, estão disponíveis estatísticas sobre os seguintes dados:

- Portas TCP mais atacadas (bytes/s e pacotes/s);
- Portas UDP mais atacadas (bytes/s e pacotes/s);
- Países de origem dos ataques (bytes/s e pacotes/s);
- Sistemas Operacionais de origem dos ataques (bytes/s e pacotes/s).

2.3.4 *Honeynet Project Walleye*

Walleye é a interface gráfica via *Web* para administração do *Honeywall*, mecanismo de contenção de tráfego malicioso criado pelo *Honeynet Project* para ser utilizado como *firewall* em *honeynets* (HONEYNET, 2005).

Esta interface provê, além de mecanismos para configuração do *Honeywall*, uma interface para análise dos dados da *honeynet*. A interface permite ter uma visão geral do tráfego de entrada e saída de uma *honeynet*, ver os fluxos de dados e escolher eventos específicos a serem analisados posteriormente com outras ferramentas (HONEYNET, 2005).

2.3.5 *HoneyStats – A Statistical View Of The Recorded Activity On a Honeynet*

O laboratório *Internet Systematics Lab* do *National Center of Scientific Research "Demokritos"*, na Grécia, mantém o *Greek Honeynet Project*, que em 17 de maio

de 2006 lançou a ferramenta "HoneyStats 1.0", versão pública de uma ferramenta inicialmente desenvolvida para gerar estatísticas sobre os registros de eventos (*logs*) registrados pelo *firewall* da *honeynet* mantida por eles.

As estatísticas que podem ser obtidas são: portas TCP ou UDP mais atacadas e tendências de crescimento ou queda de ataques por porta; países de origem dos ataques e tendências por país; e atividades por localização geográfica.

A última versão pública desta ferramenta, que data de 17 de maio de 2006, está disponível em: `http://sourceforge.net/projects/honeystats/`. Um site com uma interface para teste da ferramenta com dados reais da *honeynet* deles pode ser acessado em: `http://www.honeynet.gr/`.

2.3.6 Estatísticas do *Leurrecom.org Honeypot Project*

O *French Honeynet Project* implantou nos últimos anos o projeto Leurrecom.org, que consiste na distribuição de *honeypots* de baixa interatividade em diversos países, provendo para as instituições que instalam estes *honeypots* o acesso via *Web* a todos os dados coletados (DACIER et al., 2007).

Estes dados dão origem a estatísticas dos últimos 30 dias, mantidas no site do projeto, com dados sobre: o número de ataques, distribuído por protocolo (TCP, UDP e ICMP); o número de ataques, por agrupamentos de portas; a quantidade de pacotes de *backscatter* por país, que pode ser um indicador de tráfego vindo de vítimas de um ataque DDoS; e os tipos de ataque vindos dos 10 países com maior tráfego observado. Segundo a documentação os agrupamentos são definidos com base nos seguintes parâmetros: duração das atividades, seqüencia de portas atacadas, número de máquinas virtuais atacadas, número de pacotes enviados, ordem do ataque contra os *honeypots* e conteúdo dos pacotes.

2.3.7 Estatísticas e Visualização de Dados do *Georgia Tech Honeynet Project*

O grupo de pesquisa do *Georgia Tech Honeynet Project* desenvolveu algumas ferramentas para gerar estatísticas e para visualizar os dados coletados em sua *honeynet* (GATECH, 2006; CONTI; ABDULLAH, 2004).

A ferramenta *HoneyReport* interpreta arquivos em formato `libpcap` e gera como

saída informações em formato de *flows*. Estes *flows* são analisados para gerar estatísticas sobre as atividades mais comuns na *honeynet*, como portas mais atacadas e origens dos ataques. Não há uma versão pública desta ferramenta.

Outra ferramenta, *Rumint*, gera gráficos em coordenadas paralelas, com base no tráfego observado em uma *honeynet* ou rede de produção. Esta ferramenta também usa como entrada arquivos em formato `libpcap` (CONTI; ABDULLAH, 2004). Atualmente ela está em sua versão 2.14 e é mantida pelo seu autor, Gregory Conti, podendo ser obtida via o site `http://www.rumint.org/`.

2.3.8 *HoneySnap*

HoneySnap é uma ferramenta de linha de comando que interpreta arquivos no formato `libpcap` e produz um primeiro relatório de análise, identificando eventos significativos nos dados processados. O objetivo principal é apresentar a analistas um resumo dos dados, auxiliando na determinação do foco de análise posterior. Desta forma, uma vez identificados os eventos de interesse, podem-se usar outras ferramentas para fazer uma análise mais aprofundada.

Mais detalhes sobre seu funcionamento e seu código fonte podem ser obtidos em `https://projects.honeynet.org/honeysnap/`.

2.3.9 *HIHAT – High Interaction Honeypot Analysis Toolkit*

A ferramenta *High Interaction Honeypot Analysis Toolkit* (HIHAT) permite transformar aplicações PHP arbitrárias em *honeypots* de alta interatividade baseados em *Web*. A ferramenta, disponível em `http://hihat.sourceforge.net/`, funciona oferecendo ao atacante a funcionalidade completa da aplicação, mas realiza monitoração e registros de eventos de forma extensiva.

Adicionalmente, a ferramenta possui uma interface que suporta o monitoramento do *honeypot* e a análise de dados coletados. Entre as análises que a ferramenta fornece estão: o mapeamento dos IPs de origem dos ataques para os países de origem, com a geração de uma mapa; uma correlação entre os ataques ocorridos; e diversas estatísticas sobre o tráfego observado no *honeypot*.

2.4 Formatos de Dados Sobre Ataques e Incidentes em Redes de Computadores

Nesta seção serão discutidos os pontos fortes do uso de XML (*Extensible Markup Language*) para a definição de formato de dados, em seguida serão descritos dois formatos, definidos em XML, para a representação de dados, o primeiro daqueles capturados por sistemas de detecção de intrusão e o segundo de dados referentes a incidentes de segurança em computadores.

2.4.1 Uso de XML para Definir Formatos de Dados

O XML é uma de linguagem de marcação para documentos, aprovada e recomendada pelo W3C (*World Wide Web Consortium*), que define uma sintaxe genérica e foi projetada como um formato padrão para estruturar documentos de forma simples e legível (HAROLD; MEANS, 2004; W3C, 2008a; W3C, 2004a; KIENLE, 2001).

Os dados são incluídos em documentos XML como texto puro, cercado por marcações que descrevem estes dados. Porém, ao contrário de outras linguagens de marcação, como o HTML, o XML não possui um conjunto fixo de elementos ou marcações, permitindo que desenvolvedores criem suas próprias marcações, conforme estas forem necessárias. Estas características permitem que seja geral o suficiente para representar quaisquer tipos de dados estruturados em um arquivo texto (HAROLD; MEANS, 2004; KIENLE, 2001).

Ao mesmo tempo em que é extremamente flexível, o XML possui uma gramática com regras bastante restritas a respeito do posicionamento das marcações, definição de atributos, quais nomes de elementos são válidos e como deve ser feita a interpretação das marcações.

Apesar de ser um padrão criado para representação de documentos, o XML é hoje amplamente utilizado para armazenamento e transmissão de informações a serem usadas pelos mais diversos *softwares* e sistemas (HAROLD; MEANS, 2004), e vem se tornando o formato universal para troca de dados entre aplicações (MILO et al., 2005).

Uma das maiores vantagens no uso de XML é a possibilidade de permitir a representação multiplataforma de documentos e informações. Isto ocorre pelo fato de os dados propriamente ditos serem armazenados em formato texto puro, sendo necessário apenas possuir aplicações que entendam a sintaxe do documento para que seja

possível importar e utilizar os dados em diferentes plataformas e aplicativos (HAROLD; MEANS, 2004).

Alguns conceitos de XML, importantes para a definição de formatos de dados, são (HAROLD; MEANS, 2004; FITZGERALD, 2004):

Elementos – um elemento no XML é um conjunto de informações delimitadas por uma marcação (*tag*) inicial e uma marcação final. No seguinte exemplo:

```
<pessoa idade="35">
  <nome>José</nome>
  <sobrenome>das Couves</sobrenome>
</pessoa>
```

temos um elemento `pessoa`, cujo conteúdo, por sua vez, é formado pelos elementos `nome` e `sobrenome`, sendo o que o conteúdo de `nome` é "José" e o conteúdo de `sobrenome` é "das Couves".

Atributos – um atributo é um par de nome e valor, que é anexado à marcação inicial de um elemento XML. No exemplo apresentado anteriormente uma `pessoa` possui o atributo `idade`, cujo valor é "35". Os atributos em geral são utilizados para colocar meta-dados a respeito do conteúdo de um elemento.

Namespaces – possuem dois propósitos em XML:

- possibilitar a distinção entre elementos e atributos de diferentes vocabulários, com diferentes significados e que possam ter o mesmo nome;
- agrupar todos os elementos e atributos relacionados com uma única aplicação XML, de modo que seja possível reconhecê-los facilmente.

DTD (*Document Type Definition*) – é uma sintaxe rígida, que define precisamente onde e quais elementos e entidades podem aparecer dentro do documento (HAROLD; MEANS, 2004; FITZGERALD, 2004).

Schema – é uma descrição formal do que compreende um documento XML válido. Um *schema* é bem mais restrito do que um DTD e já possui tipos de dados pré-definidos, que devem ser respeitados pelos documentos XML que utilizem um determinado *schema*. Além disso, um *schema* leva em

consideração o *namespace* ao qual pertencem os elementos, na hora de fazer a validação (VLIST, 2002; W3C, 2008b).

2.4.2 IDMEF – *Intrusion Detection Message Exchange Format*

O IDMEF é um formato de dados desenvolvido pelo grupo de trabalho "*Intrusion Detection Exchange Format Working Group (IDWG)*", do IETF, e foi publicado como padrão experimental no documento "RFC 4765: *The Intrusion Detection Message Exchange Format (IDMEF)*" (DEBAR et al., 2007).

O IDMEF define formatos e procedimentos para o compartilhamento de informações sobre detecção de intrusão. É uma representação, orientada a objetos e implementada em XML, dos dados de alerta enviados por sensores para os sistemas que analisam estes dados (DEBAR et al., 2007).

As motivações para a construção de um modelo de dados, orientado a objetos, que permitisse a interoperabilidade entre diferentes sistemas de detecção de intrusão, foram (DEBAR et al., 2007; WOOD; ERLINGER, 2007):

- As informações de alerta geradas por sistemas de detecção de intrusão são inerentemente heterogêneas;
- Um modelo orientado a objetos é naturalmente extensível através de agregações e da definição de subclasses;
- O modelo precisa ser flexível para acomodar diferentes níveis de informação, providos por diferentes tipos de dados (tráfego de rede, registros de sistemas operacionais e aplicativos, entre outros);
- Os ambientes onde os sistemas de detecção vão gerar alertas são diferentes, e um padrão precisa levar estas diferenças em conta;
- É necessário que o modelo trate o fato de, dependendo do *software* utilizado e da característica da rede, haver diferentes necessidades do ponto de vista do tipo e da quantidade de informações que serão armazenadas e compartilhadas.

Deste modo, o objetivo é que possa ser utilizado por sistemas automatizados de notificação de alertas, possibilitando o cruzamento de informações coletadas por

sistemas de diversos fabricantes, assim como por sistemas de código livre (DEBAR et al., 2007).

2.4.3 IODEF – *Incident Object Description and Exchange Format*

Os incidentes de segurança em computadores têm sido cada vez mais distribuídos, envolvendo CSIRTs diversos e que muitas vezes estão situados em mais de um país. Somado a isto, existe o fator da sobrecarga de trabalho, aumentando a necessidade de automatizar diversas tarefas diárias dos CSIRTs, como triagem de incidentes, correlação de *logs*, entre outras (WIIK et al., 2005). Esta automatização e agilidade no processamento de informações relativas aos incidentes também é mais difícil em casos em que não existe um formato comum para a descrição dos dados (ARVIDSSON et al., 2001). Para facilitar a cooperação e troca de informações entre CSIRTs, de forma a auxiliar estes grupos no processo de automatização do tratamento e acompanhamento de incidentes, foi criado o *Incident Object Description and Exchange Format* (IODEF) (DANYLIW et al., 2007).

A definição do padrão IODEF foi iniciada em um Grupo de Trabalho dentro da TERENA (*Trans-European Research and Education Networking Association*), com o propósito de definir um formato de dados comum para a descrição e troca de informações relativas a incidentes de segurança. Este trabalho deu origem ao documento "RFC 3067: *TERENA's Incident Object Description and Exchange Format Requirements*" (ARVIDSSON et al., 2001), de fevereiro de 2001 que, pela primeira vez, descreveu os requisitos básicos para a definição de um formato para troca de dados sobre incidentes de segurança entre CSIRTs.

Os trabalhos iniciados pelo TERENA foram continuados no escopo do IETF através do Grupo de Trabalho "*Extended Incident Handling Working Group (INCH)*". Ao contrário do IDMEF, que foi publicado como um formato experimental, o IODEF foi finalizado e publicado como um *Internet Standard* pelo IETF. Este padrão está descrito no documento RFC 5070: *The Incident Object Description Exchange Format*. Esta RFC descreve o modelo de dados do IODEF, define seu *Schema* e mostra exemplos de implementação XML (DANYLIW et al., 2007).

2.4.3.1 Características do Modelo de Dados

O IODEF é uma representação orientada a objetos, implementada em XML, de dados de incidentes de segurança. Ele consegue representar informações exportadas de

sistemas utilizados por CSIRTs, facilitando a subseqüente troca destas informações entre tais grupos. O seu modelo de dados permite (DANYLIW et al., 2007):

- um aumento na automação do processamento de dados relativos a incidentes de segurança;
- uma redução nos esforços de normalização de dados vindos de diferentes fontes;
- um formato comum a partir do qual se podem construir diversas ferramentas interoperáveis para o tratamento e análise dos dados.

Para melhor entender as decisões de projeto e limitações do IODEF, é necessário levar em conta as seguintes considerações (DANYLIW et al., 2007):

- o modelo foi criado para facilitar a automação e processamento de dados através de ferramentas, não para ser de fácil leitura;
- este é um modelo para transporte de informações, não sendo um modelo ótimo para armazenamento em disco, arquivamento ou mesmo processamento em memória;
- descrever um incidente com todas as definições possíveis implicaria em um modelo de dados extremamente complexo. Deste modo, o IODEF descreve apenas aqueles dados que normalmente são intercambiados entre grupos. De qualquer modo, para permitir que seja suficientemente flexível, o IODEF prevê a possibilidade de criação de estensões ao modelo, como discutido na Seção A.2.1;
- o modelo de dados suporta a criação de objetos sobre incidentes com diferentes níveis de detalhamento;
- o modelo possui mecanismos para associar cada dado com sua fonte, de modo a atender aos requisitos usuais do processo de tratamento de incidentes.

Como, muitas vezes, os incidentes que são tratados por um CSIRT foram inicialmente observados em sistemas de controle e monitoração de redes, como os Sistemas de Detecção de Intrusão (IDS, do inglês *Intrusion Detection System*), o IODEF levou em consideração o modelo do IDMEF, descrito na Seção 2.4.2. Deste modo, algumas classes do IODEF são derivadas do IDMEF.

Uma descrição completa das classes do IODEF e de seus relacionamentos é apresentada no Apêndice A.

2.5 Limitações nas Áreas de Coleta, Análise e Formatos de Dados

Como foi visto na Seção 2.3, os trabalhos existentes na área de coleta e análise de dados de *honeypots* concentram-se em visualização e/ou correlação de dados em uma *honeynet* específica ou em conjunto de *honeypots* que utilizem tecnologias similares. Isto evidencia que os esforços atuais têm se concentrado em viabilizar a análise de dados de um subconjunto de tecnologias. Esta é uma limitação muito grande, pois com a evolução das tecnologias, a maior parte dos sistemas terá que ser reescrito ou simplesmente descontinuado, como já ocorreu com alguns sistemas de análise propostos. Outro reflexo dessa dependência que as ferramentas de análise apresentam, de somente analisar dados de uma tecnologia específica, pode acabar sendo a inibição da pesquisa por novas tecnologias para implantar *honeypots*. Isto pode ocorrer caso exista um conjunto muito pequeno de ferramentas de análise independentes de tecnologias, gerando uma tendência de novos esforços procurarem se adequar às ferramentas de análise existentes.

Embora alguns dos trabalhos apresentem características interessantes e possuam resultados promissores, nenhum leva em consideração a questão de interoperabilidade para análise de dados gerados em diferentes arquiteturas ou por diferentes tecnologias. Com isso, o que vemos hoje é que temos mais de 20 grupos realizando pesquisas na área de *honeypots* (HONEYNET, 2008), mas nenhum encontrou ainda uma forma de correlacionar dados capturados por tecnologias diferentes.

Por outro lado, a comunidade de tratamento de incidentes trabalhou para a definição de um formato de dados comum, o IODEF, que já está sendo utilizado por diversos grupos. Ele tem enfoque na interoperabilidade e possui uma estrutura que permite representar adequadamente traços de rede e informações sobre alertas de sistemas de detecção de intrusão. Porém, como pôde ser visto na Seção 2.4.3, este modelo é bastante centrado em assinaturas de ataques do ponto de vista de rede e no mapeamento da relação entre incidentes de segurança.

Do ponto de vista de pesquisa e correlação entre eventos observados por *honeypots* é necessário correlacionar também informações sobre a tecnologia sendo usada, de modo a, por exemplo, guiar a comparação entre dados de um *honeypot* de alta e

um de baixa interatividade ou determinar se a presença de filtragens em uma rede implicou em diferenças nos resultados observados, como foi discutido na Seção 2.2.4. Estas características impedem que o formato IODEF possa representar com riqueza os dados necessários para uma análise mais profunda dos dados coletados pelos diferentes tipos de *honeypots*.

De modo a tratar as limitações existentes hoje na área de análise de dados de *honeypots*, este trabalho define dois elementos para formar uma infra-estrutura que permita a análise e correlação desses dados: o HIDEF (*Honeypots Information and Data Exchange Format*), um formato para troca de dados coletados e de informações sobre a arquitetura e tecnologias usadas por *honeypots*; e o HIDEAS (*Honeypots Information and Data Exchange and Analysis System*), sistema que habilita o envio e o recebimento de informaçõees e dados em formatos como o HIDEF e o IODEF.

O Capítulo 3 apresentará o HIDEF, seus requisitos e sua modelagem, e discutirá o protótipo implementado do HIDEAS. Um estudo de caso, com a utilização do sistema para receber e tratar informações de *honeypots*, em formato HIDEF, e de incidentes de segurança, em formato IODEF, é descrito no Capítulo 4.

3 INFRA-ESTRUTURA PARA TROCA E ANÁLISE DE INFORMAÇÕES DE *HONEYPOTS* E *HONEYNETS*

Como visto no Capítulo 2, a correlação de dados obtidos por *honeypots* tem sido realizada apenas em conjuntos de dados com mesmo formato e entre *honeypots* que utilizem tecnologias similares. É importante, contudo, possibilitar a análise mais completa e robusta das atividades observadas em *honeypots* e *honeynets* utilizando diferentes tecnologias e implantados em diferentes ambientes espalhados ao redor do mundo. Além disso, é igualmente importante correlacionar esses dados com dados capturados em redes de produção ou por outras tecnologias de monitoração de redes. Tal análise forneceria uma compreensão mais profunda da distribuição dos ataques e como eles se relacionam (HONEYNET, 2004; YEGNESWARAN et al., 2005).

Este trabalho apresenta uma infra-estrutura para a representação, troca e análise de informações de *honeypots* e *honeynets*. O componente principal desta infra-estrutura é o formato definido para a representação dos dados e sua subseqüente troca entre instituições interessadas. Também é parte desta infra-estrutura um sistema que facilite o envio, recebimento e correlação dos dados. Estes dois componentes são:

HIDEF – *Honeypots Information and Data Exchange Format*: formato para representação de informações sobre e dados coletados em *honeypots*, que viabiliza a automatização da troca de informações entre as diversas implementações de *honeypots* disponíveis. Este formato, descrito em detalhes na Seção 3.2, foi definido utilizando a linguagem de marcação XML;

HIDEAS – *Honeypots Information and Data Exchange and Analysis System*: sistema que habilita o envio e o recebimento de informações e dados, em formatos como o HIDEF e o IODEF, de forma a possibilitar posterior cruzamento de todas as informações armazenadas. Este sistema é descrito na Seção 3.3.

O projeto desta infra-estrutura levou em consideração os seguintes objetivos:

- representar, de forma padronizada, o conjunto de informações de interesse, discutido na Seção 2.2.4, a ser trocado por instituições que estiverem utilizando as tecnologias de *honeypots* e *honeynets*;

- permitir a coleta, armazenamento e análise das informações;
- permitir a transmissão dos dados de formas diferentes, porém todas levando em consideração a integridade e a confidencialidade dos dados;
- permitir a correlação de informações coletadas a partir de várias implementações diferentes de *honeypots* e informações relacionadas com incidentes de segurança.

Nas próximas Seções serão discutidas as decisões de projeto e os requisitos levados em conta para a definição do formato de dados. Também serão apresentados o formato HIDEF e a arquitetura do sistema proposto, HIDEAS, seguidos da descrição do protótipo implementado.

3.1 Requisitos e Decisões de Projeto

A infra-estrutura proposta leva em conta um conjunto de requisitos que tem por objetivo permitir a representação das informações relevantes para a análise e correlação de atividades capturadas por *honeypots*.

O principal requisito para a definição do formato de dados HIDEF foi a capacidade de representar informações e dados referentes a *honeypots*, conforme discorrido na Seção 2.2.4.

A representação das informações administrativas é importante para possibilitar a correta análise, permitindo que seja levada em consideração a arquitetura utilizada e o contexto em que os dados foram coletados. Para permitir esta contextualização, é necessário que o formato permita a representação de informações sobre a configuração e sobre as tecnologias utilizadas no *honeypot*, como por exemplo: qual o tipo do *honeypot*; se está conectado a uma *honeynet*; sistemas operacionais e serviços utilizados ou sendo emulados; portas TCP, UDP e ICMP que estejam em estado "aberto" no *honeypot*; e se existem filtros que impeçam determinados tipos de tráfego de chegarem ao *honeypot*, entre outras informações levantadas na Seção 2.2.4.

Igualmente importante, é a possibilidade de representar os diferentes tipos de dados capturados, os registros das atividades dos invasores e as tentativas de ataque recebidas pelo *honeypot*. Entre estes dados cabem destacar os traços de rede, os códigos maliciosos, os artefatos gerados por atividades maliciosas e as análises das atividades

ocorridas. Porém, é necessário, também, que o formato permita a troca de informações sanitizadas, ou seja, informações que omitam detalhes sensíveis referentes às redes das instituições que pretendem trocar dados coletados por seus *honeypots*.

Outro requisito que deve ser considerado é a compatibilidade com outros formatos já definidos para outros conjuntos de dados relativos à segurança de redes. Isto é importante para possibilitar a correlação entre dados de *honeypots* com aqueles capturados por sistemas de detecção de intrusão ou com dados de incidentes de segurança detectados em redes de computadores. Para atender a este requisito é necessário que o HIDEF seja compatível com o formato IODEF (DANYLIW et al., 2007), que por sua vez é compatível com o IDMEF (DEBAR et al., 2007), conforme discutido no Capítulo 2.

Para atender a estes requisitos, foram tomadas as seguintes decisões de projeto:

O *honeypot* é a entidade principal – um único formato é utilizado para *honeypots* de baixa interatividade e para *honeypots* de alta interatividade que façam parte de uma *honeynet*. Embora, fisicamente, um *honeypot* seja colocado dentro de uma *honeynet*, isto não precisa ser refletido no formato de dados. Deste modo, os dados de uma *honeynet* são vistos como complementares. Eles podem ajudar a prover contexto na análise, e as informações sobre tecnologias de controle e coleta são importantes nestes casos. Mas, ao colocar as informações sobre uma *honeynet* como complementares, isto provê ao modelo grande flexibilidade, permitindo um formato único para tipos diferentes de *honeypots* e permitindo que os dados da *honeynet* sejam omitidos se necessário for.

A classe para representação de eventos foi derivada do IODEF – para permitir o máximo de compatibilidade com o modelo do IODEF, o HIDEF utilizou a classe EventData completa (descrita em detalhes no Apêndice A). Ao trazer esta classe completa para o HIDEF, é possível manter a maior parte dos dados coletados dos *honeypots*, principalmente os eventos de rede, totalmente compatíveis com dados de incidentes representados em formato IODEF. Algumas classes filhas de EventData, uma vez que já fazem parte do modelo HIDEF, foram utilizadas para representar outros dados dos *honeypots* em contextos diferentes. Uma descrição completa de todas as classes criadas, de como a classe EventData do IODEF foi estendida e

como ela se insere no modelo HIDEF é apresentada na Seção 3.2.

Deve haver suporte para sanitização de dados – o suporte para sanitização de dados deve permitir a representação de dados em duas situações diferentes. A primeira é permitir que uma instituição troque informações omitindo completamente dados sensíveis, como os endereços IP utilizados por *honeypots* e o bloco de rede em que o sensor está inserido. Neste caso, basta compartilhar os dados coletados já sanitizados. A segunda situação envolve instituições que sanitizam os dados no momento da coleta, mas que gostariam de compartilhar com algumas instituições de confiança os dados sobre IPs sendo utilizados e blocos de rede próximos ao *honeypot*. O modelo cobre as duas situações, ao prover uma classe para armazenar especificamente o mapeamento entre sanitizações presentes nos dados coletados e seu real valor.

Informações administrativas em diferentes graus de detalhamento – o modelo precisa permitir desde uma representação detalhada das informações administrativas descritas na Seção 2.2.4, até uma descrição minimalista. O nível de detalhamento necessário deve ser definido pela instituição que está compartilhando os dados, tendo como base suas políticas de classificação e de compartilhamento de informações.

Segurança na transmissão deve ser provida por protocolos externos – a segurança na transmissão dos dados é essencial, mas não deve ser implementada como parte interna do sistema. Esta dissociação entre a troca e a análise de dados, da segurança e dos protocolos usados para transmissão, permite reduzir a complexidade do sistema e empregar protocolos de comunicação segura em uso pela comunidade para outros fins. Dessa forma, o sistema pode aproveitar a evolução nas tecnologias de transmissão segura sem que isso exija uma alteração em seu projeto ou programação.

3.2 HIDEF – *Honeypots Information and Data Exchange Format*

Esta Seção apresenta um formato para representação e troca de dados e informações coletados em *honeypots*, o HIDEF – *Honeypots Information and Data Exchange Format*. São descritos o modelo de dados definido, incluindo os tipos de dados, e as maneiras previstas para estender este modelo.

3.2.1 Modelo de Dados

O HIDEF é uma representação, orientada a objetos, de informações relativas à configuração e tecnologia usadas por *honeypots* e de dados por eles capturados. As classes do HIDEF representam elementos XML e os seus atributos representam atributos dos elementos XML. O *Schema* XML completo do HIDEF, incluindo as definições de tipos de dados aceitos, classes e restrições, está no Apêndice B.

Na definição das classes e dos atributos, sempre que possível, foram utilizados tipos de dados nativos do XML Schema (W3C, 2004b). Porém, para permitir a representação de dados mais específicos, alguns tipos de dados próprios foram definidos. Eles foram todos derivados do IODEF para permitir a compatibilidade entre os formatos de dados. Os tipos derivados do IODEF estão descritos no Apêndice A, Seção A.1.

Ao longo desta seção, as seguintes convenções foram utilizadas: as classes representadas em cinza nas figuras são derivadas do Modelo IODEF; os nomes e atributos de todas as classes são referenciados ao longo do texto em fonte sem serifa.

Cada documento HIDEF é uma instância da classe HIDEF-Document, podendo esta instância ser composta por uma ou mais instâncias da classe Honeypot, como pode ser visto na Figura 3.1. A classe HIDEF-Document possui três atributos: version, que representa a versão do modelo HIDEF sendo utilizada; lang, que representa o idioma do documento de acordo com os valores definidos em (PHILLIPS; DAVIS, 2006); e instructions, um campo reservado para a descrição de instruções de processamento, sendo que sua semântica deve ser definida pelas instituições que estejam trocando informações.

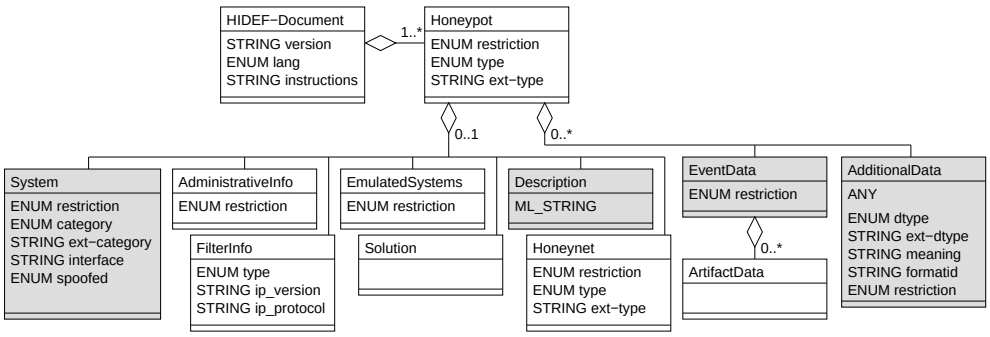

Figura 3.1 - Classe HIDEF-Document e classes agregadas.

A classe Honeypot fornece uma representação dos diversos componentes que estão relacionados com um conjunto de dados relativo a um determinado *honeypot*. Ela possui os seguintes atributos: restriction, type e ext-type. O atributo restriction, por ser também usado pelo IODEF, terá no HIDEF o mesmo significado e os mesmos possíveis valores. Este atributo indica quais as regras de divulgação de informações que devem ser observadas por quem receber o documento. Este atributo é herdado por todas as classes filhas de Honeypot. Algumas dessas classes também possuem o seu próprio atributo restriction e, nestes casos, é possível especificar novos valores. Os valores possíveis são: public, need-to-know, private e default, onde o valor default significa que deve ser utilizada uma política pré-acordada entre as partes. O atributo type determina qual o tipo do *honeypot* sendo representado, pode ter o valor low-interaction ou high-interaction. O atributo ext-type foi definido como um meio de estender o atributo type. Mais detalhes sobre como estender este e outros tipos similares são discutidos na Seção 3.2.1.1.

Cada instância da classe Honeypot pode ter zero ou uma instância das seguintes classes: System, AdministrativeInfo, EmulatedSystems, Description, FilterInfo, Solution e Honeynet. Estas classes representam as informações sobre as tecnologias utilizadas e o contexto em que os dados foram coletados. Para representar os diferentes tipos de dados capturados, os registros das atividades dos invasores e as tentativas de ataques recebidas, uma instância da classe Honeypot pode ter zero ou mais instâncias das classes EventData e AdditionalData. Estas classes e seus relacionamentos com a classe Honeypot podem ser vistos na Figura 3.1.

As classes System, Description, AdditionalData e EventData são derivadas do IODEF e tem a descrição completa de seus atributos e classes agregadas no Apêndice A. No contexto do HIDEF a classe System, quando filha da classe Honeypot, armazenará informações sobre o sistema do *honeypot*, incluindo sistema operacional, serviços, aplicativos, endereço IP versão 4 ou 6, entre outros. A classe Description pode conter uma descrição geral do *honeypot* e comentários que a instituição autora do documento HIDEF julgar relevantes. A classe AdditionalData será discutida na Seção 3.2.1.1.

A classe EventData armazenará os dados sobre as atividades maliciosas capturadas pelo *honeypot*. Esta classe, trazida do IODEF, permite representar, entre outros dados, os tempos de início e fim de uma atividade, traços de rede, protocolos envolvidos, registros de eventos, métodos de ataque e avaliações sobre o impacto do tipo de ataque registrado. Porém, esta classe não representa de maneira adequada

artefatos coletados nos *honeypots*. Deste modo, essa classe foi estendida através da adição de zero ou mais instâncias da classe ArtifactData.

A classe ArtifactData, que pode ser vista na Figura 3.2, permite representar os artefatos que venham a ser coletados nos *honeypots* de baixa ou alta interatividade, juntamente com informações relativas à sua análise e captura. Uma instância desta classe possui exatamente um artefato associado, que pode ser representado por uma destas classes: Base64Item, que permite armazenar um artefato em formato *Base64*; HexItem, que permite armazenar um artefato em formato hexadecimal; e TextItem, que permite armazenar um artefato em formato texto puro. A restrição para permitir o uso de somente uma destas classes para cada instância de ArtifactData é feita no HIDEF *Schema*. A data de captura do artefato também é única por instância de ArtifactData, sendo representada por DateTime. Em Description pode ser armazenada uma descrição geral deste artefato ou de sua análise. Cada instância de ArtifactData pode ter zero ou mais instâncias das classes Hash, URL e AdditionalData. A classe Hash permite armazenar um *hash* criptográfico ou uma assinatura digital, que podem ser usados para identificar unicamente cada artefato. Seu atributo type determina qual o tipo e pode ter os valores md5, sha1, sha256, sha512, rmd160 ou pgp-signature. A classe URL pode ser utilizada caso o artefato seja um código de domínio público e tenha uma URL associada. A classe AdditionalData será discutida na Seção 3.2.1.1.

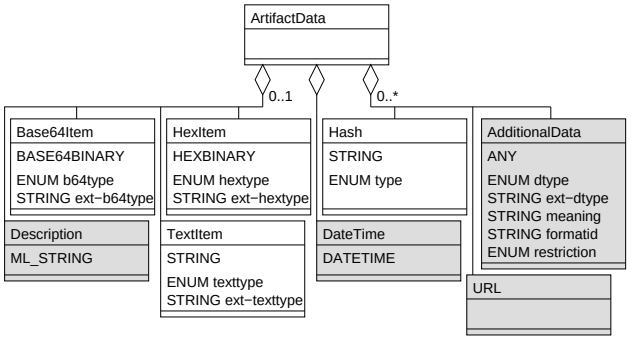

Figura 3.2 - Classe ArtifactData e classes agregadas.

A classe AdministrativeInfo, que pode ser vista na Figura 3.3, agrega as classes Hardware, EmulatedAddress, Sanitization, Contact e AdditionalData. Ela pode possuir zero ou mais instâncias destas classes, que mapeiam informações administrativas sobre o *honeypot*. No caso de um *honeypot* de baixa interatividade a classe EmulatedAddress,

que possui a mesma definição da classe Address do IODEF (DANYLIW et al., 2007), pode ser utilizada para representar a faixa de endereços que está sendo utilizada pelo *honeypot* para os sistemas sendo emulados. Hardware possibilita armazenar informações, em forma textual, que possam ser relevantes sobre o *hardware* sendo utilizado pelo *honeypot*. A classe Sanitization pode ser usada para referenciar um mapeamento entre os endereços reais do *honeypot* e possíveis endereços sanitizados que estão armazenados nas classes System e EventData. O suporte ao mapeamento da sanitização é importante porque permite que uma instituição que gere os registros de eventos já com endereços sanitizados possa eventualmente passar a informação sobre qual seria o endereço real. A classe Contact, derivada do IODEF (DANYLIW et al., 2007), permite, por exemplo, que membros de consórcios de *honeypots* ou projetos de cooperação similares, possam enviar as informações de contato da pessoa responsável pelo *honeypot* em questão. A classe AdditionalData será discutida na Seção 3.2.1.1.

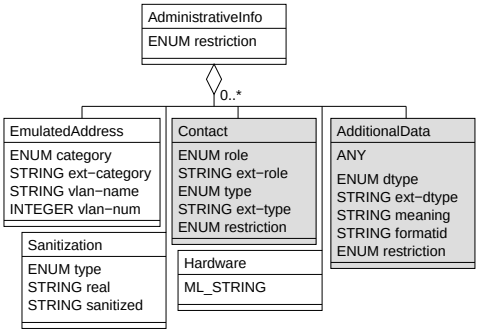

Figura 3.3 - Classe AdministrativeInfo e classes agregadas.

A classe FilterInfo (Figura 3.4) mapeia informações sobre filtros aplicados entre o tráfego da Internet e o *honeypot*. Esta informação é muito importante para o processo de correlação entre dados capturados por diferentes *honeypots*, pois sem ela não é possível saber se determinado tráfego malicioso não foi visto em um sensor porque ele não foi atacado ou porque havia um filtro impedindo a chegada deste tráfego. Se o atributo type possuir o valor open, significa que é permitido o tráfego para as portas listadas em Port e Portlist e o restante é proibido. Se o atributo type for closed, significa que é proibido o tráfego para as portas listadas em Port e Portlist e o restante é permitido. As classes Port e Portlist foram derivadas do IODEF e suas descrições completas estão em (DANYLIW et al., 2007). A classe AdditionalData será discutida na Seção 3.2.1.1.

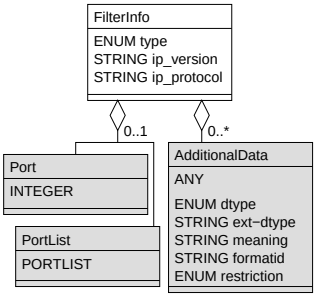

Figura 3.4 - Classe FilterInfo e classes agregadas.

A classe EmulatedSystems permite representar os diversos sistemas sendo emulados em um *honeypot* de baixa interatividade, sendo formada por uma ou mais instâncias da classe System, como pode ser visto na Figura 3.5. A classe System possui a mesma definição do IODEF (DANYLIW et al., 2007), incluindo as classes filhas Service e Application. Porém, no HIDEF a classe Application foi estendida, sendo agregada mais uma classe: Description (representada com traços na Figura). Ao permitir associar uma descrição à classe Application é possível registrar modificações feitas a aplicações de produção ou descrever implementações próprias de emuladores.

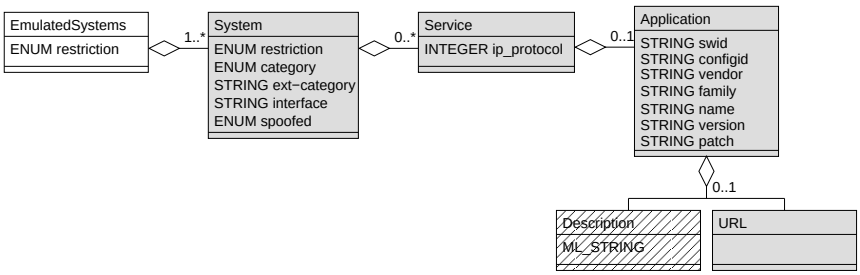

Figura 3.5 - Classe EmulatedSystems e classes agregadas.

A classe Solution representa, em *honeypots* de baixa interatividade, a solução adotada para implementar o *honeypot*. Como pode ser visto na Figura 3.6, uma instância desta classe pode ser representada por zero ou uma instância das classes Description, onde pode ser armazenada uma descrição geral, e URL, onde pode ser armazenada uma URL para o seu *site*. A classe AdditionalData, que pode armazenar informações adicionais necessárias, será discutida na Seção 3.2.1.1.

A classe Honeynet, vista na Figura 3.7 permite representar informações sobre uma

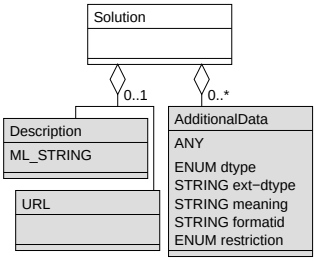

Figura 3.6 - Classe Solution e classes agregadas.

honeynet na qual possa estar um *honeypot* de alta interatividade. Seu atributo type pode ter dois valores: physical e virtual. O atributo ext-type, que permite estender o atributo type, será discutido em mais detalhes na Seção 3.2.1.1. Pode haver zero ou uma classe Description, que pode conter uma descrição geral da *honeynet*. As demais classes podem ocorrer zero ou mais vezes. A classe Address é derivada do IODEF e permite representar endereços IP, MAC (*Media Access Control*) e AS (*Autonomous System*), entre outros relacionados com a *honeynet*. As classes DataCollection e DataControl possuem estruturas semelhantes e representam dados sobre as tecnologias de coleta de dados e de controle de atividades dos invasores utilizadas na *honeynet* (HONEYNET, 2004). A classe AdditionalData será discutida na Seção 3.2.1.1.

