UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JOÃO MARCELO CERON

Arquitetura Distribuída e Automatizada para Mitigação de *Botnet* Baseada em Análise Dinâmica de *Malwares*

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dra. Liane Margarida R. Tarouco
Orientadora

Porto Alegre, Agosto de 2010

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Ceron, João Marcelo

Arquitetura Distribuída e Automatizada para Mitigação de *Botnet* Baseada em Análise Dinâmica de *Malwares* / João Marcelo Ceron. – Porto Alegre: PPGC da UFRGS, 2010.

76 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2010. Orientadora: Liane Margarida R. Tarouco.

I. Margarida R. Tarouco, Liane. II. Título.

AGRADECIMENTOS

Agradeço à minha família pelo apoio incondicional em todos os momentos difíceis que fizeram parte dessa jornada;

Agradeço a todas as pessoas que passaram pela minha vida e me auxiliaram no processo de crescimento pessoal e profissional. Em especial, gostaria de agradecer meus colegas do PoP-RS que foram muito importante numa fase decisiva da minha vida. Ao meu grande amigo e colega de trabalho, Émerson Salvadore Virti, pelas inúmeras parcerias e por todo incentivo;

Aos professores e mestres que foram fundamentais para a conclusão deste trabalho: a professora Liane Tarouco, orientadora deste trabalho; o professor Lisandro Zambenedetti Granville por valiosas lições durante o mestrado;

Aos meus colegas do Centro de Processamento de Dados da UFRGS por todo apoio e incentivo prestado;

Aos colegas do Grupo de Redes de Computadores da Universidade Federal do Rio Grande do Sul, por todo o companheirismo e apoio que manifestaram;

À Universidade Federal do Rio Grande do Sul e seu corpo docente, por terem tornado possível a realização deste e de outros trabalhos;

À todos que, direta ou indiretamente, me ajudaram a chegar até aqui.

“Work while you have the light. You are responsible for the talent that has been entrusted to you.“

— HENRI-FRÉDÉRIC AMIEL

SUMÁRIO

RESUMO

Atualmente, uma das mais sérias ameaças a segurança da Internet são as *botnets*. As *botnets* - rede de máquinas comprometidas e controladas remotamente por um atacante - caracterizam-se por serem muito dinâmicas. Frequentemente novas características são incorporadas as redes dificultando que ferramentas tradicionais tal como sistemas de antivírus e IDS sejam efetivas. Diante disso, faz-se necessário desenvolver novos mecanismos que possam complementar as atuais técnicas de defesa. Esta dissertação de mestrado apresenta uma proposta de arquitetura para uma ferramenta de mitigação e detecção de *botnets* baseada em assinatura de rede de máquinas comprometidas por *bots*. Essa arquitetura automatiza o processo de geração de assinaturas compilando informações de analisadores de *malwares* gratuitamente disponibilizados na Web. Além disso, utilizou-se de monitoração de fluxos através da solução Netflow para identificar o comportamento de rede similar aos mapeados em arquivos maliciosos analisados. Esse comportamento mapeado sinaliza uma possível infecção de máquinas na rede monitorada. Essa identificação dispara eventos na ferramenta proposta que auxiliará o gerente a mitigar a máquina comprometida. Por fim, avaliou-se a solução proposta no contexto de uma grande rede acadêmica: da própria Universidade Federal do Rio Grande do Sul (UFRGS). Os resultados alcançados por essa solução permitiram concluir que 1,5% dos controladores ficam por um longo período (52 dias) realizando atividades maliciosas e, também, observou-se um pequeno grupo de controladores responsáveis pela administração de uma grande quantidade de máquinas.

Palavras-chave: Botnet, bot, análise de malware.

An Automated and Distributed Architecture for Botnet Mitigation Based in Dynamic Malware Analysis

ABSTRACT

Currently, botnets are one of the most serious threats of Internet security. The botnets - network of compromissed machines remotely controlled by an attacker - are being very dynamic threats. Often new features are incorporated into this malicious networks making hard for traditional tools, such as antivirus and IDS, to be effective. Therefore, it is necessary to develop new mechanisms that can complement the current defense techniques. This dissertation presents an architecture for a tool for botnet mitigation and detection. The tool is based in network signature obtained from bot compromissed machine's. This architecture automates the process of signature generation compiling information from online malwares analyze tools. Furthermore, flows monitoring tools was used to identify similar behavior to those mapped in malware (bot) analyzed by the system. This mapped behavior in flows indicates possible compromissed machines, with this, the system triggers events to help the security manager to mitigate the compromissed machines. Finally, the proposed solution was evaluated in a academic network: in the Federal University of Rio Grande do Sul. The results achieved by this solution helped to observe that more than 1.5% of the botnet controllers's remain active for a long period of time (52 days) performing malicious activities. Also, was observed a small group of controllers responsible for the adminstration of a large number of compromissed machines.

Keywords: botnet, bot, malware analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AS	Autonomous System
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
DNS	Domain Name System
DoS	Denial-of-Service
DDoS	Distributed Denial-of-Service
GPL	General Public Licence
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
HTTPS	Secure Hypertext Markup Language
IRC	Internet Relay Chat
IDS	Intrusion Detection System
IP	Internet Protocol
IPFIX	IP Flow Information Export
IPS	Intrusion Prevention System
ICMP	Internet Control Message Protocol
MD5	Message-Digest algorithm 5
NAT	Network Address Translation
POP3S	Secure Post Office Protocol 3
URL	Uniform Resource Locator
RNP	Rede Nacional de Ensino Pesquisa
SSH	Secure Shell
SNMP	Simple Network Management Protocol
UDP	User Datagram Protocol
TOS	Type of Service
TCP	Transmission Control Protocol

1 INTRODUÇÃO

O contínuo crescimento e popularização da Internet está sendo acompanhado pelo aumento de novas ameaças à segurança dos sistemas computacionais. Atualmente, uma das mais sérias ameaças a segurança da Internet são as *botnets*. Uma *botnet* é definida como um conjunto de máquinas comprometidas (*bots*) que podem ser controladas remotamente por uma unidade de controle denominada de *botmaster* (GRIZZARD et al., 2007).

As ações desempenhadas pelas *botnets*, em geral, são danosas à estrutura da Internet. As *botnets* são responsáveis pelas mais diversas atividades maliciosas, tais como ataques de negação de serviço distribuído (DDoS), envio de *spams*, distribuição de *malwares* e roubo de informações. Estima-se, nos dias presentes, que exista um número superior a nove milhões de máquinas infectadas (*bots*) na Internet (KREIBICH et al., 2009). Esse número equivale a aproximadamente 40% de todas as máquinas conectadas na Internet Brasileira (VIEIRA, 2008). Um relatório recente (EHRLICH et al., 2010) apontou uma única *botnet* (Storm Worm) como responsável por 25% de todo *spam* propagado na Internet. Diante dessa disseminação, todo um modelo de negócio foi estabelecido sobre as *botnets*. Na ilegalidade, já é possível alugar *botnets*, disponibilizando a terceiros o total controle dos *bots* para as mais diversas atividades, como ataques de negação de serviço e *click fraud* (DASWANI; STOPPELMAN, 2007).