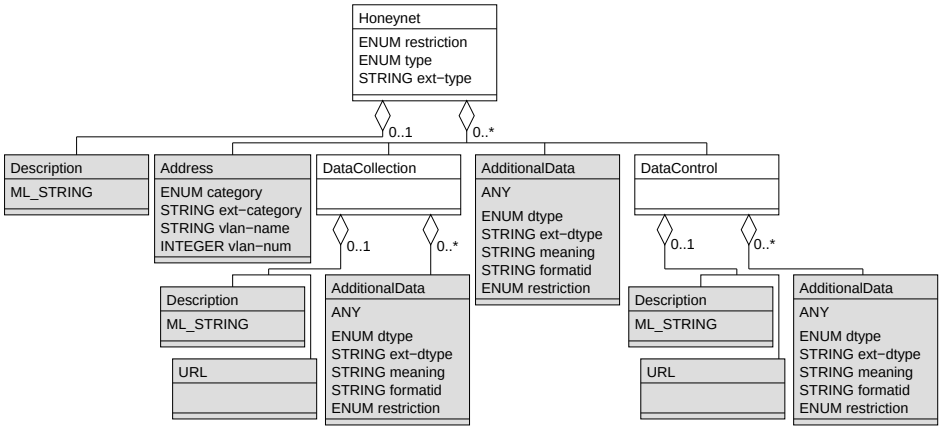

Figura 3.7 - Classe Honeynet e classes agregadas.

Para exemplificar o uso do formato HIDEF, a Figura 3.8 apresenta um documento com a representação de uma varredura, realizada contra um *honeypot* de baixa interatividade, em busca de versões vulneráveis do Serviço *Symantec Remote Man-*

agement[1].

```
<?xml version="1.0" encoding="UTF-8" ?>
<HIDEF-Document version="1.00" lang="pt-BR"
 xmlns="http://www.hideas.org/namespaces/hidef-1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.hideas.org/namespaces/hidef-1.0">
<Honeypot restriction="public" type="low-interaction">
 <System restriction="private" category="target">
  <Node>
   <NodeName>LOCAL-1</NodeName>
   <Address category="ipv4-addr">192.168.229.11</Address>
   <Location>Rio de Janeiro</Location>
  </Node>
  <OperatingSystem name="OpendBSD" vendor="OpenBSD Project" version="4.2" />
  <Description>Sistema real do honeypot de baixa interatividade da instituicao LOCAL-1.</Description>
 </System>
 <AdministrativeInfo restriction="private">
  <EmulatedAddress category="ipv4-net">192.168.229.0/24</EmulatedAddress>
  <Sanitization type="ipv4-net" real="192.168.229.0/24" sanitized="xxx.xxx.xxx." />
  <Hardware>SUNW 110 MHz, 128MB, 1GB + 2GB + 2GB (SCSI), le0 (10baseT)</Hardware>
 </AdministrativeInfo>
 <EmulatedSystems>
  <System restriction="private" category="target">
   <Node><NodeName>default</NodeName></Node>
   <Service ip_protocol="1" />
   <OperatingSystem vendor="Microsoft" name="Windows" version="XP Professional" />
  </System>
 </EmulatedSystems>
 <Solution>
  <Description>Honeyd 1.5c</Description>
  <URL>http://www.honeyd.org/</URL>
 </Solution>
 <EventData>
   <StartTime>2008-05-01T14:09:02+00:00</StartTime>
   <EndTime>2008-05-01T14:09:03+00:00</EndTime>
   <Flow>
     <System category="source">
       <Node>
         <Address category="ipv4-addr">74.219.164.130</Address>
         <Counter type="event">3</Counter>
       </Node>
       <Service ip_protocol="6">
       <Port>55914</Port>
       </Service>
       <OperatingSystem name="Windows XP SP1, Windows 2000 SP4" />
     </System>
     <System category="target">
       <Node>
         <Address category="ipv4-addr">xxx.xxx.xxx.16</Address>
         <Counter type="event">3</Counter>
       </Node>
       <Service ip_protocol="6"><Port>2967</Port></Service>
     </System>
   </Flow>
   <Record>
     <RecordData>
       <Application name="tcpdump"><URL>http://www.tcpdump.org/</URL></Application>
       <RecordItem dtype="string">
May 01 14:09:02.663002 74.219.164.130.55914 &gt; xxx.xxx.xxx.16.2967: S (src OS: Windows XP SP1, Windows 2000 SP4)
409277928:409277928(0) win 65535 &lt;mss 1460,nop,nop,sackOK&gt; (DF)
May 01 14:09:03.318392 74.219.164.130.55914 &gt; xxx.xxx.xxx.16.2967: S (src OS: Windows XP SP1, Windows 2000 SP4)
409277928:409277928(0) win 65535 &lt;mss 1460,nop,nop,sackOK&gt; (DF)
May 01 14:09:03.974287 74.219.164.130.55914 &gt; xxx.xxx.xxx.16.2967: S (src OS: Windows XP SP1, Windows 2000 SP4)
409277928:409277928(0) win 65535 &lt;mss 1460,nop,nop,sackOK&gt; (DF)
       </RecordItem>
     </RecordData>
   </Record>
 </EventData>
</Honeypot>
</HIDEF-Document>
```

Figura 3.8 - Documento HIDEF com uma varredura pelo Serviço *Symantec Remote Management*.

[1] http://www.kb.cert.org/vuls/id/404910

3.2.1.1 Extensões ao HIDEF

As áreas de segurança da informação e de *honeypots* têm uma natureza dinâmica, evoluindo conforme a tecnologia e os tipos de ataques se modificam. Por conta desta natureza o HIDEF, assim como o IODEF, prevê que possam ser feitas extensões ao modelo de dados, permitindo a inclusão de novas características. Por ser um modelo orientado a objetos, é possível estendê-lo através de herança, definindo subclasses com atributos e operações não presentes na superclasse, ou pode-se acrescentar novas classes completamente diferentes.

A classe AdditionalData, derivada do IODEF, serve como um mecanismo de extensão, permitindo a inclusão de informações que não estejam representadas no modelo de dados ou a inclusão de novas classes. Ela comporta elementos atômicos, como inteiros e cadeias de caracteres, mas também permite a representação de dados mais complexos, como novos documentos XML inteiros, derivados de outros *Schemas* (como IDMEF ou IODEF, por exemplo). Seu atributo dtype armazena a definição do tipo de dados, enquanto o atributo meaning comporta a definição de qual o significado daquele dado no contexto do documento HIDEF.

Outro mecanismo permite estender alguns atributos definidos como tipos enumerados de dados. Todos os atributos cujo nome seja da forma "ext-<nome>" são utilizados para estender o atributo de mesmo "<nome>". Um exemplo são os atributos type e ext-type, presentes na classe Honeypot (Figura 3.1). Atualmente os *honeypots* estão classificados na literatura como sendo de baixa ou alta interatividade, porém, caso esta classificação seja ampliada no futuro, o modelo pode ser estendido atribuindo-se ao atributo type o valor "`ext-type`" e ao atributo ext-type o valor "`novo-tipo`".

3.3 HIDEAS – *Honeypots Information and Data Exchange and Analysis System*

Esta Seção descreve a proposta de um sistema que habilita o envio e o recebimento de informações e dados, em formatos como HIDEF e IODEF, de forma a possibilitar o posterior cruzamento de todas as informações armazenadas. A arquitetura proposta é modular, e visa possibilitar que uma instituição interessada possa trocar informações com um número arbitrário de instituições. Para isso, é necessário apenas gerar documentos HIDEF para enviar os dados, ou implementar o sistema HIDEAS para também coletar e analisar informações de diversas fontes.

A Figura 3.9 apresenta a visão geral da arquitetura do sistema proposto, incluindo os principais módulos e as maneiras como a informação pode chegar ao sistema. Uma descrição destes módulos é apresentada a seguir. Foi utilizada a convenção de grafar seus nomes com fonte sem serifa.

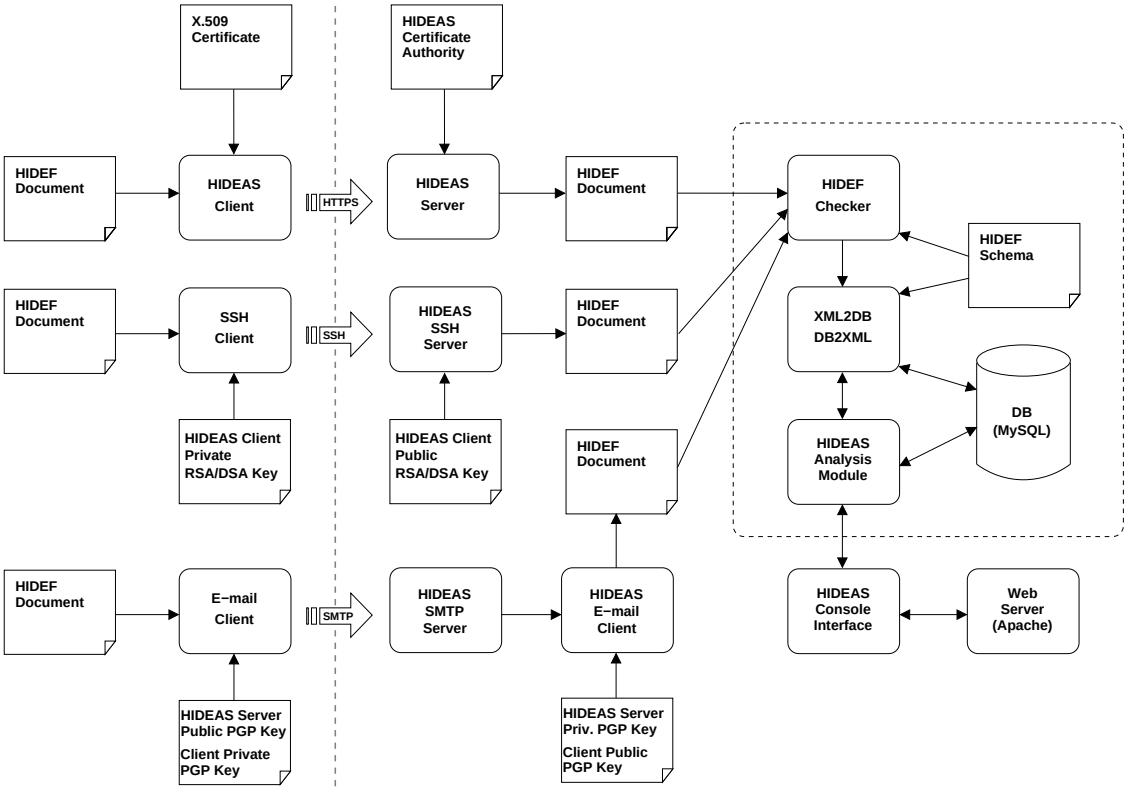

Figura 3.9 - Visão geral da arquitetura do sistema proposto.

HIDEF Document – um documento definido de acordo com o padrão HIDEF, implementado utilizando XML 1.0 (HAROLD; MEANS, 2004; W3C, 2004a), conforme descrito na Seção 3.2.

HIDEF Schema – a descrição formal dos elementos que compreendem um documento HIDEF válido. Este *Schema* possui as definições dos tipos de dados que fazem parte do padrão HIDEF e que devem ser respeitados pelos documentos HIDEF. O *Schema* completo pode ser consultado no Apêndice B.

Comunicação através de HTTPS – uma das formas de comunicação previstas é através de HTTPS (HTTP sobre SSL/TLS), com a utilização de certifi-

cados digitais X.509 (HOUSLEY et al., 1999) para autenticação dos clientes. Esta comunicação será realizada através dos seguintes módulos:

- **HIDEAS Client**: conecta-se ao HIDEAS Server através de uma conexão HTTPS, utilizando um certificado digital (X.509 Certificate) emitido pela Autoridade Certificadora (AC) do HIDEAS Server (HIDEAS Certificate Authority), e envia através desta conexão cifrada estabelecida um ou mais documentos HIDEF;
- **HIDEAS Server**: aceita conexões de clientes (HIDEAS Client) utilizando HTTPS e que possuam um certificado X.509 emitido pela AC local (HIDEAS Certificate Authority). Se a autenticação e o recebimento dos dados forem bem sucedidos envia os documentos HIDEF recebidos para o módulo HIDEF Checker.

Comunicação através de túnel SSH – outra forma de comunicação prevista é o envio de dados através de túneis SSH. Esta é uma tecnologia simples, segura e que permite transmissão de grandes volumes de dados. Para que seja possível a comunicação são necessários os seguintes componentes:

- **SSH Client**: um cliente SSH executando em um computador cliente do sistema, que precisa possuir uma chave DSA ou RSA privada para acesso ao servidor HIDEAS;
- **HIDEAS SSH Server**: servidor SSH que deve possuir uma cópia da chave pública do cliente que quer se conectar ao servidor SSH. Este par de chaves deve ser gerado exclusivamente para a comunicação com o servidor HIDEAS.

Comunicação através de SMTP – também é previsto que seja possível o envio de dados, em menor volume, atravém de *e-mails* cifrados e assinados, utilizando criptografia de chaves pública e privada via PGP. Esta comunicação será feita através dos seguintes módulos:

- **E-mail Client**: programa cliente para envio de *e-mails*, utilizado por clientes do sistema HIDEAS. Este E-mail Client pode ser um programa projetado especificamente para enviar mensagens cifradas para o HIDEAS SMTP Server, ou um cliente regular para envio de *e-mails*, desde que tenha a capacidade de enviar mensagens cifradas para o

HIDEAS SMTP Server. Estas mensagens devem conter um ou mais documentos HIDEF, que devem estar cifrados com a chave pública do servidor HIDEAS (HIDEAS Server Public PGP Key) e assinados com a chave privada do cliente (Client Private PGP Key);

- **HIDEAS SMTP Server**: servidor SMTP a ser disponibilizado pelo sistema HIDEAS para receber as mensagens de seus clientes;
- **HIDEAS E-mail Client**: pode ser um cliente de *e-mail* comum ou um programa especialmente desenvolvido para tratar mensagens recebidas pelo sistema HIDEAS. Este módulo deve ter acesso à chave privada do servidor HIDEAS (HIDEAS Server Private PGP Key) para poder decifrar mensagens cifradas com a chave pública do servidor HIDEAS (HIDEAS Server Public PGP Key). Este módulo também deve possuir a chave pública do cliente (Client Public PGP Key) de modo a poder checar se este é um cliente autorizado a enviar informações, se a mensagem foi realmente enviada por ele e se ela está íntegra.

HIDEF Checker – módulo responsável por checar se o documento HIDEF recebido está de acordo com o *Schema* XML definido (HIDEF Schema) e, em caso afirmativo, enviar o documento para o módulo XML2DB/DB2XML.

XML2DB/DB2XML – módulo responsável por converter e inserir no banco de dados as informações presentes nos documentos HIDEF, bem como gerar documentos no formato HIDEF a partir de respostas do banco de dados a *queries* SQL.

HIDEAS Analysis Module – módulo responsável por fazer a interface com o banco de dados e com o console do analista. Este módulo possuirá funções tanto de análise e correlação quanto de gerenciamento do banco de dados.

HIDEAS Console Interface – módulo responsável por realizar a interface entre o HIDEAS Analysis Module e o servidor *Web*. Age como uma camada adicional de segurança, sendo o responsável por verificar se as solicitações estão em conformidade com os parâmetros aceitos pelo sistema. É também responsável por entregar e formatar os resultados a serem exibidos pelo servidor *Web* (Web Server).

Web Server – faz parte da implementação do serviço de interface com os usuários do sistema (juntamente com o HIDEAS Console Interface). Deve estar in-

tegrado com a AC local (HIDEAS Certificate Authority) para permitir que somente usuários que possuam Certificados X.509 emitidos por esta AC local tenham acesso ao sistema.

3.3.1 Implementação do Protótipo

Para validar o sistema proposto HIDEAS, foi implementado um protótipo capaz de receber documentos HIDEF e IODEF, validá-los e inserí-los em um banco de dados para posterior correlação. A Figura 3.10 mostra o subconjunto de módulos, propostos no sistema HIDEAS, que foi implementado neste protótipo.

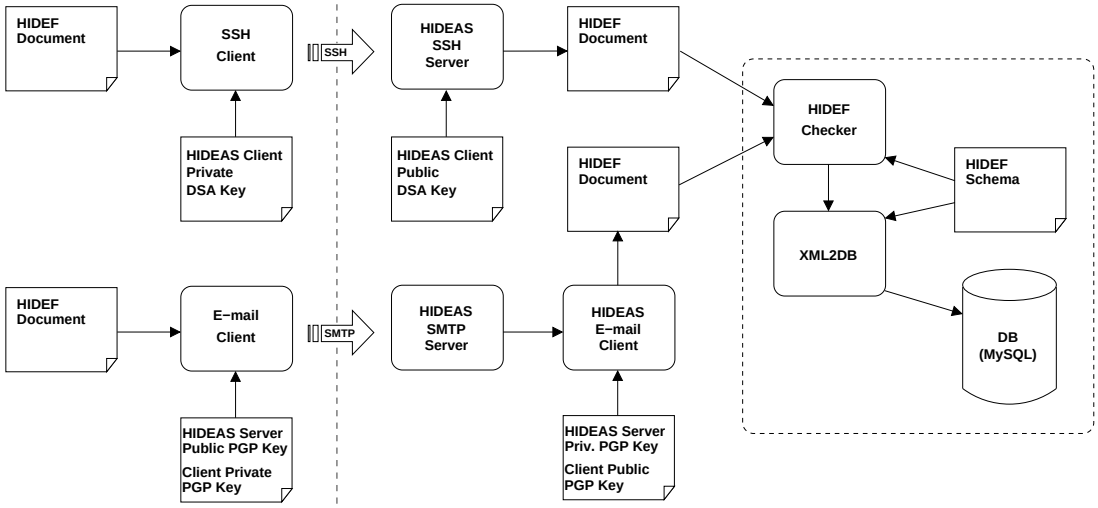

Figura 3.10 - Visão geral do protótipo implementado.

A descrição do modelo de dados do HIDEF, que define o formato do HIDEF-Document, foi apresentada na Seção 3.2 e o HIDEF Schema está disponível no Apêndice B. A seguir serão descritas as implementações dos módulos HIDEF Checker e XML2DB, e a implementação do banco de dados do sistema.

3.3.1.1 HIDEF Checker

Este módulo é responsável por verificar se os documentos HIDEF e IODEF recebidos são válidos de acordo com o *Schema* definido para cada formato. O *Schema* do IODEF está disponível em (DANYLIW et al., 2007).

A implementação foi feita em Perl, com a utilização da biblioteca `XML::LibXML` (RAY;

MCINTOSH, 2002). Foram usadas funções para fazer o *parsing* do documento e a posterior validação contra os *Schemas* do HIDEF e do IODEF.

3.3.1.2 XML2DB

Este módulo recebe os documentos HIDEF e IODEF, os processa, e importa os dados para um banco de dados MySQL (DYER, 2005). A parte principal deste módulo é um programa, desenvolvido em Perl, utilizando a biblioteca `XML::XPath` (RAY; MCINTOSH, 2002), que processa os documentos e os transforma para a inserção no banco de dados.

XPath (XML *Path Language*) é uma linguagem que permite, através do uso de expressões, referenciar elementos e atributos e acessar partes específicas de um documento XML (HAROLD; MEANS, 2004; AMIANO et al.,). A biblioteca `XML::XPath` constrói a árvore XML de cada documento sendo tratado em memória e, então, permite que esta árvore seja percorrida através de expressões XPath.

O programa processa um ou mais arquivos HIDEF, os percorre via XPath e gera, como saída, arquivos para entrada de dados no banco de dados. Uma vez terminado o processamento dos documentos, os dados são importados para um banco de dados MySQL. A próxima seção trata da implementação do banco de dados.

3.3.1.3 Banco de Dados

o banco de dados implementado foi modelado levando em conta os dados que fizeram parte do estudo de caso, que será descrito no Capítulo 4.

O primeiro passo da implementação foi o desenvolvimento de um modelo Entidade-Relacionamento (E-R) (SILBERSCHATZ et al., 1999). Na definição desse modelo, não foram mapeadas como entidades classes que atuam como *containers* de outras classes. Esse tipo de relacionamento do modelo de objetos foi mapeado como uma relação 1–N diretamente com as classes filhas da classe *container*. Também, não foram mapeadas como entidades classes cujo tipo de dados era simples, como Description e DateTime, e que poderiam ser mapeadas no Modelo E-R como atributos de outra entidade.

Esse modelo E-R foi então mapeado para um modelo Relacional (SILBERSCHATZ et al., 1999), em que foram definidas as tabelas a serem criadas no Bando de Dados

relacional MySQL.

Uma descrição completa do Modelo do Banco de Dados, incluindo o modelo E-R e o modelo Relacional, está disponível no Apêndice C.

4 ESTUDO DE CASO

Para testar e validar o formato de dados e a infra-estrutura propostas foi conduzido um estudo de caso, que coletou e correlacionou dados de *honeypots* com tecnologias diferentes e de notificações de incidentes de segurança.

Este Capítulo apresenta a visão geral do estudo de caso, incluindo a descrição das diversas fontes de dados, seguida de uma discussão dos resultados obtidos na análise dos dados coletados das diversas fontes.

4.1 Visão Geral

Este estudo de caso, que tem sua arquitetura mostrada na Figura 4.1, integrou os dados, de um período de 4 meses, das fontes descritas a seguir.

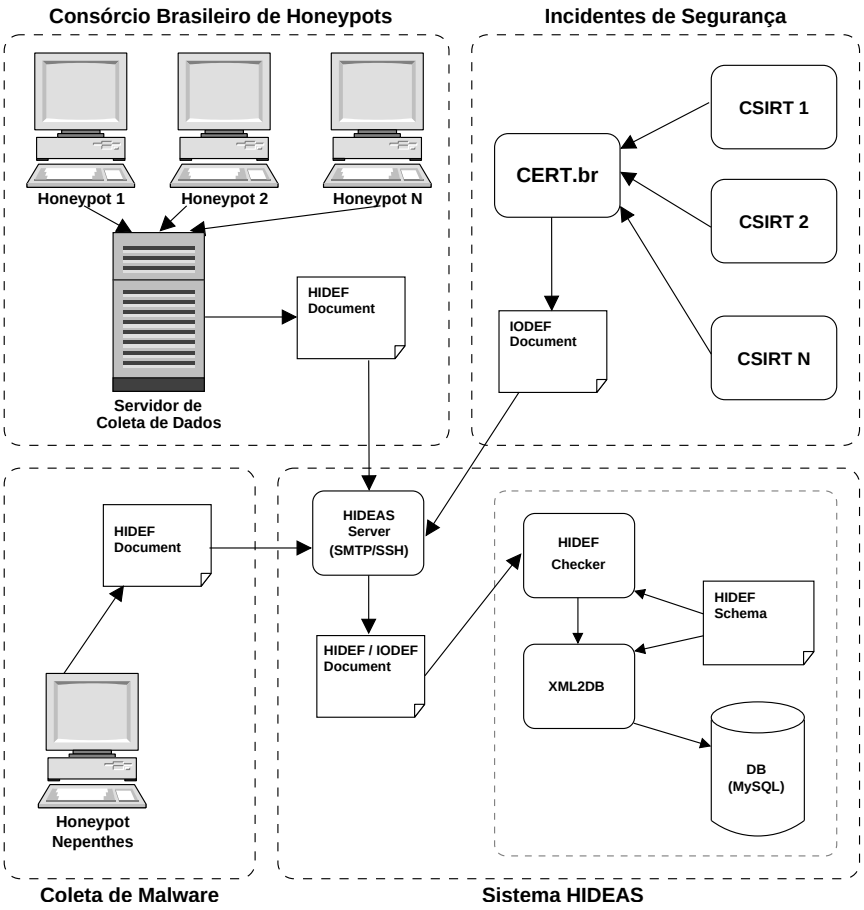

Figura 4.1 - Visão geral do estudo de caso.

Consórcio Brasileiro de *Honeypots* – Na estrutura do Consórcio, o servidor de coleta de dados executava um conjunto de programas para gerar documentos HIDEF sobre atividades maliciosas vistas em um subconjunto dos *honeypots* e, subseqüentemente, enviar estes documentos para o HIDEAS Server, via túnel SSH.

***Honeypot* Nepenthes** – Um *honeypot*, utilizando o Nepenthes como tecnologia para emulação, foi usado para capturar exemplares de códigos maliciosos (*malware*). O próprio *honeypot* executava os programas que geravam os documentos HIDEF e os enviavam para o HIDEAS Server, via túnel SSH.

Notificações de incidentes de segurança – O CERT.br colaborou com este estudo de caso identificando instituições que regularmente notificam incidentes de segurança e que se dispuseram a compartilhar estas notificações para correlação no sistema HIDEAS. Estas notificações foram enviadas em formato IODEF, via *e-mail*, para o sistema HIDEAS.

Após receber os dados enviados pelas fontes acima, o protótipo do sistema HIDEAS, descrito na Seção 3.3.1 processou todas as informações.

As próximas seções descrevem as fontes dos dados com mais detalhes, em seguida serão apresentados os resultados do processamento dos dados coletados.

4.1.1 Consórcio Brasileiro de *Honeypots*

O Consórcio Brasileiro de *Honeypots* tem como um de seus objetivos aumentar a capacidade de detecção de incidentes, correlação de eventos e determinação de tendências de ataques no espaço Internet brasileiro (STEDING-JESSEN et al., 2008). O Consórcio conta hoje com 37 instituições parceiras, que mantém cerca de 45 *honeypots* em 20 cidades brasileiras. Uma visão da arquitetura do consórcio pode ser vista na Figura 4.2.

Cada *honeypot* é configurado para responder por diferentes endereços IP, sendo que cada endereço IP pode estar emulando um sistema operacional e aplicativos diferentes. Para implementá-los são utilizados o Sistema Operacional OpenBSD como base e o Honeyd como *software* para emulação de sistemas operacionais e aplicativos (STEDING-JESSEN et al., 2008; HOEPERS et al., 2005).

Existe um conjunto de dados sobre estes *honeypots* que diz respeito à configuração

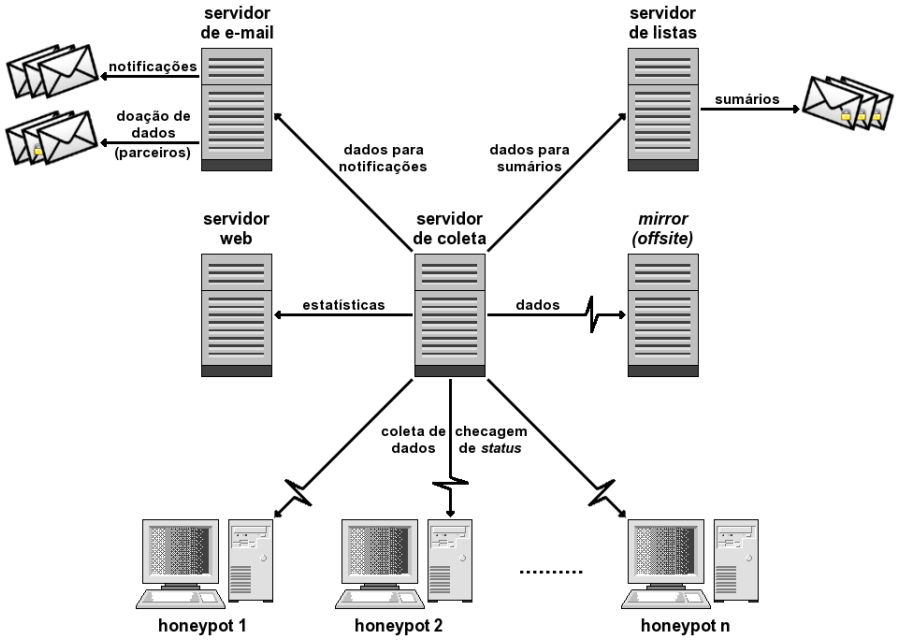

Figura 4.2 - Arquitetura do Consórcio Brasileiro de *Honeypots*.

Fonte: Consórcio Brasileiro de *Honeypots* (STEDING-JESSEN et al., 2008).

dos sistemas emulados, entre eles: sistemas operacionais e serviços sendo emulados; portas TCP, UDP e ICMP que estejam em estado "aberto" no *honeypot*; e filtros existentes que impeçam determinados tipos de tráfego de chegarem ao *honeypot*.

Além dessas informações, os *honeypots* também possuem registros de dados referentes aos ataques recebidos, como: traços de rede das tentativas de ataque recebidas; códigos maliciosos que venham a ser capturados; e possíveis artefatos gerados por atividades maliciosas.

O Consórcio conta com um servidor para coleta de dados, que centraliza todas as informações capturadas nos *honeypots*. Considerando esta estrutura já estabelecida, que pode ser vista na Figura 4.2, o estudo de caso utilizou o servidor de coleta de dados como cliente do sistema HIDEAS. O servidor do Consórcio gerou documentos HIDEF e os enviou via túneis SSH para o servidor HIDEAS.

4.1.2 *Honeypot* Nepenthes

O Nepenthes é um *software* para implantação de *honeypots* de baixa interatividade. Ele é dedicado a coletar códigos maliciosos que se propagam automaticamente, como *worms* e *bots* (PROVOS; HOLZ, 2007; GÖBEL et al., 2006).

Para capturar os códigos, ele emula o comportamento de um *software* vulnerável, decodifica o ataque e obtém uma cópia do código malicioso (GÖBEL et al., 2006).

No contexto deste estudo de caso os dados de 4 meses de captura de *malware*, realizada por um *honeypot* Nepenthes, foram convertidos para HIDEF e enviados via um túnel SSH para o servidor HIDEAS.

4.1.3 Notificações de Incidentes de Segurança

Como descrito na Seção 2.1.1, um incidente de segurança é qualquer evento adverso, confirmado ou sob suspeita, relacionado à segurança dos sistemas de computação ou das redes de computadores. Exemplos de tais eventos são: invasões de sistemas de computadores via rede; atividades de propagação de códigos maliciosos; e varreduras, via rede, à procura de vulnerabilidades em um grupo de sistemas computacionais (*scans*). Também, foi visto naquela Seção que um CSIRT é uma organização ou grupo que provê serviços e suporte, a uma comunidade bem definida, para prevenção, tratamento e resposta a incidentes de segurança em computadores e redes.

No Brasil o CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil), mantido pelo Núcleo de Informação e Coordenação do Ponto br (NIC.br), do Comitê Gestor da Internet no Brasil (CGI.br), é o CSIRT com responsabilidade nacional. Isto é, ele é o grupo responsável por receber, analisar e responder a incidentes de segurança em computadores, envolvendo redes conectadas à Internet brasileira (CERT.br, 2008b; CGI.br, 2006).

O CERT.br recebe notificações de incidentes de segurança ocorridos em diversas instituições brasileiras. Neste estudo de caso o CERT.br atuou identificando instituições que permitissem o compartilhamento das notificações de incidentes por elas detectados, com dados sanitizados, para o estudo de caso do sistema HIDEAS. O CERT.br também instalou em um de seus servidores um programa que transformava as notificações para o formato IODEF, e as enviava por *e-mail* para o servidor HIDEAS.

4.2 Resultados

O objetivo deste estudo de caso foi validar a adequação da arquitetura à sua utilização em uma situação real, bem como validar a compatibilidade do sistema com documentos no formato IODEF.

No período de 01/01/2008 a 30/04/2008 foram coletados, para inserção no banco de dados e posterior correlação, os seguintes dados:

Consórcio Brasileiro de *Honeypots*: Informações sobre todas as varreduras recebidas pelos *honeypots* das seguintes instituições:

- 1 instituição de pesquisa, com o *honeypot* monitorando o tráfego em uma rede `/24` (256 endereços IP) – nas análises este *honeypot* será referenciado como `HpotPesquisa`;
- 1 operadora de telecomunicações, com o *honeypot* monitorando o tráfego em uma rede `/24` (256 endereços IP) – nas análises este *honeypot* será referenciado como `HpotTele`;
- 1 backbone ligando várias redes acadêmicas, com o *honeypot* monitorando o tráfego em uma rede `/24` (256 endereços IP) – nas análises este *honeypot* será referenciado como `HpotBackbone`;
- 1 instituição ligada a uma rede comercial, com o *honeypot* monitorando o tráfego em uma rede `/28` (16 endereços IP) – nas análises este *honeypot* será referenciado como `HpotComercial`.

***Honeypot* Nepenthes:** Todos os códigos maliciosos coletados, incluindo informações de IP de origem, possível tipo do código e assinatura (*hash* criptográfico) – nas análises este *honeypot* será referenciado como `Nepenthes`.

Notificações de Incidentes de Segurança: Incidentes identificados e notificados em formato IODEF, providos pelo CERT.br – nas análises os dados provenientes destas notificações em formato IODEF serão referenciados como `Notificações`.

A Tabela 4.1 mostra, para cada tabela criada no banco de dados MySQL, o seu número de linhas e o volume de dados inseridos. A descrição completa dos campos de cada tabela pode ser vista no Apêndice C, Seção C.2.

Tabela 4.1 - Tabelas do banco de dados MySQL, ordenadas pelos seus nomes, com respectivos números de linhas e volume de dados armazenados.

Tabela	Número de Linhas	Tamanho
artifactdata	3.467	777,1 K
contact	5	256 B
event_system	6.391.567	227.6 M
eventdata	417.936	15.1 M
hash	6.934	460.4 K
honeypot	5	460 B
hpot_contact	5	152 B
hpot_emulated_system	32	1.1 K
hpot_emulatednet	4	104 B
hpot_event	416.329	9.5 M
hpot_real_system	5	188 B
ipv4addr_cc_asn	264.913	6.4 M
record_app	414.469	10.3 M
recorddata	414.469	45.1 M
sanitization	5	216 B
service	6.391.685	221.6 M
service_app	53	1.9 K
service_port	13.099.383	441.2 M
software	91	3.8 K
solution	5	256 B
system	6.391.604	274.8 M
system_asn	5	184 B
system_ipv4addr	6.391.654	246.8 M
system_nodename	32	1.1 K
system_os	180.580	4.7 M

Avaliando a Tabela 4.1, é possível verificar, através do número de linhas da tabela `eventdata`, que durante os 4 meses de dados sendo analisados, tivemos 417.936 eventos registrados ao todo. Também é possível observar, pelo número de linhas da tabela `hpot_event`, que destes eventos registrados, a maioria foi observada em *honeypots*: 416.329. Ainda com base no número de linhas das tabelas do banco de dados, podemos ver que o total de endereços IP únicos, que geraram as atividades maliciosas registradas, foi de 264.913 (tabela `ipv4addr_cc_asn`).

No restante desta seção são apresentados resultados obtidos através de consultas e cruzamentos realizados nos dados presentes no banco.

A Tabela 4.2 mostra o número de endereços IP únicos que originaram tráfego para

cada um dos *honeypots* do Consórcio, os IPs únicos que injetaram códigos maliciosos no *honeypot* Nepenthes e aqueles que foram reportados ao CERT.br.

Tabela 4.2 - Endereços IP únicos que geraram tráfego malicioso, por fonte de dados.

Fonte	IPs únicos
`HpotPesquisa`	18.568
`HpotTele`	126.000
`HpotBackbone`	98.645
`HpotComercial`	32.629
Intersecção dos *honeypots* do Consórcio	210
`Nepenthes`	1.235
`Notificações`	1.607

É interessante, e de certa forma esperada, a diferença entre o total de IPs únicos presentes em incidentes notificados e aqueles vistos individualmente por *honeypots* do Consórcio. Isso pode ter ocorrido porque as notificações, como são feitas voluntariamente por administradores de redes, em geral representam somente um subconjunto de atividades maliciosas, que não necessariamente englobam todos os IPs que geram atividades maliciosas contra as redes que notificaram incidentes. Também era esperado que o `Nepenthes`, como registrou apenas os endereços IP que injetaram códigos maliciosos, apresentasse um número bem menor de IPs únicos do que os *honeypots* do Consórcio, que registraram um conjunto de atividades maior.

Nota-se, também, que o número de IPs emulados por um *honeypot* não necessariamente influencia a quantidade de IPs únicos por ele vistos. O `HpotPesquisa`, apesar de emular 256 endereços, registrou menos endereços IPs originando atividades que o `HpotComercial`, que emula apenas 16 endereços IP. Outro dado interessante é que, ao cruzarmos os dados sobre os IPs únicos vistos em cada *honeypot* do Consórcio, somente 210 IPs foram vistos por todos os 4 *honeypots*.

A Tabela 4.3 apresenta alguns cruzamentos adicionais feitos entre os endereços IP de origem das atividades vistas nas fontes de dados. A Tabela também informa (entre parênteses) a distância entre os blocos CIDR `/8` aos quais os IPs das fontes pertencem. Por exemplo, a distância entre os blocos CIDR `211.0.0.0/8` e `216.0.0.0./8` é 5.

Como podemos ver na Tabela 4.3, se cruzarmos os IPs vistos pelos 4 *honeypots*

Tabela 4.3 - Número de IPs de origem em comum e distância, em blocos /8, entre as diversas fontes.

	Nepenthes	(/8)	Notificações	(/8)
HpotComercial	435	(0)	1.320	(0)
HpotTele	130	(1)	565	(1)
HpotBackbone	8	(8)	148	(8)
HpotPesquisa	4	(50)	145	(50)
Nepenthes	–	–	0	(0)
Intersecção dos IPs do Consórcio	1	–	18	–

do Consórcio com os IPs que injetaram códigos maliciosos no *honeypot* Nepenthes, obtemos apenas 1 endereço IP que é visto em comum por estes 5 *honeypots*. Se cruzarmos os dados do Consórcio com os dados de Notificações, 18 endereços IP em comum são encontrados. Já, se cruzarmos os dados de todas as fontes, não há nenhum endereço IP em comum na origem de atividades maliciosas, pois o Nepenthes não registrou nenhum IP em comum com as Notificações.

A medida da distância entre IPs emulados pelos *honeypots* e os IPs das redes que notificaram incidentes evidencia um padrão na quantidade de IPs em comum entre as fontes. Quanto maior a distância entre os endereços IP, menor o número de IPs de origem em comum registrados. Considerando que cada um desses *honeypots* possui endereços IP em blocos bem distintos, e está em um AS diferente, essa pode ser a razão do número tão baixo de IPs em comum entre os sensores. Esse comportamento pode refletir diferentes estratégias de propagação de códigos maliciosos, por exemplo, que tendem a concentrar suas ações nas imediações dos IPs infectados. Esse resultado pode indicar que, realmente, para ter informações mais precisas do tráfego malicioso na Internet, é necessário coletar informações em diferentes pontos da rede.

A única exceção é entre o Nepenthes e as Notificações, que estão no mesmo bloco /8 mas não tiveram nenhum IP em comum. Isto pode ser pelo fato de o Nepenthes, por ser um *honeypot* especializado na coleta de *malware*, registrar um tipo de tráfego muito diferente daquele das notificações. Nas Notificações, os administradores de redes das instituições identificadas tenderam a não notificar atividades relacionadas com propagação de *malware*.

A Tabela 4.4 mostra, para todos os endereços IP de origem, os 10 *Country Codes* (CC) e os 10 ASs com maior número de IPs únicos encontrados.

Tabela 4.4 - Países e ASs com maior número de endereços IP de origem únicos.

Países		*Autonomous Systems*			
CC	IPs	ASN	Nome		IPs
US	41.919	4134	CHINANET-BACKBONE	(CN)	20.336
CN	37.117	7738	Tele Norte Leste Participações S.A.	(BR)	10.083
BR	28.962	4837	CHINA169-BACKBONE CNCGROUP	(CN)	9.687
DE	18.567	3320	DTAG Deutsche Telekom AG	(DE)	9.433
AR	8.922	7132	SBIS-AS - AT&T Internet Services	(US)	8.944
IT	8.898	3269	ASN-IBSNAZ TELECOM ITALIA	(IT)	7.885
PL	8.246	5617	TPNET Polish Telecom commercial IP network	(PL)	6.245
JP	7.948	7303	Telecom Argentina S.A.	(AR)	4.828
ES	7.891	8167	Brasil Telecom S/A	(BR)	4.684
KR	7.024	3462	HINET Data Communication Business Group	(TW)	4.520

Vale destacar que todos os ASs, listados entre os 10 que mais tiveram IPs únicos originando tráfego para alguma das fontes, são de redes que fornecem serviços de banda larga para usuários finais. Essa informação é consistente com as tendências atuais, de as máquinas de usuários finais comprometidas por *bots* estarem entre as originadoras de tráfego malicioso na Internet.

Se fizermos uma análise dos 210 IPs em comum nos *honeypots* do Consórcio, vemos que estes IPs são de 38 países e 122 ASs distintos. A Tabela 4.5 mostra os 10 países e ASs que mais tiveram IPs, dentre estes 210.

Tabela 4.5 - Países e ASs com maior número de endereços IP, dentre os 210 IPs que geraram atividades contra todos os *honeypots* do Consórcio.