O aumento das ameaças de segurança deve-se primariamente a mudanças da motivação básica por trás dos ataques. No passado, a maioria dos ataques era realizada por atacantes inexperientes buscando o próprio reconhecimento. Recentemente, entretanto, centros de pesquisas constatam um grande número de atividades com objetivos financeiros, incluindo roubo de identidade e extorsões comandadas por grupos organizados (LAUINGER et al., 2010). A motivação econômica impulsiona o desenvolvimento de novas funcionalidades desempenhadas pelas *botnets* tornando, assim, a tarefa de combater essa ameaça ainda mais complexa.

Neste contexto, a comunidade científica de segurança da informação vem estudando meios de prevenção e detecção de novas ameaças de segurança. A literatura apresenta diferentes métodos (GU et al., 2008) (GOEBEL; HOLZ, 2007) (GU et al., 2007) para detectar e mitigar *botnets*. No entanto, a maioria das metodologias e técnicas estão relacionadas a tipos específicos de *botnets*, como por exemplo, *botnets* que utilizam estrutura centralizada e baseadas em certos protocolos de comunicação.

Os principais métodos de detecção de *botnets* baseados em rede podem ser classificados em duas abordagens: correlação horizontal e correlação vertical (WURZINGER et al., 2009). Na correlação vertical, busca-se identificar o comportamento específico de um *bot* analisando padrões comportamentais que caracterizam uma *botnet*. Por exemplo, uma solução que identifique assinaturas (padrões) em datagramas de conexão de uma *botnet* específica. Outra forma de implementar a correlação vertical envolve a avaliação de

malwares, capturados com auxílio de *honeypots*, a fim de identificar nodos controladores (BARFORD; BLODGETT, 2007) (BACHER et al., 2005). A desvantagem dessa abordagem, entretanto, é a necessidade de um conhecimento prévio das características de cada *botnet*, para que só assim padrões sejam identificados.

Na correlação horizontal, diferentemente da abordagem já descrita, busca-se correlacionar o padrão comportamental de duas ou mais máquinas provavelmente infectadas. Uma solução possível é onde fluxos de rede - com características semelhantes - são agrupados e analisados segundo uma heurística que identifica indícios de comprometimento por *bots* (EHRLICH et al., 2010). Essa abordagem, contudo, necessita de dois ou mais *bots* na rede analisada para mapear similaridades de comportamento.

O presente trabalho propõe-se a especificar e implementar uma arquitetura para detecção e mitigação de *botnets* utilizando a abordagem de correlação vertical explicada acima. A metodologia implementada busca identificar *botnets* tanto de estruturas centralizadas quanto distribuídas, tornando-se mais genérica que a maioria das soluções atuais.

Considerando tal cenário, este trabalho apresenta uma arquitetura para, automaticamente, gerar assinaturas de *botnets* tendo como base a análise de arquivos maliciosos. Para isso, a arquitetura emprega algumas estratégias para gerar assinaturas por meio de análise dinâmica de arquivos maliciosos - coletados por uma *honeynet* - e analisados em ferramentas *on-line* conhecidas por *sandboxes*. Após a identificação das máquinas comprometidas, a arquitetura especifica o emprego de mecanismos para efetivamente isolar as máquinas comprometidas. Esse procedimento é fundamental para a segurança da rede, pois não basta que os *bots* não recebam mais instruções do *botmaster*, mas também é necessário informar para o responsável pela máquina comprometida que o seu sistema está infectado.

Como contribuição indireta, a arquitetura implementa uma base de dados com informações de todos os *malwares* analisados, tais como assinaturas de *botnets*; estatísticas de detecção; *botnets* mais ativas; e *bots* detectados na rede local. Todos esses dados são disponibilizados ao administrador de rede, para que dessa forma, sejam obtidas conclusões mais precisas em relação as ameaças de segurança na rede local. Além da especificação e implementação da arquitetura de mitigação e detecção de *botnets*, o presente trabalho discute detalhes da implementação e emprego da solução no contexto da Universidade Federal do Rio Grande do Sul, considerando um período de 8 meses.

Esta dissertação está organizada como segue. Os trabalhos relacionados discutindo as diferentes técnicas de detecção e mitigação de *botnets* são apresentados no capítulo 2. Os conceitos relativos ao processo de identificação de *botnets* são discutidos no capítulo 3. No capítulo 4, é apresentada a solução proposta por este trabalho. Os detalhes e particularidades da implementação são descritos no capítulo 5. A validação da arquitetura mostrando um estudo de caso realizado na rede acadêmica da Universidade Federal do Rio Grande do Sul é apresentada no capítulo 6. Finalmente, esta dissertação é encerrada no capítulo 7, onde são apresentadas as conclusões e os trabalhos futuros.

2 ESTADO DA ARTE

Os *malwares*, em particular os *bots*, representam uma significante ameaça a segurança da Internet atual. Os *bots* são responsáveis por uma série de atividades maliciosas, geralmente também observadas em *worms*, *rootkits*, e cavalos de tróia. Infelizmente, a complexidade dos *malwares* mais recentes requer que novas ferramentas e abordagens sejam avaliadas com o objetivo de desenvolver soluções adequadas de defesa.

As atuais ferramentas de segurança possuem um papel fundamental para defender o usuário de um computador das ameaças de segurança da rede. A defesa mais importante e popular contra códigos maliciosos são os próprios sistemas de antivírus. Os antivírus, tipicamente, têm o seu funcionamento estabelecido numa base de dados de assinaturas previamente conhecida, a qual é determinada segundo características do conteúdo de arquivos maliciosos. Logo, a eficiência dos sistemas de antivírus está diretamente ligada a assinaturas de *malwares* presente na base de dados do sistema. Entretanto, desenvolver novas assinaturas - especialmente para *malwares* mais recentes que empregam técnicas sofisticadas - é uma tarefa complexa e suscetível a imprecisão.

Para complementar as ferramentas que atuam no contexto da máquina do usuário - tais como os sistemas de antivírus - é desejável a utilização de outras metodologias de defesa que atuam em contextos diferenciados. Ferramentas baseadas em análise de tráfego de rede, por exemplo, podem ser utilizadas para suprir as deficiências de outros métodos e identificar máquinas comprometidas na rede. Os IDS's (*Intrusion Detection Systems*) e IPS's (*Intrusion Prevention Systems*) são ferramentas que caracterizam-se por monitorar o tráfego de rede em busca de uma atividade tipicamente maliciosa. A capacidade de inspecionar datagramas de um enlace permite que os IDS's utilizem assinaturas comportamentais (como número de datagramas de uma conexão) e também assinaturas baseadas no conteúdo de datagramas (palavras chaves). A efetividade dos sistemas de detecção de intrusão reside no mecanismo de identificação de anomalias na rede, como por exemplo, um comportamento de tráfego fora do padrão usual.

No entanto, apenas a utilização de sistemas tradicionais como antivírus e IDS são insuficientes para deter os danos causados pelas novas ameaças a segurança de redes. As novas ameaças, tais como as *botnets*, caracterizam-se por utilizar arquiteturas complexas e implementar funcionalidades que conseguem subverter os atuais mecanismos de segurança. Pela própria natureza intrínseca das *botnets*, o processo de mitigação é uma tarefa árdua uma vez que as máquinas infectadas podem estar dispersas em domínios administrativos diferentes. Além da necessidade de coordenação dos esforços, as *botnets* em si são bastante dinâmicas: constantemente novas funcionalidades são incorporadas as redes maliciosas o que torna o processo de detecção ainda mais sofisticado.

Neste contexto, observa-se uma intensificação nos esforços para desenvolver técni-

cas ou mecanismos que possam identificar uma *botnet*, ou ao menos diminuir os danos causados por uma rede de máquinas infectadas. A literatura apresenta diferentes técnicas para entender a estrutura de uma *botnet* e formular uma estratégia de controle. Segundo o estudo realizado por Peter Wurzinger e outros (WURZINGER et al., 2009), a tarefa de detecção de *botnets* pode ser classificada em duas grandes abordagens principais: correlação horizontal e correlação vertical. Na sequência, serão discutidos os trabalhos relacionados com o tema detecção e mitigação de *botnets* tendo em vista a classificação acima citada.

2.1 Correlação Horizontal

A correlação horizontal busca relacionar padrões de comunicação entre máquinas que podem revelar um comportamento típico de *bots*. Máquinas integrantes de uma *botnet* tendem a atuar de forma semelhante e sincronizada, realizando as instruções solicitadas pelo controlador da rede, como por exemplo, envio de *spam* ou varredura por vulnerabilidades. Dessa forma, a análise de padrões de tráfego torna possível a identificação de similaridades comportamentais de duas ou mais máquinas comprometidas numa mesma rede monitorada.

A identificação de similaridades de tráfego e padrões de comunicação é tipicamente realizada por ferramentas de monitoração de fluxos e anomalias de rede. A monitoração de fluxos torna possível observar um conjunto de características relativas a uma comunicação entre pares de máquinas. Os aspectos monitorados podem determinar a existência de uma anomalia ou uma exceção no tráfego na rede, tal como um alto número de datagramas originado por um serviço suspeito. Em complementação, podem ser desenvolvidas heurísticas ou algoritmos mais complexos para analisar características de fluxos e agrupar as mesmas segundo similaridades, caracterizando possíveis ações maliciosas.

Paralelamente a análise de fluxos, outras ferramentas de detecção de anomalias podem ser empregadas na tarefa de identificar tráfego suspeito. Tais ferramentas, como os sistemas de detecção de intrusão, podem correlacionar eventos na rede e os classificar como um comportamento típico de uma máquina comprometida.

É possível observar um conjunto de trabalhos que implementam técnicas baseadas na abordagem de correlação horizontal. Por exemplo, na ferramenta `Botsniffer` (GU; ZHANG; LEE, 2008), os autores especificam uma metodologia para identificar controladores de *botnets* baseada na correlação de padrões de comunicação (similaridade de tráfego). Para isso, a ferramenta emprega um algoritmo de correlação que atua em *botnets* de estrutura centralizada e baseadas em protocolos específicos (IRC e HTTP). Nesse algoritmo, são observados *bots* pertencentes a mesma *botnet* que demonstram uma sincronização muito forte em relação às suas mensagens de respostas na rede. Através de diferentes análises, a ferramenta propõe uma correlação espaço-temporal do tráfego e identifica nodos de *botnet* com baixa taxa de falsos positivos. O Botsniffer mostra-se relativamente eficaz. No entanto, é apenas aplicável a um conjunto restrito de *botnets* baseado no protocolo IRC e HTTP.

Em outro trabalho (KARASARIDIS; REXROAD; HOEFLIN, 2007) os autores apresentam um algoritmo passivo para detecção de anomalias de rede tendo como escopo *botnets* baseados no protocolo IRC. O algoritmo apresentado atua em duas etapas: a) agregação de fluxos nos quais são observados eventos relativos a atividades suspeitas, como envio de *spams*, varreduras, DoS. Nesta etapa, adicionalmente, é feito um mapeamento nos fluxos agregados enumerando dados tais como IP de origem e IP destino, bem como porta origem e porta destino nas comunicações. Logo após, essas informações são

submetidas para uma heurística que sumariza dados de possíveis comunicações entre máquinas de uma *botnet*; Na segunda etapa (b) do algoritmo, as possíveis comunicações de *botnets* - identificadas na etapa anterior - passam por uma análise temporal. Essa análise busca identificar os padrões de comunicação em fluxos coletados ou já armazenados no sistema de coleta da própria ferramenta. Como resultado, o algoritmo identifica máquinas comprometidas e seus respectivos controladores de *botnets*.

Na arquitetura de detecção `BotMiner` (GU et al., 2008), é apresentado um algoritmo de detecção de *botnets* mais robusto, capaz de identificar características de *botnets* independente do protocolo utilizado pela rede maliciosa. Essa arquitetura define um agrupamento de comunicações de tráfego em dois níveis: a) similaridades de tráfego de comunicação; b) tráfego possivelmente malicioso. Logo após, é realizado uma correlação entre os agrupamentos para identificar similaridades, tendo em vista comunicação entre pares de máquinas. Dessa forma, analisando apenas os fluxos de rede - certas características do cabeçalho dos datagramas - é possível identificar máquinas comprometidas. Devido as suas características, o BotMiner pode atuar no combate de *botnets* independentes de estrutura e independente de protocolos, incluindo redes baseadas em protocolos HTTP, IRC e redes P2P.

Conforme demonstrado, em alguns trabalhos a análise de tráfego e detecção de anomalias (como alto volume de tráfego, latência, e similaridades) podem ser úteis para identificar a presença de certas *botnets*. Entretanto, para que as anomalias sejam observadas é necessário que a *botnet* esteja ativa e realizando atividade com os seus *bots*. Logo, os métodos de detecção baseados em anomalias apresentam limitações para redes maliciosas em estado ocioso, ou aguardando instruções para realizar ações. Diante disso, os autores Binkley e Singh apresentaram uma solução (BINKLEY; SINGH, 2006) que visa resolver tal deficiência. No referido trabalho, é apresentado uma heurística que combina estatísticas temporais de mensagens do protocolo IRC com comunicações suspeitas realizadas em pares. Com isso, o sistema consegue identificar *botnets* que operam com o protocolo IRC com baixa taxa de falso positivo. Muito embora seja uma solução eficiente, esse trabalho não é efetivo para *botnets* que utilizam mensagens cifradas.