Países		*Autonomous Systems*			
CC	IPs	ASN	Nome		IPs
US	61	4134	CHINANET-BACKBONE	(CN)	26
CN	52	4837	CHINA169-BACKBONE CNCGROUP	(CN)	12
GB	11	3462	HINET Data Communication Business Group	(TW)	8
KR	8	30083	SERVER4YOU - Server4You Inc.	(US)	8
TW	8	13333	NAUTICOM-NET - Pinnatech, Inc.	(US)	7
DE	7	2914	NTT-COMMUNICATIONS - NTT America, Inc.	(US)	5
HK	5	209	ASN-QWEST - Qwest	(US)	5
FR	5	4766	KIXS-AS-KR Korea Telecom	(KR)	4
BR	4	4	ISI-AS - University of Southern California	(US)	4
ES	3	21327	H3GUK Hutchison 3G UK core network	(GB)	4

É interessante notar que neste caso há muito pouca concentração de IPs por AS de origem e, com exceção dos 3 primeiros ASs, os outros não fazem parte da Tabela 4.4. Já do ponto de vista de *Country Code* de origem há maior concentração, com cerca de 60% dos IPs em comum nos *honeypots* alocados para apenas 3 países: Estados Unidos, China e Inglaterra.

Porém, nem todas as atividades registradas foram necessariamente maliciosas. Quatro IPs do AS da *University of Southern California*, listado na Tabela 4.5, fazem parte do projeto *ANT Censuses of the Internet Address Space*[1]. Este projeto realiza *probes* aleatórios na Internet para medir a sua topologia e determinar lentidão em *links*, entre outros objetivos. De maneira similar foram identificados três endereços IP pertencentes ao projeto ARK (*Archipelago Measurement Infrastructure*) do CAIDA (*Cooperative Association for Internet Data Analysis*)[2].

Outra análise realizada foi a investigação de quais serviços foram os mais atingidos por atividades maliciosas. Para essa análise foram contabilizadas as portas TCP e UDP que mais receberam pacotes ao longo dos 4 meses. A Tabela 4.6 mostra as portas TCP que mais receberam pacotes em todo o período de análise.

Tabela 4.6 - Quinze portas TCP que mais receberam pacotes.

TCP		
Serviço Usualmente Associado	Porta	Pacotes
Microsoft-SQL	1433	3.777.404
Microsoft-DS	445	3.596.549
NETBIOS *Session Service*	139	3.332.085
DCE *Endpoint Resolution*	135	3.152.328
Symantec Remote Management	2967	1.789.409
SSH (*Secure Shell*)	22	1.321.945
Proxy HTTP	8080	951.780
VNC (*Virtual Network Computing*)	5900	856.308
SOCKS	1080	855.009
—	9988	846.696
Proxy AnalogX	6588	845.415
Proxy Squid	3128	810.493
VNC	5800	702.283
Symantec Remote Management	2968	682.506
HTTP	80	618.048

[1] http://www.isi.edu/ant/address/
[2] http://www.caida.org/projects/ark/

Dentre as portas registradas, a 22/TCP (SSH) e as portas 5800/TCP e 5900/TCP (VNC), por estarem relacionadas com serviços de *login* remoto, normalmente recebem ataques de força bruta na tentativa de encontrar contas com senhas fracas. As portas 1433/TCP, 2967/TCP e 2968/TCP estão relacionadas com serviços que, nos últimos anos, possuíram versões com vulnerabilidades que permitem o acesso remoto, se exploradas. Ambas foram amplamente exploradas e atacantes ainda procuram por versões vulneráveis conectadas à Internet. As portas TCP 8080, 1080, 3128 e 6588 são portas associadas a serviços de *proxy*. Varreduras por estas portas normalmente estão associadas à procura de *proxies* que possam ser abusados para o envio de *spam*. A porta 80/TCP é padrão para servidores *Web*. Varreduras por esta porta podem indicar procura por servidores vulneráveis ou, em muitos casos, por servidores HTTP que se comportem como *proxies* abertos. As portas 445/TCP, 139/TCP e 135/TCP são, comumente, utilizadas por códigos maliciosos para explorar diversas vulnerabilidades em sistemas operacionais Windows. A porta 9988/TCP não possui um serviço oficial associado, embora alguns serviços de P2P a utilizem para tranferência de arquivos.

A Tabela 4.7 apresenta as portas UDP que mais receberam tráfego no período analisado.

Tabela 4.7 - Quinze portas UDP que mais receberam pacotes.

UDP		
Serviço Usualmente Associado	Porta	Pacotes
Windows Messaging Service	1026	862.791
Windows Messaging Service	1027	763.431
Microsoft-SQL-Monitor	1434	298.429
—	45306	263.259
NETBIOS *Name Service*	137	248.439
—	4515	202.999
—	1432	98.973
—	13226	62.152
—	13120	50.954
—	41524	48.895
—	13322	22.866
—	7592	22.431
—	13229	17.906
—	60350	17.822
—	28608	11.804

As portas 1026/UDP e 1027/UDP são, normalmente, associadas pelo sistema operacional Windows ao serviço *Windows Messaging Service*, que permite que sejam enviadas janelas do tipo *pop-up* para um computador conectado à Internet sem a proteção de um *firewall*. Esse serviço tem sido abusado para o envio de mensagens indesejadas em formato de *pop-up*, também conhecidas como *pop-up spam* (PANG et al., 2004). A porta 1434/UDP está associada ao servidor SQL da Microsoft, que possui versões vulneráveis, que podem ser exploradas via esta porta. A porta 137/UDP é utilizada por sistemas Windows para obter e prover informações sobre serviços de rede disponíveis em um sistema. As demais portas UDP não possuem serviços oficiais associados.

A Tabela 4.8 apresenta uma visão das portas TCP e UDP mais freqüentes, de forma agregada.

Tabela 4.8 - Quinze portas TCP e UDP que mais receberam pacotes.

TCP + UDP		
Serviço Usualmente Associado	Porta	Pacotes
Microsoft-SQL	1433	3.777.404
Microsoft-DS	445	3.600.253
NETBIOS *session service*	139	3.332.085
DCE *endpoint resolution*	135	3.152.481
Symantec Remote Management	2967	1.793.865
SSH (*Secure Shell*)	22	1.321.945
Proxy HTTP	8080	951.791
Windows Messaging (UDP) ou `nterm` (TCP)	1026	880.515
VNC (*Virtual Network Computing*)	5900	856.308
SOCKS	1080	855.009
—	9988	846.699
Proxy AnalogX	6588	845.419
Proxy Squid	3128	810.497
Windows Messaging (UDP) ou MS NT Directory Interface (TCP)	1027	766.936
VNC	5800	702.283

É interessante notar que as portas 1026 e 1027 nesta tabela apresentam mais pacotes do que na tabela de portas UDP. Isto indica que, além da atividade associada com o *Windows Messaging Service*, também foram vistos pacotes direcionados para os serviços TCP associados a essas portas.

Uma análise nos códigos maliciosos coletados pelo *honeypot* Nepenthes também foi

realizada. A Tabela 4.9 apresenta algumas estatísticas gerais sobre o conjunto de dados coletados.

Tabela 4.9 - Estatísticas sobre os códigos maliciosos capturados pelo *honeypot* Nepenthes.

Códigos maliciosos capturados (com repetição)	3.467
Endereços IP (únicos)	1.235
Hashes criptográficos (únicos)	582
Assinaturas de antivírus encontradas (únicas)	95
Exemplares sem nenhuma assinatura conhecida (com repetição)	582

Vale notar que o número de códigos maliciosos capturados, com repetição, é maior que o de endereços IP únicos vistos. Isso mostra que endereços IP foram vistos mais de uma vez propagando códigos maliciosos.

Na Tabela 4.10 podemos ver, para cada tipo de *malware* identificado, quantos exemplares ao total tentaram infectar o *honeypot* e, deste total, quantos eram únicos, ou seja, com o mesmo *hash* criptográfico. O tipo de *malware* foi identificado no momento da coleta, através da checagem de assinaturas no serviço *online* do antivírus Kaspersky[3].

Tabela 4.10 - Tipos de códigos maliciosos capturados.

Tipo de *Malware*	Exemplares Totais	Exemplares Únicos
`bot`	1.935	82
`undefined`	582	66
`worm`	505	313
`trojan`	245	16
`virus`	126	88
`backdoor`	74	12

Como era de se esperar, o maior número de códigos maliciosos que chegaram ao *honeypot* foi da categoria *bot*. Do total de 1.935 existem apenas 82 variantes. A categoria com menos variantes foi a de *worms*. Também foi interessante notar a quantidade de códigos que foram classificados como cavalos de tróia (*trojans*). Destes, 2 possuem a

[3] `http://www.kaspersky.com/scanforvirus`

assinatura "`Trojan-Downloader.Win32.Banload.kgg`" – a *string* `Banload` identifica que o *payload* é específico para coletar dados bancários da vítima.

5 CONCLUSÕES

Como discutido ao longo desta tese, os trabalhos existentes na área de coleta e análise de dados de *honeypots* concentram-se na correlação de dados de *honeypots* que utilizem tecnologias similares. Porém, é importante analisar o tráfego observado entre *honeypots* que utilizem diferentes tecnologias e implantados em diferentes ambientes e, complementarmente, correlacionar esses dados com aqueles capturados em redes de produção ou por outras tecnologias de monitoração de redes.

A principal contribuição deste trabalho foi a definição do HIDEF, um formato para representação e troca de dados e informações produzidas por instituições que estejam utilizando *honeypots* e *honeynets*. Estes dados são representados de forma estruturada e com uma semântica apropriada, o que permite uma localização rápida, facilita a automatização do processamento e a interoperabilidade entre diferentes tecnologias. O formato proposto, ao manter-se compatível com o padrão IODEF, permite que os dados coletados por *honeypots* possam ser correlacionados facilmente com dados de incidentes de segurança em redes ou com os dados coletados por sistemas de detecção de intrusão. Adicionalmente, os dados coletados por *honeypots* e representados no formato HIDEF podem facilmente ser exportados para o formato IODEF e notificados como um incidente de segurança.

Outra vantagem da compatibilidade com o padrão IODEF se deve ao fato do documento que o define, a RFC 5070, ser uma RFC do tipo *Standard*. Ou seja, ela define um padrão de tecnologia para a Internet. RFCs nesta categoria, em geral, são adotadas por desenvolvedores de *software* para a implementação de sistemas ligados à Internet que necessitem maior interoperabilidade. Com isso, há grande possibilidade de serem desenvolvidas, em um futuro próximo, ferramentas para importar e exportar dados de ataques em formato IODEF. Quando ferramentas deste tipo ficarem disponíveis, elas podem ser integradas a um sistema de geração de documentos HIDEF, como o sistema proposto HIDEAS.

O sistema HIDEAS, outra contribuição desta tese, habilita o envio e o recebimento de informações e dados, em formatos como o HIDEF e o IODEF, de forma a possibilitar o posterior cruzamento de todas as informações armazenadas. Foi implementado um protótipo, para permitir a validação da sua aplicabilidade através de um estudo de caso, que contou com os módulos chave para geração, transporte e importação dos dados em um banco de dados relacional.

O estudo de caso, conduzido para validar o formato de dados e a infra-estrutura sendo propostas, coletou e correlacionou dados de *honeypots* com tecnologias diferentes e de notificações de incidentes de segurança. Este estudo integrou os dados, de um período de 4 meses, de 4 *honeypots* do Consórcio Brasileiro de *Honeypots*, de um *honeypot* específico para coleta de códigos maliciosos, bem como de incidentes de segurança notificados ao CERT.br.

Os resultados discutidos no Capítulo 4 mostraram que, mesmo com o banco de dados modelado para armazenar os dados conforme o formato HIDEF, foi transparente a inserção dos dados de incidentes em formato IODEF. Os resultados também evidenciaram que as diversas fontes capturaram informações diferentes no que diz respeito às origens e quantidades de atividades maliciosas. As correlações realizadas reforçaram a percepção da necessidade de fontes distintas de dados para obter uma visão mais completa do tráfego malicioso na Internet.

De forma geral, o formato HIDEF e o protótipo implementado foram eficientes ao representar e manipular dados reais, relativos a 4 meses de dados coletados em sensores e em incidentes de segurança. O estudo de caso também confirmou que resultados interessantes são obtidos ao serem cruzadas informações de fontes diferentes, cruzamento este que seria mais difícil caso os dados não estivessem representados em um formato comum.

Possíveis desdobramentos deste trabalho incluem:

- conduzir um estudo de caso com dados coletados por *honeynets* e por *honeypots* implementados com outras tecnologias, além das utilizadas neste trabalho. Embora o modelo tenha sido definido de modo a permitir a representação de dados coletados em *honeynets*, por *honeyclients* ou por outros *softwares* para implementação de *honeypots*, não foram realizados testes relativos à representação destes dados;

- completar a implementação do sistema HIDEAS, através do desenvolvimento dos módulos previstos para realizar a análise dos dados (HIDEAS Analysis Module e HIDEAS Console Interface). É possível a criação de diversos módulos de análise, utilizando diferentes técnicas que operam em um mesmo conjunto de dados coletados pelo sistema;

- implantar o sistema em um ambiente de produção, recebendo e correlaci-

onando continuamente dados de *honeypots* diversos, de outros sensores de rede e de incidentes de segurança;

- estudar uma possível otimização da modelagem relacional utilizada no banco de dados, de forma aumentar a performance nas consultas;

- como a sanitização de dados, na maior parte das vezes, envolve o uso de caracteres (como "`xxx`"), os campos relativos a endereços IP foram definidos como *strings* e não como inteiros de 32 bits (ou 128, no caso de IPv6). Seria interessante testar a viabilidade do mapeamento da sanitização, presente no IP recebido via HIDEF, para endereços de redes reservadas, que poderiam ser armazenados como inteiros de 32 bits. Isto economizaria espaço e, potencialmente, aumentaria a performance. Porém, seria adicionada uma complexidade a mais, ao ser necessário manter um mapeamento entre os valores originais do documento HIDEF e os valores inseridos no banco de dados;

- avaliar qual seria o desempenho caso fossem utilizados bancos de dados orientados a objetos ou nativos XML;

- a utilização de Perl para a implementação do protótipo foi escolhida por ser uma linguagem simples, com bons recursos para o tratamento texto e por ter amplo suporte a XML. Porém, para aumento da performace, seria interessante implementar todo o sistema em outra linguagem, que não seja de interpretação como o Perl.

REFERÊNCIAS BIBLIOGRÁFICAS

ACTIVEWORX. **Honeynet security console**. 2007. Disponível em: <http://www.activeworx.org/programs/hsc/>. Acesso em: 08 mar. 2008. 41

ALBERTS, C.; DOROFEE, A.; KILLCRECE, G.; RUEFLE, R.; ZAJICEK, M. **Defining incident management processes for CSIRTs**: a work in progress. Pittsburgh, Pennsylvania: CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, 2004. 249 p. CMU/SEI-2004-TR-015. Disponível em: <http://www.cert.org/archive/pdf/04tr015.pdf>. Acesso em: 08 mar. 2008. 30, 31, 32

ALLEN, J. H. **The CERT guide to system and network security practices**. 1. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2001. 480 p. (SEI series in software engineering). ISBN 978-0-201-73723-3. 32

ALLMAN, M.; BLANTON, E.; PAXSON, V.; SHENKER, S. Fighting coordinated attackers with cross-organizational information sharing. In: HOTNETS V: WORKSHOP ON HOT TOPICS IN NETWORKS, 5., Irvine, California, USA. **Proceedings...** [S.l.], 2006. p. 121–126. 23

ALLMAN, M.; PAXSON, V.; TERRELL, J. A brief history of scanning. In: IMC '07: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT, 7., San Diego, California, USA. **Proceedings...** New York, NY, USA: ACM, 2007. p. 77–82. ISBN 978-1-59593-908-1. 23

AMIANO, M.; D'CRUZ, C.; ETHIER, K.; THOMAS, M. D. **XML**: problem - design - solution. [S.l.: s.n.]. 69

ARVIDSSON, J.; CORMACK, A.; DEMCHENKO, Y.; MEIJER, J. **RFC 3067**: TERENA's incident object description and exchange format requirements. Feb. 2001. Disponível em: <http://www.ietf.org/rfc/rfc3067.txt>. Acesso em: 08 mar. 2008. 23, 27, 28, 48

BACE, R. G. **Intrusion detection**. 1. ed. Indianopolis, Indiana: Sams, 2000. 368 p. ISBN 978-1578701858. 23

BAILEY, M.; COOKE, E.; JAHANIAN, F.; NAZARIO, J.; WATSON, D. The internet motion sensor: a distributed blackhole monitoring system. In: NDSS'05:

ANNUAL NETWORK AND DISTRIBUTED SYSTEM SECURITY SYMPOSIUM, 12., San Diego, California, USA. **Proceedings...** Reston, Virginia, USA: Internet Society, 2005. ISBN 1-891562-19-3. 23, 33

BELLOVIN, S. M. There be dragons. In: USENIX SECURITY SYMPOSIUM, 3., Baltimore, Maryland, USA. **Proceedings...** [S.l.], 1992. p. 1–16. 33

BISHOP, M. **Computer security**: art and science. 1. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2002. 1136 p. ISBN 978-0201440997. 23

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML guia do usuário**. Rio de Janeiro, RJ: Campus, 2000. 492 p. ISBN 85-352-0562-4. 101

BROWN, M. J. W.; STIKVOORT, D.; KOSSAKOWSKI, K. P.; KILLCRECE, G.; RUEFLE, R.; ZAJICEK, M. **Handbook for computer security incident response teams (CSIRTs)**. 2. ed. Pittsburgh, Pennsylvania: CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, 2003. 223 p. CMU/SEI-2003-HB-002. Disponível em: <http://www.sei.cmu.edu/publications/documents/03.reports/03hb002.html>. Acesso em: 08 mar. 2008. 27, 28, 32

BROWNLEE, N.; GUTTMAN, E. **RFC 2350**: expectations for computer security incident response. June 1998. Disponível em: <http://www.ietf.org/rfc/rfc2350.txt>. Acesso em: 08 mar. 2008. 27

CAIN, P.; JEVANS, D. **Extensions to the IODEF-Document class for phishing, fraud, and other crimeware**. Oct. 2007. Expires on Apr. 26, 2008. Disponível em: <http://www.ietf.org/internet-drafts/draft-cain-post-inch-phishingextns-03.txt>. Acesso em: 08 mar. 2008. 116

CENTRO DE ESTUDOS, RESPOSTA E TRATAMENTO DE INCIDENTES DE SEGURANÇA NO BRASIL (CERT.br). **Estatísticas dos incidentes reportados ao CERT.br**. 2008. Disponível em: <http://www.cert.br/stats/incidentes/>. Acesso em: 08 mar. 2008. 23

_____. **Missão do CERT.br**. 2008. Disponível em: <http://www.cert.br/missao.html>. Acesso em: 08 mar. 2008. 74

CERT COORDINATION CENTER. **Advisory CA-2001-19 "Code Red"worm exploiting buffer overflow in IIS indexing service DLL**. July 2001.

Disponível em: <http://www.cert.org/advisories/CA-2001-19.html>. Acesso em: 08 mar. 2008. 116

CERT Coordination Center. **CERT/CC statistics 1988-2008**. 2008. Disponível em: <http://www.cert.org/stats/cert_stats.html>. Acesso em: 08 mar. 2008. 23

CERT COORDINATION CENTER, SOFTWARE ENGINEERING INSTITUTE, CARNEGIE MELLON UNIVERSITY. **CSIRT FAQ**. 2006. Tradução autorizada, feita pelo CERT.br. Disponível em: <http://www.cert.br/certcc/csirts/csirt_faq-br.html>. Acesso em: 08 mar. 2008. 27, 28, 32

CHESWICK, W. R. An evening with berferd in which a cracker is lured, endured, and studied. In: WINTER 1992 USENIX CONFERENCE, San Francisco, California, USA. **Proceedings...** [S.l.], 1992. p. 163–174. 33

COHEN, F. Deception toolkit. **Risks Digest**, v. 19.62, Mar. 1998. 34

COMITÊ GESTOR DA INTERNET NO BRASIL. **Atribuições e atividades do CGI.br**. 2006. Disponível em: <http://www.cgi.br/sobre-cg/>. Acesso em: 08 mar. 2008. 74

CONTI, G.; ABDULLAH, K. Passive visual fingerprinting of network attack tools. In: VizSEC/DMSEC '04: 2004 ACM Workshop on Visualization and Data Mining for Computer Security, Washington DC, USA. **Proceedings...** [S.l.]: ACM Press, 2004. p. 45–54. ISBN 1-58113-974-8. 43, 44

COOKE, E.; BAILEY, M.; MAO, Z. M.; WATSON, D.; JAHANIAN, F.; MCPHERSON, D. Toward understanding distributed blackhole placement. In: WORM '04: 2004 ACM WORKSHOP ON RAPID MALCODE, Washington DC, USA. **Proceedings...** New York, NY, USA: ACM, 2004. p. 54–64. ISBN 1-58113-970-5. 23, 33

DACIER, M.; POUGET, F.; PHAM, V.-H. **Leurrecom.org honeypot project**. 2007. Disponível em: <http://www.leurrecom.org/>. Acesso em: 08 mar. 2008. 43

DANYLIW, R.; MEIJER, J.; DEMCHENKO, Y. **RFC 5070**: the incident object description exchange format. Dec. 2007. Disponível em:

<http://www.ietf.org/rfc/rfc5070.txt>. Acesso em: 08 mar. 2008. 23, 48, 49, 55, 60, 61, 68, 101, 102, 103, 104, 105, 107, 110, 111, 114, 115, 116, 117

DEBAR, H.; CURRY, D.; FEINSTEIN, B. **RFC 4765**: The intrusion detection message exchange format (IDMEF). Mar. 2007. Disponível em: <http://www.ietf.org/rfc/rfc4765.txt>. Acesso em: 08 mar. 2008. 47, 48, 55

DYER, R. **MySQL in a nutshell**. 1. ed. Sebastopol, California: O'Reilly, 2005. 348 p. ISBN 0-596-00789-2. 69

FILHO, A. B.; AMARAL, A. S. M. S.; MONTES, A.; HOEPERS, C.; STEDING-JESSEN, K.; FRANCO, L. H.; CHAVES, M. H. P. C. Honeynet.br: desenvolvimento e implantação de um sistema para avaliação de atividades hostis na internet brasileira. In: . [S.l.: s.n.]. 24, 34

FITZGERALD, M. **XML hacks**. 1. ed. Sebastopol, California: O'Reilly, 2004. 478 p. ISBN 0-596-00711-6. 46

GEORGIA INSTITUTE OF TECHNOLOGY, NETWORK SECURITY AND ARCHITECTURE LABORATORY. **The georgia tech honeynet**. 2006. Disponível em: <http://www.nsa.gatech.edu/honeynet/>. Acesso em: 08 mar. 2008. 43

GÖBEL, J.; HEKTOR, J.; HOLZ, T. Advanced honeypot-based intrusion detection. **;login: The USENIX Magazine**, v. 31.6, p. 17–25, Dec. 2006. 35, 74

GRANCE, T.; KENT, K.; KIM, B. **Computer security incident handling guide**. [S.l.], Jan. 2004. Special Publication 800-61. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf>. Acesso em: 08 mar. 2008. 27, 28

HAROLD, E. R.; MEANS, W. S. **XML in a nutshell**. 3. ed. Sebastopol, California: O'Reilly and Associates, Inc., 2004. 712 p. ISBN 0-596-00764-7. 45, 46, 65, 69

HARTMEIER, D. Design and performance of the openbsd stateful packet filter (pf). In: FREENIX TRACK: 2002 USENIX ANNUAL TECHNICAL CONFERENCE (FREENIX '02), Monterey, California, USA. **Proceedings...** [S.l.], 2002. 42

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., Singapore. **Proceedings...** [S.l.], 2005. 23, 33, 34, 42, 72

HOEPERS, C.; STEDING-JESSEN, K.; MONTES, A. Honeynets applied to the csirt scenario. In: . [S.l.: s.n.]. 24

HOUSEHOLDER, A.; HOULE, K.; DOUGHERTY, C. Computer attack trends challenge internet security. **IEEE Computer Magazine (Supplement to Computer: Preview of Upcoming Security & Privacy Magazine)**, v. 35, n. 4, p. 5–7, Apr. 2002. ISSN: 0018-9162. Disponível em: <http://csdl.computer.org/comp/mags/co/2002/04/r4s05.pdf>. Acesso em: 13 abr. 2008. 23

HOUSLEY, R.; FORD, W.; POLK, W.; SOLO, D. **RFC 2459**: Internet x.509 public key infrastructure. Jan. 1999. Disponível em: <http://www.ietf.org/rfc/rfc2459.txt>. Acesso em: 08 mar. 2008. 66

HOWARD, J. D. **An analysis of security incidents on the internet**: 1989 - 1995. Tese (PhD Thesis) — Carnegie Mellon University, Pittsburgh, Pennsylvania, Apr. 1997. Disponível em: <http://www.cert.org/research/JHThesis/>. Acesso em: 08 mar. 2008. 27

HOWARD, J. D.; LONGSTAFF, T. A. **A common language for computer security incidents**. [S.l.], Oct. 1998. SAND98-8667. Disponível em: <http://www.cert.org/research/taxonomy_988667.pdf>. Acesso em: 08 mar. 2008. 27, 28, 29

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **International standard**: codes for the representation of currencies and funds, ISO 4217:2001. Aug. 2001. 113

KIENLE, H. M. Exchange format bibliography. **SIGSOFT Softw. Eng. Notes**, ACM Press, New York, NY, USA, v. 26, n. 1, p. 56–60, 2001. ISSN 0163-5948. 45

KLYNE, G.; NEWMAN, C. **RFC 3339**: date and time on the internet - timestamps. July 2002. Disponível em: <http://www.ietf.org/rfc/rfc3339.txt>. Acesso em: 08 mar. 2008. 104, 106

MILO, T.; ABITEBOUL, S.; AMANN, B.; BENJELLOUN, O.; NGOC, F. D. Exchanging intensional xml data. **ACM Trans. Database Syst.**, ACM Press, New York, NY, USA, v. 30, n. 1, p. 1–40, 2005. ISSN 0362-5915. 45

MORIARTY, K. M. **Real-time inter-network defense**. Feb. 2008. Expires on Aug. 22, 2008. Disponível em: <http://www.ietf.org/internet-drafts/draft-moriarty-post-inch-rid-05.txt>. Acesso em: 08 mar. 2008. 116

OETIKER, T. **RRDtool – round robin database tool**. 2008. Disponível em: <http://oss.oetiker.ch/rrdtool/>. Acesso em: 08 mar. 2008. 42

ORCAWARE TECHNOLOGIES. **ORCA – a performance and trend analysis package**. 2006. Disponível em: <http://www.orcaware.com/orca/>. Acesso em: 08 mar. 2008. 42

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. 23, 33, 82

PHILLIPS, A.; DAVIS, M. **RFC 4646**: tags for identifying languages. Sep. 2006. Disponível em: <http://www.ietf.org/rfc/rfc4646.txt>. Acesso em: 08 mar. 2008. 57, 102, 103, 109, 110, 112

PROVOS, N. A virtual honeypot framework. In: USENIX SECURITY SYMPOSIUM, 13., San Diego, CA, USA. **Proceedings...** 2004. p. 1–14. Disponível em: <http://www.usenix.org/publications/library/proceedings/sec04/tech/provos.html>. Acesso em: 17 ago. 2006. 33, 35

PROVOS, N.; HOLZ, T. **Virtual honeypots**: from botnet tracking to intrusion detection. 1. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2007. 480 p. ISBN 978-0-321-33632-3. 23, 24, 33, 34, 35, 74

RAY, E. T.; MCINTOSH, J. **Perl and XML**. 1. ed. Sebastopol, California: O'Reilly, 2002. 216 p. ISBN 0-596-00205-X. 69

SHIREY, R. **RFC 4949**: internet security glossary, version 2. Aug. 2007. http://www.ietf.org/rfc/rfc4949.txt. 27

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHA, S. **Sistema de banco de dados**. São Paulo, SP: Makron Books, 1999. 812 p. ISBN 85-346-1073-8. 69, 153

SPITZNER, L. **Honeypots**: tracking hackers. 1. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2002. 480 p. ISBN 978-0-321-10895-1. 24, 33, 34

_____. **Honeypots**: simple, cost-effective detection. Apr. 2003. Disponível em: <http://www.securityfocus.com/infocus/1690>. Acesso em: 08 mar. 2008. 34

SPITZNER, L.; RANUM, M. Honeypots: tracking hackers. In: SANS 2002 ANNUAL CONFERENCE, Orlando, Florida, USA. **Proceedings...** [S.l.], 2002. 33

STEDING-JESSEN, K.; HOEPERS, C.; MONTES, A. **Consórcio brasileiro de honeypots – projeto honeypots distribuídos**. 2008. Disponível em: <http://www.honeypots-alliance.org.br/index-po.html>. Acesso em: 08 mar. 2008. 42, 72, 73

STOLL, C. Stalking the wily hacker. **Communications of the ACM**, v. 31, p. 484–497, May 1988. 33

_____. **The cuckoo's egg**: tracking a spy through the maze of computer espionage. Garden City, New York: Doubleday, 1989. 326 p. ISBN 0-385-24946-2. 33

THE HONEYNET PROJECT. **Know your enemy**: sebek. Nov. 2003. Disponível em: <http://www.honeynet.org/papers/sebek.pdf>. Acesso em: 08 mar. 2008. 41

_____. **Know your enemy**: learning about security threats. 2. ed. Upper Saddle River, New Jersey: Addison Wesley Professional, 2004. 800 p. ISBN 978-0-321-16646-3. 24, 34, 35, 36, 38, 39, 40, 53, 62

_____. **The honeynet project**: to learn the tools, tactics, and motives of the blackhat community & share the lessons learned. 2008. Disponível em: <http://www.honeynet.org/>. Acesso em: 08 mar. 2008. 34, 38, 40, 50

THE HONEYNET PROJECT & RESEARCH ALLIANCE. **Know your enemy**: honeywall cdrom roo, 3rd generation technology. Aug. 2005. Disponível em: <http://www.honeynet.org/papers/cdrom/roo/>. Acesso em: 08 mar. 2008. 42

THE WORLD WIDE WEB CONSORTIUM (W3C). **Extensible markup language (XML) 1.0, W3C recommendation**. Feb. 2004. Disponível em: <http://www.w3.org/TR/2004/REC-xml-20040204/>. Acesso em: 08 mar. 2008. 45, 65

_____. **XML schema part 2**: datatypes second edition – w3c recommendation. Oct. 2004. Disponível em: <http://www.w3.org/TR/xmlschema-2/>. Acesso em: 08 mar. 2008. 57, 101, 102

_____. **Extensible markup language (XML)**. 2008. Disponível em: <http://www.w3.org/XML/>. Acesso em: 08 mar. 2008. 45

_____. **XML schema**. 2008. Disponível em: <http://www.w3.org/XML/Schema>. Acesso em: 08 mar. 2008. 47

VLIST, E. van der. **XML schema**. 1. ed. Sebastopol, California: O'Reilly and Associates, Inc., 2002. 400 p. ISBN 0-596-00252-1. 47

WIIK, J.; GONZALEZ, J. J.; KOSSAKOWSKI, K.-P. Limits to effectiveness in computer security incident response teams. In: INTERNATIONAL CONFERENCE OF THE SYSTEM DYNAMICS SOCIETY, 23., Boston, MA, USA. **Proceedings...** 2005. Disponível em: <http://www.systemdynamics.org/conf2005/proceed/index.htm>. 48

WOOD, M.; ERLINGER, M. **RFC 4766**: intrusion detection message exchange requirements. Mar. 2007. Disponível em: <http://www.ietf.org/rfc/rfc4766.txt>. Acesso em: 08 mar. 2008. 47

WYK, K. R. van; FORNO, R. **Incident response**. 1. ed. Sebastopol, California: O'Reilly and Associates, Inc., 2001. 234 p. ISBN 0-596-00130-4. 23, 27, 28

YEGNESWARAN, V.; BARFORD, P.; PAXSON, V. Using honeynets for internet situational awareness. In: HOTNETS IV: WORKSHOP ON HOT TOPICS IN NETWORKS, 4., College Park, Maryland, USA. **Proceedings...** [S.l.], 2005. 23, 24, 33, 53

GLOSSÁRIO

Alvo O alvo de um ataque, que pode ser uma máquina (*host*), uma conta de usuário, um programa de computador, um endereço de rede, uma pessoa ou uma organização.

Artefato Qualquer informação deixada por um invasor em um sistema comprometido.

Assinatura Característica específica de um ataque, que possui padrões únicos e pode ser comparada com uma base de regras, de modo a identificar este ataque.

Ataque Série de eventos, causados direta ou indiretamente por uma origem, que violam uma política de segurança de um alvo. Estas violações podem incluir um comprometimento de uma conta privilegiada, uma negação de serviço, furto de informações, entre outros.

Autonomous System (AS) É um conjunto de roteadores sob uma mesma administração técnica, que apresente para outros ASs uma visão consistente sobre suas políticas internas de roteamento e sobre quais destinos são alcançáveis em seu interior. Ou seja, é um conjunto de redes que possuem a mesma política de roteamento.

Autonomous System Number (ASN) É um identificador único, de 16 bits, para um AS. É normalmente alocado pelos *Regional Internet Registries* (RIRs).

Bloco CIDR É identificado de maneira similar a um endereço IP, porém, a porção inicial, composta por números decimais separados por pontos, é seguida de uma barra ("/") e um número de 0 a 32. A porção decimal é interpretada como um conjunto de 32 bits quebrados em octetos. A parte que segue a barra define o tamanho do prefixo, ou seja, quantos bits iniciais são compartilhados pelos IPs daquele bloco CIDR.

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

Confidencialidade Garantia de que informações ou recursos só serão disponibilizados àqueles que possuam autorização.

CSIRT Do inglês *Computer Security Incident Response Team*, Grupo de Resposta a Incidentes de Segurança em Computadores. É o termo que designa o indivíduo, ou grupo de indivíduos, que dá suporte e coordena a resposta a

incidentes para um público alvo bem definido. Um CSIRT cria, processa e mantém uma notificação de incidente.

Disponibilidade Refere-se à habilidade de garantir o uso de informações ou recursos, quando necessário.

Evento Uma ocorrência, em um sistema ou rede, que pode ser de interesse e necessitar de atenção. Um evento pode ou não ser deliberado ou malicioso.

Exploit Programa ou parte de um programa malicioso projetado para explorar uma vulnerabilidade existente em um *software*.

Falso negativo Situação na qual um IDS falha em identificar um ataque quando na verdade ele ocorreu.

Falso positivo Situação na qual um IDS identifica ataques quando não há nenhum.

Flood Uma ação na qual um atacante procura provocar uma falha em um sistema ou rede através do envio de um volume maior de informações do que o sistema ou rede pode tratar.

Gerenciamento de incidentes Conjunto de processos para controlar ou administrar tarefas associadas com incidentes de segurança em computadores. Estes processos habilitam a realização de serviços pró-ativos e reativos, de modo a auxiliar o tratamento de incidentes.

Honeynet Rede projetada especificamente para ser comprometida, e que contém mecanismos de controle para prevenir que seja utilizada como base de ataques contra outras redes.

Honeypot Recurso de segurança especialmente configurado para coletar informações sobre ataques e cujo valor reside em ser sondado, atacado ou comprometido.

IANA (*Internet Assigned Numbers Authority*) Entidade que coordena, globalmente, os servidores DNS raiz, a alocação de endereços IP para os RIRs e a designação de números e nomes aos protocolos Internet.

IDS Do inglês *Intrusion Detection System*, Sistema de Detecção de Intrusões. Sistema que monitora e analiza eventos, com o propósito de alertar sobre atividades maliciosas ou anômalas.

IETF *The Internet Engineering Task Force*, é uma comunidade internacional de operadores de redes, fabricantes de tecnologias e pesquisadores preocupados com a evolução da arquitetura da Internet e sua operação sem problemas.

Impacto Conseqüência de um incidente em um alvo, expressado em termos relevantes para a comunidade de usuários.

Incidente de segurança Qualquer evento adverso, confirmado ou sob suspeita, relacionado à segurança dos sistemas de computação ou das redes de computadores. Ato de violar uma política de segurança, explícita ou implícita.

Integridade Confiabilidade de dados ou recursos. Geralmente, refere-se à habilidade de impedir modificações não autorizadas.

Internet-Draft Documento em construção, sendo desenvolvido por um grupo de trabalho do IETF. Normalmente é um documento que se tornará uma RFC após revisão final e aprovação pelo IESG.

Notificação de incidente Coleção de informações que descrevem um incidente e, normalmente, são enviadas por e/ou para um CSIRT.

Origem A origem de um ataque. Pode ser uma máquina (*host*), uma conta de usuário, um programa de computador, um endereço de rede, uma pessoa ou uma organização.

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.

Probe Uma ação tomada por um atacante na qual ele tenta um acesso a um sistema, de forma a obter alguma informação sobre este sistema.

Resposta a incidentes Ações que devem ser tomadas para se recuperar ou mitigar um incidente e para coordenar e disseminar informações. Muitas vezes também envolve definir estratégias para evitar que outros incidentes similares ocorram.

RFC *Request for Comments* – série de documentos do IETF em que são publicadas boas práticas e padrões usados na Internet. Através de RFCs foram determinados os padrões em uso hoje na Internet, como o TCP/IP, HTTP, DNS, SMTP, entre muitos outros essenciais à sua operação. Após a descrição de uma tecnologia em uma RFC ela está pronta para ser considerada para adoção por diversos fabricantes. Além das RFCs que definem padrões existem também RFCs que discutem problemas, sugerem soluções ou apontam boas práticas.

Regional Internet Registry (RIR) Entidade responsável por gerenciar a distribuição de endereços IP em uma determinada região do mundo. São 5 os

RIRs exitentes: AfriNIC, região da África; APNIC, região da Ásia-Pacífico; ARIN, América do Norte e parte do Caribe; LACNIC, América Latina e parte do Caribe; e RIPE NCC, Europa, Partes da Ásia e do Oriente Médio.

Sanitização de dados Processo de remover ou modificar partes sensíveis de informações que serão enviadas para terceiros. Normalmente as modificações são referentes a configurações de servidores, endereços IP, topologia de redes, entre outras que possam ser consideradas privilegiadas por uma instituição.

***Schema* XML** Descrição formal do que compreende um documento XML válido. Possui tipos de dados pré-definidos, que devem ser respeitados pelos documentos XML que utilizem um determinado *schema*.

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

Spoof Tipo de ação na qual uma entidade ganha acesso a um sistema ou executa uma ação maliciosa ao se fazer passar por uma entidade autorizada.

Scan Ver varredura.

Tratamento de incidentes Conjunto de processos para detecção, notificação, triagem, análise e resposta a incidentes de segurança em computadores.

Varredura Uma ação na qual um atacante envia diversas requisições (*probes*) a um conjunto de endereços em um *host* ou rede, com o objetivo de obter informações sobre o sistema ou sistemas.

VLAN (*Virtual* LAN) Uma separação virtual dentro de um *switch*, que provê uma separação lógica entre LANs, de modo que cada uma se comporta como se estivesse configurada em um *switch* físico separado.

A APÊNDICE A DETALHAMENTO DO MODELO DE DADOS DO IODEF

Neste Apêndice é apresentado o modelo de dados do IODEF, descrito no documento "*RFC 5070: The Incident Object Description Exchange Format*" (DANYLIW et al., 2007). Assim como na RFC 5070, os objetos são representados em UML (BOOCH et al., 2000), porém neste Apêndice o enfoque foi de agrupar visualmente as classes que compõem conjuntos mais específicos de dados. Desta forma é possível ter uma visão mais geral do modelo e do relacionamento entre as classes. É importante destacar, também, que todas as classes representam elementos XML, e seus atributos representam atributos dos elementos XML.

A seguir são apresentados os tipos de dados e as classes do IODEF, seguidos das formas de extensão do modelo e de um exemplo de documento IODEF em XML.

A.1 Tipos de Dados

Na definição das classes e dos atributos do IODEF, sempre que possível, foram utilizados tipos de dados nativos do XML Schema (DANYLIW et al., 2007; W3C, 2004b). Porém, para permitir a representação de dados mais específicos, alguns tipos próprios foram definidos. São eles:

ML_STRING – este tipo é utilizado para representar todos os textos relativos a análises, descrições genéricas ou outras informações que possam ser escritas em múltiplos idiomas. É uma cadeia de caracteres (`xs:string` (W3C, 2004b)) que possui como atributo um idioma (`xs:language` (W3C, 2004b)).

PORTLIST – este tipo representa uma lista de portas de serviços de rede. O formato permite que se listem portas (N) e seqüências de portas (N-M), separadas por vírgulas. Este tipo é uma cadeia de caracteres (`xs:string` (W3C, 2004b)) restringida pela seguinte expressão regular:
`\d+(\-\d+)?(,\d+(\-\d+)?)*`.

ENUM – em diversos atributos são utilizados dados enumerados. No *Schema* do IODEF eles foram definidos como seqüências do tipo de dados NMTOKEN (`xs:NMTOKEN` (W3C, 2004b)). A descrição de cada um será feita juntamente com a descrição da classe que os utiliza pela primeira vez.

HEXBIN – nos casos em que é necessária a representação de dados binários é utilizado formato binário hexadecimal, implementado através do tipo de dados

`xs:hexBinary` do *Schema* XML (W3C, 2004b).

A.2 Classes

Um documento IODEF é uma instância da classe IODEF-Document, podendo esta instância ser composta por uma ou mais instâncias da classe Incident. Este relacionamento pode ser visto na Figura A.1. Os seus atributos obrigatórios são: version, que, segundo a especificação, deve ser "1.00"; e lang, que deve ser um código válido de acordo com a RFC 4646 (PHILLIPS; DAVIS, 2006) e representa o idioma principal usado no documento. Algumas classes agregadas também possuem o atributo lang, permitindo a ocorrência de comentários e descrições em múltiplos idiomas. O atributo formatid é opcional, seu conteúdo é um texto livre, com instruções sobre o processamento. A semântica deste atributo deve ser negociada previamente pelos CSIRTs que estiverem trocando os dados.

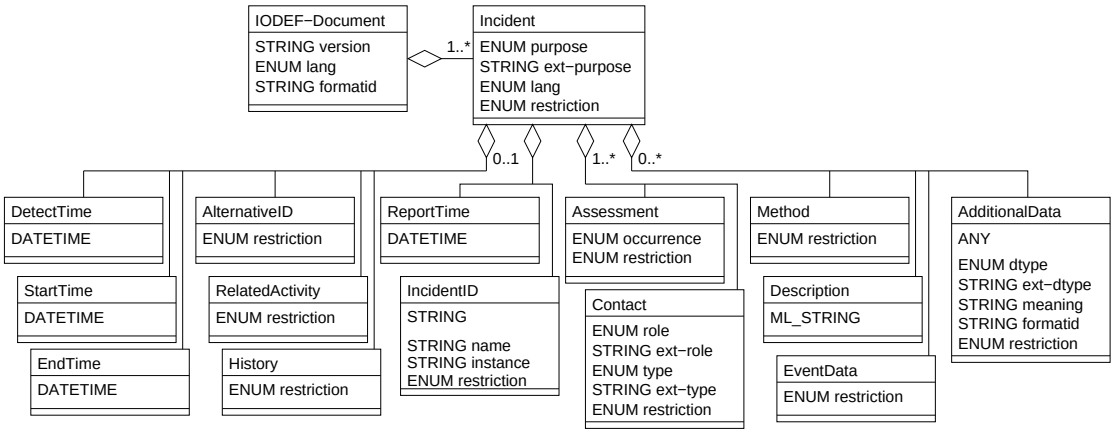

Figura A.1 - Classes IODEF-Document, Incident e classes agregadas.

Fonte: RFC 5070 (DANYLIW et al., 2007).

Cada incidente único é representado por uma instância da classe Incident, que também pode ser vista na Figura A.1. Essa classe fornece uma representação padronizada dos dados mais comumente compartilhados durante o tratamento de um incidente. A classe Incident possui quatro atributos:

- purpose: obrigatório, apresenta a razão pela qual o documento foi criado. Pode ter os valores: traceback, mitigation, reporting, other e ext-value;

- ext-purpose: opcional, usado quando purpose possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1;

- lang: opcional, um código válido de acordo com a RFC 4646 (PHILLIPS; DAVIS, 2006) que representa o idioma usado nesta classe;

- restriction: opcional, indica quais as regras de divulgação de informações que devem ser observadas, por quem receber o documento, com relação ao incidente em análise. Este atributo é herdado por todas as classes filhas de Incident. Algumas dessas classes também declaram o atributo restriction e, nestes casos, é possível especificar novos valores. Os valores possíveis são: public, need-to-know, private e default. Se o valor for default significa que deve ser utilizada uma política pré-acordada entre as partes. Se o atributo não for declarado, deve-se considerá-lo como sendo private.

Um texto com a descrição geral do incidente pode ser informado na classe Description. Já a classe IncidentID associa a cada incidente um identificador, que é relativo ao CSIRT que criou o documento. Tem como atributo obrigatório name, que é o nome do CSIRT que o criou. São opcionais os atributos restriction e instance, este último podendo referenciar um subconjunto específico do incidente em questão.

Segundo o modelo, cada incidente pode ter, além do identificador representado em IncidentID, outros identificadores alternativos, normalmente usados por outros CSIRTs para o mesmo incidente e representados por AlternativeID, que pode ser vista Figura A.2. Esta classe é formada por uma ou mais instâncias de IncidentID e possui como atributo restriction.

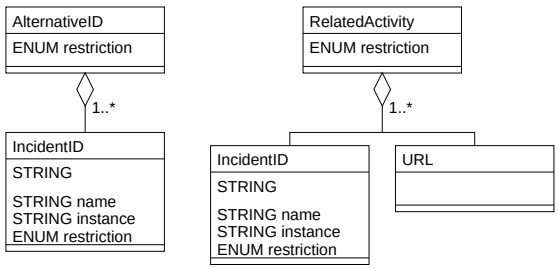

Figura A.2 - Classes AlternativeID, RelatedActivity e classes agregadas.

Fonte: RFC 5070 (DANYLIW et al., 2007).

De maneira similar, é possível ter atividades relacionadas, como outros incidentes que estejam associados com aquele representado na instância Incident em questão. A classe RelatedActivity representa esta informação e pode ter uma ou mais instâncias de IncidentID ou uma ou mais instâncias de URL, onde são informadas URLs de atividades relacionadas. A classe RelatedActivity pode ser vista com detalhes na Figura A.2.

Como apresentado na Figura A.1, cada incidente pode ter a ele associados quatro tempos distintos, representados pelas seguintes classes:

ReportTime: uma ocorrência, data e horário em que o incidente foi reportado;

DetectTime: zero ou uma ocorrência, data e horário em que o incidente foi detectado pela primeira vez;

StartTime: zero ou uma ocorrência, data e horário em que o incidente teve início;

EndTime: zero ou uma ocorrência, data e horário em que o incidente terminou.

O modelo recomenda que o horário presente em ReportTime seja aquele da criação do documento IODEF. As classes referentes a tempo no IODEF seguem o padrão definido na RFC 3339 (KLYNE; NEWMAN, 2002).

A classe EventData, mostrada juntamente com suas classes agregadas na Figura A.3, descreve os eventos relacionados a um conjunto de máquinas ou redes envolvidas em um incidente.

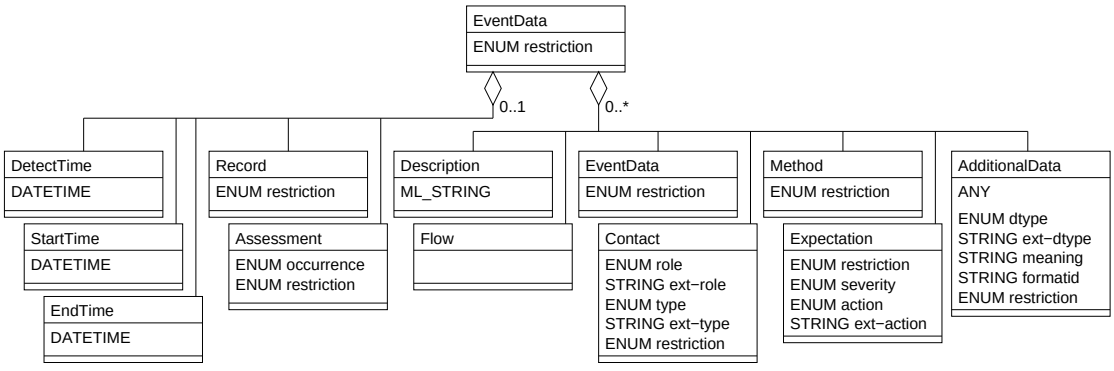

Figura A.3 - Classe EventData e classes agregadas.

Fonte: RFC 5070 (DANYLIW et al., 2007).

A classe EventData é recursiva, de modo poder a atuar como um *container* de eventos relacionados com um incidente, onde cada instância mais específica de EventData contém as informações de um único evento, como *logs* e dados sobre os sistemas envolvidos. Cada instância de EventData também possui seus próprios tempos associados (StartTime, EndTime, DetectTime), mas neste contexto eles dizem respeito a cada evento, e não ao incidente como um todo. Cada evento pode ter uma restrição diferente, representada pelo atributo restriction e descrições associadas, representadas pela classe Description.

A classe agregada Record, vista em detalhes na Figura A.4, é composta por uma ou mais instâncias da classe RecordData, sendo que estas instâncias armazenam informações de *logs* e de auditoria dos sistemas envolvidos no incidente. Estas informações são o *timestamp* (DateTime) do item associado (RecordItem), um padrão pelo qual procurar nos dados (RecordPattern), a aplicação que gerou os dados (Application) uma descrição geral sobre estes dados (Description) e um conjunto de informações adicionais (AdditionalData), que também podem ser utilizadas para estender o modelo, como será visto na seção A.2.1.

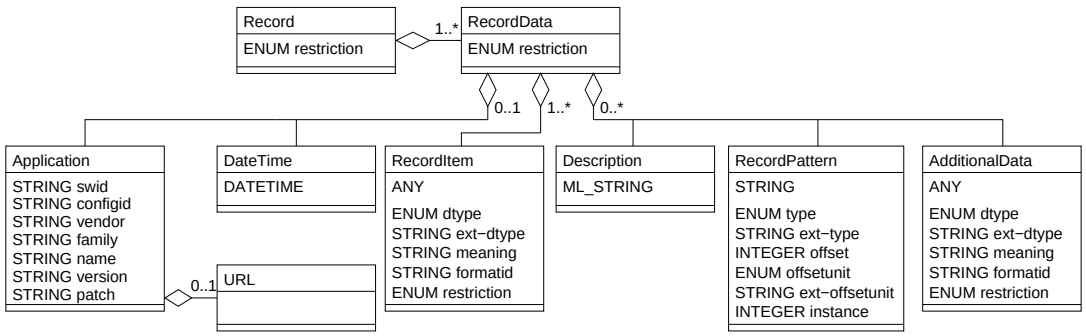

Figura A.4 - Classe Record e classes agregadas.

Fonte: RFC 5070 (DANYLIW et al., 2007).

A classe RecordItem permite incorporar ao documento IODEF *logs*, trilhas de auditoria ou outros dados relevantes para dar suporte ao tratamento do incidente. Esta classe permite o encapsulamento de dados ou a indicação de um local onde estes dados estejam armazenados, e possui cinco atributos. O único atributo obrigatório é dtype, através do qual é declarado o tipo do dado contido em RecordItem. Este

atributo pode ter um dos seguintes valores: boolean; byte; character; date-time (uma data e hora, conforme a RFC 3339 (KLYNE; NEWMAN, 2002)); integer; portlist (uma lista de portas); real; string; file (um arquivo binário em formato *base64*); frame (um *frame* de dados, nível 2 de rede, em formato HEXBIN); packet (um pacote de dados, nível 3 de rede, em formato HEXBIN); ipv4-packet (o conteúdo de um pacote ipv4, em formato HEXBIN); ipv6-packet (o conteúdo de um pacote ipv6, em formato HEXBIN); path (caminho completo de um arquivo); url; csv (uma lista de valores separados por vírgulas); winreg (uma *string* contendo um *Windows Registry key*); xml (um documento XML, de acordo com um *Schema* externo); e ext-value. Os atributos opcionais de RecordItem são: ext-dtype, usado quando dtype possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1; meaning, uma descrição do significado do conteúdo; formatid, um identificador que referencia o formato e a semântica do conteúdo; e restriction.

A classe RecordPattern permite definir um padrão pelo qual procurar em RecordItem. Os seus atributos são: type, obrigatório, que pode ter os valores regex (uma expressão regular), binary (um padrão do tipo HEXBIN), xpath (XML *Path*) e ext-value; ext-type, opcional, usado quando type possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1; offset, opcional, indica um número de unidades (especificadas em offsetunit) a avançar em RecordItem antes de encontrar o padrão procurado; offsetunit, opcional, unidade do atributo offset, podendo ter os valores line, binary (número de bytes) e ext-value (o padrão é "line"); ext-offsetunit, opcional, usado quando offsetunit possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1; instance, opcional, indica o número de vezes que o padrão deve ser aplicado ao dado presente em RecordItem.

Na classe Application podem-se armazenar informações sobre a aplicação usada para gerar o dado armazenado em RecordItem. Ela pode ter zero ou uma classe URL agregada, indicando onde podem ser encontradas mais informações sobre a aplicação. Seus atributos, todos opcionais, são: swid, um número identificador; configid, um número identificador que pode ser associado a uma configuração particular; vendor, o nome do fabricante ou desenvolvedor do *software*; family, uma determinada família de *software*; name, o nome do *software*; version a versão do *software*; e patch, indica as correções aplicadas ao *software*.

A classe Flow, que pode ser vista na Figura A.5, agrupa os sistemas origem e destino em um evento, representados pela classe System, que por sua vez, descreve um

computador ou rede envolvido em um evento.

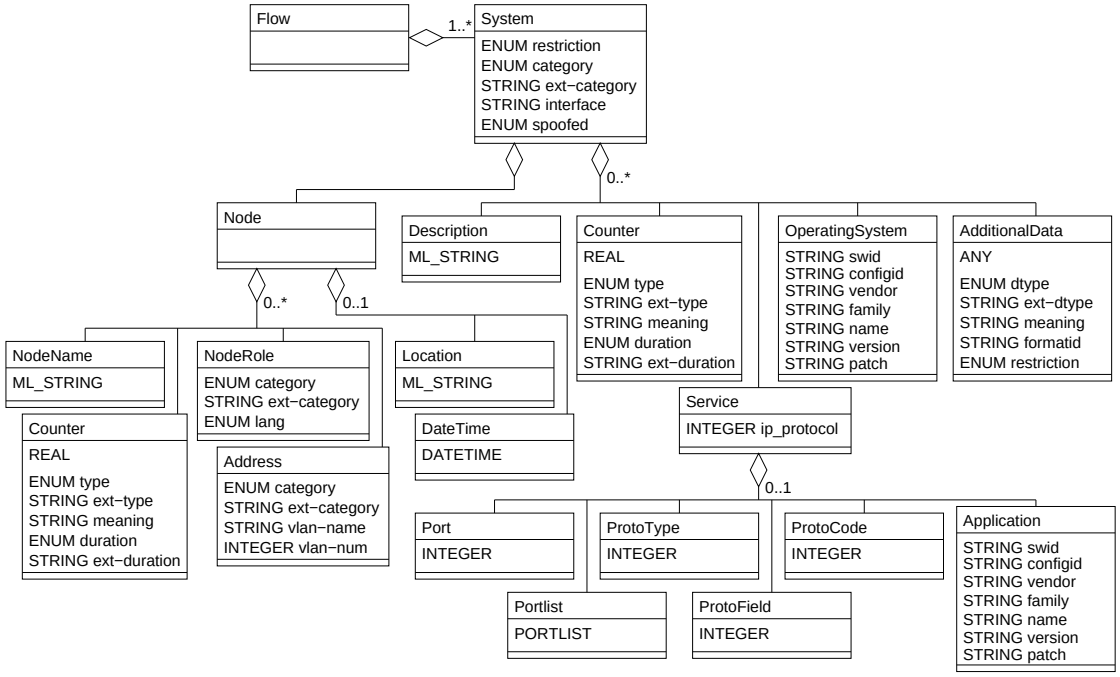

Figura A.5 - Classes Flow, System e classes agregadas.

Fonte: RFC 5070 (DANYLIW et al., 2007).

Os sistemas ou redes descritos pela classe System são categorizados de acordo com seu papel no incidente, representado pelo atributo obrigatório category. Este atributo pode ter um dos valores: source (a origem do evento), target (o alvo do evento), intermediate (um sistema intermediário), sensor (um sistema monitorando o evento), infrastructure (o sistema faz parte da infra-estrutura) e ext-value. Os outros atributos, opcionais, são: restriction; ext-category, opcional, usado quando category possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1; interface, utilizado caso o sistema seja um computador, define a interface de rede onde os eventos se originaram; spoofed, indicação da confiança de que o sistema ou rede é mesmo a origem ou alvo do evento, pode ter os valores unknown, yes ou no.

Na classe Description é possível colocar um texto com uma descrição geral do sistema. Já a classe OperatingSystem, que possui definição idêntica à da classe Application, permite descrever os detalhes de um Sistema Operacional executando em um sistema.

A classe Counter permite sumarizar ocorrências múltiplas de um evento, contagens ou taxas sobre informações como pacotes, seções, etc. A sua semântica depende da classe à qual está agregada e, em System, a classe Counter, que pode ocorrer zero ou mais vezes, é usada para sumarizar propriedades do *host* ou rede sendo representado. O seu atributo type, obrigatório, especifica a unidade do conteúdo de Counter, seus valores possíveis são: byte, packet, flow, session, alert, message, event, host, site, organization e ext-value. O atributo ext-type, opcional, é usado quando type possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1. O atributo meaning, opcional, pode conter uma descrição geral do conteúdo. Se o atributo duration estiver presente, significa que a classe Counter representa uma taxa, com seu conteúdo sendo o numerador e o conteúdo deste atributo o denominador desta taxa. Seus possíveis valores são second, minute, hour, day, month, quarter, year e ext-value. O atributo ext-duration, opcional, é usado quando duration possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1.

Em Node é feita a identificação da máquina, rede ou dispositivo envolvido no evento. Esta classe é derivada do IDMEF, porém, foi estendida com a adição das classes DateTime, NodeRole e Counter. As classes agregadas de Node são:

NodeName: zero ou mais, representa as maneiras como uma máquina pode ser chamada, como por exemplo o nome DNS. Esta classe deve estar presente caso não exista ao menos uma instância de Address.

Address: zero ou mais, representa um endereço físico (nível 2), de rede (nível 3) ou de aplicação (nível 7). Esta classe deve estar presente caso não exista ao menos uma instância de NodeName. O atributo category é obrigatório, define qual o tipo de endereço que a classe contém, os valores possíveis são asn, atm, e-mail, ipv4-addr, ipv4-net, ipv4-net-mask, ipv6-addr, ipv6-net, ipv6-net-mask, mac ou ext-value. O atributo ext-category, opcional, é usado quando category possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1. Os outros dois atributos opcionais são vlan-name e vlan-num, utilizados para designar a VLAN (*Virtual LAN*) a que um sistema pertence.

Location: zero ou mais, uma descrição da localização física da máquina.

DateTime: zero ou uma, *timestamp* do momento em que e foi feita a resolução entre nome e endereço, sendo recomendado que esse dado seja informado caso NodeName e Address sejam especificados.

NodeRole: zero ou mais, o papel desempenhado pela máquina descrita em Node. O atributo category, obrigatório, indica a sua funcionalidade e pode ter um dos seguintes valores: client, server-internal, server-public, www, mail, messaging, streaming, voice, file, ftp, p2p, name (e.g. DNS, WINS), directory, credential (e.g. Kerberos), print, application, database, infra (e.g. roteador, *firewall*, DHCP), log ou ext-value. O atributo ext-category, opcional, é usado quando category possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1. O atributo lang, opcional, é um código válido de acordo com a RFC 4646 (PHILLIPS; DAVIS, 2006) que representa o idioma usado nesta classe.

Counter: zero ou mais, permite sumarizar propriedades da máquina, rede ou dispositivo representado em Node.

Service pode ocorrer zero ou mais vezes e descreve um serviço de rede. No atributo ip_protocol deve ser informado o número do protocolo junto ao IANA. As classes agregadas de Service, que podem ocorrer zero ou uma vez, são:

Port: uma porta associada a um serviço.

Portlist: um conjunto de portas associadas a um serviço.

ProtoCode: código de um protocolo (nível 4).

ProtoType: tipo de um protocolo (nível 4).

ProtoField: campo de um protocolo (nível 4), como por exemplo *flags* TCP.

Application: uma aplicação associada à porta ou portas informadas em Port ou Portlist.

A classe Contact, vista na Figura A.6, faz parte tanto de Incident quanto de EventData. Esta classe descreve as informações de contato de pessoas ou organizações envolvidas no incidente, permitindo a identificação de seu papel. Pessoas podem ser associadas com organizações através do uso da recursão.

O atributo obrigatório role, que indica o papel do contato, pode ter um destes valores: creator (quem gerou o documento), admin (contato administrativo), tech (contato técnico), irt (CSIRT que está tratando o incidente), cc (contato que deve ser mantido informado sobre o progresso) ou ext-value. O atributo ext-role, opcional, é usado quando role possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1. O atributo type, obrigatório, indica o tipo de contato descrito na classe,

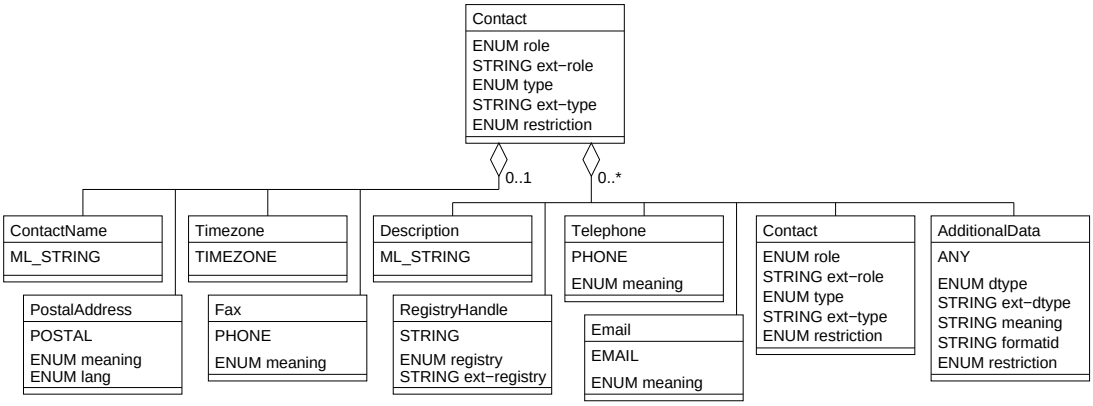

Figura A.6 - Classe Contact e classes agregadas.

Fonte: RFC 5070 (DANYLIW et al., 2007).

pode ter os seguintes valores: person, organization ou ext-value. O atributo ext-type, opcional, é usado quando type possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1. A classe Contact é formada pelas seguintes classes agregadas:

ContactName: zero ou uma, o nome do contato, que pode ser uma organização ou uma pessoa.

Description: zero ou mais, uma descrição do contato ou, em casos de pessoas, seu cargo ou função dentro da organização.

RegistryHandle: zero ou mais, representa a identificação na base de dados de um *Internet Registry* ou em uma base local. O atributo registry, obrigatório, indica qual é o *Registry* e pode ter os valores internic, apnic, arin, lacnic, ripe, afrinic, local ou ext-value. O atributo ext-registry, opcional, é usado quando registry possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1.

PostalAddress: zero ou uma, o endereço postal do contato. O atributo meaning, opcional, pode conter uma descrição geral do conteúdo. lang, atributo obrigatório, é um código válido de acordo com a RFC 4646 (PHILLIPS; DAVIS, 2006) que representa o idioma usado nesta classe.

Email: zero ou mais, um endereço de *e-mail* do contato. O atributo meaning, opcional, pode conter uma descrição geral do conteúdo.

Telephone: zero ou mais, um telefone do contato. O atributo meaning, opcional, pode conter uma descrição geral do conteúdo ou observações, como horários em que o contato estará disponível naquele telefone.

Fax: zero ou uma, um número de FAX do contato. O atributo meaning, opcional, pode conter uma descrição geral do conteúdo ou observações.

Timezone: zero ou uma, o fuso horário em que o contato se encontra.

AdditionalData: zero ou mais, tem definição igual à da classe RecordItem e permite representar informações adicionais, bem como estender o modelo, como será visto na seção A.2.1.

A classe Expectation, mostrada na Figura A.7, está agregada à classe EventData e informa, a quem estiver recebendo a informação sobre o incidente, qual a ação sendo requisitada por quem criou o documento IODEF.

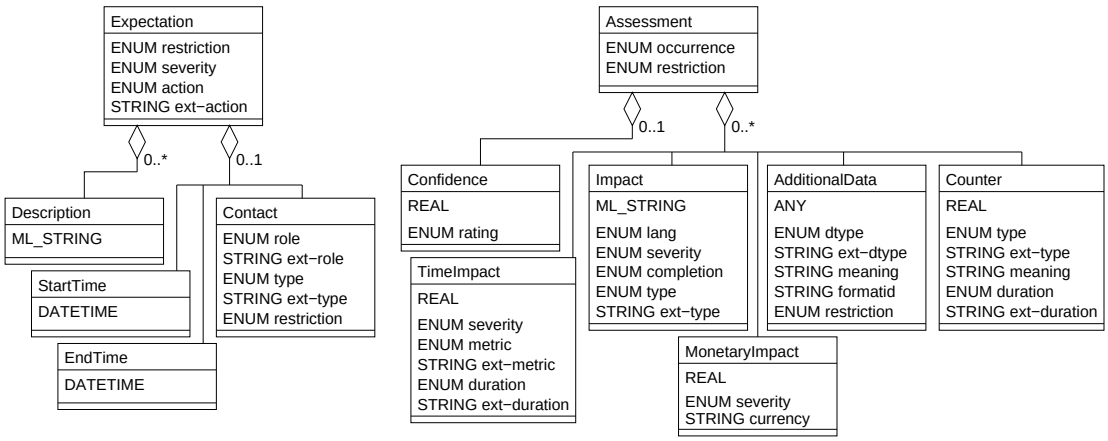

Figura A.7 - Classes Expectation, Assessment e classes agregadas.

Fonte: RFC 5070 (DANYLIW et al., 2007).

Seus atributos, todos opcionais, são: restriction; severity, indica a prioridade que deve ser dada ao incidente, seu valor pode ser low, medium ou high; action, classifica o tipo de ação requerida, seu valor pode ser nothing, contact-source-site, contact-target-site, contact-sender, investigate, block-host, block-network, block-port, rate-limit-host, rate-limit-network, rate-limit-port, remediate-other, status-triage, status-new-info, other e ext-value; O atributo ext-action é usado quando action possui o valor ext-value, permite estender

o modelo, como será visto na seção A.2.1. A classe Expectation é formada pelas seguintes classes agregadas:

Description: zero ou mais, pode ter uma descrição mais detalhada das ações esperadas.

StartTime: zero ou uma, uma data e horário em que se espera que a ação seja realizada. Caso a data seja anterior àquela presente em ReportTime de Incident, isto significa que a ação deve ser realizada o mais rápido possível. A ausência deste elemento deixa a execução à escolha do destinatário.

EndTime: zero ou uma, uma data e horário limite para a execução da ação. Caso a ação não tenha sido realizada até esta hora, não necessita mais ser realizada.

Contact: zero ou uma, quem se espera que realize esta ação.

Na classe Assessment, vista na Figura A.7, são descritas as repercussões técnicas e não técnicas de um incidente. Quando esta classe está ligada diretamente à classe Incident ela se refere ao impacto geral do incidente, já quando está ligada à classe EventData se refere aos impactos específicos de cada evento que faz parte do incidente. Seus atributos, opcionais, são: restriction; e ocurrence, que especifica se os impactos descritos ocorreram ou são resultados em potencial, pode ter os valores actual e potential. A classe Assessment é formada pelas seguintes classes agregadas:

Impact: zero ou mais, permite categorizar e descrever o impacto técnico de um incidente na rede de uma organização. O atributo lang, obrigatório, é um código válido de acordo com a RFC 4646 (PHILLIPS; DAVIS, 2006) que representa o idioma usado nesta classe. O atributo opcional severity permite estimar a severidade de uma notificação, podendo ter os valores low, medium ou high. O atributo completion é opcional e indica se o ataque teve sucesso ou não, seus valores possíveis são failed ou succeeded. No atributo type, obrigatório, é informada a categoria de ataque ocorrido. Os valores possíveis são: admin, dos, file, info-leak, misconfiguration, policy, recon, social-engineering, user unknown, ou ext-value. O atributo ext-type, opcional, é usado quando type possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1.

TimeImpact: zero ou mais, descreve o impacto em função do tempo em que houve interrupção de atividades e do tempo de recuperação. A semântica é dada

pelos seus atributos. O atributo opcional severity define a severidade e pode ter os valores low, medium ou high. Em metric, obrigatório, é possível representar com base em qual fator foi feito o cálculo, as opções são: labor, elapsed, downtime ou ext-value. O atributo ext-metric, opcional, é usado quando type possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1. O atributo obrigatório duration define uma unidade de tempo que, combinada com o valor do atributo metric, descreve uma métrica do impacto que será representado no elemento da classe TimeImpact, seus valores podem ser second, minute, hour, day, month, quarter, year ou ext-value. O atributo ext-duration, opcional, é usado quando duration possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1.

MonetaryImpact: zero ou mais, valor numérico que representa o impacto financeiro de um incidente sobre uma organização, possui dois atributos: severity, opcional, define a severidade e pode ter os valores low, medium ou high; e currency, obrigatório, a moeda em que o valor está expresso, cujos valores permitidos são aqueles definidos na norma ISO 4217:2001 (ISO, 2001).

Counter: zero ou mais, um contador com o qual é possível sumarizar a magnitude da atividade.

Confidence: zero ou uma, descreve uma estimativa da validade e corretude da informação sobre o impacto. O atributo rating, obrigatório, é uma avaliação da validade dos dados fornecidos na classe Assessment, podendo ter os valores low, medium, high ou numeric. Se o valor for numeric o conteúdo de Confidence é um número que expressa a confiança. Se o valor for um dos outros, o valor de Confidence deve ser vazio.

AdditionalData: zero ou mais, tem definição igual à da classe RecordItem e permite representar informações adicionais, bem como estender o modelo, como será visto na seção A.2.1.

A classe Method, representada na Figura A.8, descreve a metodologia usada pelo atacante para perpetrar os eventos que fazem parte do incidente. Quando esta classe está ligada a Incident ela representa um resumo ou informações sobre a metodologia mais relevante usada pelo atacate. Quanto estiver ligada a EventData representará a metodologia específica para cada evento ligado ao incidente.

Esta classe possui um único atributo, opcional, que é restriction, permitindo definir uma restrição diferente para a divulgação de seus dados. As classes agregadas que

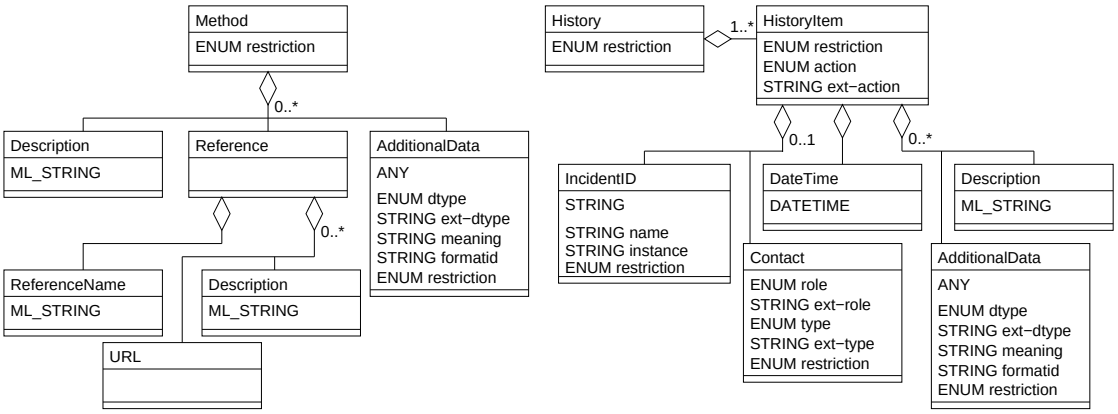

Figura A.8 - Classes Method, History e classes agregadas.

Fonte: RFC 5070 (DANYLIW et al., 2007).

compõem Method são:

Reference: zero ou mais, contém uma referência à vulnerabilidade, alerta, exemplar de código malicioso, advisory ou técnica de ataque. Essa referência é mapeada nas seguintes classes agregadas: ReferenceName (nome da referência); URL (URL a ela associada); e Description (uma descrição geral).

Description: zero ou mais, uma descrição geral do método utilizado.

AdditionalData: zero ou mais, tem definição igual à da classe RecordItem e permite representar informações adicionais, bem como estender o modelo, como será visto na seção A.2.1.

A classe History (Figura A.8) está associada à classe Incident e agrega registros de ações tomadas pelas partes envolvidas ao longo da resolução do incidente. Este histórico é composto de uma ou mais instâncias da classe HistoryItem, que possui os seguintes atributos: restriction, opcional; action, obrigatório, classifica a ação ou ocorrência documentada em HistoryItem, podendo ter os valores nothing, contact-source-site, contact-target-site, contact-sender, investigate, block-host, block-network, block-port, rate-limit-host, rate-limit-network, rate-limit-port, remediate-other, status-triage, status-new-info, other e ext-value; e ext-action, opcional, que é usado quando action possui o valor ext-value, permite estender o modelo, como será visto na seção A.2.1. HistoryItem é composta pelas seguintes classes agregadas:

DateTime: uma, a data e horário em que estes dados foram incluídos no histórico do incidente.

IncidentID: zero ou uma, pode ser utilizada quando mais de um CSIRT for responsável pelas ações, deste modo o ID associado ao incidente por aquele CSIRT pode ser colocado aqui para identificação.

Contact: zero ou uma, contato da pessoa que realizou as ações descritas.

Description: zero ou mais, uma descrição das ações realizadas.

AdditionalData: zero ou mais, tem definição igual à da classe RecordItem e permite representar informações adicionais, bem como estender o modelo, como será visto na seção A.2.1.

A.2.1 Extensões ao Modelo

O IODEF prevê que possam ser feitas extensões, permitindo a representação de novos dados sem ser necessário modificar o modelo. Caso sejam criadas extensões bem documentadas e amplamente utilizadas pela comunidade, elas poderão vir a fazer parte de uma próxima versão do modelo (DANYLIW et al., 2007).

Existem basicamente duas maneiras de estender o modelo do IODEF:

Extensão de valores enumerados de atributos: todos os atributos enumerados, cujo nome seja da forma "ext-<nome>", são utilizados para estender o atributo de respectivo "<nome>". Um exemplo são os atributos type e ext-type, presentes em diversas classes do IODEF. Caso os tipos que eles definem possuam sua classificação ampliada no futuro, o modelo pode ser estendido atribuindo-se ao atributo type o valor "ext-value" e ao atributo ext-type o valor do novo tipo.

Extensão das classes: a classe AdditionalData, que está agregada a diversas classes do IODEF, pode ser utilizada como um mecanismo de extensão, permitindo a inclusão de informações que não estejam representadas no modelo de dados ou a inclusão de novas classes. Ela comporta elementos atômicos, como inteiros e cadeias de caracteres, mas também permite a representação de dados mais complexos, como novos documentos XML inteiros. Quando o conteúdo desta classe é do tipo XML, é possível encapsular documentos que sejam referentes a outros *Schemas*, como por exemplo um documento IDMEF.

Atualmente já existem duas extensões sendo discutidas no âmbito do IETF:

- ***Real-time Inter-network Defense*** – propõe uma extensão ao IODEF para permitir coletar dados que facilitem um método de comunicação pró-ativa, para trocar informações e integrar os mecanismos de provedores de *backbone*, permitindo o rastreamento de ataques e a identificação de sua origem. Esta extensão propõe o uso do *namespace* iofef-rid (MORIARTY, 2008).

- ***Extension to the IODEF-Document Class for Phishing, Fraud, and Other Crimeware*** – propõe uma extensão que permita a notificação de incidentes como fraude, *spam*, *phishing* e outros que não necessariamente estejam associados a eventos de rede. Ele é amplo para permitir que outros tipos de crimes cometidos através da Internet possam também ser representados. Esta extensão propõe o uso do *namespace* phish (CAIN; JEVANS, 2007).

A.3 Exemplo de Documento IODEF

O IODEF utiliza o padrão XML 1.0, define o *namespace* "iodef" e um *Schema* próprio, cuja definição completa está disponível em (DANYLIW et al., 2007). Para ilustrar uma notificação de incidente usando o IODEF é mostrado na Figura A.9 um exemplo da RFC 5070 (DANYLIW et al., 2007), que ilustra uma notificação de um incidente relativo a uma máquina infectada pelo *worm* Code Red (CERT/CC, 2001).