Além da monitoração de fluxos, outras técnicas já foram estudadas a fim de mapear o comportamento anômalo de máquinas infectadas. A monitoração de requisições a serviços de resolução de nomes (DNS) tem apresentado bons resultados (CHOI et al., 2007). As técnicas de detecção baseadas em DNS, predominantemente, baseiam-se na observação de dados específicos de resolução de nomes solicitados por uma *botnet*. Como detalhado no capítulo 3, um *bot* típico inicia uma conexão com um *servidor de controle e comando* para receber instruções. Para acessar o controlador, o *bot* faz uma requisição a um serviço de nomes a fim de localizar o respectivo endereço IP onde o mesmo deve conectar. Logo, torna-se possível observar requisições de DNS para detectar possíveis requisições originadas por *bots*. No trabalho apresentado por Dagon (DAGON, 2005), o autor propõe um mecanismo para identificar controladores de *botnets* através da observação de certas características de consulta aos servidores de nomes, tais como a) domínios com número anormal de acesso; b) domínio com alta concentração de consultas; e c) concentração de consultas segundo análise temporal. A proposta de Dagon, entretanto, demonstrou gerar um alto número de falsos positivos, além de ser facilmente evitada através da utilização de falsas consultas a domínios.

Em 2007, foi proposto um mecanismos de detecção de *botnets* através da monitoração de consultas coordenadas aos servidores de DNS (CHOI et al., 2007). As máquinas que consultam simultaneamente certos registros de nomes foram avaliadas segundo um

algoritmo de classificação desenvolvido pelos autores. A metodologia apresenta bons resultados, uma vez que as consultas ao servidor de DNS aparecem em várias etapas do ciclo de vida de uma *botnet*. Os autores também desenvolveram um mecanismos que permite identificar migração de controladores de *botnets* centralizados. Essa solução é mais robusta que outras metodologias baseadas em análise de tráfego DNS e pode detectar *botnets* centralizadas que utilizem canais criptografados, uma vez que são utilizas informações do cabeçalho IP, e não do conteúdo do datagrama. A principal limitação dessa metodologia, entretanto, é o alto tempo de processamento requerido para analisar todas as requisições do servidor DNS.

Diferentemente das técnicas apresentadas, o trabalho descrito por Strayer e outros autores (STRAYER et al., 2007) propõe uma solução baseada em análise de tráfego de tráfego de rede utilizando técnicas de inteligência artificial. Através de uma classificação realizada em diversas etapas, os autores demonstram que é possível identificar comunicações entre controladores extraindo características de fluxos de rede. De forma simplificada, a solução proposta faz a distinção de fluxos IRC e após identifica tráfego de controladores de *botnet* nos fluxos anteriormente selecionados. Embora a solução descrita seja efetiva para detectar certos tipos de *botnets* (IRC), o sistema apresentou um alto número de falsos positivos. Segundo os autores, para que a solução apresente uma maior precisão seria necessário analisar o conteúdo dos datagramas, e não apenas certas características do cabeçalho dos pacotes.

O conjunto de trabalhos e soluções descritos acima buscam identificar *botnets* de forma passiva, analisando anomalias e características do próprio tráfego de rede. De forma complementar, existem técnicas que atuam com diferentes abordagens, como descrito na sequência.

2.2 Correlação Vertical

As soluções que utilizam a abordagem de correlação vertical buscam primeiro identificar as ações de um *bot* para depois mapeá-las no tráfego da rede monitorada. As funcionalidades de um arquivo malicioso ou de uma máquina comprometida por um *bot* podem revelar o comportamento típico de uma *botnet* específica. Para que essa técnica seja bem sucedida, evidentemente, é necessário um conhecimento prévio das ações desempenhadas por cada *bot*. Para isso, existe um conjunto de ferramentas que podem ser utilizadas na tarefa de identificação de características das máquinas infectadas, assim como os analisadores de tráfego de rede (*sniffers*).

Os analisadores de tráfego podem revelar particularidades de uma *botnet*, como por exemplo, dados em datagramas com instruções maliciosas instanciadas pelo controlador da rede. Na ferramenta `Rishi` (GOEBEL; HOLZ, 2007), os autores identificam características de tráfego de máquinas comprometidas por *botnets* baseadas no protocolo IRC. Com base a essa análise, foi observado um padrão no conteúdo dos datagramas que pode ser mapeado na rede. As informações mapeadas correspondem a características de conexão como *nicknames* típicos de *botnet* (definido por padrões pré-construídos) e servidores IRC em portas incomuns. Através de um sistema probabilístico, a ferramenta se habilita a detectar *bots* utilizando canais de comunicação não usuais, localizando comunicações encapsuladas no protocolo HTTP. Uma vez que a detecção obtenha êxito, a ferramenta determina o endereço IP do controlador da *botnet*, bem como o canal de comunicação do servidor. Adicionalmente, podem ser observados os parâmetros de conexão de cada *bot*, como por exemplo, senha do canal IRC e sintaxe do *nickname*. Infelizmente essa ferra-

menta de detecção é eficiente somente para *botnets* centralizadas cujas características - padrão de *nicknames* - sejam pré-conhecidas.

Além dos analisadores de tráfego, os *honeypots* - detalhados na seção 3.3 - são ferramentas eficientes na tarefa de identificar características de máquinas infectadas. Algumas implementações de *honeypots* possuem um nível maior de interação com ataques e conseguem obter informações mais detalhadas sobre atividades maliciosas. Tais informações podem, por exemplo, refletir peculiaridades das *botnets* assim como dados para acesso ao canal de controle e comando, e características de datagramas de rede (tamanho, conteúdo específico, entre outros). No trabalho intitulado *Know your Enemy:Tracking Botnets* (BACHER et al., 2005), os autores apresentam uma técnica - baseada no conceito de *honeypots* - para detectar *botnets* utilizando análise de arquivos maliciosos. Para isso, foi desenvolvida uma ferramenta automatizada - *mwcollect2* (JOHN et al., 2009) - para coletar *malwares* propagados dinamicamente pela rede. Através dessa ferramenta foi possível analisar informações do canal de controle presente em arquivos binários (*bots*) e infiltrar-se na rede da *botnet*. Mesmo com uma análise simplificada dos *malwares*, a técnica demonstrou-se eficientes para um subconjunto de *botnets* baseadas no protocolo IRC. Como resultado, os autores apresentam uma análise das redes nas quais foi feita a infiltração, apresentando informações como tamanho da rede, tipos de comandos (instruções), e particularidades de cada máquina infectada. Apesar de atuar apenas em *botnets* baseadas no protocolo IRC, o trabalho acima descrito tem importância histórica, pois trata-se do primeiro relato de infiltração em *botnets*, apresentando os comandos e informações comportamentais dos ataques observados.

Holz (HOLZ et al., 2008a), por outro lado, apresenta uma técnica para mitigar a *botnet* `Storm Worm`. A Storm Worm é uma rede maliciosa distribuída baseada em protocolo P2P, cujo sua principal funcionalidade é o envio de *spams*. Trata-se de uma rede bastante popular e, segundo estatísticas (HOLZ et al., 2008a), já foi considerada responsável por mais de 10% de todo *spam* gerado na Internet. No trabalho, os autores analisaram uma grande quantidade de *spams* e foram aptos a capturar exemplares de arquivos maliciosos. Logo após, foi feito uma engenharia reversa nos arquivos maliciosos visando entender os vetores de propagação e os mecanismos de acesso a rede de controle da *botnet*. Como resultado, os autores decifraram o protocolo de comunicação e infiltraram-se na rede da própria *botnet*. Um estudo mais intensivo possibilitou identificar algumas limitações do protocolo de comunicação e controle da *botnet*, como por exemplo, a falta de autenticação para a troca de mensagens. De posse dessas informações, os autores conseguiram enviar mensagens na rede - utilizando o protocolo da própria *botnet* - solicitando o descadastramento dos integrantes (máquinas infectadas).

Uma solução semelhante a acima é aplicada para as *botnets* do tipo `Walowdac` (STOCK et al., 2009). A rede maliciosa Walowdac é referida como sucessora da Storm Worm. No entanto, a rede Walowdac utiliza uma estrutura mais descentralizada de propagação de mensagens (STOCK et al., 2009), bem como novos protocolos de comunicação. No referido trabalho foi demonstrada uma metodologia de infiltração na rede para a obtenção de dados das máquinas comprometias, como por exemplo, o endereço IP, pares (máquinas) de origem, sistema operacional, tempo na rede (*uptime*) e também atualização de mecanismos de controle. Esses dados auxiliaram no estudo da rede maliciosa e no processo de desenvolvimento de mecanismos de mitigação da rede *Walowdac*.

Embora os trabalhos acima sejam eficientes para obter mais informações sobre as *botnets*, os mesmos se aplicam apenas a um conjunto limitado de rede (Storm Worm e Walowdac), não sendo genéricos para os demais tipos de *botnets* disseminados na rede.

Adicionalmente, existem outras técnicas para obter informações de uma *botnet*, por exemplo, através de uma correlação de eventos entre ferramentas de monitoração (IDS, IPS). A ferramenta `Bothunter` (GU et al., 2007) caracteriza-se por utilizar diferentes sistemas de detecção de intrusão para fazer uma correlação entre eventos. O modelo de detecção é baseado numa série de eventos análogos, sendo eles: a) varredura de portas (*port scan*); b) sondagem a uma vulnerabilidade; c) *download* de arquivo binário; d) comunicação com um controlador de rede; e) e varredura de portas externas a rede. As diferentes combinações de alarmes ocorridas num período específico de tempo referente a um mesmo endereço IP satisfazem a condição para identificar uma máquina infectada.

Por fim, a solução que mais se assemelha com a solução proposta por este trabalho é apresentado por Peter Wurzinger e demais autores (WURZINGER et al., 2009). Na solução é descrita uma técnica de detecção de *botnets* baseada em assinatura (tráfego de rede) de máquinas infectadas. Para isso, um conjunto de *malwares* é analisado num ambiente controlado - num *sandbox* comercial - e suas atividades de rede monitoradas. A arquitetura proposta em nosso trabalho, entretanto, utiliza um conjunto de serviços *online* gratuitos - tal como analisadores de arquivos - para compor assinaturas de *botnets*. Além disso, a arquitetura é concebida de forma modular possibilitando com que novos componentes sejam implementados como, por exemplo, novas metodologias para análise de *malwares*. Adicionalmente, a solução proposta por esta dissertação permite localizar máquinas comprometidas num contexto temporal, localizando assinaturas no tráfego armazenado pela solução `Netflow` (Cisco, 2010). Desta forma, máquinas comprometidas mas não ativas no momento podem ser identificadas. Finalmente, a arquitetura proposta visa construir colaborativamente com novas técnicas para detectar *botnets*, com base nas análises dos *malwares* armazenados na base de dados do sistema.

Dada as soluções descritas acima é possível notar que a identificação de características de um *bot* é essencial no processo de detecção de redes maliciosas. A identificação permite que outras máquinas com as mesmas características sejam mapeadas e rapidamente mitigadas. Na sequência, as soluções descritas acima são comparadas considerando suas características comuns.

2.3 Comparativo das Técnicas de Detecção

Esta seção visa traçar com comparativo das principais técnicas de mitigação descritas nas duas seções anteriores. Para isso, os trabalhos descritos foram sumarizados na tabela 2.1 visando facilitar a comparação das características e limitações de cada metodologia.

Na tabela 2.1 é traçado um paralelo entre as soluções tendo em vista uma série de quesitos:

- a) *Bots Desconhecidos*: habilidade de detectar *bots* desconhecidos, ou seja, não previamente mapeados pela solução;
- b) *Independente de Protocolo e Estrutura*: aptidão para identificar *botnets* com diferentes estruturas (centralizada ou distribuída) e utilizando protocolos distintos (HTTP, P2P, IRC);
- c) *Bots Criptográficos*: identificação eficiente para redes que utilizem canal de comunicação criptografado.

A maioria das soluções que utilizam correlação horizontal conseguem identificar *bots* desconhecidos, que ainda não foram mapeados por um sistema de assinatura. Isso acon-

Solução Proposta	*Bots* Desconhecidos	Independente de Protocolo e Estrutura	*Bots* Criptográficos
(GU; ZHANG; LEE, 2008)	Sim	Não	Não
(KARASARIDIS; REXROAD; HOEFLIN, 2007)	Não	Não	Não
(GU et al., 2008)	Sim	Sim	Sim
(BINKLEY; SINGH, 2006)	Não	Não	Não
(DAGON, 2005)	Sim	Não	Sim
(CHOI et al., 2007)	Sim	Sim	Sim
(STRAYER et al., 2007)	Não	Não	Não
(GOEBEL; HOLZ, 2007)	Não	Não	Não
(BACHER et al., 2005)	Não	Não	Não
(HOLZ et al., 2008a)	Não	Não	Sim
(STOCK et al., 2009)	Não	Não	Sim
(GU et al., 2007)	Sim	Sim	Não
(WURZINGER et al., 2009)	Sim	Sim	Sim

Correlação Horizontal Correlação Vertical

Tabela 2.1: Comparativo das diferentes técnicas de detecção de *botnets*.

tece, basicamente, porque tais técnicas analisam o comportamento na rede de dois ou mais *bots* e detectam indícios de tráfego típico de máquinas comprometidas. Por outro lado, nota-se que boa parte dos métodos é dependente de protocolos ou arquiteturas, como por exemplo, redes baseadas em protocolos *botnets* P2P e *IRC*. Logo, a maioria das técnicas é eficiente para determinados tipos e configuração. No entanto, para novas *botnets* com diferentes estruturas e protocolos as soluções descritas são limitadas. No tocante a redes que utilizam comunicação ou protocolos criptografados, um pequeno conjunto de soluções é eficiente. Como a maioria das técnicas são baseadas na análise de dados, é difícil estabelecer uma relação entre as diferentes ações maliciosas realizadas.