No documento da Figura A.9 os dados do incidente estão organizados de forma a facilitar o processamento automatizado dos dados. Ele apresenta uma expectativa, que é impedir que a máquina infectada continue fazendo varreduras na rede (`<Expectation action="block-host" />`), possui um trecho do *log* em RecordItem e indica em outra instância de RecordItem em qual URL podem ser encontrados *logs* adicionais. Além disso, para fins de importação para um banco de dados do CSIRT de destino, os dados permitem identificar facilmente a origem e o destino do ataque, bem como quem mais já foi contatado.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This example demonstrates a report for a very old worm (Code Red) -->
<IODEF-Document version="1.00" lang="en"
  xmlns="urn:ietf:params:xml:ns:iodef-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:schema:iodef-1.0">
  <Incident purpose="reporting">
    <IncidentID name="csirt.example.com">189493</IncidentID>
    <ReportTime>2001-09-13T23:19:24+00:00</ReportTime>
    <Description>Host sending out Code Red probes</Description>
    <!-- An administrative privilege was attempted, but failed -->
    <Assessment><Impact completion="failed" type="admin"/></Assessment>
    <Contact role="creator" type="organization">
      <ContactName>Example.com CSIRT</ContactName>
      <RegistryHandle registry="arin">example-com</RegistryHandle>
      <Email>contact@csirt.example.com</Email>
    </Contact>
    <EventData>
      <Flow>
        <System category="source">
          <Node>
            <Address category="ipv4-addr">192.0.2.200</Address>
            <Counter type="event">57</Counter>
          </Node>
        </System>
        <System category="target">
          <Node>
            <Address category="ipv4-net">192.0.2.16/28</Address>
          </Node>
          <Service ip_protocol="6"><Port>80</Port></Service>
        </System>
      </Flow>
      <Expectation action="block-host" />
      <!-- <RecordItem> has an excerpt from a log -->
      <Record>
        <RecordData>
          <DateTime>2001-09-13T18:11:21+02:00</DateTime>
          <Description>Web-server logs</Description>
          <RecordItem dtype="string">
          192.0.2.1 - - [13/Sep/2001:18:11:21 +0200] "GET /default.ida?
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          </RecordItem>
          <!-- Additional logs -->
          <RecordItem dtype="url">http://mylogs.example.com/logs/httpd_access</RecordItem>
        </RecordData>
      </Record>
    </EventData>
    <History>
      <!-- Contact was previously made with the source network owner -->
      <HistoryItem action="contact-source-site">
        <DateTime>2001-09-14T08:19:01+00:00</DateTime>
        <Description>Notification sent to constituency-contact@192.0.2.200</Description>
      </HistoryItem>
    </History>
  </Incident>
</IODEF-Document>
```

Figura A.9 - Exemplo de notificação de incidente em formato IODEF.

Fonte: RFC 5070 (DANYLIW et al., 2007).

B APÊNDICE B HIDEF SCHEMA

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema targetNamespace="http://www.hideas.org/namespaces/hidef-1.0"
3             xmlns="http://www.hideas.org/namespaces/hidef-1.0"
4             xmlns:hidef="http://www.hideas.org/namespaces/hidef-1.0"
5             xmlns:xs="http://www.w3.org/2001/XMLSchema"
6             elementFormDefault="qualified"
7             attributeFormDefault="unqualified">
8
9    <xs:annotation>
10     <xs:documentation>
11     Honeypots Information and Data Exchange Format (HIDEF) -- v1.00
12     </xs:documentation>
13   </xs:annotation>
14
15 <!-- ********************************************************** -->
16 <!-- ********************************************************** -->
17 <!-- *** Declaration of classes defined in HIDEF              *** -->
18 <!-- ********************************************************** -->
19 <!-- ********************************************************** -->
20 <!--
21 The System, Description, EventData, AdditionalData and their child
22 classes are derived from IODEF for compatibility purposes.
23 The full description of the IODEF Standard is provided in the document
24 "RFC 5070: The Incident Object Description Exchange Format", available
25 at: http://www.ietf.org/rfc/rfc5070.txt
26 -->
27 <!-- ********************************************************** -->
28 <!-- *** HIDEF-Document Class                                 *** -->
29 <!-- ********************************************************** -->
30
31   <xs:element name="HIDEF-Document">
32     <xs:complexType>
33       <xs:sequence>
34         <xs:element ref="hidef:Honeypot"
35                     maxOccurs="unbounded"/>
36       </xs:sequence>
37       <xs:attribute name="version"
38                     type="xs:string" fixed="1.00"/>
39       <xs:attribute name="lang"
40                     type="xs:language" use="required"/>
41       <xs:attribute name="instructions"
42                     type="xs:string"/>
43     </xs:complexType>
44   </xs:element>
```

```
45
46  <!-- ********************************************************** -->
47  <!-- *** Honeypot Class                                     *** -->
48  <!-- ********************************************************** -->
49
50    <xs:element name="Honeypot">
51      <xs:complexType>
52        <xs:sequence>
53          <!-- *** Zero or one occurrence *** -->
54          <xs:element ref="hidef:System"
55                      minOccurs="0"/>
56          <xs:element ref="hidef:AdministrativeInfo"
57                      minOccurs="0"/>
58          <xs:element ref="hidef:FilterInfo"
59                      minOccurs="0"/>
60          <xs:element ref="hidef:EmulatedSystems"
61                      minOccurs="0"/>
62          <xs:element ref="hidef:Solution"
63                      minOccurs="0"/>
64          <xs:element ref="hidef:Description"
65                      minOccurs="0"/>
66          <xs:element ref="hidef:Honeynet"
67                      minOccurs="0"/>
68          <!-- *** Zero or more occurrences *** -->
69          <xs:element ref="hidef:EventData"
70                      minOccurs="0" maxOccurs="unbounded"/>
71          <xs:element ref="hidef:AdditionalData"
72                      minOccurs="0" maxOccurs="unbounded"/>
73        </xs:sequence>
74        <!-- *** Attributes *** -->
75        <xs:attribute name="restriction"
76                      type="hidef:restriction-type" default="private"/>
77        <xs:attribute name="type" use="required">
78          <xs:simpleType>
79            <xs:restriction base="xs:NMTOKEN">
80              <xs:enumeration value="low-interaction"/>
81              <xs:enumeration value="high-interaction"/>
82              <xs:enumeration value="ext-value"/>
83            </xs:restriction>
84          </xs:simpleType>
85        </xs:attribute>
86        <xs:attribute name="ext-type"
87                      type="xs:string" use="optional"/>
88      </xs:complexType>
89    </xs:element>
90
91  <!-- ********************************************************** -->
```

```
92  <!-- *** System Class                                                  *** -->
93  <!-- ************************************************************** -->
94
95    <xs:element name="System">
96      <xs:complexType>
97        <xs:sequence>
98          <!-- *** Exactly one occurrence *** -->
99          <xs:element ref="hidef:Node"/>
100         <!-- *** Zero or more occurrences *** -->
101         <xs:element ref="hidef:Service"
102                     minOccurs="0" maxOccurs="unbounded"/>
103         <xs:element ref="hidef:OperatingSystem"
104                     minOccurs="0" maxOccurs="unbounded"/>
105         <xs:element ref="hidef:Counter"
106                     minOccurs="0" maxOccurs="unbounded"/>
107         <xs:element ref="hidef:Description"
108                     minOccurs="0" maxOccurs="unbounded"/>
109         <xs:element ref="hidef:AdditionalData"
110                     minOccurs="0" maxOccurs="unbounded"/>
111       </xs:sequence>
112       <!-- *** Attributes *** -->
113       <xs:attribute name="restriction"
114                     type="hidef:restriction-type"/>
115       <xs:attribute name="interface"
116                     type="xs:string"/>
117       <xs:attribute name="category">
118         <xs:simpleType>
119           <xs:restriction base="xs:NMTOKEN">
120             <xs:enumeration value="source"/>
121             <xs:enumeration value="target"/>
122             <xs:enumeration value="intermediate"/>
123             <xs:enumeration value="sensor"/>
124             <xs:enumeration value="infrastructure"/>
125             <xs:enumeration value="ext-value"/>
126           </xs:restriction>
127         </xs:simpleType>
128       </xs:attribute>
129       <xs:attribute name="ext-category"
130                     type="xs:string" use="optional"/>
131       <xs:attribute name="spoofed"
132                     default="unknown">
133         <xs:simpleType>
134           <xs:restriction base="xs:NMTOKEN">
135             <xs:enumeration value="unknown"/>
136             <xs:enumeration value="yes"/>
137             <xs:enumeration value="no"/>
138           </xs:restriction>
```

```
139         </xs:simpleType>
140       </xs:attribute>
141     </xs:complexType>
142   </xs:element>
143
144 <!-- ********************************************************** -->
145 <!-- *** Node class                                          *** -->
146 <!-- ********************************************************** -->
147
148   <xs:element name="Node">
149     <xs:complexType>
150       <xs:sequence>
151         <!-- *** Zero or more occurrences *** -->
152         <xs:choice maxOccurs="unbounded">
153           <xs:element name="NodeName"
154                       type="hidef:MLStringType" minOccurs="0"/>
155           <xs:element ref="hidef:Address"
156                       minOccurs="0" maxOccurs="unbounded"/>
157         </xs:choice>
158         <!-- *** Zero or one occurrences *** -->
159         <xs:element ref="hidef:Location"
160                     minOccurs="0"/>
161         <xs:element ref="hidef:DateTime"
162                     minOccurs="0"/>
163         <!-- *** Zero or more occurrences *** -->
164         <xs:element ref="hidef:NodeRole"
165                     minOccurs="0" maxOccurs="unbounded"/>
166         <xs:element ref="hidef:Counter"
167                     minOccurs="0" maxOccurs="unbounded"/>
168       </xs:sequence>
169     </xs:complexType>
170   </xs:element>
171
172 <!-- ********************************************************** -->
173 <!-- *** Address Class                                       *** -->
174 <!-- ********************************************************** -->
175
176   <xs:element name="Address">
177     <xs:complexType>
178       <xs:simpleContent>
179         <xs:extension base="xs:string">
180           <!-- *** Attributes *** -->
181           <xs:attribute name="category" default="ipv4-addr">
182             <xs:simpleType>
183               <xs:restriction base="xs:NMTOKEN">
184                 <xs:enumeration value="asn"/>
185                 <xs:enumeration value="atm"/>
```

```
186                 <xs:enumeration value="e-mail"/>
187                 <xs:enumeration value="mac"/>
188                 <xs:enumeration value="ipv4-addr"/>
189                 <xs:enumeration value="ipv4-net"/>
190                 <xs:enumeration value="ipv4-net-mask"/>
191                 <xs:enumeration value="ipv6-addr"/>
192                 <xs:enumeration value="ipv6-net"/>
193                 <xs:enumeration value="ipv6-net-mask"/>
194                 <xs:enumeration value="ext-value"/>
195               </xs:restriction>
196             </xs:simpleType>
197           </xs:attribute>
198           <xs:attribute name="ext-category"
199                         type="xs:string" use="optional"/>
200          <xs:attribute name="vlan-name"
201                         type="xs:string"/>
202           <xs:attribute name="vlan-num"
203                         type="xs:integer"/>
204         </xs:extension>
205       </xs:simpleContent>
206     </xs:complexType>
207   </xs:element>
208
209 <!-- ********************************************************** -->
210 <!-- *** Location Class                                     *** -->
211 <!-- ********************************************************** -->
212
213   <xs:element name="Location"
214               type="hidef:MLStringType"/>
215
216 <!-- ********************************************************** -->
217 <!-- *** NodeRole Class                                     *** -->
218 <!-- ********************************************************** -->
219
220   <xs:element name="NodeRole">
221     <xs:complexType>
222       <xs:simpleContent>
223         <xs:extension base="hidef:MLStringType">
224           <!-- *** Attributes *** -->
225           <xs:attribute name="category" use="required">
226             <xs:simpleType>
227               <xs:restriction base="xs:NMTOKEN">
228                 <xs:enumeration value="client"/>
229                 <xs:enumeration value="server-internal"/>
230                 <xs:enumeration value="server-public"/>
231                 <xs:enumeration value="www"/>
232                 <xs:enumeration value="mail"/>
```

```
                <xs:enumeration value="messaging"/>
                <xs:enumeration value="streaming"/>
                <xs:enumeration value="voice"/>
                <xs:enumeration value="file"/>
                <xs:enumeration value="ftp"/>
                <xs:enumeration value="p2p"/>
                <xs:enumeration value="name"/>
                <xs:enumeration value="directory"/>
                <xs:enumeration value="credential"/>
                <xs:enumeration value="print"/>
                <xs:enumeration value="application"/>
                <xs:enumeration value="database"/>
                <xs:enumeration value="infra"/>
                <xs:enumeration value="log"/>
                <xs:enumeration value="ext-value"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="ext-category"
                        type="xs:string" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

<!-- ********************************************************** -->
<!-- *** Service Class                                      *** -->
<!-- ********************************************************** -->

  <xs:element name="Service">
    <xs:complexType>
      <xs:sequence>
        <!-- *** Zero or one occurrence *** -->
        <xs:choice minOccurs="0">
          <xs:element name="Port"
                      type="xs:integer"/>
          <xs:element name="Portlist"
                      type="hidef:PortlistType"/>
        </xs:choice>
        <xs:element name="ProtoType"
                    type="xs:integer" minOccurs="0"/>
        <xs:element name="ProtoCode"
                    type="xs:integer" minOccurs="0"/>
        <xs:element name="ProtoField"
                    type="xs:integer" minOccurs="0"/>
        <xs:element ref="hidef:Application"
                    minOccurs="0"/>
```

```
      </xs:sequence>
      <!-- *** Attributes *** -->
      <xs:attribute name="ip_protocol"
                    type="xs:integer" use="required"/>
    </xs:complexType>
  </xs:element>

<!-- ********************************************************** -->
<!-- *** Counter Class                                      *** -->
<!-- ********************************************************** -->

  <xs:element name="Counter">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:double">
          <!-- *** Attributes *** -->
          <xs:attribute name="type" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="byte"/>
                <xs:enumeration value="packet"/>
                <xs:enumeration value="flow"/>
                <xs:enumeration value="session"/>
                <xs:enumeration value="event"/>
                <xs:enumeration value="alert"/>
                <xs:enumeration value="message"/>
                <xs:enumeration value="host"/>
                <xs:enumeration value="site"/>
                <xs:enumeration value="organization"/>
                <xs:enumeration value="ext-value"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="ext-type"
                        type="xs:string" use="optional"/>
          <xs:attribute name="meaning"
                        type="xs:string" use="optional"/>
          <xs:attribute name="duration"
                        type="hidef:duration-type"/>
          <xs:attribute name="ext-duration"
                        type="xs:string" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

<!-- ********************************************************** -->
```

```
<!-- *** Record Class                                                *** -->
<!-- ************************************************************** -->

  <xs:element name="Record">
    <xs:complexType>
      <xs:sequence>
        <!-- *** One or more occurrences *** -->
        <xs:element ref="hidef:RecordData"
                    maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- *** Attribute *** -->
      <xs:attribute name="restriction"
                    type="hidef:restriction-type"/>
    </xs:complexType>
  </xs:element>

<!-- ************************************************************** -->
<!-- *** RecordData Class                                            *** -->
<!-- ************************************************************** -->

  <xs:element name="RecordData">
    <xs:complexType>
      <xs:sequence>
        <!-- *** Zero or one occurrence *** -->
        <xs:element ref="hidef:DateTime"
                    minOccurs="0"/>
        <!-- *** Zero or more occurrences *** -->
        <xs:element ref="hidef:Description"
                    minOccurs="0" maxOccurs="unbounded"/>
        <!-- *** Zero or one occurrence *** -->
        <xs:element ref="hidef:Application"
                    minOccurs="0"/>
        <!-- *** Zero or more occurrences *** -->
        <xs:element ref="hidef:RecordPattern"
                    minOccurs="0" maxOccurs="unbounded"/>
        <!-- *** One or more occurrences *** -->
        <xs:element ref="hidef:RecordItem"
                    maxOccurs="unbounded"/>
        <!-- *** Zero or more occurrences *** -->
        <xs:element ref="hidef:AdditionalData"
                    minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- *** Attribute *** -->
      <xs:attribute name="restriction"
                    type="hidef:restriction-type"/>
    </xs:complexType>
  </xs:element>
```

```
374
375 <!-- ********************************************************** -->
376 <!-- *** RecordPattern Class                                 *** -->
377 <!-- ********************************************************** -->
378
379   <xs:element name="RecordPattern">
380     <xs:complexType>
381       <xs:simpleContent>
382         <xs:extension base="xs:string">
383         <!-- *** Attributes *** -->
384           <xs:attribute name="type" use="required">
385             <xs:simpleType>
386               <xs:restriction base="xs:NMTOKEN">
387                 <xs:enumeration value="regex"/>
388                 <xs:enumeration value="binary"/>
389                 <xs:enumeration value="xpath"/>
390                 <xs:enumeration value="ext-value"/>
391               </xs:restriction>
392             </xs:simpleType>
393           </xs:attribute>
394           <xs:attribute name="ext-type"
395                         type="xs:string" use="optional"/>
396           <xs:attribute name="offset"
397                         type="xs:integer" use="optional"/>
398           <xs:attribute name="offsetunit"
399                         use="optional" default="line">
400             <xs:simpleType>
401               <xs:restriction base="xs:NMTOKEN">
402                 <xs:enumeration value="line"/>
403                 <xs:enumeration value="byte"/>
404                 <xs:enumeration value="ext-value"/>
405               </xs:restriction>
406             </xs:simpleType>
407           </xs:attribute>
408           <xs:attribute name="ext-offsetunit"
409                         type="xs:string" use="optional"/>
410           <xs:attribute name="instance"
411                         type="xs:integer" use="optional"/>
412         </xs:extension>
413       </xs:simpleContent>
414     </xs:complexType>
415   </xs:element>
416
417 <!-- ********************************************************** -->
418 <!-- *** RecordItem Class                                    *** -->
419 <!-- ********************************************************** -->
420
```

```
421   <xs:element name="RecordItem"
422               type="hidef:ExtensionType"/>
423
424 <!-- ********************************************************** -->
425 <!-- *** Description Class                                  *** -->
426 <!-- ********************************************************** -->
427
428   <xs:element name="Description"
429               type="hidef:MLStringType"/>
430
431 <!-- ********************************************************** -->
432 <!-- *** URL Class                                          *** -->
433 <!-- ********************************************************** -->
434
435   <xs:element name="URL"
436               type="xs:anyURI"/>
437
438 <!-- ********************************************************** -->
439 <!-- *** Application                                        *** -->
440 <!-- ********************************************************** -->
441
442   <xs:element name="Application"
443               type="hidef:SoftwareType"/>
444
445 <!-- ********************************************************** -->
446 <!-- *** OperatingSystem                                    *** -->
447 <!-- ********************************************************** -->
448
449   <xs:element name="OperatingSystem"
450               type="hidef:SoftwareType"/>
451
452 <!-- ********************************************************** -->
453 <!-- *** AdditionalData                                     *** -->
454 <!-- ********************************************************** -->
455
456   <xs:element name="AdditionalData"
457               type="hidef:ExtensionType"/>
458
459 <!-- ********************************************************** -->
460 <!-- *** AdministrativeInfo Class                           *** -->
461 <!-- ********************************************************** -->
462
463   <xs:element name="AdministrativeInfo">
464     <xs:complexType>
465       <xs:sequence>
466         <!-- *** Zero or more occurrences *** -->
467         <xs:element ref="hidef:EmulatedAddress"
```

```
468                     minOccurs="0" maxOccurs="unbounded"/>
469         <xs:element ref="hidef:Sanitization"
470                     minOccurs="0" maxOccurs="unbounded"/>
471         <xs:element ref="hidef:Hardware"
472                     minOccurs="0" maxOccurs="unbounded"/>
473         <xs:element ref="hidef:Contact"
474                     minOccurs="0" maxOccurs="unbounded"/>
475         <xs:element ref="hidef:AdditionalData"
476                     minOccurs="0" maxOccurs="unbounded"/>
477       </xs:sequence>
478       <!-- *** Attribute *** -->
479       <xs:attribute name="restriction"
480                     type="hidef:restriction-type"/>
481     </xs:complexType>
482   </xs:element>
483
484 <!-- ********************************************************** -->
485 <!-- *** EmulatedAddress Class                                *** -->
486 <!-- ********************************************************** -->
487
488   <xs:element name="EmulatedAddress">
489     <xs:complexType>
490       <xs:simpleContent>
491         <xs:extension base="xs:string">
492           <!-- *** Attributes *** -->
493           <xs:attribute name="category" default="ipv4-addr">
494             <xs:simpleType>
495               <xs:restriction base="xs:NMTOKEN">
496                 <xs:enumeration value="asn"/>
497                 <xs:enumeration value="atm"/>
498                 <xs:enumeration value="e-mail"/>
499                 <xs:enumeration value="mac"/>
500                 <xs:enumeration value="ipv4-addr"/>
501                 <xs:enumeration value="ipv4-net"/>
502                 <xs:enumeration value="ipv4-net-mask"/>
503                 <xs:enumeration value="ipv6-addr"/>
504                 <xs:enumeration value="ipv6-net"/>
505                 <xs:enumeration value="ipv6-net-mask"/>
506                 <xs:enumeration value="ext-value"/>
507               </xs:restriction>
508             </xs:simpleType>
509           </xs:attribute>
510           <xs:attribute name="ext-category"
511                         type="xs:string" use="optional"/>
512           <xs:attribute name="vlan-name"
513                         type="xs:string"/>
514           <xs:attribute name="vlan-num"
```

```
                        type="xs:integer"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>

<!-- ********************************************************** -->
<!-- *** Sanitization Class                                  *** -->
<!-- ********************************************************** -->

    <xs:element name="Sanitization">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <!-- *** Attributes *** -->
            <xs:attribute name="type" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                  <xs:enumeration value="asn"/>
                  <xs:enumeration value="atm"/>
                  <xs:enumeration value="e-mail"/>
                  <xs:enumeration value="mac"/>
                  <xs:enumeration value="ipv4-addr"/>
                  <xs:enumeration value="ipv4-net"/>
                  <xs:enumeration value="ipv4-net-mask"/>
                  <xs:enumeration value="ipv6-addr"/>
                  <xs:enumeration value="ipv6-net"/>
                  <xs:enumeration value="ipv6-net-mask"/>
                  <xs:enumeration value="ext-value"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="ext-type"
                          type="xs:string" use="optional"/>
            <xs:attribute name="real"
                          type="xs:string"/>
            <xs:attribute name="sanitized"
                          type="xs:string"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>

<!-- ********************************************************** -->
<!-- *** Hardware Class                                      *** -->
<!-- ********************************************************** -->

```

```
562   <xs:element name="Hardware"
563               type="hidef:MLStringType"/>
564
565 <!-- ********************************************************** -->
566 <!-- *** Contact Class                                      *** -->
567 <!-- ********************************************************** -->
568
569   <xs:element name="Contact">
570     <xs:complexType>
571       <xs:sequence>
572         <!-- *** Zero or one occurrence *** -->
573         <xs:element ref="hidef:ContactName"
574                     minOccurs="0"/>
575         <!-- *** Zero or more occurrences *** -->
576         <xs:element ref="hidef:Description"
577                     minOccurs="0" maxOccurs="unbounded"/>
578         <xs:element ref="hidef:RegistryHandle"
579                     minOccurs="0" maxOccurs="unbounded"/>
580         <!-- *** Zero or one occurrence *** -->
581         <xs:element ref="hidef:PostalAddress"
582                     minOccurs="0"/>
583         <!-- *** Zero or more occurrences *** -->
584         <xs:element ref="hidef:Email"
585                     minOccurs="0" maxOccurs="unbounded"/>
586         <xs:element ref="hidef:Telephone"
587                     minOccurs="0" maxOccurs="unbounded"/>
588         <!-- *** Zero or one occurrence *** -->
589         <xs:element ref="hidef:Fax"
590                     minOccurs="0"/>
591         <xs:element ref="hidef:Timezone"
592                     minOccurs="0"/>
593         <!-- *** Zero or more occurrences *** -->
594         <xs:element ref="hidef:Contact"
595                     minOccurs="0" maxOccurs="unbounded"/>
596         <xs:element ref="hidef:AdditionalData"
597                     minOccurs="0" maxOccurs="unbounded"/>
598       </xs:sequence>
599       <!-- *** Attributes *** -->
600       <xs:attribute name="role" use="required">
601         <xs:simpleType>
602           <xs:restriction base="xs:NMTOKEN">
603             <xs:enumeration value="creator"/>
604             <xs:enumeration value="admin"/>
605             <xs:enumeration value="tech"/>
606             <xs:enumeration value="irt"/>
607             <xs:enumeration value="cc"/>
608             <xs:enumeration value="ext-value"/>
```

```
609             </xs:restriction>
610           </xs:simpleType>
611         </xs:attribute>
612         <xs:attribute name="ext-role"
613                       type="xs:string" use="optional"/>
614         <xs:attribute name="type" use="required">
615           <xs:simpleType>
616             <xs:restriction base="xs:NMTOKEN">
617               <xs:enumeration value="person"/>
618               <xs:enumeration value="organization"/>
619               <xs:enumeration value="ext-value"/>
620             </xs:restriction>
621           </xs:simpleType>
622         </xs:attribute>
623         <xs:attribute name="ext-type"
624                       type="xs:string" use="optional"/>
625         <xs:attribute name="restriction"
626                       type="hidef:restriction-type"/>
627       </xs:complexType>
628     </xs:element>
629
630 <!-- ********************************************************** -->
631 <!-- *** ContactName Class                                  *** -->
632 <!-- ********************************************************** -->
633
634     <xs:element name="ContactName"
635                 type="hidef:MLStringType"/>
636
637 <!-- ********************************************************** -->
638 <!-- *** RegistryHandle Class                               *** -->
639 <!-- ********************************************************** -->
640
641     <xs:element name="RegistryHandle">
642       <xs:complexType>
643         <xs:simpleContent>
644           <xs:extension base="xs:string">
645             <xs:attribute name="registry">
646               <xs:simpleType>
647                 <xs:restriction base="xs:NMTOKEN">
648                   <xs:enumeration value="internic"/>
649                   <xs:enumeration value="apnic"/>
650                   <xs:enumeration value="arin"/>
651                   <xs:enumeration value="lacnic"/>
652                   <xs:enumeration value="ripe"/>
653                   <xs:enumeration value="afrinic"/>
654                   <xs:enumeration value="local"/>
655                   <xs:enumeration value="ext-value"/>
```

```
656               </xs:restriction>
657             </xs:simpleType>
658           </xs:attribute>
659           <xs:attribute name="ext-registry"
660                         type="xs:string" use="optional"/>
661         </xs:extension>
662       </xs:simpleContent>
663     </xs:complexType>
664   </xs:element>
665
666 <!-- ********************************************************** -->
667 <!-- *** PostalAddress Class                                  *** -->
668 <!-- ********************************************************** -->
669
670   <xs:element name="PostalAddress">
671     <xs:complexType>
672       <xs:simpleContent>
673         <xs:extension base="hidef:MLStringType">
674           <xs:attribute name="meaning"
675                         type="xs:string" use="optional"/>
676         </xs:extension>
677       </xs:simpleContent>
678     </xs:complexType>
679   </xs:element>
680
681 <!-- ********************************************************** -->
682 <!-- *** Email Class                                          *** -->
683 <!-- ********************************************************** -->
684
685   <xs:element name="Email"
686               type="hidef:ContactMeansType"/>
687
688 <!-- ********************************************************** -->
689 <!-- *** Telephone Class                                      *** -->
690 <!-- ********************************************************** -->
691
692   <xs:element name="Telephone"
693               type="hidef:ContactMeansType"/>
694
695 <!-- ********************************************************** -->
696 <!-- *** Fax Class                                            *** -->
697 <!-- ********************************************************** -->
698
699   <xs:element name="Fax"
700               type="hidef:ContactMeansType"/>
701
702 <!-- ********************************************************** -->
```

```
703 <!-- *** FilterInfo Class                                          *** -->
704 <!-- ************************************************************ -->
705
706   <xs:element name="FilterInfo">
707     <xs:complexType>
708       <xs:sequence>
709         <!-- *** Zero or one occurrence *** -->
710         <xs:choice minOccurs="0">
711           <xs:element name="Port"
712                       type="xs:integer"/>
713           <xs:element name="Portlist"
714                       type="hidef:PortlistType"/>
715         </xs:choice>
716         <!-- *** Zero or more occurrences *** -->
717         <xs:element ref="hidef:AdditionalData"
718                     minOccurs="0" maxOccurs="unbounded"/>
719       </xs:sequence>
720       <!-- *** Attributes *** -->
721       <xs:attribute name="type" use="required">
722         <xs:simpleType>
723           <xs:restriction base="xs:NMTOKEN">
724             <xs:enumeration value="open"/>
725             <xs:enumeration value="closed"/>
726           </xs:restriction>
727         </xs:simpleType>
728       </xs:attribute>
729       <xs:attribute name="ip_version" default="ipv4">
730         <xs:simpleType>
731           <xs:restriction base="xs:NMTOKEN">
732             <xs:enumeration value="ipv4"/>
733             <xs:enumeration value="ipv6"/>
734             <xs:enumeration value="ext-value"/>
735           </xs:restriction>
736         </xs:simpleType>
737       </xs:attribute>
738       <xs:attribute name="ext-category"
739                     type="xs:string" use="optional"/>
740       <xs:attribute name="ip_protocol"
741                     type="xs:integer" use="required"/>
742     </xs:complexType>
743   </xs:element>
744
745 <!-- ************************************************************ -->
746 <!-- *** EmulatedSystems Class                                    *** -->
747 <!-- ************************************************************ -->
748
749   <xs:element name="EmulatedSystems">
```

```
750       <xs:complexType>
751         <xs:sequence>
752           <!-- *** One or more occurrences *** -->
753           <xs:element ref="hidef:System"
754                       maxOccurs="unbounded"/>
755         </xs:sequence>
756         <!-- *** Attribute *** -->
757         <xs:attribute name="restriction"
758                       type="hidef:restriction-type"/>
759       </xs:complexType>
760     </xs:element>
761
762   <!-- ********************************************************** -->
763   <!-- *** Solution Class                                     *** -->
764   <!-- ********************************************************** -->
765
766     <xs:element name="Solution">
767       <xs:complexType>
768         <xs:sequence>
769           <!-- *** Zero or one occurrence *** -->
770           <xs:element ref="hidef:Description"
771                       minOccurs="0"/>
772           <xs:element ref="hidef:URL"
773                       minOccurs="0"/>
774           <!-- *** Zero or more occurrences *** -->
775           <xs:element ref="hidef:AdditionalData"
776                       minOccurs="0" maxOccurs="unbounded"/>
777         </xs:sequence>
778       </xs:complexType>
779     </xs:element>
780
781   <!-- ********************************************************** -->
782   <!-- *** Honeynet Class                                     *** -->
783   <!-- ********************************************************** -->
784
785     <xs:element name="Honeynet">
786       <xs:complexType>
787         <xs:sequence>
788           <!-- *** Zero or one occurrence *** -->
789           <xs:element ref="hidef:Description"
790                       minOccurs="0"/>
791           <!-- *** Zero or more occurrences *** -->
792           <xs:element ref="hidef:Address"
793                       minOccurs="0" maxOccurs="unbounded"/>
794           <xs:element ref="hidef:DataCollection"
795                       minOccurs="0" maxOccurs="unbounded"/>
796           <xs:element ref="hidef:DataControl"
```

```
                    minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="hidef:AdditionalData"
                    minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- *** Attributes *** -->
      <xs:attribute name="restriction"
                    type="hidef:restriction-type"/>
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="physical"/>
            <xs:enumeration value="virtual"/>
            <xs:enumeration value="ext-value"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="ext-type"
                    type="xs:string" use="optional"/>
    </xs:complexType>
  </xs:element>

<!-- ********************************************************** -->
<!-- *** DataCollection Class                                *** -->
<!-- ********************************************************** -->

  <xs:element name="DataCollection">
    <xs:complexType>
      <xs:sequence>
        <!-- *** Zero or one occurrence *** -->
        <xs:element ref="hidef:Description"
                    minOccurs="0"/>
        <xs:element ref="hidef:URL"
                    minOccurs="0"/>
        <!-- *** Zero or more occurrences *** -->
        <xs:element ref="hidef:AdditionalData"
                    minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

<!-- ********************************************************** -->
<!-- *** DataControl Class                                  *** -->
<!-- ********************************************************** -->

  <xs:element name="DataControl">
    <xs:complexType>
      <xs:sequence>
```

```
844         <!-- *** Zero or one occurrence *** -->
845         <xs:element ref="hidef:Description"
846                     minOccurs="0"/>
847         <xs:element ref="hidef:URL"
848                     minOccurs="0"/>
849         <!-- *** Zero or more occurrences *** -->
850         <xs:element ref="hidef:AdditionalData"
851                     minOccurs="0" maxOccurs="unbounded"/>
852       </xs:sequence>
853     </xs:complexType>
854   </xs:element>
855
856 <!-- ********************************************************** -->
857 <!-- *** ArtifactData Class                                 *** -->
858 <!-- ********************************************************** -->
859
860   <xs:element name="ArtifactData">
861     <xs:complexType>
862       <xs:sequence>
863         <!-- *** One occurrence *** -->
864         <xs:element ref="hidef:DateTime"/>
865         <!-- *** Zero or one occurrence *** -->
866         <xs:element ref="hidef:Description"
867                     minOccurs="0"/>
868         <!-- *** Exactly one of the following three *** -->
869         <xs:choice>
870           <xs:element ref="Base64Item"/>
871           <xs:element ref="HexItem"/>
872           <xs:element ref="TextItem"/>
873         </xs:choice>
874         <!-- *** Zero or more occurrences *** -->
875         <xs:element ref="hidef:Hash"
876                     minOccurs="0" maxOccurs="unbounded"/>
877         <xs:element ref="hidef:URL"
878                     minOccurs="0" maxOccurs="unbounded"/>
879         <xs:element ref="hidef:AdditionalData"
880                     minOccurs="0" maxOccurs="unbounded"/>
881       </xs:sequence>
882     </xs:complexType>
883   </xs:element>
884
885 <!-- ********************************************************** -->
886 <!-- *** Base64Item Class                                   *** -->
887 <!-- ********************************************************** -->
888
889   <xs:element name="Base64Item">
890     <xs:complexType>
```

```
891         <xs:simpleContent>
892           <xs:extension base="xs:base64Binary">
893             <!-- *** Attributes *** -->
894             <xs:attribute name="b64type" use="required">
895               <xs:simpleType>
896                 <xs:restriction base="xs:NMTOKEN">
897                   <xs:enumeration value="virus"/>
898                   <xs:enumeration value="worm"/>
899                   <xs:enumeration value="bot"/>
900                   <xs:enumeration value="spyware"/>
901                   <xs:enumeration value="trojan"/>
902                   <xs:enumeration value="backdoor"/>
903                   <xs:enumeration value="exploit"/>
904                   <xs:enumeration value="rootkit"/>
905                   <xs:enumeration value="gen-toolkit"/>
906                   <xs:enumeration value="irc-log"/>
907                   <xs:enumeration value="undefined"/>
908                   <xs:enumeration value="ext-value"/>
909                 </xs:restriction>
910               </xs:simpleType>
911             </xs:attribute>
912             <xs:attribute name="ext-b64type"
913                           type="xs:string" use="optional"/>
914           </xs:extension>
915         </xs:simpleContent>
916       </xs:complexType>
917     </xs:element>
918
919   <!-- ********************************************************** -->
920   <!-- *** HexItem Class                                      *** -->
921   <!-- ********************************************************** -->
922
923     <xs:element name="HexItem">
924       <xs:complexType>
925         <xs:simpleContent>
926           <xs:extension base="xs:hexBinary">
927             <!-- *** Attributes *** -->
928             <xs:attribute name="hextype" use="required">
929               <xs:simpleType>
930                 <xs:restriction base="xs:NMTOKEN">
931                   <xs:enumeration value="virus"/>
932                   <xs:enumeration value="worm"/>
933                   <xs:enumeration value="bot"/>
934                   <xs:enumeration value="spyware"/>
935                   <xs:enumeration value="trojan"/>
936                   <xs:enumeration value="backdoor"/>
937                   <xs:enumeration value="exploit"/>
```

```
938                 <xs:enumeration value="rootkit"/>
939                 <xs:enumeration value="gen-toolkit"/>
940                 <xs:enumeration value="irc-log"/>
941                 <xs:enumeration value="undefined"/>
942                 <xs:enumeration value="ext-value"/>
943               </xs:restriction>
944             </xs:simpleType>
945           </xs:attribute>
946           <xs:attribute name="ext-hextype"
947                         type="xs:string" use="optional"/>
948         </xs:extension>
949       </xs:simpleContent>
950     </xs:complexType>
951   </xs:element>
952
953 <!-- ********************************************************** -->
954 <!-- *** TextItem Class                                     *** -->
955 <!-- ********************************************************** -->
956
957   <xs:element name="TextItem">
958     <xs:complexType>
959       <xs:simpleContent>
960         <xs:extension base="xs:string">
961           <!-- *** Attributes *** -->
962           <xs:attribute name="texttype" use="required">
963             <xs:simpleType>
964               <xs:restriction base="xs:NMTOKEN">
965                 <xs:enumeration value="virus"/>
966                 <xs:enumeration value="worm"/>
967                 <xs:enumeration value="bot"/>
968                 <xs:enumeration value="spyware"/>
969                 <xs:enumeration value="trojan"/>
970                 <xs:enumeration value="backdoor"/>
971                 <xs:enumeration value="exploit"/>
972                 <xs:enumeration value="rootkit"/>
973                 <xs:enumeration value="gen-toolkit"/>
974                 <xs:enumeration value="irc-log"/>
975                 <xs:enumeration value="winreg"/>
976                 <xs:enumeration value="logs"/>
977                 <xs:enumeration value="email"/>
978                 <xs:enumeration value="undefined"/>
979                 <xs:enumeration value="ext-value"/>
980               </xs:restriction>
981             </xs:simpleType>
982           </xs:attribute>
983           <xs:attribute name="ext-texttype"
984                         type="xs:string" use="optional"/>
```

```
985           </xs:extension>
986         </xs:simpleContent>
987       </xs:complexType>
988     </xs:element>
989
990   <!-- ********************************************************** -->
991   <!-- *** Hash Class                                         *** -->
992   <!-- ********************************************************** -->
993
994     <xs:element name="Hash">
995       <xs:complexType>
996         <xs:simpleContent>
997           <xs:extension base="xs:string">
998             <!-- *** Attributes *** -->
999             <xs:attribute name="htype"
1000                           type="hidef:hash-type" use="required"/>
1001            <xs:attribute name="ext-htype"
1002                          type="xs:string" use="optional"/>
1003          </xs:extension>
1004        </xs:simpleContent>
1005      </xs:complexType>
1006    </xs:element>
1007
1008  <!-- ********************************************************** -->
1009  <!-- *** EventData Class                                    *** -->
1010  <!-- ********************************************************** -->
1011
1012    <xs:element name="EventData">
1013      <xs:complexType>
1014        <xs:sequence>
1015          <!-- *** Zero or more occurrences *** -->
1016          <xs:element ref="hidef:Description"
1017                      minOccurs="0" maxOccurs="unbounded"/>
1018          <!-- *** Zero or one occurrence *** -->
1019          <xs:element ref="hidef:DetectTime"
1020                      minOccurs="0"/>
1021          <xs:element ref="hidef:StartTime"
1022                      minOccurs="0"/>
1023          <xs:element ref="hidef:EndTime"
1024                      minOccurs="0"/>
1025          <!-- *** Zero or more occurrences *** -->
1026          <xs:element ref="hidef:Contact"
1027                      minOccurs="0" maxOccurs="unbounded"/>
1028          <!-- *** Zero or one occurrence *** -->
1029          <xs:element ref="hidef:Assessment"
1030                      minOccurs="0"/>
1031          <!-- *** Zero or more occurrences *** -->
```

```
1032          <xs:element ref="hidef:Method"
1033                      minOccurs="0" maxOccurs="unbounded"/>
1034          <xs:element ref="hidef:Flow"
1035                      minOccurs="0" maxOccurs="unbounded"/>
1036          <xs:element ref="hidef:Expectation"
1037                      minOccurs="0" maxOccurs="unbounded"/>
1038          <!-- *** Zero or one occurrence *** -->
1039          <xs:element ref="hidef:Record"
1040                      minOccurs="0"/>
1041          <!-- *** Zero or more occurrences *** -->
1042          <xs:element ref="hidef:EventData"
1043                      minOccurs="0" maxOccurs="unbounded"/>
1044          <xs:element ref="hidef:ArtifactData"
1045                      minOccurs="0" maxOccurs="unbounded"/>
1046          <xs:element ref="hidef:AdditionalData"
1047                      minOccurs="0" maxOccurs="unbounded"/>
1048        </xs:sequence>
1049        <!-- *** Attribute *** -->
1050        <xs:attribute name="restriction"
1051                      type="hidef:restriction-type" default="default"/>
1052      </xs:complexType>
1053    </xs:element>
1054
1055  <!-- ********************************************************** -->
1056  <!-- *** Flow Class                                         *** -->
1057  <!-- ********************************************************** -->
1058
1059    <xs:element name="Flow">
1060      <xs:complexType>
1061        <xs:sequence>
1062          <!-- *** One or more occurrences *** -->
1063          <xs:element ref="hidef:System"
1064                      maxOccurs="unbounded"/>
1065        </xs:sequence>
1066      </xs:complexType>
1067    </xs:element>
1068
1069  <!-- ********************************************************** -->
1070  <!-- *** Expectation Class                                  *** -->
1071  <!-- ********************************************************** -->
1072
1073    <xs:element name="Expectation">
1074      <xs:complexType>
1075        <xs:sequence>
1076          <!-- *** Zero or more occurrences *** -->
1077          <xs:element ref="hidef:Description"
1078                      minOccurs="0" maxOccurs="unbounded"/>
```

```
1079          <!-- *** Zero or one occurrence *** -->
1080          <xs:element ref="hidef:StartTime"
1081                      minOccurs="0"/>
1082          <xs:element ref="hidef:EndTime"
1083                      minOccurs="0"/>
1084          <xs:element ref="hidef:Contact"
1085                      minOccurs="0"/>
1086        </xs:sequence>
1087        <!-- *** Attributes *** -->
1088        <xs:attribute name="restriction"
1089                      type="hidef:restriction-type" default="default"/>
1090        <xs:attribute name="severity"
1091                      type="hidef:severity-type"/>
1092        <xs:attribute name="action"
1093                      type="hidef:action-type" default="other"/>
1094        <xs:attribute name="ext-action"
1095                      type="xs:string" use="optional"/>
1096      </xs:complexType>
1097    </xs:element>
1098
1099  <!-- ********************************************************** -->
1100  <!-- *** Method Class                                       *** -->
1101  <!-- ********************************************************** -->
1102
1103    <xs:element name="Method">
1104      <xs:complexType>
1105        <xs:sequence>
1106          <!-- *** One or more occurrences *** -->
1107          <xs:choice maxOccurs="unbounded">
1108            <xs:element ref="hidef:Reference"/>
1109            <xs:element ref="hidef:Description"/>
1110          </xs:choice>
1111          <!-- *** Zero or more occurrences *** -->
1112          <xs:element ref="hidef:AdditionalData"
1113                      minOccurs="0" maxOccurs="unbounded"/>
1114        </xs:sequence>
1115        <!-- *** Attribute *** -->
1116        <xs:attribute name="restriction"
1117                      type="hidef:restriction-type"/>
1118      </xs:complexType>
1119    </xs:element>
1120
1121  <!-- ********************************************************** -->
1122  <!-- *** Reference Class                                    *** -->
1123  <!-- ********************************************************** -->
1124
1125    <xs:element name="Reference">
```

```
1126      <xs:complexType>
1127        <xs:sequence>
1128          <!-- *** One occurrence *** -->
1129          <xs:element name="ReferenceName"
1130                      type="hidef:MLStringType"/>
1131          <!-- *** Zero or more occurrences *** -->
1132          <xs:element ref="hidef:URL"
1133                      minOccurs="0" maxOccurs="unbounded"/>
1134          <xs:element ref="hidef:Description"
1135                      minOccurs="0" maxOccurs="unbounded"/>
1136        </xs:sequence>
1137      </xs:complexType>
1138    </xs:element>
1139
1140  <!-- ********************************************************** -->
1141  <!-- *** Assessment Class                                   *** -->
1142  <!-- ********************************************************** -->
1143
1144    <xs:element name="Assessment">
1145      <xs:complexType>
1146        <xs:sequence>
1147          <!-- *** One or more occurrences *** -->
1148          <xs:choice maxOccurs="unbounded">
1149            <xs:element ref="hidef:Impact"/>
1150            <xs:element ref="hidef:TimeImpact"/>
1151            <xs:element ref="hidef:MonetaryImpact"/>
1152          </xs:choice>
1153          <!-- *** Zero or more occurrences *** -->
1154          <xs:element ref="hidef:Counter"
1155                      minOccurs="0" maxOccurs="unbounded"/>
1156          <!-- *** Zero or one occurrence *** -->
1157          <xs:element ref="hidef:Confidence" minOccurs="0"/>
1158          <!-- *** Zero or more occurrences *** -->
1159          <xs:element ref="hidef:AdditionalData"
1160                      minOccurs="0" maxOccurs="unbounded"/>
1161        </xs:sequence>
1162        <!-- *** Attributes *** -->
1163        <xs:attribute name="occurrence">
1164          <xs:simpleType>
1165            <xs:restriction base="xs:NMTOKEN">
1166              <xs:enumeration value="actual"/>
1167              <xs:enumeration value="potential"/>
1168            </xs:restriction>
1169          </xs:simpleType>
1170        </xs:attribute>
1171        <xs:attribute name="restriction"
1172                      type="hidef:restriction-type"/>
```

```
1173      </xs:complexType>
1174    </xs:element>
1175
1176  <!-- ********************************************************** -->
1177  <!-- *** Impact Class                                       *** -->
1178  <!-- ********************************************************** -->
1179
1180    <xs:element name="Impact">
1181      <xs:complexType>
1182        <xs:simpleContent>
1183          <xs:extension base="hidef:MLStringType">
1184          <!-- *** Attributes *** -->
1185            <xs:attribute name="severity"
1186                          type="hidef:severity-type"/>
1187            <xs:attribute name="completion">
1188              <xs:simpleType>
1189                <xs:restriction base="xs:NMTOKEN">
1190                  <xs:enumeration value="failed"/>
1191                  <xs:enumeration value="succeeded"/>
1192                </xs:restriction>
1193              </xs:simpleType>
1194            </xs:attribute>
1195            <xs:attribute name="type"
1196                          use="optional" default="unknown">
1197              <xs:simpleType>
1198                <xs:restriction base="xs:NMTOKEN">
1199                  <xs:enumeration value="admin"/>
1200                  <xs:enumeration value="dos"/>
1201                  <xs:enumeration value="extortion"/>
1202                  <xs:enumeration value="file"/>
1203                  <xs:enumeration value="info-leak"/>
1204                  <xs:enumeration value="misconfiguration"/>
1205                  <xs:enumeration value="recon"/>
1206                  <xs:enumeration value="policy"/>
1207                  <xs:enumeration value="social-engineering"/>
1208                  <xs:enumeration value="user"/>
1209                  <xs:enumeration value="unknown"/>
1210                  <xs:enumeration value="ext-value"/>
1211                </xs:restriction>
1212              </xs:simpleType>
1213            </xs:attribute>
1214            <xs:attribute name="ext-type"
1215                          type="xs:string" use="optional"/>
1216          </xs:extension>
1217        </xs:simpleContent>
1218      </xs:complexType>
1219    </xs:element>
```

```
1220
1221 <!-- ********************************************************** -->
1222 <!-- *** TimeImpact Class                                   *** -->
1223 <!-- ********************************************************** -->
1224
1225   <xs:element name="TimeImpact">
1226     <xs:complexType>
1227       <xs:simpleContent>
1228         <xs:extension base="hidef:PositiveFloatType">
1229           <!-- *** Attributes *** -->
1230           <xs:attribute name="severity"
1231                         type="hidef:severity-type"/>
1232           <xs:attribute name="metric"
1233                         use="required">
1234             <xs:simpleType>
1235               <xs:restriction base="xs:NMTOKEN">
1236                 <xs:enumeration value="labor"/>
1237                 <xs:enumeration value="elapsed"/>
1238                 <xs:enumeration value="downtime"/>
1239                 <xs:enumeration value="ext-value"/>
1240               </xs:restriction>
1241             </xs:simpleType>
1242           </xs:attribute>
1243           <xs:attribute name="ext-metric"
1244                         type="xs:string" use="optional"/>
1245           <xs:attribute name="duration"
1246                         type="hidef:duration-type"/>
1247           <xs:attribute name="ext-duration"
1248                         type="xs:string" use="optional"/>
1249         </xs:extension>
1250       </xs:simpleContent>
1251     </xs:complexType>
1252   </xs:element>
1253
1254 <!-- ********************************************************** -->
1255 <!-- *** MonetaryImpact Class                               *** -->
1256 <!-- ********************************************************** -->
1257
1258   <xs:element name="MonetaryImpact">
1259     <xs:complexType>
1260       <xs:simpleContent>
1261         <xs:extension base="hidef:PositiveFloatType">
1262           <!-- *** Attributes *** -->
1263           <xs:attribute name="severity"
1264                         type="hidef:severity-type"/>
1265           <xs:attribute name="currency"
1266                         type="xs:string"/>
```

```
1267          </xs:extension>
1268        </xs:simpleContent>
1269      </xs:complexType>
1270    </xs:element>
1271
1272  <!-- ********************************************************** -->
1273  <!-- *** Confidence Class                                   *** -->
1274  <!-- ********************************************************** -->
1275
1276    <xs:element name="Confidence">
1277      <xs:complexType mixed="true">
1278        <!-- *** Attributes *** -->
1279        <xs:attribute name="rating" use="required">
1280          <xs:simpleType>
1281            <xs:restriction base="xs:NMTOKEN">
1282              <xs:enumeration value="low"/>
1283              <xs:enumeration value="medium"/>
1284              <xs:enumeration value="high"/>
1285              <xs:enumeration value="numeric"/>
1286              <xs:enumeration value="unknown"/>
1287            </xs:restriction>
1288          </xs:simpleType>
1289        </xs:attribute>
1290      </xs:complexType>
1291    </xs:element>
1292
1293  <!-- ********************************************************** -->
1294  <!-- *** Date and Time related Classes                      *** -->
1295  <!-- ********************************************************** -->
1296
1297    <xs:element name="DateTime"
1298                type="xs:dateTime"/>
1299    <xs:element name="DetectTime"
1300                type="xs:dateTime"/>
1301    <xs:element name="StartTime"
1302                type="xs:dateTime"/>
1303    <xs:element name="EndTime"
1304                type="xs:dateTime"/>
1305    <xs:element name="Timezone"
1306                type="hidef:TimezoneType"/>
1307    <xs:simpleType name="TimezoneType">
1308      <xs:restriction base="xs:string">
1309        <xs:pattern value="Z|[\+\-](0[0-9]|1[0-4]):[0-5][0-9]"/>
1310      </xs:restriction>
1311    </xs:simpleType>
1312
1313  <!-- ********************************************************** -->
```

```
1314  <!-- ********************************************************** -->
1315  <!-- *** Declaration of types defined in HIDEF                *** -->
1316  <!-- ********************************************************** -->
1317  <!-- ********************************************************** -->
1318
1319  <!-- ********************************************************** -->
1320  <!-- *** restriction-type                                     *** -->
1321  <!-- ********************************************************** -->
1322
1323    <xs:simpleType name="restriction-type">
1324      <xs:restriction base="xs:NMTOKEN">
1325        <xs:enumeration value="default"/>
1326        <xs:enumeration value="public"/>
1327        <xs:enumeration value="need-to-know"/>
1328        <xs:enumeration value="private"/>
1329      </xs:restriction>
1330    </xs:simpleType>
1331
1332  <!-- ********************************************************** -->
1333  <!-- *** PortlistType                                         *** -->
1334  <!-- ********************************************************** -->
1335
1336    <xs:simpleType name="PortlistType">
1337      <xs:restriction base="xs:string">
1338        <xs:pattern value="\d+(\-\d+)?(,\d+(\-\d+)?)*"/>
1339      </xs:restriction>
1340    </xs:simpleType>
1341
1342  <!-- ********************************************************** -->
1343  <!-- *** MLStringType                                         *** -->
1344  <!-- ********************************************************** -->
1345
1346    <xs:complexType name="MLStringType">
1347      <xs:simpleContent>
1348        <xs:extension base="xs:string">
1349          <xs:attribute name="lang"
1350                        type="xs:language" use="optional"/>
1351        </xs:extension>
1352      </xs:simpleContent>
1353    </xs:complexType>
1354
1355  <!-- ********************************************************** -->
1356  <!-- *** ExtensionType                                        *** -->
1357  <!-- ********************************************************** -->
1358
1359    <xs:complexType name="ExtensionType" mixed="true">
1360      <xs:sequence>
```

```
1361        <xs:any namespace="##any" processContents="lax"
1362                minOccurs="0" maxOccurs="unbounded"/>
1363      </xs:sequence>
1364      <xs:attribute name="dtype"
1365                    type="hidef:dtype-type" use="required"/>
1366      <xs:attribute name="ext-dtype"
1367                    type="xs:string" use="optional"/>
1368      <xs:attribute name="meaning"
1369                    type="xs:string"/>
1370      <xs:attribute name="formatid"
1371                    type="xs:string"/>
1372      <xs:attribute name="restriction"
1373                    type="hidef:restriction-type"/>
1374    </xs:complexType>
1375
1376  <!-- ********************************************************** -->
1377  <!-- *** hash-type                                          *** -->
1378  <!-- ********************************************************** -->
1379
1380    <xs:simpleType name="hash-type">
1381      <xs:restriction base="xs:string">
1382        <xs:enumeration value="md5"/>
1383        <xs:enumeration value="sha1"/>
1384        <xs:enumeration value="sha256"/>
1385        <xs:enumeration value="sha512"/>
1386        <xs:enumeration value="rmd160"/>
1387        <xs:enumeration value="pgp-signature"/>
1388        <xs:enumeration value="ext-value"/>
1389      </xs:restriction>
1390    </xs:simpleType>
1391
1392  <!-- ********************************************************** -->
1393  <!-- *** dtype-type                                         *** -->
1394  <!-- ********************************************************** -->
1395
1396    <xs:simpleType name="dtype-type">
1397      <xs:restriction base="xs:NMTOKEN">
1398        <xs:enumeration value="boolean"/>
1399        <xs:enumeration value="byte"/>
1400        <xs:enumeration value="character"/>
1401        <xs:enumeration value="date-time"/>
1402        <xs:enumeration value="integer"/>
1403        <xs:enumeration value="ntpstamp"/>
1404        <xs:enumeration value="portlist"/>
1405        <xs:enumeration value="real"/>
1406        <xs:enumeration value="string"/>
1407        <xs:enumeration value="file"/>
```

```
1408        <xs:enumeration value="path"/>
1409        <xs:enumeration value="frame"/>
1410        <xs:enumeration value="packet"/>
1411        <xs:enumeration value="ipv4-packet"/>
1412        <xs:enumeration value="ipv6-packet"/>
1413        <xs:enumeration value="url"/>
1414        <xs:enumeration value="csv"/>
1415        <xs:enumeration value="winreg"/>
1416        <xs:enumeration value="xml"/>
1417        <xs:enumeration value="ext-value"/>
1418      </xs:restriction>
1419    </xs:simpleType>
1420
1421  <!-- ********************************************************** -->
1422  <!-- *** SoftwareType                                       *** -->
1423  <!-- ********************************************************** -->
1424
1425    <xs:complexType name="SoftwareType">
1426      <xs:sequence>
1427        <xs:element ref="hidef:Description"
1428                    minOccurs="0"/>
1429        <xs:element ref="hidef:URL"
1430                    minOccurs="0"/>
1431      </xs:sequence>
1432      <xs:attribute name="swid"
1433                    type="xs:string" default="0"/>
1434      <xs:attribute name="configid"
1435                    type="xs:string" default="0"/>
1436      <xs:attribute name="vendor"
1437                    type="xs:string"/>
1438      <xs:attribute name="family"
1439                    type="xs:string"/>
1440      <xs:attribute name="name"
1441                    type="xs:string"/>
1442      <xs:attribute name="version"
1443                    type="xs:string"/>
1444      <xs:attribute name="patch"
1445                    type="xs:string"/>
1446    </xs:complexType>
1447
1448  <!-- ********************************************************** -->
1449  <!-- *** ContactMeansType                                   *** -->
1450  <!-- ********************************************************** -->
1451
1452    <xs:complexType name="ContactMeansType">
1453      <xs:simpleContent>
1454        <xs:extension base="xs:string">
```

```
1455          <xs:attribute name="meaning"
1456                        type="xs:string" use="optional"/>
1457        </xs:extension>
1458      </xs:simpleContent>
1459    </xs:complexType>
1460
1461  <!-- ********************************************************** -->
1462  <!-- *** duration-type                                      *** -->
1463  <!-- ********************************************************** -->
1464
1465    <xs:simpleType name="duration-type">
1466      <xs:restriction base="xs:NMTOKEN">
1467        <xs:enumeration value="second"/>
1468        <xs:enumeration value="minute"/>
1469        <xs:enumeration value="hour"/>
1470        <xs:enumeration value="day"/>
1471        <xs:enumeration value="month"/>
1472        <xs:enumeration value="quarter"/>
1473        <xs:enumeration value="year"/>
1474        <xs:enumeration value="ext-value"/>
1475      </xs:restriction>
1476    </xs:simpleType>
1477
1478  <!-- ********************************************************** -->
1479  <!-- *** severity-type                                      *** -->
1480  <!-- ********************************************************** -->
1481
1482    <xs:simpleType name="severity-type">
1483      <xs:restriction base="xs:NMTOKEN">
1484        <xs:enumeration value="low"/>
1485        <xs:enumeration value="medium"/>
1486        <xs:enumeration value="high"/>
1487      </xs:restriction>
1488    </xs:simpleType>
1489
1490  <!-- ********************************************************** -->
1491  <!-- *** action-type                                        *** -->
1492  <!-- ********************************************************** -->
1493
1494    <xs:simpleType name="action-type">
1495      <xs:restriction base="xs:NMTOKEN">
1496        <xs:enumeration value="nothing"/>
1497        <xs:enumeration value="contact-source-site"/>
1498        <xs:enumeration value="contact-target-site"/>
1499        <xs:enumeration value="contact-sender"/>
1500        <xs:enumeration value="investigate"/>
1501        <xs:enumeration value="block-host"/>
```

```
1502        <xs:enumeration value="block-network"/>
1503        <xs:enumeration value="block-port"/>
1504        <xs:enumeration value="rate-limit-host"/>
1505        <xs:enumeration value="rate-limit-network"/>
1506        <xs:enumeration value="rate-limit-port"/>
1507        <xs:enumeration value="remediate-other"/>
1508        <xs:enumeration value="status-triage"/>
1509        <xs:enumeration value="status-new-info"/>
1510        <xs:enumeration value="other"/>
1511        <xs:enumeration value="ext-value"/>
1512      </xs:restriction>
1513    </xs:simpleType>
1514
1515  <!-- ********************************************************** -->
1516  <!-- *** PositiveFloatType                                  *** -->
1517  <!-- ********************************************************** -->
1518
1519    <xs:simpleType name="PositiveFloatType">
1520      <xs:restriction base="xs:float">
1521        <xs:minExclusive value="0"/>
1522      </xs:restriction>
1523    </xs:simpleType>
1524
1525  <!-- ********************************************************** -->
1526
1527  </xs:schema>
1528
1529  <!-- ********************************************************** -->
1530  <!-- *** HIDEF Schema ends here                             *** -->
1531  <!-- ********************************************************** -->
```