De modo geral, as técnicas que utilizam a abordagem horizontal apresentam limitações no que tange a identificação de *bots* em redes com grande quantidade de tráfego ou com um grande volume de dados de comunicação entre máquinas. Além disso, é necessário que as máquinas comprometidas estejam ativas na rede - gerando tráfego malicioso - para que o mesmo seja observado e analisado. Por fim, a principal desvantagem dessa técnica é que a mesma não consegue identificar *bots* individuais, ou seja, faz-se necessário que duas ou mais máquinas apresentem comportamentos similares - integrantes de uma mesma *botnet* - na rede monitorada.

Por outro lado, os trabalhos que utilizam a *correlação vertical* possuem uma abordagem mais específica, atuando diretamente nas ações de cada *bot*. A maioria das soluções apresentadas atuam em redes específicas desenvolvendo métodos diretos - sem processamento extra - para a identificação na rede. Nota-se uma grande quantidade de trabalhos que busca atuar em redes tradicionais como as baseadas no protocolo IRC e mais atualmente em P2P. Uma exceção é o trabalho de Peter Wurzinger e demais autores (WUR-

ZINGER et al., 2009) que implementa detecção automatizada de assinaturas de vários tipos de *botnets*.

A principal limitação da abordagem correlação vertical é a necessidade de um pré-conhecimento das atividades dos *bots*. No entanto, trata-se de uma abordagem bastante flexível que pode ser replicada de forma rápida para a identificação de *bots* em outras redes (com base nas assinaturas desenvolvidas). Por fim, ambas as abordagens de pesquisa (correlação horizontal ou vertical) são eficientes, em certos contextos, para detectar redes maliciosas. No entanto, nota-se que as soluções baseadas em assinaturas de rede vem ganhando evidência em pesquisas recentes. Em particular, a correlação vertical possui características mais flexíveis: a) não necessita de dois ou mais *bots* para identificar uma *botnet*; b) não necessita que a rede maliciosa esteja ativa (realizando atividades).

Independente da abordagem utilizada, observa-se que as soluções atuais de detecção e mitigação de *botnets* continuam apresentando desafios significantes para os pesquisadores. A natureza fluida dessas ameaças requerem que os pesquisadores desenvolvam técnicas cada vez mais flexíveis a fim de identificar novas funcionalidades e ações das futuras *botnets*.

3 MITIGAÇÃO DE BOTNETS

Este capítulo apresenta conceitos relacionados com solução proposta, ou seja, da arquitetura de mitigação e detecção de *botnets*. Os conceitos apresentados neste capítulo visam facilitar o entendimento e dar embasamento para melhor compreender todos os elementos que compõe a arquitetura proposta. Por questões organizacionais, este capítulo está disposto nas seguintes seções: *Anatomia de Botnets*, que descreve o funcionamento elementar de uma *botnet*; *Honeypots* e *Honeynets*, responsável por apresentar os conceitos básicos de tais ferramentas; *Fluxos de Rede*, que descreve a tecnologia de monitoração de fluxos; e, por fim, o capítulo é encerrado na seção *Análise de Malwares* que discute o processo de análise de informações contidas em arquivos binários maliciosos.

3.1 Anatomia de *Botnets*

As *botnets* podem ser definidas como um conjunto de máquinas comprometidas que podem ser controladas remotamente por uma entidade administrativa. Essa entidade administrativa é representada por um humano denominado *botmaster* que tem a capacidade de gerenciar toda a estrutura de uma rede maliciosa. Com isso, o *botmaster* utiliza a estrutura sob seu controle para ações relacionadas com as mais diversas atividades ilícitas, em especial, com objetivos financeiros (BACHER et al., 2005).

O uso das *botnets* como uma estrutura para lançar ataques maliciosos a diferentes redes é muito frequente. Cada máquina comprometida apresenta um conjunto de funcionalidades pré-programadas que podem ser invocadas remotamente pelo *botmaster*. Além disso, os *bots* possuem, na maioria dos casos, a opção de atualização remota. Com isso, novas funcionalidades podem ser incorporadas à rede sem a necessidade de renovação da estrutura ou do comprometimento de novas máquinas. As atividades maliciosas mais frequentes observadas na rede compreendem: roubo de identidade, baseado nas credenciais obtidas no sistema comprometido; roubo de dados sensíveis (espionagem industrial); hospedagem de sites falsos; propagação de *malwares*; sondagens e ataques a sistemas computacionais; campanhas de envio de *spams*; negação de serviço distribuído (SCHOOF; KONING, 2007); entre outras;

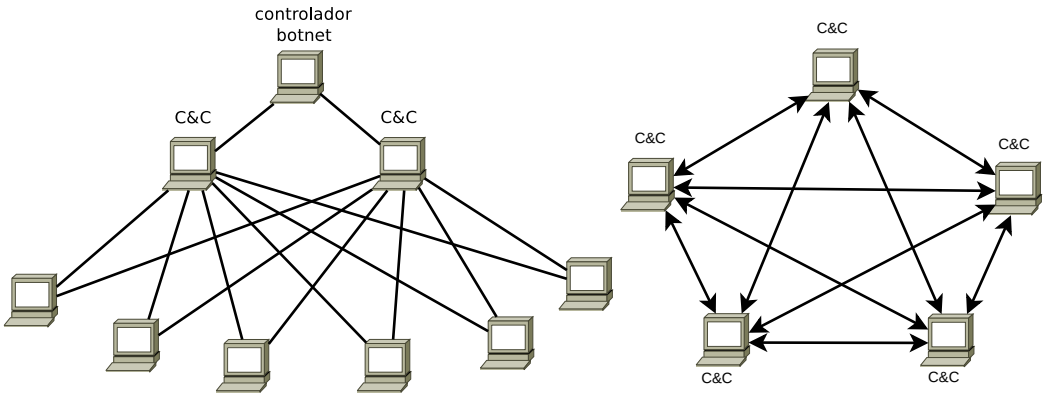
Observa-se uma grande heterogeneidade em relação às *botnets* disseminadas na rede. Pesquisadores já identificaram *botnets* dos mais diferentes tamanhos, com poucas centenas de *bots* e outras com milhares de *bots* (RAJAB et al., 2007). No entanto, estimar o tamanho médio de uma *botnet* é uma tarefa complexa devido à grande quantidade de variáveis presentes na constituição das mesmas. Fatores como migração temporária entre canais de comunicação, clone de canais de controle e estruturas ocultas são as principais dificuldades no processo de enumerar a população de uma *botnet* (RAJAB et al.,