C APÊNDICE C MODELAGEM DO BANCO DE DADOS DO PROTÓTIPO DO SISTEMA HIDEAS

Neste Apêndice, é apresentada a modelagem Entidade-Relacionamento (E-R) (SILBERSCHATZ et al., 1999) do Protótipo do Sistema HIDEAS, seguido do mapeamento para o Modelo Relacional (SILBERSCHATZ et al., 1999). Estes modelos foram utilizados para a implementação do Banco de Dados utilizado no protótipo.

C.1 Modelo Entidade-Relacionamento

Nesta seção serão apresentadas as descrições das entidades identificadas no mapeamento entre as classes e o Modelo E-R e as descrições de seus atributos.

Na definição do Modelo E-R, não foram mapeadas como entidades classes que atuam como *containers* de outras classes. Esse tipo de relacionamento do modelo de objetos foi mapeado como uma relação 1–N diretamente com as classes filhas da classe *container*. Também não foram mapeadas como entidades classes cujo tipo de dados era simples e que poderiam ser mapeadas no Modelo E-R como atributos de uma entidade.

Na Figura C.1, estão representadas as entidades identificadas quando feito o mapeamento da classe EventData, derivada do IODEF para o HIDEF, para o Modelo E-R. As principais entidades identificadas foram EventData, Contact, Expectation, Assessment, System, Method e RecordData. Também na Figura C.1 estão representadas outras entidades identificadas no momento do mapeamento das classes filhas de EventData para o Modelo E-R e os relacionamentos entre elas.

Seguem as descrições das entidades da Figura C.1 e de seus atributos.

EventData – dados sobre um evento sendo reportado ou capturado por um *honeypot*; Atributos:

DetecTime: momento em que o evento foi detectado;

StartTime: momento em que o evento teve início;

EndTime: momento em que o evento se encerrou;

Description: descrição geral do evento.

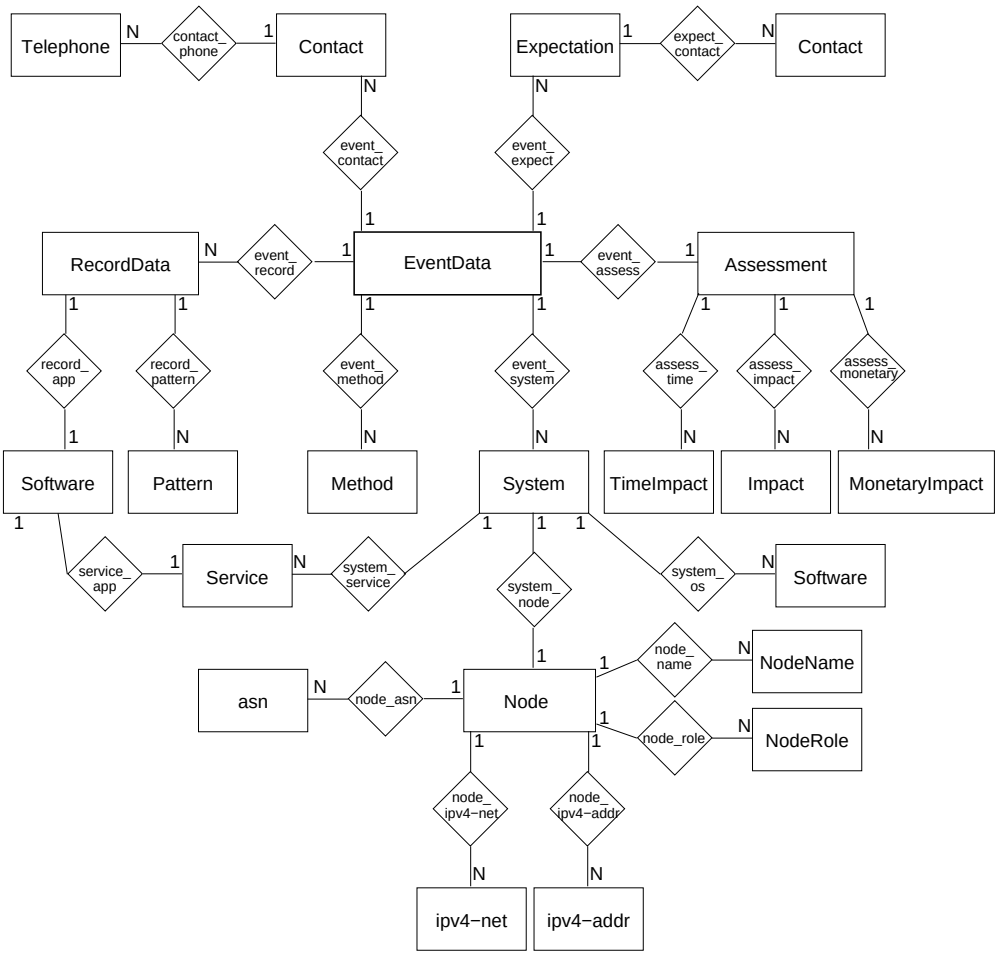

Figura C.1 - Diagrama E-R da entidade EventData e seus relacionamentos com outras entidades.

RecordData – *logs* e informações de auditoria dos sistemas envolvidos em uma atividade maliciosa; Atributos:

Restriction: regras que devem ser observadas quanto à divulgação das informações;

DateTime: *timestamp* do item associado;

Description: descrição geral do conteúdo da entidade.

Location – localização dos dados, seu conteúdo pode ser um *path* para um diretório do sistema ou uma URL para um *site* com os dados.

Software – esta entidade representa informações sobre *softwares* utilizados; Atributos:

Type: o tipo de *software*, que pode ter os seguintes valores: OS ou Application;

swid: um número identificador;

configid: um número identificador que pode ser associado a uma configuração em particular;

Vendor: o fabricante ou desenvolvedor do *software*;

Family: uma determinada família do *software*;

Name: o nome do *software*;

Version: a versão do *software*;

Patch: correções aplicadas ao *software*;

URL: uma URL para *download* do *software*, para o *site* do fabricante ou para uma página com mais informações;

Description: uma descrição geral do *software*.

Pattern – padrão pelo qual procurar dentro dos dados; Atributos:

Type: tipo do padrão sendo fornecido, que pode ser uma expressão regular (regex), um padrão do tipo HEXBIN (binary) ou um XML *Path* (xpath);

Offset: um número de unidades que se deve avançar antes de chegar ao padrão desejado;

OffsetUnit: a unidade do atributo Offset; pode ter os valores line, para um número de linhas a avançar, ou binary, para um número de bytes a avançar.

System – sistemas envolvidos na atividade representada pelo conjunto de entidades relacionadas com EventData; Atributos:

Restriction: regras que devem ser observadas quanto à divulgação das informações;

Category: papel do sistema descrito, pode ser a origem (source), destino (target), um sistema monitorando o evento (sensor), parte da infraestrutura (infrastructure) ou um sistema intermediário na atividade maliciosa em questão (intermediate);

Interface: caso o sistema seja um computador, designa a interface de rede onde os eventos descritos se originaram;

Spoofed: indicação da confiança de que o sistema ou rede é mesmo a origem ou alvo do evento, pode ter os valores unknown, yes ou no;

Description: texto com a descrição geral do sistema;

Counter: sumariza características de um *host* ou rede sendo representado;

CounterType: unidade do conteúdo de Counter, os valores possíveis são byte, packet, flow, session, alert, message, event, host, site ou organization;

CounterMeaning: descrição geral do conteúdo de Counter;

CounterDuration: se possuir um valor, significa que counter é uma taxa, cujo conteúdo é o numerador, e o valor presente em CounterDuration é o denominador; seus possíveis valores são second, minute, hour, day, month, quarter ou year.

Service – descreve um serviço de rede; Atributos:

ip_protocol: o número do protocolo junto ao IANA;

PortList: uma porta ou um conjunto de portas associadas a um serviço;

ProtoType: tipo de um protocolo (nível 4);

ProtoCode: código de um protocolo (nível 4);

ProtoField: campos ou *flags* de um protocolo (nível 4).

Node – identificação da máquina, rede ou dispositivo envolvido no evento; Atributos

Location: descrição da localização física do equipamento ou rede;

DateTime: momento em que foi feita a resolução entre nome e endereço, representados respectivamente pelas entidades NomeName e uma das entidades que representam endereços;

Counter: sumariza propriedades do dispositivo ou rede sendo representado;

CounterType: unidade do conteúdo de Counter, os valores possíveis são byte, packet, flow, session, alert, message, event, host, site ou organization;

CounterMeaning: descrição geral do conteúdo de Counter;

CounterDuration: se possuir um valor, significa que counter é uma taxa, cujo conteúdo é o numerador, e o valor presente em CounterDuration é o denominador; seus possíveis valores são second, minute, hour, day, month, quarter ou year.

NodeName: representa as maneiras como o dispositivo pode ser chamado, como por exemplo nomes DNS;

NodeRole: indica o papel desempenhado pelo dispositivo, pode ter os valores client, server-internal, server-public, www, mail, messaging, streaming, voice, file, ftp, p2p, name, print, application, database, infra ou log.

ipv4-addr: endereço IPv4; Atributos:

vlan-name: nome da rede virtual (`vlan`) à qual o endereço IP pertence;

vlan-num: número da rede virtual (`vlan`) à qual o endereço IP pertence.

ipv4-net: endereço de rede (CIDR) IPv4; Atributos:

vlan-name: nome da rede virtual (`vlan`) à qual o CIDR pertence;

vlan-num: número da rede virtual (`vlan`) à qual o CIDR pertence.

asn: *Autonomous System Number* associado à rede ou dispositivo.

Contact – informações de contato de pessoas ou organizações envolvidas na atividade sendo registrada; Atributos:

Restriction: regras que devem ser observadas quanto à divulgação das informações;

Lang: idioma em que o endereço está escrito.

Role: papel do contato, pode ter os valores admin, tech, irt ou cc;

Type: tipo de contato, pode ter os valores person ou organization;

Name: o nome do contato;

Timezone: o fuso horário onde esse contato se encontra;

PostalAddress: o endereço de correspondência do contato;

Description: uma descrição geral do contato;

Email: o endereço de *e-mail* do contato.

Telephone – telefone associado a um contato; Atributos:

Type: tipo do telefone, pode ter os valores phone ou fax;

Number: número do telefone.

Method – metodologia utilizada pelo atacante; Atributos:

Restriction: regras que devem ser observadas quanto à divulgação das informações;

Description: descrição geral da metodologia;

Reference: o nome de uma referência associada à metologia;

ReferenceURL: uma URL associada à referência;

ReferenceDescription: uma descrição geral da referência.

Expectation – qual ação se espera que seja tomada com relação ao evento documentado; Atributos:

Restriction: regras que devem ser observadas quanto à divulgação das informações;

Severity: prioridade que deve ser dada ao tratamento do evento, pode ter os valores low, medium ou high;

Action: qual a ação esperada, pode ter os valores nothing, contact-source-site, contact-target-site, network, rate-limit-port, remediate-other, status-triage, status-new-info;

Description: descrição detalhada do que é esperado;

StartTime: data e horário em que se espera que a ação seja realizada;

EndTime: data limite para a execução da ação.

Assessment – descrição das repercussões dos eventos documentados; Atributos:

Restriction: regras que devem ser observadas quanto à divulgação das informações;

Ocurrence: define se os impactos listados ocorreram ou são impactos em potencial, pode ter os valores actual ou potential;

Confidence: confiança na avaliação, pode ter os valores low, medium ou high;

Counter: permite sumarizar a magnitude da atividade;

CounterType: unidade do conteúdo de Counter, os valores possíveis são byte, packet, flow, session, alert, message, event, host, site ou organization;

CounterMeaning: descrição geral do conteúdo de Counter;

CounterDuration: se possuir um valor, significa que counter é uma taxa, cujo conteúdo é o numerador, e o valor presente em CounterDuration é o denominador; seus possíveis valores são second, minute, hour, day, month, quarter ou year.

Impact – impacto técnico de um incidente; Atributos

Lang: idioma dos dados representados;

Severity: permite estimar a severidade da atividade, pode ter os valores low, medium ou high;

Completion: indica se o ataque, parte da atividade, teve sucesso ou não, pode ter os valores failed ou succeeded;

Type: categoria de ataque ocorrido, pode ter os valores admin, dos, file, info-leak, misconfiguration, policy, recon, social-engineering ou user unknown.

TimeImpact – tempo de interrupção das atividades e de recuperação; Atributos:

Severity: permite estimar a severidade da atividade, pode ter os valores low, medium ou high;

Metric: fator usado para o cálculo do tempo, pode ter os valores labor, elapsed ou downtime;

Duration: unidade de tempo que, combinada com o valor de Metric, descreve o impato, pode ter os valores second, minute, hour, day, month, quarter ou year.

MonetaryImpact – impacto financeiro da atividade; Atributos:

Severity: permite estimar a severidade da atividade, pode ter os valores low, medium ou high;

Currency: modeda em que o valor está expresso.

Na Figura C.2 estão representados os relacionamentos entre a entidade Honeypot e as entidades Honeynet, Solution, Contact, Sanitization, asn, ipv4-addr, ipv4-net, FilterInfo,

System (representando o(s) sistema(s) emulado(s)), System (representando o sistema real), EventData e ArtifactData. Estão também representados os relacionamentos entre ArtifactData e Hash, e Honeynet, DataCollection,DataControl asn, ipv4-addr e ipv4-net.

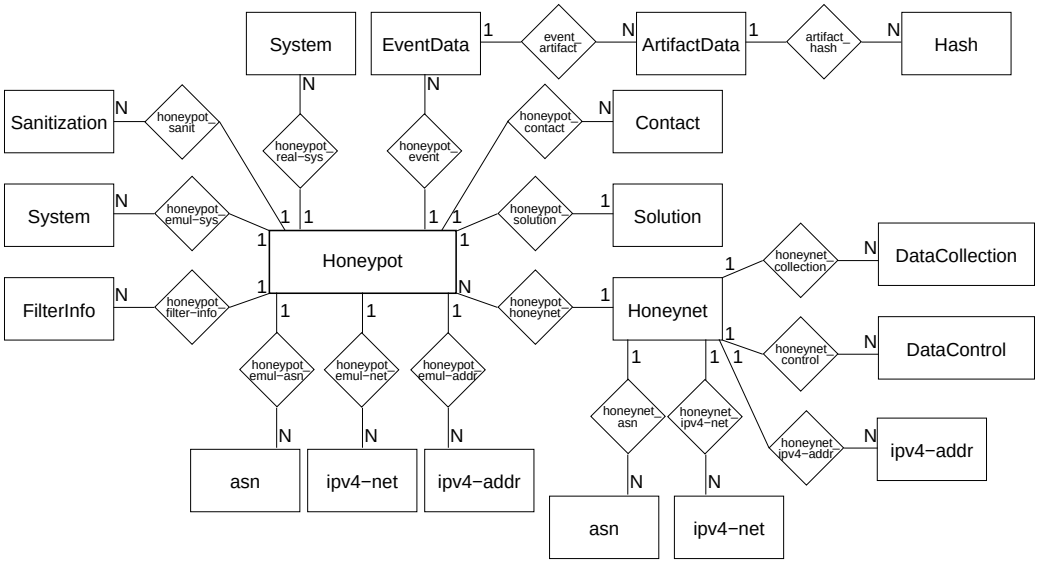

Figura C.2 - Diagrama E-R da entidade Honeypot e seus relacionamentos com outras entidades.

Seguem as descrições das entidades da Figura C.2 e de seus atributos. As entidades já descritas anteriormente não serão descritas novamente.

Honeypot – características gerais de um *honeypot*; Atributos:

Restriction: regras que devem ser observadas quanto à divulgação das informações;

Type: tipo do *honeypot*, pode possuir os valores low-interaction ou high-interaction;

Hardware: descrição do *hardware* utilizado;

Description: texto livre com uma descrição geral do *honeypot*.

Sanitization – detalhes sobre a sanitização usada nos dados representados em EventData; Atributos:

Type: indica que tipo de sanitização está representada, pode ter os valores ipv4-addr ou ipv4-net;

Real: contém o endereço ou rede real sendo utilizada pelo *honeypot*.

Sanitized: forma escolhida para sanitização.

FilterInfo – representa filtros aplicados entre o tráfego da Internet e o *honeypot*; Atributos:

Type: tipo do filtro, se o valor for open, significa que é permitido o tráfego para as portas listadas no atributo Portlist, se o valor for closed, significa que é proibido o tráfego para as portas listadas em Portlist e o restante é permitido;

ip_protocol: número do protocolo junto ao IANA;

PortList: uma porta ou um conjunto de portas associadas a um serviço de rede.

Solution – o *software* ou tecnologia sendo utilizada para implementar o *honeypot*; Atributos:

URL: URL para um página com mais informações ou com o *software* para *download*;

Description: nome ou uma descrição geral da solução adotada.

ArtifactData – dados sobre um artefato capturado; Atributos:

Description: descrição geral do artefato coletado;

DateTime: data e horário da captura;

Location: localização de um artefato capturado, pode ser um *path* para um diretório do sistema ou uma URL para um *site* com os dados;

URL: URL para um site com uma análise ou com uma versão pública do artefato.

Hash – um *hash* criptográfico ou uma assinatura digital, que podem ser usados para identificar unicamente um artefato; Atributos:

Type: tipo do *hash*, pode ter os valores md5, sha1, sha256, sha512, rmd160 ou pgp-signature.

Value: valor do *hash* ou assinatura.

Honeynet – *honeynet* onde um *honeypot* pode estar localizado; Atributos:

Type: tipo da *honeynet*, pode ter os valores physical ou virtual;

Restriction: regras que devem ser observadas quanto à divulgação das informações;

Description: descrição geral da *honeynet*.

DataCollection – tecnologia de coleta de dados usada; Atributos:

Description: nome ou descrição geral da tecnologia;

URL: URL para um página com mais informações ou com o *software* para *download*.

DataControl – tecnologia usada para controle das atividades dos invasores; Atributos:

Description: nome ou descrição geral da tecnologia;

URL: URL para um página com mais informações ou com o *software* para *download*.

C.2 Mapeamento para o Modelo Relacional

As entidades e relacionamentos do Modelo E-R, descrito na Seção C.1, necessários para a implementação do Estudo de Caso discutido no Capítulo 3, foram mapeados para as Tabelas do Modelo Relacional apresentadas nesta Seção.

Nas tabelas `artifactdata` e `recorddata` foi criado um campo `location`, onde será armazenada a localização do artefato e do *log*, respectivamente, em disco. Essa decisão foi tomada para permitir armazenar no banco de dados apenas as informações sobre o artefato e sobre o *log*, mas não seus conteúdos completos. Com isso, evita-se criar tabelas com um volume exagerado de dados. As consultas são efetuadas com base nas informações armazenadas e, se for necessário obter dados mais detalhados, isso pode ser feito diretamente no artefato ou no *log* armazenado em disco.

A seguir cada tabela é apresentada em conjunto com os tipos de dados utilizados na sua implementação no Banco de Dados MySQL.

Tabela **honeypot**

hpot_id	restriction	hpot_type	hardware	description

```
hpot_id             VARCHAR(20) NOT NULL
restriction         ENUM('default', 'public', 'need-to-know', 'private')
hpot_type           ENUM('low-interaction', 'high-interaction') NOT NULL
hardware            VARCHAR(255)
description         VARCHAR(255)
PRIMARY KEY         (hpot_id)
```

Tabela **sanitization**

hpot_id	sanitization_type	real_addr	sanitized

```
hpot_id             VARCHAR(20) NOT NULL
sanitization_type   ENUM('asn', 'atm', 'e-mail', 'mac', 'ipv4-addr',
                    'ipv4-net', 'ipv4-net-mask') NOT NULL
real_addr           VARCHAR(20)
sanitized           VARCHAR(20)
PRIMARY KEY         (hpot_id,real_addr)
CONSTRAINT          fk_sanitization_honeypot FOREIGN KEY (hpot_id)
                    REFERENCES honeypot (hpot_id)
```

Tabela **hpot_emulatednet**

hpot_id	emulated_addr

```
hpot_id             VARCHAR(20) NOT NULL
emulated_addr       VARCHAR(20) NOT NULL
PRIMARY KEY         (hpot_id,emulated_addr)
CONSTRAINT          fk_hpot_emulatednet FOREIGN KEY (hpot_id)
                    REFERENCES honeypot (hpot_id)
```

Tabela **system**

system_id	restriction	category	interface	spoofed	description	location

date_time	counter	counter_type	counter_meaning	counter_duration

```
system_id           VARCHAR(50) NOT NULL
restriction         ENUM('default', 'public', 'need-to-know', 'private')
category            ENUM('source', 'target', 'intermediate', 'sensor',
                    'infrastructure')
interface           VARCHAR(10)
```

```
spoofed             ENUM('unknown', 'yes', 'no')
description         VARCHAR(255)
location            VARCHAR(60)
date_time           DATETIME
counter             DOUBLE
counter_type        ENUM('byte', 'packet', 'flow', 'session', 'event',
                    'alert', 'message', 'host', 'site', 'organization')
counter_meaning     VARCHAR(60)
counter_duration    VARCHAR(60)
PRIMARY KEY         (system_id)
```

Tabela **hpot_real_system**

hpot_id	system_id

```
hpot_id             VARCHAR(20) NOT NULL
system_id           VARCHAR(50) NOT NULL
PRIMARY KEY         (hpot_id,system_id)
CONSTRAINT          fk_hpot_real_system_honeypot FOREIGN KEY (hpot_id)
                    REFERENCES honeypot (hpot_id)
CONSTRAINT          fk_hpot_real_system_system FOREIGN KEY (system_id)
                    REFERENCES system (system_id)
```

Tabela **system_ipv4addr**

system_id	ipv4_addr

```
system_id           VARCHAR(50) NOT NULL
ipv4_addr           VARCHAR(20) NOT NULL
PRIMARY KEY         (system_id,ipv4_addr)
CONSTRAINT          fk_system_ipv4addr FOREIGN KEY (system_id)
                    REFERENCES system (system_id)
```

Tabela **system_asn**

system_id	asn

```
system_id           VARCHAR(50) NOT NULL
asn                 VARCHAR(20) NOT NULL
PRIMARY KEY         (system_id,asn)
CONSTRAINT          fk_system_asn FOREIGN KEY (system_id)
                    REFERENCES system (system_id)
```

Tabela **software**

sw_id	type	config_id	vendor	family	name	version	patch	url	description

```
sw_id               VARCHAR(50) NOT NULL
type                ENUM('OS', 'application', 'emulatedOS',
                    'fingerprintOS') NOT NULL
config_id           VARCHAR(10)
vendor              VARCHAR(50)
family              VARCHAR(50)
name                VARCHAR(60)
version             VARCHAR(60)
patch               VARCHAR(60)
url                 VARCHAR(255)
description         VARCHAR(255)
PRIMARY KEY         (sw_id,type)
```

Tabela **system_os**

system_id	sw_id

```
system_id           VARCHAR(50) NOT NULL,
sw_id               VARCHAR(50) NOT NULL,
PRIMARY KEY         (system_id,sw_id),
CONSTRAINT          fk_system_os_system FOREIGN KEY (system_id)
                    REFERENCES system (system_id),
CONSTRAINT          fk_system_os_software FOREIGN KEY (sw_id)
                    REFERENCES software (sw_id)
```

Tabela **contact**

email	name	role	type	restriction	lang	postal_address	description	timezone

```
email               VARCHAR(255) NOT NULL
name                VARCHAR(255)
role                ENUM('creator','admin','tech','irt','cc')
type                ENUM('person','organization')
restriction         ENUM('default', 'public', 'need-to-know', 'private')
lang                VARCHAR(10)
postal_address      VARCHAR(255)
description         VARCHAR(255)
timezone            VARCHAR(6)
PRIMARY KEY         (email)
```

Tabela **hpot_contact**

hpot_id	email

```
hpot_id             VARCHAR(20) NOT NULL
email               VARCHAR(255) NOT NULL
PRIMARY KEY         (hpot_id,email)
```

```
CONSTRAINT          fk_hpot_contact_honeypot FOREIGN KEY (hpot_id)
                    REFERENCES honeypot (hpot_id)
CONSTRAINT          fk_hpot_contact_contact FOREIGN KEY (email)
                    REFERENCES contact (email)
```

Tabela **hpot_emulated_system**

hpot_id	system_id

```
hpot_id             VARCHAR(20) NOT NULL
system_id           VARCHAR(50) NOT NULL
PRIMARY KEY         (hpot_id,system_id)
CONSTRAINT          fk_hpot_emulated_system_honeypot FOREIGN KEY (hpot_id)
                    REFERENCES honeypot (hpot_id)
CONSTRAINT          fk_hpot_emulated_system_system FOREIGN KEY (system_id)
                    REFERENCES system (system_id)
```

Tabela **system_nodename**

system_id	node_name

```
system_id           VARCHAR(50) NOT NULL
node_name           VARCHAR(255) NOT NULL
PRIMARY KEY         (system_id,node_name)
CONSTRAINT          fk_system_nodename_system FOREIGN KEY (system_id)
                    REFERENCES system (system_id)
```

Tabela **service**

service_id	system_id	ip_protocol	proto_type	proto_code	proto_field

```
service_id          VARCHAR(10) NOT NULL
system_id           VARCHAR(50) NOT NULL
ip_protocol         SMALLINT UNSIGNED NOT NULL
proto_type          SMALLINT UNSIGNED
proto_code          SMALLINT UNSIGNED
proto_field         SMALLINT UNSIGNED
PRIMARY KEY         (service_id,system_id)
CONSTRAINT          fk_service_system FOREIGN KEY (system_id)
                    REFERENCES system (system_id)
```

Tabela **service_port**

system_id	service_id	port

```
system_id           VARCHAR(50) NOT NULL,
service_id          VARCHAR(10) NOT NULL,
port                SMALLINT UNSIGNED NOT NULL,
PRIMARY KEY         (system_id,service_id,port),
CONSTRAINT          fk_service_port_system FOREIGN KEY (system_id)
                    REFERENCES system (system_id),
CONSTRAINT          fk_service_port_service FOREIGN KEY (service_id)
                    REFERENCES service (service_id)
```

Tabela **service_app**

service_id	system_id	sw_id

```
service_id          VARCHAR(10) NOT NULL,
system_id           VARCHAR(50) NOT NULL,
sw_id               VARCHAR(50) NOT NULL,
PRIMARY KEY         (service_id,system_id,sw_id),
CONSTRAINT          fk_service_app_system FOREIGN KEY (system_id)
                    REFERENCES system (system_id),
CONSTRAINT          fk_service_app_service FOREIGN KEY (service_id)
                    REFERENCES service (service_id),
CONSTRAINT          fk_service_app_software FOREIGN KEY (sw_id)
                    REFERENCES software (sw_id)
```

Tabela **solution**

solution_id	hpot_id	url	description

```
solution_id         ENUM('nepenthes', 'honeyd') NOT NULL,
hpot_id             VARCHAR(20),
url                 VARCHAR(255),
description         VARCHAR(255),
PRIMARY KEY         (solution_id,hpot_id),
CONSTRAINT          fk_solution_honeypot FOREIGN KEY (hpot_id)
                    REFERENCES honeypot (hpot_id)
```

Tabela **eventdata**

event_id	date_time	start_time	end_time	description

```
event_id            VARCHAR(50) NOT NULL,
date_time           DATETIME,
start_time          DATETIME,
end_time            DATETIME,
description         VARCHAR(255),
PRIMARY KEY         (event_id)
```

Tabela **hpot_event**

hpot_id	event_id

```
hpot_id             VARCHAR(20) NOT NULL,
event_id            VARCHAR(50) NOT NULL,
PRIMARY KEY         (hpot_id,event_id),
CONSTRAINT          fk_hpot_event_honeypot FOREIGN KEY (hpot_id)
                    REFERENCES honeypot (hpot_id),
CONSTRAINT          fk_hpot_event_eventdata FOREIGN KEY (event_id)
                    REFERENCES eventdata (event_id)
```

Tabela **event_system**

event_id	system_id

```
event_id            VARCHAR(50) NOT NULL,
system_id           VARCHAR(50) NOT NULL,
PRIMARY KEY         (event_id,system_id),
CONSTRAINT          fk_event_system_eventdata FOREIGN KEY (event_id)
                    REFERENCES eventdata (event_id),
CONSTRAINT          fk_event_system_system FOREIGN KEY (system_id)
                    REFERENCES system (system_id)
```

Tabela **recorddata**

record_id	event_id	restriction	date_time	description	location

```
record_id           VARCHAR(50) NOT NULL,
event_id            VARCHAR(50) NOT NULL,
restriction         ENUM('default', 'public', 'need-to-know', 'private'),
date_time           DATETIME,
description         VARCHAR(255),
location            VARCHAR(255),
PRIMARY KEY         (record_id),
CONSTRAINT          fk_recorddata_eventdata FOREIGN KEY (event_id)
                    REFERENCES eventdata (event_id)
```

Tabela **record_app**

record_id	sw_id

```
record_id           VARCHAR(50) NOT NULL,
sw_id               VARCHAR(50) NOT NULL,
PRIMARY KEY         (record_id,sw_id),
CONSTRAINT          fk_record_app_record FOREIGN KEY (record_id)
```

```
                    REFERENCES recorddata (record_id),
    CONSTRAINT      fk_record_app_software FOREIGN KEY (sw_id)
                    REFERENCES software (sw_id)
```

Tabela **artifactdata**

artifact_id	event_id	description	date_time	location	url	type

```
    artifact_id     VARCHAR(50) NOT NULL,
    event_id        VARCHAR(50) NOT NULL,
    description     VARCHAR(255),
    date_time       DATETIME,
    location        VARCHAR(255),
    url             VARCHAR(255),
    type            ENUM('virus', 'worm', 'bot', 'spyware', 'trojan',
                    'backdoor', 'exploit', 'rootkit', 'gen-toolkit',
                    'irc-log', 'undefined', 'ext-value') NOT NULL,
    PRIMARY KEY     (artifact_id),
    CONSTRAINT      fk_artifactdata_eventdata FOREIGN KEY (event_id)
                    REFERENCES eventdata (event_id)
```

Tabela **hash**

artifact_id	type	value

```
    artifact_id     VARCHAR(50) NOT NULL,
    type            ENUM('md5', 'sha1', 'sha256', 'sha512', 'rmd160',
                    'pgp-signature') NOT NULL,
    value           VARCHAR(255),
    PRIMARY KEY     (artifact_id,type),
    CONSTRAINT      fk_hash_artifactdata FOREIGN KEY (artifact_id)
                    REFERENCES artifactdata (artifact_id)
```

Além das tabelas já apresentadas, que foram criadas a partir do modelo E-R, foi criada também uma tabela adicional: `ipv4addr_cc_asn`.