2007). Mesmo assim, é factível realizar uma aproximação quanto ao tamanho das redes. Segundo o *Internet Storm Center* - organização especializada em detectar, analisar e disseminar informações relacionadas com problemas de segurança da Internet, já foram observadas redes com mais de 100,000 computadores infectados (HOLZ et al., 2008a).

Embora existam diferentes tamanhos e arquiteturas de *botnets*, as mesmas são constituídas por entidades básicas de funcionamento. O entendimento de uma *botnet* passa pela identificação dos seus elementos funcionais que são compostos por: *bots*, *botmaster* e *canal de controle e comando*. Um *bot* é definido como um programa de computador instalado em uma máquina a qual permite que um atacante execute comandos arbitrários no sistema infectado (STINSON; MITCHELL, 2008a). O termo *bot* também é utilizado para referir-se a própria máquina comprometida e integrante de uma *botnet*. Assim como outros tipos de *malwares* (*worms* e vírus), os *bots* são disseminados pelos tradicionais vetores de propagação, como por exemplo: a utilização de arquivos maliciosos embutidos em e-mails; *malwares* compartilhados em redes P2P; *scripts* maliciosos no corpo de páginas HTML que exploram vulnerabilidades de navegadores Web; e outros tipos de ataques (COVA; KRUEGEL; VIGNA, 2008) (NAZARIO; HOLZ, 2008).

A administração de uma *botnet* passa por uma estrutura de controle - denominada de *canal de controle e comando* - a qual permite que comandos sejam enviados às máquinas comprometidas. Da mesma forma, essa estrutura possibilita que atividades sejam automatizadas pelo administrador da *botnet*. A estrutura de controle pode ser bastante complexa, pois compreende protocolos de comunicação, servidores de controle e um conjunto de regras para o estabelecimento de uma conexão (SCHOOF; KONING, 2007). Além disso, é comum encontrar nas redes maliciosas técnicas que visam preservar o anonimato de um *botmaster*. Mecanismos como o uso de *proxy*, tunelamento, e canais cifrados são empregados com frequência pelos controladores das *botnets* com o objetivo de ocultar a sua identidade.

O *canal de controle e comando* é um elemento crítico na arquitetura de uma *botnet*, afinal o mesmo define como a *botnet* será estruturada logicamente e como se dará a intercomunicação entre os elementos da rede. É possível observar, nas *botnets* atuais, diferentes implementações do *canal de controle e comando* tendo em vista a sua arquitetura. Estudos apontam que a grande parte das *botnets* utilizam arquitetura centralizada (HOLZ et al., 2008a) (STINSON; MITCHELL, 2008b). Esse fato decorre-se de questões históricas: as primeiras estruturas de controle das *botnets* foram herdadas de servidores IRC, onde diversos clientes conectavam em um servidor para comunicação em tempo real. Devido a grande quantidade de máquinas conectadas, os servidores de IRC disponibilizavam de um conjunto de ferramentas para automatizar certas tarefas, como envio de notícias e registros de nomes. Essas ferramentas eram denominadas de robôs (*robot*) e tornaram-se bastante popular nessas redes, como é o caso do *EggDrop* (GRIZZARD et al., 2007). Sendo assim, as *botnets* de estrutura centralizada - representada na figura 3.1(a) - utilizando o protocolo de comunicação IRC ganharam popularidade e continuam sendo utilizadas até os tempos atuais com bastante eficácia. Nota-se, porém, algumas variações nessas arquiteturas buscando tornar o canal de comunicação menos perceptível: a utilização de outros protocolos de comunicação, como por exemplo, o HTTP. O protocolo HTTP possui vantagens em relação a outros protocolos: tem livre trânsito, isto é, o tráfego Web geralmente é permitido na maioria das redes; e também pode facilmente encapsular dados em seus datagramas.



(a) *Botnet* com controle centralizado.

(b) *Botnet* com controle descentralizado.

Figura 3.1: Diferentes arquiteturas utilizadas pelas *botnets*.

Embora as estruturas centralizadas sejam eficientes, as mesmas sofrem de um problema intrínseco de toda estrutura centralizada: um ponto central de falha. A desarticulação da estrutura de controle de comando inviabiliza o controle da *botnet* por parte do atacante, perdendo assim a sua efetividade. Com o passar do tempo novas arquiteturas de *botnet* foram implementadas, como por exemplo, arquiteturas descentralizadas que utilizam protocolos *Peer-to-Peer* (P2P) para comunicação. A figura 3.1(b) ilustra uma *botnet* utilizando arquitetura descentralizada para troca de mensagens. Em tais arquiteturas não existe um ponto central de falhas. Os nodos que compõe redes do tipo P2P atuam como clientes e servidores. Dessa forma, não existe uma coordenação centralizada que possa ser incapacitada. Se algum nodo controlador da rede for desabilitado outro nodo assume as funcionalidades e a rede continua a operar sob o controle do *botmaster*. Tais estruturas, ainda observadas em menor escala, são mais desafiadoras e necessitam de diferentes abordagens para a desarticulação das redes maliciosas.

Independentemente da arquitetura utilizada e metodologias implementadas, o entendimento dos elementos *bots*, *canal de controle e comando*, e *botmasters* são fundamentais para desenvolver métodos de desarticulação de uma *botnet*. Sendo assim, faz-se necessário desenvolver técnicas robusta de detecção e mitigação de *botnets* tendo em vista as particularidades de cada entidade da *botnet* em si.

3.2 Fluxos de Rede

Existem diferentes maneiras de monitorar o tráfego de uma rede de computadores: uma delas é utilizar a tecnologia de monitoração baseada em fluxos de rede. Essa tecnologia busca coletar informações de datagramas da rede com o objetivo de mapear em tempo real a situação do tráfego.

O conjunto de informações disponíveis numa rede é muito amplo. Existem diferentes tipos de dados relacionados a cada datagrama que podem ser monitorados. Para isso, foram definidas especificações que padronizam os dados a serem observados numa rede. As especificações dos protocolos `NetFlow` (Cisco, 2010) e `Sflow` (WANG; LI; LI, 2004) são exemplos de padronizações bastante disseminadas em equipamentos de rede.