Tabela **ipv4addr_cc_asn**

ipv4_addr	cc	asn

```
    ipv4_addr       VARCHAR(20) NOT NULL,
    cc              VARCHAR(2),
    asn             SMALLINT UNSIGNED,
    PRIMARY KEY     (ipv4_addr)
```

Essa tabela foi criada posteriormente, a partir de uma consulta que retornou todos

os endereços IP únicos armazenados no banco de dados. De posse dessa lista foi gerada esta tabela com informações sobre o *Country Code* e o ASN (Autonomous System Number) relativos a cada IP.

Estas informações são utilizadas para gerar algumas correlações relativas aos contatos de rede e aos países de origem dos IPs.

D APÊNDICE D ARTIGOS

D.1 Artigo em Periódico Indexado

O artigo entitulado "*HIDEF: a Data Exchange Format for Information Collected in Honeypots and Honeynets*", foi publicado no "*INFOCOMP Journal of Computer Science*", Volume 7, Número 1, páginas 87–96, ISSN 1807-4545. Atendendo a exigência do Regimento do Curso de Pós-Graduação em Computação Aplicada, este periódico possui avaliação B Nacional, na Área de Atuação Multidisciplinar, junto à CAPES.

Também conforme requerido pelo Regimento do Curso de Pós-Graduação em Computação Aplicada, nas próximas páginas consta o artigo na íntegra, conforme publicado no periódico.

HIDEF: a Data Exchange Format for Information Collected in Honeypots and Honeynets

CRISTINE HOEPERS[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)
cristine@acm.org
[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. The deployment of honeypots is one of the methods used to collect data about attack trends in computer networks. The lack of a standard format for data representation makes the exchange and centralization of data generated by different technologies difficult. This also restricts the correlation and analysis of this information. This paper presents the HIDEF (Honeypots Information and Data Exchange Format), a proposal for a format to enable the representation and exchange of data and information produced by organizations using honeypots and honeynets.

Keywords: network security, honeypots, honeynets, XML, data exchange format

(Received November 02, 2007 / Accepted February 02, 2008)

1 Introduction

In the past few years the number of computer security incidents on Internet connected networks has continuously increased [3, 4]. As a result, there is an increasing need for attack data correlation and tools to help understand attacks and identify trends.

The deployment of sensors in computer networks to gather malicious traffic is one of the methods used by researchers and security professionals to collect data for attack trends analysis. One type of sensor that has been used is a honeypot, a security resource whose value lies in being probed, attacked or compromised [17, 16]. Another type of sensor used is a honeynet, a network specifically designed for the purpose of being compromised, that has control mechanisms to prevent it from being used as a base of attacks against other networks [18, 12]. The honeypot technologies have considerably developed in the past few years, mainly because of the increase in research activities and the development of new ways to collect data about attacks [16].

However, it is very important to enable a more robust and complete analysis of the traffic captured by honeypots and honeynets that use different technologies and are deployed in different locations around the world. This analysis would make it possible to better understand the attacks' distribution and how they interrelate [18]. To allow the correlation of information from different honeypot and honeynet implementations, it is not only necessary to have a system to collect and analyze these data, but also to have a format to represent this information. The existence of a standard format would facilitate the exchange of data, because it would enable the development of tools to automate the generation and retrieval of

information from different sources.

The existing research in the area of honeypot data collection and analysis is concentrated in visualizing and correlating data from a unique honeynet or from a set of honeypots using similar technologies [18, 16]. However, none of them considers the interoperability issue of analyzing data generated in different architectures by different technologies. On the other hand, standard formats to represent data related to attacks and intrusion detection, such as the Intrusion Detection Message Exchange Format (IDMEF) [7] and the Incident Object Description Exchange Format (IODEF) [6], are not adequate to represent data from honeypots.

To contribute to this research area this work proposes HIDEF (Honeypots Information and Data Exchange Format), a format to represent and exchange data and information produced by organizations using honeypots and honeynets. This proposed format intends to solve the interoperability issue among different honeypot implementations, while preserving compatibility with other standards like IDMEF and IODEF.

The remainder of this paper is organized as follows. Section 2 describes the different honeypots technologies and types, the different data analysis approaches and the different data formats for representing data about attacks in computer networks. In this Section some limitations of the approaches and formats shown are also raised. In Section 3 the HIDEF format is discussed, along with its structure and an example of data captured in a low-interaction honeypot represented in HIDEF format. In Section 4 the conclusions and future work are presented.

2 Honeypots and Data Formats

This Section initially presents honeypot types and technologies, as well as the existing approaches to collect and analyze their data. Then, two data formats for the representation of attacks in computer networks will be presented. Finally, the limitations of the methods presented, regarding data interoperability for honeypots using different technologies, will be discussed.

2.1 Honeypots

Honeypots are security resources specially configured to collect information about attacks and whose value lies in being probed, attacked or compromised [17, 16]. In general, they are computers specially configured to register all activities directed to them, such as probes, attacks and intrusions [15]. Also, as a honeypot is not a production system, all traffic directed to it is most likely malicious or anomalous. This makes the data collected in honeypots really valuable, because there are no false positives and it is easy to extract signatures for malicious activities [17].

2.1.1 Types of Honeypots

Honeypots can be classified basically into two types: low-interaction and high-interaction [17, 16].

Low-interaction honeypots are configured just to emulate certain systems and services. When an attacker interacts with one of these honeypots, she is not interacting directly with the system, but with a program or set of programs designed to emulate the system's characteristics, like operating system (OS), application version, vulnerabilities, etc. This way the honeypot is not actually compromised, but is still able to capture several pieces of information about the type of attack being perpetrated, like scan trends and new malicious code, among others [16, 11].

High-interaction honeypots are real systems, usually located in a network with malicious traffic control mechanisms, known as honeynet. These systems, once attacked, can be compromised and the intruder can completely take over the machine. In this kind of honeypot is possible to observe the intruder installing tools, attempting attacks against other networks and modifying the system configuration [18, 16].

Nowadays, honeypots and honeynets are high-end security resources used both to better understand the intruders actions and to help in activities like intrusion detection [16]. It is also a consensus that they can capture valuable information about different attacks on the Internet, as well as help computer security incident handling activities and provide details about the attackers' tactics and motives [17, 18, 12].

2.1.2 Honeypot Technologies

There are several technologies available for the implementation of low- and high-interaction honeypots. In this Section we will present the technologies that are more mature and widely adopted.

The most used programs to implement low-interaction honeypots are Honeyd and Nepenthes [16]. Honeyd is a framework that allows the emulation of hundreds of different systems in several different IP (Internet Protocol) addresses. For each IP address it is possible to specify a different OS to be emulated, and which applications will be emulated in each OS. To emulate the applications and services it is possible to use Honeyd native subsystems or external programs [16, 15]. Honeyd can run in different operating systems, and some logs are generated by the host system. Nepenthes is a honeypot designed to collect samples of malware that propagate automatically, like worms and bots [16, 8]. It emulates the behaviour of a vulnerable software, decodes the attack and retrieves a copy of the malware [8].

As high-interaction honeypots are in fact real systems

deployed in a controlled environment or in a honeynet, the technologies used to implement them have a greater degree of complexity. Each honeypot may have a different OS and this makes the information available to the analysis, after a compromise, very diverse. This information includes OS logs, application logs, process states, kernel related information, files left by the intruder, among several other data. The capture and control mechanisms can also be implemented through different technologies. One of the widely adopted technologies is the honeywall, comprised of a set of integrated tools that restrict the outgoing traffic generated by intruders, therefore stopping them from attacking third party networks, but still allowing the intruder interactions with the honeypot [18, 16]. Honeywall also has mechanisms that enable the capture of any traffic generated by the attackers, and the aggregation and preliminary analysis of the data collected in a given honeynet.

2.2 Data Collection Performed by Groups Studying Honeypots

This Section discusses the approach taken by three well known groups that study and deploy honeypots, to implement a central data collection infrastructure for further data analysis.

2.2.1 Honeynet Project

The Honeynet Project maintains several honeynets around the world, whose data are all stored in a central server. To allow further correlation of the data, a set of requirements that must be fulfilled during the data capture in all honeynets was defined [18]:

- a record with the configuration of all active honeypots in the honeynet must be maintained;
- the data captured by the firewall, router or IDS (Intrusion Detection System) must be stored in GMT (Greenwich Mean Time) timezone;
- each honeynet must have a unique identifier, a name convention and a mapping, that allow to identify its location and configuration;

Although the description of data collection to a central server is available, there is no information about how this server is structured or which type of correlation is performed.

2.2.2 Honeynet Research Alliance

The Honeynet Research Alliance is a forum where the participants are organizations from several countries, that perform research on honeypots and honeynets [18].

Up to now there is no method available for these institutions to share data or analysis among them. However, some organizations send data to a central server maintained by the Honeynet Project. This central server can deal only with data generated by this specific set of capture and control tools: the `libpcap` library binary files and Linux `iptables` firewall logs [18]. This implies that only organizations using these technologies can send data to the central server.

It is also important to notice that this architecture does not enable the exchange of information directly among the organizations. There is also no available information about which kind of data analysis or correlation is being done with the data.

2.2.3 Brazilian Honeypots Alliance

The Brazilian Honeypots Alliance publishes daily statistics about the activities observed in the distributed honeypots that are part of its infrastructure [11]. The data used to generate the statistics are logs in `libpcap` format generated by the OpenBSD packet filter (pf) [10].

Once all the honeypots' data are available at the data collection server, these data are converted into flow format, which stores a summary about the traffic between two IPs, taking into account the ports and protocol used. These flows are processed by ORCA (`http://www.orcaware.com/orca/`) and by RRDtool (`http://oss.oetiker.ch/rrdtool/`), which are the tools used to generate the daily statistics and graphics in the project's website.

2.3 Other Data Collection and Analysis Initiatives

2.3.1 Honeynet Security Console

The Honeynet Security Console (HSC) is a tool developed by Jeff Bell, from the Florida Honeynet Project, to correlate events from a local network or honeynet. This tool allows storing and querying the following types of data: `libpcap` files generated by `tcpdump`, `syslog` logs, firewall logs stored by `syslog` and logs from the Sebek tool, which captures all commands executed by an intruder in a high-interaction honeypot [18].

The HSC authors have chosen to store the data as simply as possible in a SQL database. This was achieved by using programs already available to convert data from each application to a relational database. Each one of these programs creates its own table structure, with different mapping and data types. Thus similar data created by different applications will be represented differently. To solve these inconsistencies HSC needs to perform data type conversions and correlate some data after it performs the queries to the database. In some of the cases if the data were mapped differently, the correlations could be done directly as queries to the database.

2.3.2 A Statistical View of The Recorded Activity On a Honeynet – HoneyStats

The Internet Systematics Lab, from the National Center of Scientific Research "Demokritos" in Greece, maintains the Greek Honeynet Project. In May 2006 they released a tool called "HoneyStats 1.0", which creates statistics about logs generated by the laboratory's honeynet firewall. A demo version of this tool is available at `http://www.honeynet.gr/`.

2.3.3 Georgia Tech Honeynet Project Statistics and Data Visualization

The Georgia Tech Honeynet Project has developed some tools to generate statistics and data visualization from the data collected in their honeynet [5].

The tool HoneyReport parses files in `libpcap` format and generates flows as output. These flows are then analysed to generate statistics about the top activities in the honeynet, like most scanned ports and source of attacks. The tools Rumint and SecVis generate graphics in parallel coordinates based on the traffic captured by the honeynet.

2.4 Data Exchange Formats

XML (Extensible Markup Language) is widely adopted to store and transmit information used by different software and systems [1, 9]. It is actually becoming a universal format for data exchange between applications [13].

In the following Sections we are going to present two XML data formats to represent data captured by IDSs and data from computer security incidents.

2.4.1 Intrusion Detection Message Exchange Format

The IETF (Internet Engineering Task Force) standard described in RFC 4765: "The Intrusion Detection Message Exchange Format (IDMEF)" [7] defines procedures and a data format to share information about intrusion detection. More specifically, it defines a data format to be used by automated alert notification systems [7]. One of its objectives is to allow the correlation of information collected by systems from different vendors and from open source tools.

The IDMEF is an object-oriented representation, implemented in XML, of alert data sent by IDS sensors to their data analysis systems. The definition of the data model has these main requirements: allow the interoperability among different IDSs and accommodate different levels of information provided by different data types, like network traffic, OS logs and application logs [7, 19].

2.4.2 Incident Object Description Exchange Format

The IODEF (Incident Object Description Exchange Format) is a standard format for the representation of data and statistics usually required to effectively respond to a security incident. The IODEF data model is described in the RFC 5070 "The Incident Object Description Exchange Format", from December 2007 [6]. This document also shows an XML implementation of the data model.

As it is common for an incident to be initially observed in control and monitoring systems like IDSs, the IODEF kept its definition compatible with the IDMEF format. This way, data generated in IDMEF format can be easily imported to an IODEF document.

2.5 Limitations in the Collection, Analysis and Data Format Areas

As we could see in the previous Sections the existing studies in the areas of collection and analysis of honeypots' data are focused in visualization and/or correlation of data in a given honeynet or in a set of honeypots that use similar technologies [18]. Although some of these studies present interesting characteristics and have promising results, none of them takes into consideration the issue of interoperability to analyze data generated by different technologies in different architectures.

On the other hand, the IODEF and IDMEF standards have a focus in interoperability and a structure that allows representing network traces and information about IDSs alerts adequately. However, these models are too focused on network attack signatures and on mapping relationships among security incidents. From the point of view of research and data correlation of events observed in a honeypot, it is necessary to also exchange information about the technology being used. This would allow, for example, the comparison between data from low- and high-interaction honeypots or determining if filters applied to a network have implicated in differences in the results.

To address the existing limitations of the honeypot research area, this paper proposes the HIDEF, an exchange format for data collected in honeypots, and information about the architecture and technologies used by honeypots. Section 3 will present HIDEF, its requisites and data model.

3 HIDEF

As seen in Section 2, honeypot data correlation has been performed only in data with the same format and among honeypots using similar technologies. However, it is very important to enable a more robust and complete analysis of the traffic captured by honeypots and honeynets that use different technologies and are deployed in different

locations around the world. This analysis would make possible to better understand the attacks' distribution and how they interrelate [18].

This work contributes to this research area in proposing a data format for the exchange of information and data collected in honeypots. This format is the HIDEF – Honeypots Information and Data Exchange Format.

3.1 Requirements

The HIDEF format takes into account some requirements for the representation of relevant information to analyze and correlate activities captured by honeypots.

To perform a correct analysis it is necessary to consider the architecture used and the context were the data was captured. To enable this, it is necessary that the format allows representing data about the honeypots configuration and the technologies used, such as: what is the type of the honeypot; if it is connected to a honeynet; which OS and services are being used or emulated; open TCP, UDP and ICMP ports in the honeypot; and if there are filters that prevent certain types of traffic to reach the honeypot.

It is equally important to be able to represent different types of data captured, like intruder activity logs and attack attempts received by the honeypot. Among these data it is important to highlight the network traces, malicious codes, artifacts resulting from malicious activities and the analysis performed. It is also necessary that the format allows exchanging sanitized information, that is, information that omits sensitive network details from the organizations that are interested in exchanging data collected by their honeypots.

Another requirement that needs to be considered is the compatibility with standard formats already defined for other sets of data related to network security. This is important to enable the correlation among data from honeypots and data from IDSs or from computer and network security incidents. To fulfill this requirement it is necessary that the HIDEF remains compatible with the IDMEF [7] and the IODEF [6] formats, discussed in Section 2.4.

3.2 Data Types

Whenever possible, XML Schema native data types [2] are used in the definition of the HIDEF classes and attributes. However, to represent some specific data we defined a few data types. These data types are derived from IODEF for compatibility reasons. The types defined are listed below.

ML_STRING – this type is derived from IODEF. It is used to represent any text related to analysis, general descriptions or any other information that may be written in multiple languages. It is represented by a string (`xs:string` [2]) that has a language (`xs:language` [2]) as an attribute.

PORTLIST – this type is also derived from IODEF. It represents a list of network services ports. This format allows representing ports (N) and port sequences (N-M) separated by commas. This type is represented by a string (`xs:string` [2]) restricted by the following regular expression:
`\d+(\-\d+)?(,\d+(\-\d+)?)*`.

ENUM – enumerated data, also derived from IODEF, are used in several attributes. They are defined as sequences of NMTOKEN (`xs:NMTOKEN` [2]). The description of each specific enumerated type will be made together with the description of the class that uses it for the first time.

3.3 Data Model

The HIDEF is an object-oriented representation of information related to the configuration and the technologies used by honeypots, as well as data captured by them. The HIDEF classes represent XML elements and their attributes represent XML attributes. Throughout this Section the following conventions are used: the classes represented in gray are derived from the IODEF model; the names of the attributes and all classes are written in sans-serif font.

Each HIDEF document is an instance of the HIDEF-Document class, which is comprised of one or more instances of the Honeypot class, as we can see in Figure 1. The HIDEF-Document class has three attributes: version, that represents the version of the HIDEF model being used; lang, that represents the language of the document according to the values defined in [14]; and instructions, a field reserved to contain parsing instructions, whose semantic must be defined by the organizations exchanging information.

The Honeypot class provides a representation of the several components related to the data from a given honeypot. It has the following attributes: restriction, type and ext-type. The restriction attribute, as it is used also in IODEF, will have the same meaning in HIDEF, as well as the same possible values. This attribute is inherited by all classes that are children of Honeypot. Some of these classes also have a restriction attribute, in which case it is possible to assign new values to them. The possible values are: public, need-to-know, private and default. When the value is default it means that a policy predefined by the organizations involved should be used. The type attribute determines which type of honeypot is being represented, and can have the values low-interaction, high-interaction or ext-value. The ext-type attribute is optional and was

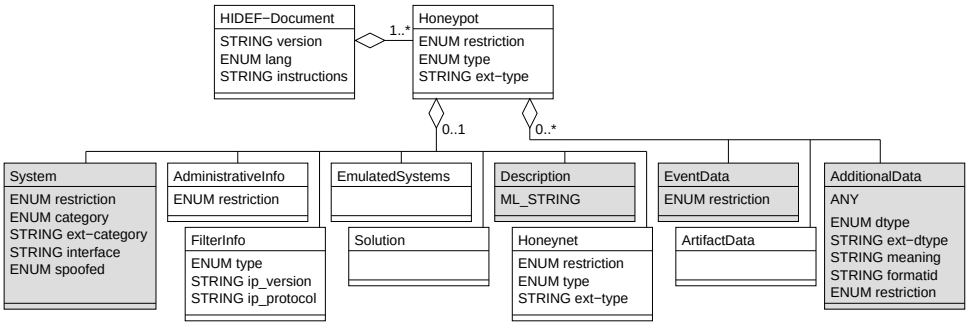

Figure 1: HIDEF-Document and its aggregated classes.

defined as a mean to extend the type attribute. More details about how to extend this and other similar types is discussed in Section 3.4.

Each Honeypot class instance may have zero or one instance of the following classes: System, AdministrativeInfo, EmulatedSystems, Description, FilterInfo, Solution and Honeynet. These classes represent the information about the technologies used and the context in which the data was collected. To represent different types of data captured, the logs of the intruders activities and attack attempts received, each instance of the Honeypot class may have zero or more instances of the classes EventData, ArtifactData and AdditionalData. These classes and their relationship with the Honeypot class can be seen in Figure 1.

The System, Description, EventData and AdditionalData classes are derived from IODEF and their full description, including attributes and aggregated classes, is available in [6]. In the HIDEF context a System class will hold information about the honeypot system, including operating system, services, applications, IP address version 4 or 6, among others. The Description class may hold a general description of the honeypot and comments that the organization creating the document considers relevant. EventData is the class that will store the data about most of the malicious activities captured by the honeypot. This class allows representing the start and end time of an activity, network traces, involved protocols, logs, attack methods, and assessment about the impact of the attack registered, among other data. The AdditionalData will be discussed in Section 3.4.

The AdministrativeInfo class, seen in Figure 2, aggregates the classes Hardware, EmulatedAddress, Sanitization, Contact and AdditionalData. It can have zero or more instances of these aggregated classes, which hold administrative information about the honeypot. In case of a low-interaction honeypot the EmulatedAddress class, which has the same definition of the IODEF Address class [6], can be used to represent the network range being used by the honeypot for the emulated systems. Hardware enables to store relevant text information about the hardware being used by the honeypot. The Sanitization class can be used to reference the relation between the honeypot real addresses and the sanitized addresses stored in the System and EventData classes. It is very important to keep this relation because it allows an institution generating the logs with the addresses already sanitized to share the information about the real addresses. The Contact class is derived from IODEF [6] and permits, for example, that members of groups studying honeypots or similar projects share information about who is responsible for a given honeypot. The AdditionalData class will be discussed in Section 3.4.

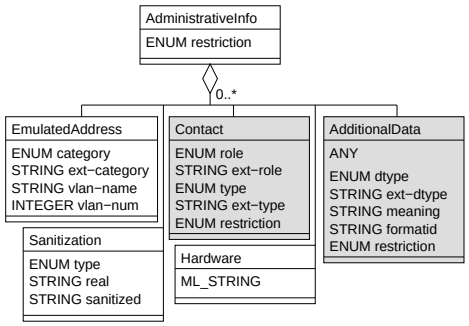

Figure 2: AdministrativeInfo and its aggregated classes.

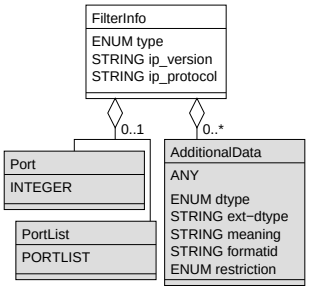

Figure 3: FilterInfo and its aggregated classes.

The FilterInfo class (Figure 3) maps the information about filters applied between the Internet traffic and the honeypot. This information is very important for the correlation of data captured in different honeypots. Without

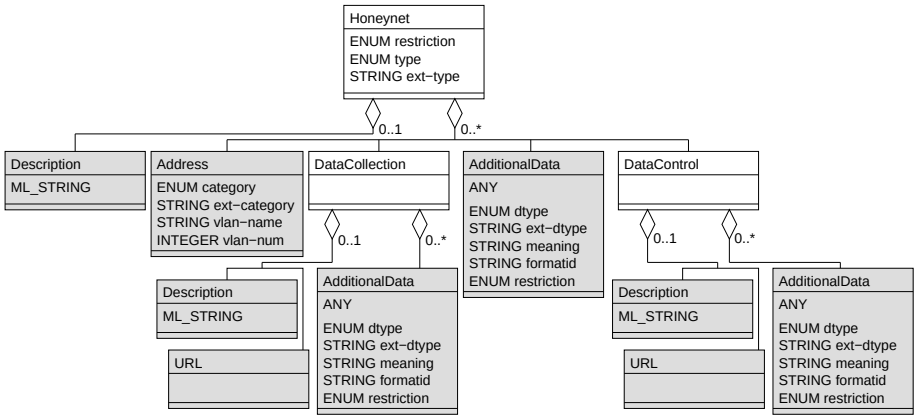

Figure 5: Honeynet and its aggregated classes.

this information it is not possible to know if a given malicious traffic was not seen because the honeypot was not attacked or the malicious traffic was blocked. If the type attribute has the value open this means that the traffic to the ports listed in Port and Portlist is allowed and everything else is denied. If the type attribute is closed, this means that the traffic to the ports listed in Port and Portlist is blocked and everything else is allowed. The Port and Portlist classes are derived from IODEF and their complete description is available in [6]. The AdditionalData class will be discussed in Section 3.4.

The EmulatedSystems class represents different systems being emulated in a low-interaction honeypot. It is formed by one or more instances of the System class, as we can see in Figure 4. The System class has basically the same definition as in IODEF [6], but in HIDEF it has been extended to aggregate one more class: the EmulatedApplication. This class allows storing specific information about the programs used to emulate the applications. Each instance is formed by a software description, represented in Description, and an URL (Uniform Resource Locator) pointing to its web page, represented in the URL class.

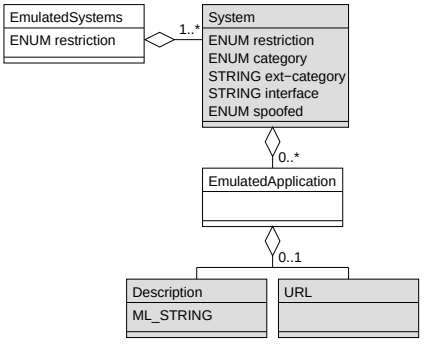

Figure 4: EmulatedSystems and its aggregated classes.

The Honeynet class, that we can see in Figure 5, allows representing information about the honeynet where a high-interaction honeypot may be located. Its type attribute may have three values: physical, virtual and ext-value. The ext-type attribute permits the extension of the type attribute, as will be discussed in Section 3.4. The Honeynet class may have zero or one Description class, that can contain a general description of the honeynet. The other classes may occur zero or more times. The Address class is derived from IODEF and represents the following types of addresses: IP, MAC (Media Access Control), AS (Autonomous System), among others related to a honeynet. The DataCollection and DataControl classes have similar structures and represent data about the technologies used in a honeynet to collect data and control the intruders' activities [18]. The AdditionalData class will be discussed in Section 3.4.

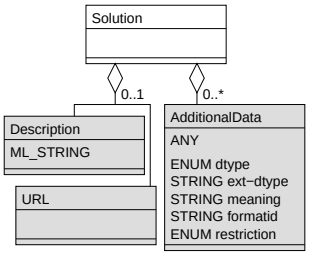

Figure 6: Solution and its aggregated classes.

The Solution class represents, in low-interaction honeypots, the solution used to implement the honeypot. As we can see in Figure 6 an instance of this class is comprised of zero or one instances of the Description class, where we can store a general description of the solution, and of the URL class, that can hold a URL to the application site. The AdditionalData class can hold additional information needed and will be discussed in Section 3.4.

The ArtifactData, seen in Figure 7, represents artifacts

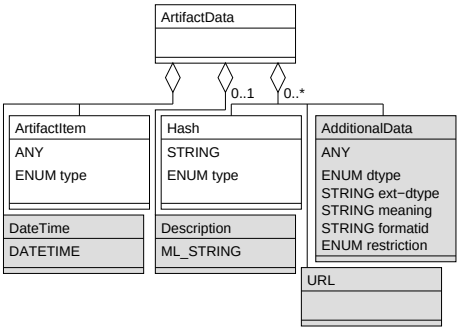

Figure 7: ArtifactData and its aggregated classes.

collected in low- or high-interaction honeypots, together with their capture and analysis information. An instance of this class has exactly one ArtifactItem, where the artifact itself is stored, and the timestamp of the capture, represented in DateTime. In Description we can store a general description of the artifact or its analysis. Each ArtifactData instance may also have zero or more instances of the Hash, URL and AdditionalData classes. The Hash class may store a cryptographic hash or a digital signature for the artifact, this data can be used to uniquely identify each artifact. Its type attribute determines what kind of hash was used, and may have one of the following values: md5, sha1, sha256, sha512, rmd160 or pgp-signature. The URL class can be used in case the artifact is already in public domain and has a URL associated to it. The AdditionalData class will be discussed in Section 3.4.

To exemplify the use of HIDEF, Figure 8 presents a document with the representation of an SSH scan against a low-interaction honeypot.

3.4 HIDEF Extensions

Taking into account the dynamic nature of the computer security area, HIDEF has means for the definition of extensions to the model allowing the inclusion of new characteristics to the data model. As it is an object-oriented model it is always possible to extend it through inheritance, defining subclasses with attributes not present in the super-class. As we will see below, it is also possible to include new classes.

The AdditionalData class, derived from IODEF, acts as an extension mechanism. It allows including information not originally represented in the data model or including new classes. This class permits the inclusion of atomic elements, like integers and strings, or more complex data, like complete XML documents, derived from other Schemas (like IDMEF or IODEF). Its type attribute holds the definition of the data type, and its meaning attribute defines which is the meaning of that data in the HIDEF document context.

```
<?xml version="1.0" encoding="UTF-8" ?>
<HIDEF-Document version="1.00" lang="en"
 xmlns="hidef-1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Honeypot restriction="public" type="low-interaction">
 <System restriction="private" category="target">
  <Node>
   <Address category="ipv4-addr">192.0.2.1</Address>
  </Node>
  <OperatingSystem vendor="OpenBSD" version="3.9" />
  <Description>
   Low-interaction honeypot real OS.
  </Description>
 </System>
 <AdministrativeInfo restriction="private">
  <EmulatedAddress category="ipv4-net">
   10.0.0.0/26</EmulatedAddress>
  <Sanitization type="ipv4-addr" real="10.0.0.1"
                sanitized="xxx.xxx.xxx.1" />
 </AdministrativeInfo>
 <EmulatedSystems>
  <System restriction="private" category="target">
   <Node>
    <Address category="ipv4-addr">10.0.0.1</Address>
   </Node>
   <Service ip_protocol="6"><Port>22</Port></Service>
   <OperatingSystem vendor="Linux" version="2.4.7" />
   <EmulatedApplication>
    <Description>
     SSH emulator derived from the dropbear server.
    </Description>
    <URL>http://matt.ucc.asn.au/dropbear/dropbear.html
    </URL>
   </EmulatedApplication>
  </System>
 </EmulatedSystems>
 <Solution>
  <Description>Honeyd</Description>
  <URL>http://www.honeyd.org/</URL>
 </Solution>
 <EventData>
  <StartTime>2006-11-03T16:12:31Z</StartTime>
  <EndTime>2006-11-03T16:12:45Z</EndTime>
  <Flow>
   <System category="source">
    <Node>
     <Address category="ipv4-addr">
      192.168.8.2</Address>
    </Node>
    <Service ip_protocol="6">
     <Portlist>2458,2535,2598,2752</Portlist>
    </Service>
   </System>
   <System category="target">
    <Node>
     <Address category="ipv4-addr">
      xxx.xxx.xxx.1</Address>
    </Node>
    <Service ip_protocol="6"><Port>22</Port></Service>
   </System>
  </Flow>
  <Record>
   <RecordData>
    <RecordItem dtype="string">
    Nov 03 16:12:31 192.168.8.2.2458 > xxx.xxx.xxx.1.22
    Nov 03 16:12:34 192.168.8.2.2535 > xxx.xxx.xxx.1.22
    Nov 03 16:12:37 192.168.8.2.2598 > xxx.xxx.xxx.1.22
    Nov 03 16:12:45 192.168.8.2.2752 > xxx.xxx.xxx.1.22
    </RecordItem>
   </RecordData>
  </Record>
 </EventData>
</Honeypot>
</HIDEF-Document>
```

Figure 8: HIDEF document with data about an SSH scan against a low-interaction honeypot.

There is another mechanism that enable the extension of some attributes defined as enumerated data types. All attributes whose name is of the form “ext-<name>” can be used to extend the attributes of the same “<name>”. An example are the attributes type and ext-type from the Honeypot class (Figure 1). Nowadays the honeypots are

classified in the literature as low- and high-interaction. However, this classification may be expanded in the future. If this happens the type attribute can be extended by assigning to it the value "ext-value", and assigning a "new-value" to the optional attribute ext-type, representing a new type of honeypot.

4 Conclusions and Future Work

The data collected by honeypots and honeynets, and the information about their technologies, are complex. This requires an equally complex data structure to represent them. In the HIDEF model these data are represented in a structured way, giving the data appropriate semantics. This enables quick access to the data and facilitates the interoperability and the processing automation.

The proposed format is compatible with the IODEF and IDMEF standards, allowing data collected by honeypots to be more easily correlated with data from IDSs and network security incidents. Additionally, the data collected from honeypots and represented in HIDEF format can easily be exported to the IODEF format, allowing them to be notified as a security incident. This is possible mainly because HIDEF uses a class derived from the IODEF EventData to represent network traces and logs.

The next steps include the development of a client-server system that will allow the creation and exchange of HIDEF documents in a secure way. This system will also validate and import these data into a database system. A case study will be implemented to exchange and correlate data from a low-interaction honeypot network and a honeynet, both already operational.

References

[1] Extensible Markup Language (XML). http://www.w3.org/XML/. Access date: Oct 27, 2007.

[2] XML Schema Part 2: Datatypes Second Edition – W3C Recommendation. http://www.w3.org/TR/xmlschema-2/, October 2004. Access date: Oct 27, 2007.

[3] Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br). Incidentes Reportados ao CERT.br. http://www.cert.br/stats/incidentes/. Access date: Oct 27, 2007.

[4] CERT Coordination Center. CERT/CC Statistics 1988-2007. http://www.cert.org/stats/cert_stats.html. Access date: Oct 27, 2007.

[5] Conti, G. and Abdullah, K. Passive visual fingerprinting of network attack tools. In *Proceedings...*, pages 45–54. VizSEC/DMSEC '04: 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM Press, October 2004.

[6] Danyliw, R., Meijer, J., and Demchenko, Y. RFC 5070: the incident object description exchange format. http://www.ietf.org/rfc/rfc5070.txt, December 2007. Access date: Feb 2, 2008.

[7] Debar, H., Curry, D., and Feinstein, B. RFC 4765: The intrusion detection message exchange format (IDMEF). http://www.ietf.org/rfc/rfc4765.txt, March 2007. Access date: Oct 27, 2007.

[8] Göbel, J., Hektor, J., and Holz, T. Advanced honeypot-based intrusion detection. *;login: The USENIX Magazine*, 31(6):17–25, December 2006.

[9] Harold, E. R. and Means, W. S. *XML in a Nutshell, A Desktop Quick Reference*. O'Reilly and Associates, Inc., 2nd edition, June 2002. ISBN-10: 0-596-00292-0.

[10] Hartmeier, D. Design and Performance of the OpenBSD Stateful Packet Filter (pf). In *Proceedings...* FREENIX Track: 2002 USENIX Annual Technical Conference (FREENIX '02), June 2002.

[11] Hoepers, C., Steding-Jessen, K., Cordeiro, L. E. R., and Chaves, M. H. P. C. A National Early Warning Capability Based on a Network of Distributed Honeypots. In *Proceedings...* Annual FIRST Conference on Computer Security Incident Handling, June 2005.

[12] Hoepers, C., Steding-Jessen, K., and Montes, A. Honeynets Applied to the CSIRT Scenario. In *Proceedings...* Annual FIRST Conference on Computer Security Incident Handling, June 2003.

[13] Milo, T., Abiteboul, S., Amann, B., Benjelloun, O., and Ngoc, F. D. Exchanging intensional XML data. *ACM Trans. Database Syst.*, 30(1):1–40, 2005.

[14] Phillips, A. and Davis, M. RFC 4646: Tags for identifying languages. http://www.ietf.org/rfc/rfc4646.txt, September 2006. Access date: Oct 27, 2007.

[15] Provos, N. A virtual honeypot framework. In *Proceedings...*, pages 1–14. USENIX Security Symposium, August 2004.

[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] Spitzner, L. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 1st edition, September 2002. ISBN-10: 0321108957.

[18] The Honeynet Project. *Know your Enemy: Learning About Security Threats*. Addison-Wesley Professional, 2nd edition, May 2004. ISBN: 0-321-16646-9.

[19] Wood, M. and Erlinger, M. RFC 4766: Intrusion detection message exchange requirements. `http://www.ietf.org/rfc/rfc4766.txt`, March 2007. Access date: Oct 27, 2007.

D.2 Artigos em Simpósios e Conferências

O Artigo entitulado "*Honeynets Applied to the CSIRT Scenario*", foi publicado e apresentado na Conferência "*15th Annual Computer Security Incident Handling Conference*", em junho de 2003, em Ottawa, Canadá.

O Artigo entitulado "*A National Early Warning Capability Based on a Network of Distributed Honeypots*", foi publicado e apresentado na Conferência "*17th Annual Computer Security Incident Handling Conference*", em junho de 2005, em Cingapura, Cingapura.

Conforme requerido pelo Regimento dos Cursos de Pós-Graduação em Computação Aplicada, nas próximas páginas constam os artigos na íntegra, conforme publicados nos anais das Conferências.

Honeynets Applied to the CSIRT Scenario

Cristine Hoepers
NIC BR Security Office – NBSO
cristine@nic.br

Klaus Steding-Jessen
NIC BR Security Office – NBSO
jessen@nic.br

Antonio Montes
National Institute for Space Research – INPE
montes@lac.inpe.br

Abstract

A honeynet is a research tool consisting of a network specifically designed for the purpose of being compromised, with control mechanisms that prevent this network from being used as a base for launching attacks against other networks. Once compromised, the honeynet can be used to observe the intruders' activities, collect tools and determine new trends in network attacks. In this paper we discuss the implementation of a honeynet, based entirely on open source software, that meet the requirements listed above. We present its topology, the tools developed and the results achieved. We also discuss how valuable a honeynet can be to better understand the threats to the constituency of a Computer Security Incident Response Team (CSIRT).

1 Introduction

Due to the necessity of understanding both the attacks to Internet connected networks and the profile of the intruders, some research groups[1] have joined to develop, deploy and monitor honeynets.

Honeynets are research tools consisting of a network specifically designed for the purpose of being compromised [1,2]. Once compromised, the honeynet can be used to observe the intruders' activities and behavior. This allows a detailed analysis of the tools used by the intruders as well as the vulnerabilities used to compromise the honeypots.

To join the efforts of the honeynet community we have implemented a Honeynet in Brazil with the intention of helping us monitor the malicious activities from our country's point of view and share this data with people from other countries and honeynets. One of our main concerns was to implement a low cost honeynet and still have a high quality data control in place. With this in mind we started developing our data control architecture and tools all based on open source free software. The tools we developed are also freely available. After a test phase we put the Honeynet.BR in operation as a cooperative research work between NIC BR Security Office (NBSO) and the Brazilian National Institute for Space Research (INPE). The Honeynet.BR Project provides the means to observe ocurring attacks and intrusions, collect data and develop new tools to improve the honeynet technology.

In this paper we initially present the adopted architecture and methodologies applied to the Honeynet deployed. We also discuss the mechanisms implemented to contain the outgoing malicious traffic, capture data and generate alerts. The activities observed and the usefulness of a Honeynet to a CSIRT are discussed as well.

2 The Honeynet Components

The initial plans for the Honeynet.BR implementation were made in December 2001 and it started operations in March 2002. In the initial phase the main project decisions were taken regarding its topology, operating systems and tools to be used, and development of other tools to analyse and contain the traffic.

[1] http://www.honeynet.org/alliance/

The Honeynet.BR topology, shown in Fig. 1, is divided into two distinct parts: the Administrative Network and the Honeynet itself.

The Administrative Network has the main function of containing the outgoing malicious traffic and monitoring all the incoming and outgoing traffic. This network is completely transparent to both the Honeynet and the Internet, and it is comprised of:

- A Firewall that allows all incoming traffic to the Honeynet and blocks malicious outgoing traffic. This control is exerted in the layer 2 level with the Firewall operating as a bridge. The Firewall functionalities will be discussed in depth in section 2.2;
- A Hogwash machine configured to block the outgoing traffic that has well-known malicious content. This machine also operates as a bridge. More details in section 2.2.4;
- An IDS that captures and analyses all traffic related to the Honeynet, and sends alerts in case of a compromise. It also generates daily summaries about all the activities. Details about its implementation are available in sections 2.3 and 2.4;
- A file server (named Forensics) dedicated to the storage of artifacts and disk images of the Honeynet hosts. Details are further discussed in section 2.1.1.

The main mechanisms used to contain traffic and generate alerts were developed by the Honeynet.BR Team using OpenBSD as the operating system platform.

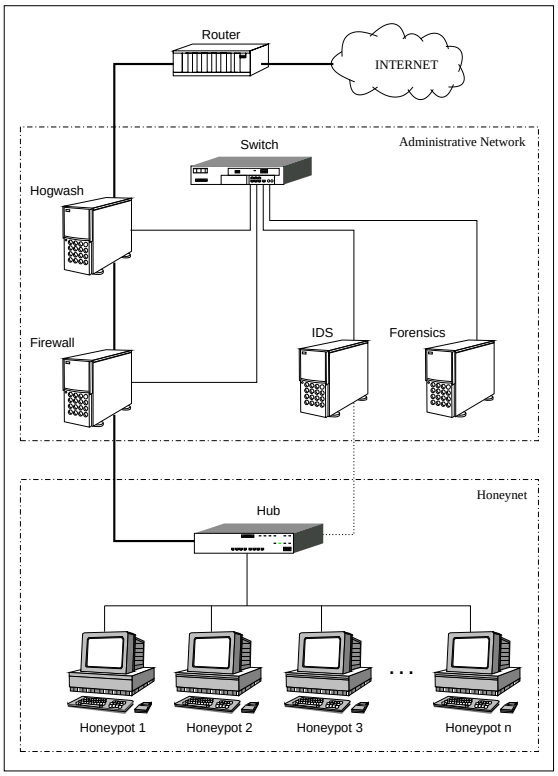

Figure 1: Honeynet.BR Topology

2.1 Honeypots

The Honeynet itself is composed of several hosts (the honeypots) running different operating systems and services. One of this hosts is the nameserver of the Honeynet and the central logserver. The honeypots installation details are discussed in the next section.

2.1.1 Procedures Applied to Honeypots

During the honeypots deployment process a set of procedures is followed to maintain a record of the system and services installed in each honeypot, as well as to prevent data related to previous intrusions to remain in the disk.

The basic steps taken during the honeypots deployment process are:

1. The previous disk data is erased by writing zeroes over the entire disk. This procedure also has the advantage of providing a better compression rate of the disk image [3];
2. The honeypot operating system is installed and its services configured;
3. A compressed disk image of the newly installed honeypot is stored in the Forensics machine;
4. This entire process is recorded in a logbook;
5. The honeypot is connected into the Honeynet.

The honeypot is closely monitored after its deployment. Once compromised, all the malicious activities related to the honeypot and the artifacts col-

lected are recorded and analyzed. In general the monitoring process of the honeypot consists of:

1. Recording in another logbook all activities observed while the intrusions last;
2. Storing all tools used by the intruders in the Forensics machine;
3. Disconnecting, when appropriate, the honeypot from the Honeynet, making an image from the compromised disk and storing it in the Forensics machine;
4. Restarting the deployment process.

2.2 Data Control

One of the most important requirements of a Honeynet is to contain the malicious outgoing traffic. This guarantees we can observe the intrusion and not let the intruder use the Honeynet as a base to launch new attacks.

In the following sections we describe the methods developed to contain outgoing traffic in the Honeynet.BR.

2.2.1 Firewall Rules

The Firewall is configured as a bridge, this means it does not have an IP address and does not decrement the TTL (*Time to Live*) of the IP packets that pass through it, reducing the chances of the Firewall being noticed.

The rules are implemented using the OpenBSD Stateful Packet Filter [4] (`pf`). They allow any incoming traffic and drop all the potentially malicious outgoing traffic. Some examples of the potentially malicious traffic it drops are:

- spoofed IPs;
- some ICMP packets;
- UDP packets depending on its source and destination;
- TCP traffic directed to well-known vulnerable services or with unusual characteristics.[2]

[2]For example when the source port is the same as the destination.

2.2.2 Outgoing Traffic Normalization

There are some attacks that overlap IP fragments to confuse intrusion detection systems and firewalls, as well as some scanning tools that use invalid TCP flag combinations to achieve the same goal. These invalid flags are also used for remote TCP/IP fingerprinting.

To contain this kind of malicious packets the `pf` normalization mechanism is used to reassemble and discard invalid packets.

In addition to the usual normalization made by `pf`, we have also modified the `pf` source code to discard packets with both `SYN` and `FIN` flags on, feature that is frequently used by scanning tools.

2.2.3 `sessionlimit` Implementation

The OpenBSD `pf` is a stateful packet filter and is configured to create sessions for the outgoing packets that are not dropped by the rules mentioned before. By inspecting the `pf` state table it is possible to obtain some information about each created session, for example:

- direction (in or out);
- protocol;
- source and destination IPs and ports;
- total number of bytes and packets;
- creation and expiration time;
- status (`SYN_SENT`, `ESTABLISHED`, etc);

Based on these characteristics we have built an open source tool called `sessionlimit`. This tool continuously monitors the state table entries and interacts with `pf` inserting and removing rules as necessary.

`Sessionlimit` can block the traffic related to a specific host based on one of the following criteria:

1. when the number of states related to a source IP is increasing too fast;
2. when the source IP has reached a predefined limit of outgoing connections (the default is 20 connections).
3. when the number of bytes associated with an ICMP state has reached a predefined limit.

If any of these conditions is satisfied a rule blocking all outgoing traffic from this host is inserted in the current `pf` rules, and all outgoing sessions for this IP are removed.

It is important to note that the blocking rule inserted by `sessionlimit` affects the outgoing traffic only. All incoming sessions already established to this host are not affected. This typically includes the intruder's interactive session.

`Sessionlimit` removes a blocking rule from the list of active rules in the Firewall after a predefined timeout.

All actions taken by `sessionlimit` are logged via `syslog`, as shown in Fig. 2.

```
Jul 15 04:42:49 fw sessionlimit[29832]: starting

[...]

Jul 23 19:58:29 fw sessionlimit[29832]: \
    ICMP: blocking xxx.xxx.xxx.xxx (65016 bytes)
Jul 23 19:58:29 fw sessionlimit[29832]: \
    11 state(s) killed from xxx.xxx.xxx.xxx
Jul 23 20:28:29 fw sessionlimit[29832]: \
    expiring xxx.xxx.xxx.xxx after 1800 seconds

[...]

Oct  2 13:04:27 fw sessionlimit[29832]: \
     blocking xxx.xxx.xxx.xxx (20 states)
Oct  2 13:04:27 fw sessionlimit[29832]: \
     20 state(s) killed from xxx.xxx.xxx.xxx
Oct  2 13:34:27 fw sessionlimit[29832]: \
     expiring xxx.xxx.xxx.xxx after 1800 seconds
```

Figure 2: `sessionlimit` log excerpts. Some lines are broken to facilitate reading. The real IPs were removed.

2.2.4 Outgoing Content Filters

Besides the methods described above, that are implemented by the Firewall, malicious outgoing packets can be dropped by the Hogwash machine based on its contents.

We use the open source tool `hogwash`[3] to do this analysis. The packets are dropped by this tool whenever the contents match a well-known attack signature.

`Hogwash` uses the same rules as the `snort`[4] Intrusion Detection tool. In our Honeynet `hogwash` is executed in a machine assigned for this purpose, as shown in Fig. 1.

One of the advantages of using `hogwash` is the facility to update its signatures. They can be obtained from the security community or can be created based on attacks previously observed in the Honeynet.

2.2.5 Bandwidth Limitation

As an additional measure to contain malicious traffic we have decided to restrict the available outgoing bandwidth by using `ALTQ` (*Alternate Queueing*)[5].

The goal is to limit the intensity of a Denial of Service attack originated in the Honeynet, in case the other data control mechanisms fail.

2.3 Data Capture

All incoming and outgoing traffic, as well as the traffic inside the Honeynet, is captured and stored. The data is captured in two places:

1. Firewall

   The Firewall stores all incoming and outgoing traffic in `tcpdump` binary format[6]. This is done through the `pf` logging mechanism. The use of this standard format facilitates the data manipulation, sice it allows the use of well-known tools like `tcpdump`, `ethereal`, `ngrep`, etc.

2. IDS

   The IDS has a network interface, with no IP address assigned to it, which captures all data circulating between the honeypots and the incoming and outgoing data.

   The data capture is made by a script that uses `tcpdump` to read the data and store it in files named by year, date and time of the beginning of the capture.

[3] `http://hogwash.sourceforge.net/`

[4] `http://www.snort.org/`

[5] `www.csl.sony.co.jp/person/kjc/software.html`

[6] With the exception of the spoofed traffic, typically used in Denial of Service attacks, because of the huge volume of logs it generates

The data captured by the IDS is used by the alerting mechanisms and also used to generate the daily summaries. This is described in detail in section 2.4.

2.3.1 Data Rotation and Compression

All the data captured by the Firewall and by the IDS are rotated and compressed every 24 hours. The name of the generated files follows our convention, which is year, month and day of the generation. After a period of 30 days each file is moved to an off-line storage media.

2.4 Alerts and Summaries

2.4.1 Alerts

The generation of alerts follows the principle that any traffic observed in the Honeynet is malicious. Outgoing traffic from the Honeynet is a clear indication of a compromise.

The alerts can be generated as follows:

1. Outgoing traffic

   A script using `tcpdump`, running in the IDS machine, filters the captured data. Any outgoing packet originating from the Honeynet, that is not in response to an incoming packet, generates an alert. All alerts are grouped and sent by email periodically.

2. Shell commands

   The Honeynet Unix machines have a modified shell that sends the commands history to the log server via the syslog service. A script running in the IDS machine monitors the traffic and produces an alert if the logs generated by the shell are detected.

An example of an alert generated by a shell command is shown in Fig. 3.

One single alert can contain any of the types described above. A copy of all alerts is maintained in the IDS machine for future reference.

The generation of alerts can be easily configured to have different levels of sensibility. This is important to reduce the number of false positives.

```
2002/09/22 02:41:01 host:514 -> loghost:514
HISTORY: UID=48 rm -rf /tmp/.unlock.uu
/tmp/.unlock.c /tmp/.update.c /tmp/httpd
/tmp/update /tmp/.unlock;

2002/09/22 02:41:01 host:514 -> loghost:514
HISTORY: UID=48 cat > /tmp/.unlock.uu <<
__eof__;

2002/09/22 02:41:10 host:514 -> loghost:514
HISTORY: UID=48 uudecode -o /tmp/.unlock
/tmp/.unlock.uu; tar xzf /tmp/.unlock -C /tmp/;
gcc -o /tmp/httpd /tmp/unlock.c -lcrypto; gcc -o
/tmp/update /tmp/.update.c;
```

Figure 3: Example of an alert generated because of the capture of a shell activity by the IDS. Some lines were edited because of legibility. The real IPs were removed.

In addition to email, the alerts can be sent by pager or mobile phone.

2.4.2 Summaries

A summary is issued daily containing the activity observed in the Honeynet the day before. This summary is sent by email and also stored in the IDS machine. The input data for each summary is the IDS compressed data captured during the period of one day. The output contains:

1. Statistics

   - total number of packets captured;
   - number of packets by protocol and corresponding percentage;
   - hosts that originated most TCP, UDP and ICMP traffic;
   - TCP and UDP ports with more incoming traffic.

2. `snort` alerts

   The `snort` program is used to read the captured data and to generate alerts, which are listed in the summary.

3. Incoming traffic

   We include a condensed `tcpdump` output of the incoming traffic only. This is particularly useful to verify if a certain behavior, like a scan, happened only against one host or in the whole honeynet.

3 Activities Observed

During the observation period several malicious activities were detected. This allowed us to collect tools and monitor both the vulnerabilities explored and the exchange of information between the intruders. In this section we discuss the various types of attacks and trends we have observed so far.

3.1 Top Scanned Services

The scans in their majority were targeted at the `http` (80/TCP) and `ftp` (21/TCP) services. These were, by far, the most scanned services in the Honeynet. The other scans were directed mostly to the following ports: 22/TCP, 23/TCP, 111/TCP, 515/TCP and 6112/TCP.

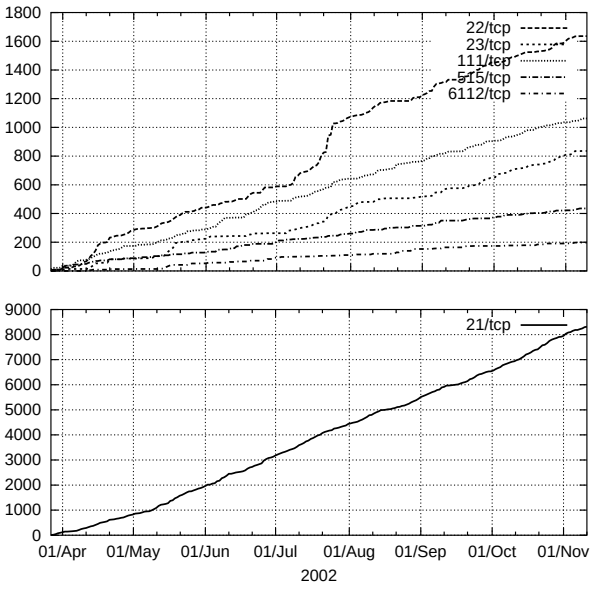

Figure 4: Scanned ports – cumulative

In Fig. 4 we see the cumulative number of scans, with the exception of port 80/TCP, directed to these ports. It is interesting to notice that the scans for `ssh` port show a marked growth in mid July, nearly a month after the release of the CERT® Advisory CA-2002-18[7].

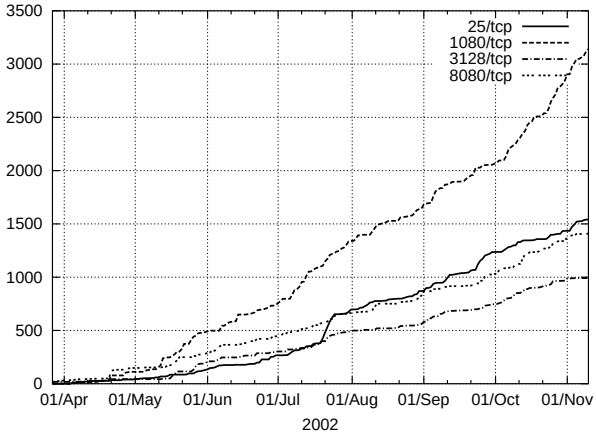

Figure 5: Scan for open proxies and relays – cumulative

We also detected that searches for open proxies and poorly configured email servers that permit email relay have been constant, as shown in Fig 5. Sometimes the activity is more intense than the search for vulnerable services.

3.2 DoS and DDoS Attacks

We observed the intruders trying to use several different Denial of Service tools and techniques. This varied from the trivial "`ping -f`" command, to tools using UDP fragmented packets.

We have also seen coordinated distributed attacks launched via IRC. They have used the tool `kaiten`[8], that is an IRC based DDoS client. This client connects to a specific server and receives commands via an IRC channel.

3.3 Worms

We have detected a high number of attacks made by worms. This attacks, in general, were targeted against web servers.

[7] `http://www.cert.org/advisories/CA-2002-18.html`
[8] `http://packetstormsecurity.nl/irc/kaiten.c`

In Fig. 6 we can see incoming traffic to ports associated with worm activity. It is possible to observe that scans for port 80/TCP (Nimda, CodeRed, and others) have shown a constant rate since the beginning. The scans for port 1433/TCP (SQL Worm) started in May, around the same time CERT®/CC released its "Incident Note IN-2002-04"[9]. And the scans for port 443/TCP (Slapper and variants) had a small increase in activity starting in the first half of September, when CERT®/CC released its "CERT® Advisory CA-2002-27"[10], but the major increase occurs after the release of the worm variants about September 20th.

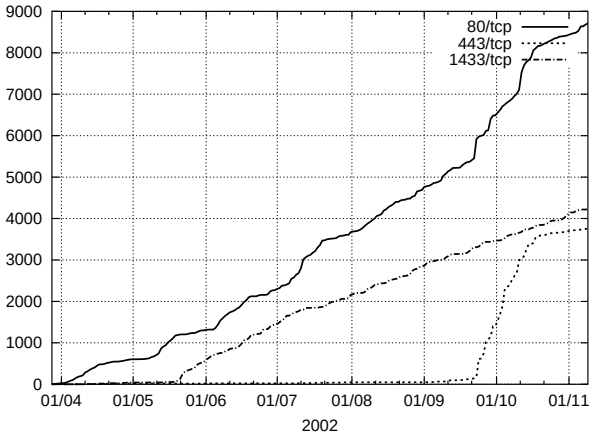

Figure 6: Worm related activity – cumulative

3.4 Intruders Profile

The intruder's profiles were very similar. Almost all of them, after obtaining privileged access, installed scan tools, exploits, massrooters, rootkits and IRC related programs. In some cases Denial of Service tools were also installed, as discussed above.

All the backdoors installed in the Honeynet made use of some cryptographic mechanism, preventing the capture of the traffic related to these sessions. Because of this, the observation of the intruders' sessions remained restricted to the data obtained through the modified shell.

Most of the intruders have launched the exploits from machines located outside Brazil, even in the cases where we identified the intruders as Brazilians.

Observing the IRC traffic it was possible to identify that some intruders were Brazilians, although the great majority was formed by Romanians.

Regarding the motives, we have determined that their intention was mostly the use of the machines as IRC bouncers, lauch points to other attacks and stepping-stones. In one case we also identified that the intruders were involved in credit card fraud.

In Fig. 7 we put together some statistics correlating countries to the sources of scans, exploits and backdoor access. The scans statistics were made considering only the ports associated with services that had received any exploit attempt.

To define to which country an IP belongs we considered to which country the IP block is assigned, not taking into account DNS lookups.

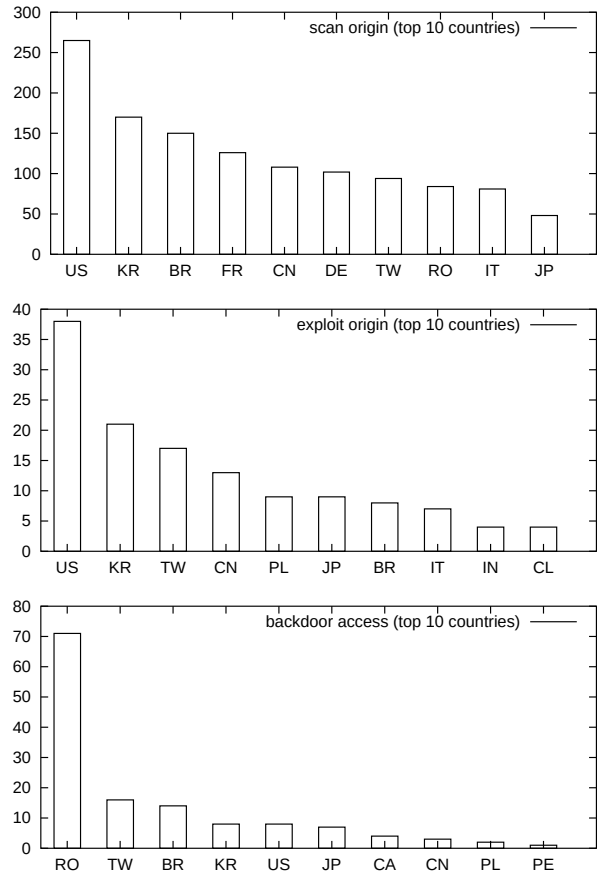

Figure 7: Intruders origin – top 10 countries

[9] http://www.cert.org/incident_notes/IN-2002-04.html

[10] http://www.cert.org/advisories/CA-2002-27.html

3.5 Defacements

The Honeynet has no Web server and was not our intention to attract defacers, but even so they came. They compromised a honeypot, which did not have Web services running, via an OpenSSH vulnerability. Once they gained privileged access they started the `httpd` daemon and made the "defacement" only to announce it to a mirror Web site.

4 Usefulness to CSIRTs

During the observation period of the Honeynet it has proved itself a valuable tool to a CSIRT. Some of the advantages of having such technology in place are discussed next.

4.1 Detection of Attacks

The NBSO constituency is the Brazilian Internet connected networks. At first we thought that having only one Honeynet, placed in a small part of the Brazilian address space, would show only limited data about what is really happening. But we were positively surprised when we realized that the data captured in the Honeynet reflected the activity we were seeing through incident reports coming from various parts of the country.

We have followed some attacks in the Honeynet that were very specific and with some caractheristics that made clear they were peculiar to the Brazilian address space. In some cases this data was correlated to incident reports in order to have a better understanding of the overall picture.

4.2 Source of Training Material

Once the Honeynet is up and running, and the methodologies for deploying the honeypots and preserving evidences is in place, it becomes a great source of material to learn how to perform artifact analysis and forensics.

This is very interesting as a way to maintain the team in close contact with material coming from intrusions, and ready to act in case of a real compromise inside the constituency.

It is also a great tool to train new incident handlers to deal with log analysis and intrusions, and to get them experienced with the process of preserving evidence.

4.3 Helping the Community

Now and then it is common to see a scan, or even a compromise, coming from an IP that looks like a compromised machine. Every time we identify any malicious traffic as coming from a compromised machine we send an email to its Whois contact advising about the problem.

This email is sent by NBSO exactly in the same way we send emails advising about activities observed in networks that do not want their names to be disclosed. It contains sanitized logs and some advice on how to check if the machine is compromised and how to recover from an intrusion.

It is very common to receive positive followups from these emails. In most cases the administrators were not aware of the intrusion and our email gave them the first warning to look at the network and find out about the problem.

Some new rootkits were collected and provided to the authors of the open source tool `chkrootkit`[11] to update the tool. In this way people that make use of the tool can benefit from our findings.

5 Future Work

The use of encrypted sessions is largely disseminated through the blackhat community. This makes very important the development of new techniques to monitor their activities. We plan to work in the improvement of new tools to do keylogging at kernel level, or using system libraries.

We also plan to improve `sessionlimit` in order to achieve better performance in high bandwidth networks.

6 Conclusions

Implementing a Honeynet is a challenge, but all the work of developing data control mechanisms and data analysis tools is rewarded by the valuable information it provides.

[11] `http://www.chkrootkit.org/`

To a CSIRT it is very important to be in contact with new tools and techniques used by intruders. In some cases people report an incident but they do not have all the information about the attack, or they do not know how to get that information. Correlate incident notifications with data captured in the Honeynet can clarify some attacks or add more information to them.

The correlation between data from the Honeynet and from Incident Reports can also help to distinguish between real threats and false positives.

Acknowledgments

This project is sponsored by both the NIC BR Security Office (NBSO) and the National Institute for Space Research (INPE).

Several other people and institutions have helped to set up the project. We would like to give our special thanks to the Honeynet.BR Team[12] for their support and ideias, and to the Ministry of Science and Technology and The State of São Paulo Research Foundation (FAPESP) for their support and donations.

We also would like to thank Monica Rubly for revising this paper.

References

[1] The Honeynet Project, *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Addison-Wesley, 1st ed., August 2001. ISBN 0-201-74613-1.

[2] L. Spitzner, "Learning the Tools and the Tactics of the Enemy with Honeynets," in *Proceedings of the 12th Annual Computer Security Incident Handling Conference*, (Chicago, Illinois, USA), June 2000.

[3] D. Farmer and W. Venema, "Being Prepared for Intrusion," *Dr. Dobb's Journal*, vol. 26, April 2001.

[4] D. Hartmeier, "Design and Performance of the OpenBSD Stateful Packet Filter (pf)," in *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference (FREENIX '02)*, (Monterey, California, USA), June 2002.

[5] A. B. Filho, A. S. M. S. Amaral, A. Montes, C. Hoepers, K. Steding-Jessen, L. H. Franco, and M. H. P. C. Chaves, "Honeynet.BR: Desenvolvimento e Implantação de um Sistema para Avaliação de Atividades Hostis na Internet Brasileira," in *Anais do IV Simpósio sobre Segurança em Informática (SSI'2002)*, (São José dos Campos, SP), pp. 19–25, Novembro 2002. `http://www.lac.inpe.br/security/honeynet/papers/hnbr-ssi2002.pdf`.

[6] W. R. Cheswick, "An Evening with Berferd in Which a Cracker is Lured, Endured, and Studied," in *Proceedings of the Winter 1992 USENIX Conference*, (San Francisco, California, USA), pp. 163–174, 1992.

[7] S. M. Bellovin, "There Be Dragons," in *Proceedings of the Third Usenix Security Symposium*, 1992.

[8] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*. Garden City, NY: Doubleday, 1989. ISBN 0-385-24946-2.

[9] C. Stoll, "Stalking the Wily Hacker," *Communications of the ACM*, vol. 31, pp. 484–497, May 1988.

[12] `http://www.lac.inpe.br/security/honeynet/`

A National Early Warning Capability Based on a Network of Distributed Honeypots

Cristine Hoepers, Klaus Steding-Jessen, Luiz E. R. Cordeiro, Marcelo H. P. C. Chaves
NIC BR Security Office – NBSO
Brazilian Computer Emergency Response Team
{cristine,jessen,cordeiro,mhp}@nic.br

Abstract

We present here the work developed by NBSO/Brazilian CERT, in the "Brazilian Honeypots Alliance – Distributed Honeypots Project", to centralize the data gathered in several honeypots and to process this data to be used for early warning and incident response. We shortly describe how the honeypots are deployed and how the data is centralized, then focus on how the data is being used by NBSO to generate statistics and to notify networks potentially compromised or infected.

1 The Distributed Honeypots Project

Honeypots have been pointed as tools that can play a significant role in gaining early warning of attacks [1], although most of the effort made by the community has focused on using honeypots and honeynets to observe the motives and methods of the attackers [2,3,4].

With the objective of using low-interaction honeypots [2] as tools for early warning and trend analysis, the NBSO/Brazilian CERT and the CenPRA Research Center deployed the "Brazilian Honeypots Alliance – Distributed Honeypots Project"[1]. The main objective of this project is to increase the capacity of early warning, event correlation and trend analysis in the Brazilian Internet space.

The Brazilian Honeypots Alliance is coordinated by NBSO and CenPRA, and has around 30 partner institutions from academia, government and private sector, as we can see in Figure 1 and Table 1. Each institution is responsible for the maintenance of at least one honeypot and for providing a network range[2] to be used by the honeypot.

All honeypots run Honeyd[3] on the OpenBSD operating system, and are configured according to the project's standards. The goals of these standards are to guarantee that all honeypots are hardened to minimize the chances of a compromise, and to facilitate the maintenance. These honeypots generate Honeyd and `pf` [5] firewall logs. The firewall is configured to log the payload of the packets. These logs are then collected to a central location, as described in section 2.

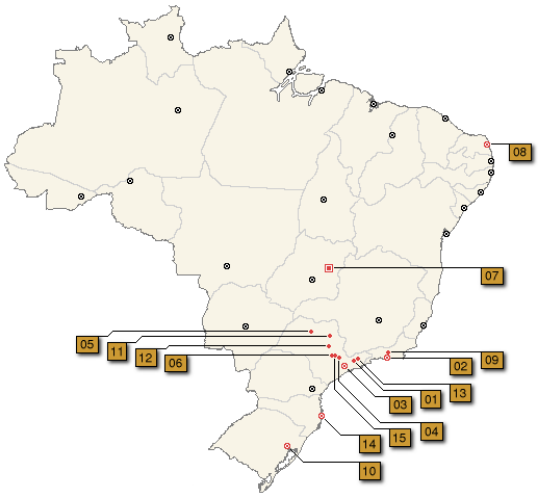

Figure 1: Brazilian map showing the cities where there are honeypots deployed, as of April 2005. See details in Table 1.

[1] Project site: `http://www.honeypots-alliance.org.br/`
[2] Usually a `/24` per institution, but this can vary.
[3] `http://www.honeyd.org/`

#	City	Alliance Members
01	S. José dos Campos	INPE, ITA
02	Rio de Janeiro	CBPF, Fiocruz, PUC-RIO, RedeRio, UFRJ
03	São Paulo	ANSP, Diveo, Durand, NBSO/Brazilian CERT, UNESP, USP
04	Campinas	CenPRA, HP Brasil, UNICAMP
05	S. José do Rio Preto	UNESP
06	Piracicaba	USP
07	Brasília	Brasil Telecom, TCU, UNB LabRedes
08	Natal	UFRN
09	Petrópolis	LNCC
10	Porto Alegre	CERT-RS
11	Ribeirão Preto	USP
12	São Carlos	USP
13	Taubaté	UNITAU
14	Florianópolis	UFSC DAS
15	Americana	VIVAX AMR

Table 1: Members of the Brazilian Honeypots Alliance grouped by city, as of April 2005.

2 Data Collection and Status Check

NBSO has developed several tools that collect the data gathered by each honeypot. The honeypots maintain their logs locally for 24 hours. Every day the central server connects to each honeypot and transfers the data using `ssh`.

In order to ensure that all the honeypots are operating correctly, a remote status checking mechanism was developed. By running this several times a day from a centralized location, it is possible to quickly determine that all honeypots are running according to the project's standards.

The tests are performed using a ssh connection to each honeypot. Some of the items checked are listed below:

- connectivity;
- processes running;
- amount of free disk space;
- clock synchronization, using NTP;
- uptime.

3 Data Analysis

Every day, project members receive a sanitized summary of the activities observed in all honeypots. This summary is distributed only to the members, through an encrypted mailing list, and includes information about ports, protocols, IPs and Source OSs that generated the major part of the activities observed.

Besides these internal summaries, we also maintain public daily statistics based on network flow data directed to the honeypots of the Brazilian Honeypots Alliance.

We maintain statistics for the categories below:

- Destination TCP Ports (bytes/s)
- Destination TCP Ports (packets/s)
- Destination UDP Ports (bytes/s)
- Destination UDP Ports (packets/s)
- Source Country Codes (bytes/s)
- Source Country Codes (packets/s)
- Source Operating Systems (bytes/s)
- Source Operating Systems (packets/s)

To explain how the graphics are generated, we will use the "Destination TCP Ports" category as an example. We extract all the destination TCP ports seen in the network flow data and compute the quantity of bytes or packets received by each port. The top 10 TCP ports related to the highest quantities of bytes or packets are selected to be displayed on the graphic. The quantities of bytes or packets of the remaining ports are summed in a data set called "Others". Then, this information is used to plot the packet or byte rate per second for the top 10 TCP ports and the "Others" data set. The other categories follow the same criteria.

Each image presented in the statistics contains a stack area graphic, which means that each data set of the graphic is stacked on top of the previous one.

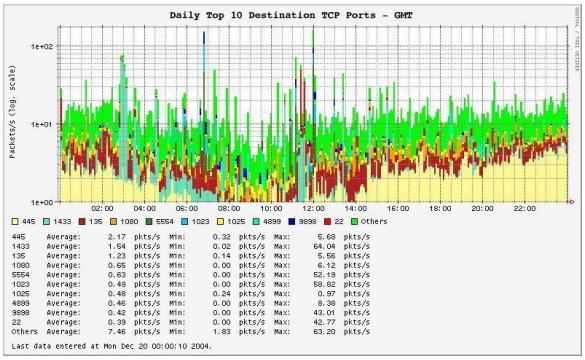

Figure 2: Top TCP ports (packets).

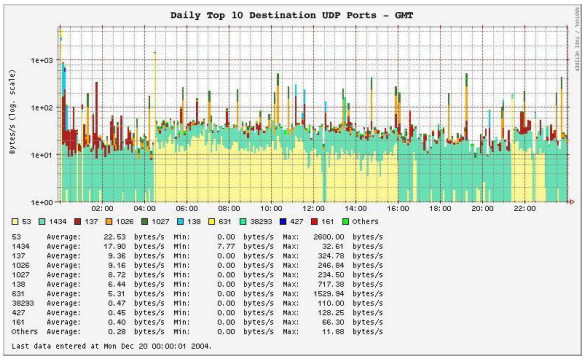

Figure 3: Top UDP ports (bytes).

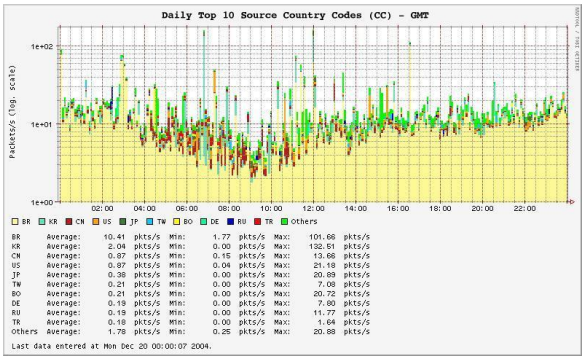

Figure 4: Top Country Codes (packets).

Some sample graphics of data collected in December 19th, 2004 are presented in Figures 2, 3, 4, and 5. Figure 2 shows the top 10 TCP ports that received the highest packet quantities. Figure 3 shows the top 10 UDP ports that received the highest byte quantities. Figure 4 shows the top 10 countries related to the majority of source network flows directed

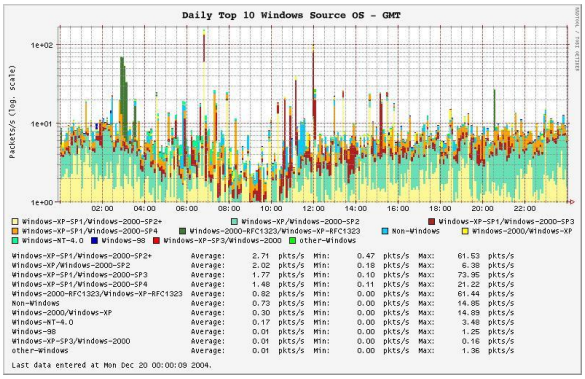

Figure 5: Top Windows Source Operating Systems (packets).

to the honeypots. The country code[4] values are obtained from RIR[5] statistics files. And, finally, Figure 5 shows the top 10 Windows Operating Systems related to the majority of source network flows directed to the honeypots. Source operating system guessing is performed by passive fingerprinting. Non-Windows Operating Systems statistics are also generated and available in the project's web site.

4 Use for Incident Response

All data collected is analyzed in order to identify signatures of well known malicious activities, for example: bots, worms and scans for ports known to run vulnerable services. Once these kind of activity is identified, we separate this data in 2 categories: Brazilian IP addresses and foreign IP addresses.

A portion of the sanitized data related to malicious activities coming from foreign IP addresses is donated to The Team Cymru[6], so the networks that use the data provided by their projects can benefit from the traffic seen in our honeypots.

NBSO processes all data related to malicious activities coming from Brazilian IP addresses and analyse the packets and their payloads looking for signatures of attacks.

We then find the contacts and/or CSIRTs for all the IPs and send to each one an email with the sani-

[4] ISO 3166 2-letter code.

[5] AfriNIC, APNIC, ARIN, LACNIC, and Ripe NCC.

[6] `http://www.cymru.com/`

tized logs and guidelines about how to deal with that information. The guidelines are usually technical tips to recover from a compromise or a worm infection.

Some categories of malicious activities identified are:

- generic scans
  - 21/TCP, 22/TCP, 111/TCP, etc
- worms/viruses
  - blaster, codered, dabber, mydoom, nimda, slammer, etc
- bots
- spam activity
  - open proxy scans
  - pop-up spam

Acknowledgments

This project is sponsored by both the NBSO/Brazilian CERT and the CenPRA Research Center.

Several other people and institutions have helped to set up the project. We would like to give our special thanks to all the Honeypots Alliance members[7], who maintain their own honeypots and allow us to collect data in their networks. We would also like to thank the Honeynet.BR Team[8] for their support and ideas.

References

[1] H. Lipson, "Tracking and Tracing Cyber-Attacks: Technical Challenges and Global Policy Issues," Tech. Rep. CMU/SEI-2002-SR-009, CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, November 2002.

[2] The Honeynet Project, *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Addison-Wesley, 1st ed., August 2001. ISBN 0-201-74613-1.

[3] L. Spitzner, "Learning the Tools and the Tactics of the Enemy with Honeynets," in *Proceedings of the 12th Annual Computer Security Incident Handling Conference*, (Chicago, Illinois, USA), June 2000.

[4] C. Hoepers, K. Steding-Jessen, and A. Montes, "Honeynets Applied to the CSIRT Scenario," in *Proceedings of the 15th Annual Computer Security Incident Handling Conference*, (Ottawa, Canada), June 2003.

[5] D. Hartmeier, "Design and Performance of the OpenBSD Stateful Packet Filter (pf)," in *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference (FREENIX '02)*, (Monterey, California, USA), June 2002.

[7] http://www.honeypots-alliance.org.br/

[8] http://www.honeynet.org.br/

ÍNDICE

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.