No contexto do protocolo NetFlow, mas especificamente na versão 5 desse protocolo, um fluxo de rede é definido por uma tupla de sete informações: endereço IP de origem, en-

dereço IP de destino, porta de origem de origem, porta de destino, protocolo, interface, IP ToS. Além das especificações do NetFlow versão 5, existem outras versões com maior nível de detalhamento como é o caso da própria versão 9 do protocolo. Mais recentemente, um novo protocolo denominado IPFIX (DIETZ et al., 2010) - baseado em NetFlow - vem sendo especificado a fim de padronizar universalmente um conjunto de informações que podem ser exportadas por dispositivos aptos a exportarem fluxos do protocolo IP.

Independente do tipo de protocolo utilizado, a arquitetura de monitoração de fluxos possui uma estrutura própria que é composta de elementos básicos funcionais. Nessa arquitetura, representada na figura 3.2, dois elementos são destacados: o *Exportador de fluxos*, que é responsável por obter informações do enlace monitorado; e o *Coletor de fluxos*, que armazena os dados exportados de forma persistente.

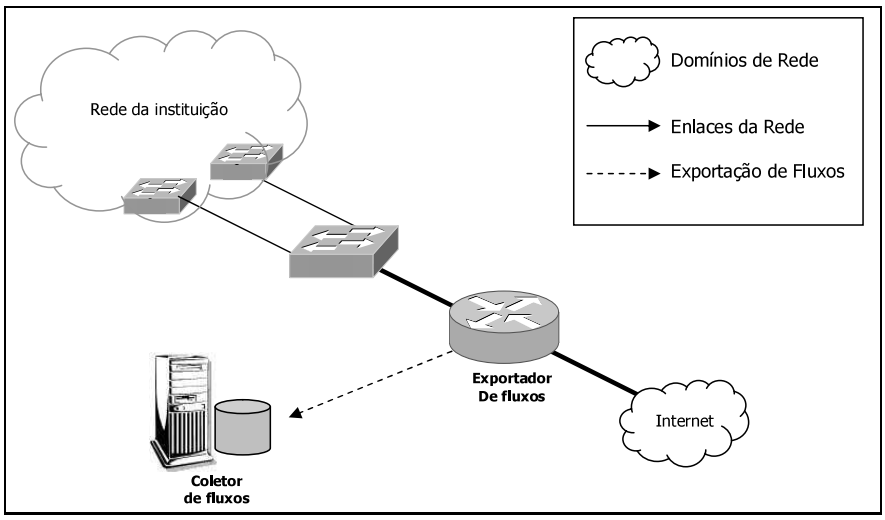


Figura 3.2: Arquitetura de monitoração de fluxos de rede.

Os fluxos são exportados por dispositivos posicionados em posição estratégia na rede, geralmente na borda da rede ou em enlaces que se deseja monitorar. Tais dispositivos - tipicamente roteadores ou comutadores de datagramas - coletam informações da rede e as exportam para o *Coletor de fluxos*. Dessa forma, todos os dados são encapsulados em datagramas UDP e devidamente encaminhados para o seu destino de análise.

Atuando de forma complementar na arquitetura, o *Coletor de fluxos*, por sua vez, encarrega-se de receber os fluxos exportados e armazená-los numa base de dados persistente. Essa atividade é um tanto dispendiosa quanto ao processamento. No entanto, existem softwares especializados na tarefa de processamento e armazenamento de fluxos de rede. Os softwares Nfdump (NFDUMP, 2010) e Flow-tools (ROMIG, 2000) são ferramentas amplamente utilizadas e são aptos a receber os datagramas UDP gerados pelo *Exportador de fluxos* e os armazenando de forma persistente numa base de informações.

O armazenamento dessas informações é fundamental, sobretudo no processo de análise de dados. Através dos softwares especializados é possível processar as informações armazenadas no *Coletor de Fluxos* e tecer conclusões a respeito do tráfego monitorado. A grande parte das ferramentas de análise de fluxos disponibiliza interfaces - gráficas ou via linha de comando - através das quais diversas estatísticas podem ser traçadas. A figura 3.3 disponibiliza uma listagem de fluxos de rede (NetFlow) obtidas com auxílio da ferramenta Nfdump.

Src IP Addr:Port		Dst IP Addr:Port	Packets	Bytes	Flows
201.21.224.[illegible]:0	->	143.54.100.[illegible]:0	28889	3.0 M	4831
143.54.1.[illegible]:0	->	201.21.224.[illegible]:0	32679	10.1 M	4673
201.21.224.[illegible]:0	->	143.54.1.[illegible]:0	4673	224304	4673
143.54.1.[illegible]:0	->	189.6.128.[illegible]:0	3254	620236	1072
189.6.128.[illegible]:0	->	143.54.1.[illegible]:0	5327	752204	1072
66.67.33.[illegible]:0	->	143.54.15[illegible]:0	353	21180	353
66.67.33.[illegible]:0	->	143.54.15.[illegible]:0	351	21060	350
189.10.153.[illegible]:0	->	143.54.100.[illegible]:0	2040	740482	325
143.54.1.[illegible]:0	->	189.10.153.[illegible]:0	1798	259739	324
189.10.153.[illegible]:0	->	143.54.1.[illegible]:0	308	14784	308

Figura 3.3: Informações de fluxos exportados numa rede monitorada.

O conjunto de informações, exibidos na figura 3.3, apresenta 5 colunas onde cada linha fornece dados relativos a um único fluxo coletado na rede. As colunas representam respectivamente: o endereço IP origem; o IP destino; o número de datagramas da conexão; o número de *bytes*; e o número de fluxos ou conexões. Com esses dados é possível derivar informações sobre o tráfego da rede (total de *bytes*, total de datagramas) e também estatísticas de acessos para cada máquina (comunicações e comportamentos de máquinas). Em suma, o conjunto de informações disponibilizadas pelos fluxos fornece uma visão muito aproximada da real situação da rede e permite que investigações sejam realizadas.

Por fim, a monitoração de fluxos é um importante recurso de gerenciamento e segurança das mais diversas redes. Por ser escalável e possuir alto desempenho, a tecnologia de fluxos é utilizada em redes heterogenias com diferentes quantidades de datagramas por segundo. A popularização dessa tecnologia - suportada cada vez mais por dispositivos de rede - tende a auxiliar e complementar os métodos de gerenciamento mais tradicionais, como é o caso da arquitetura SNMP.

3.3 *Honeypots* e *Honeynets*

Os *honeypots* são sistemas computacionais dedicados a serem sondados e comprometidos (MOKUBE; ADAMS, 2007). Trivialmente esses sistemas são implementados por ferramentas que emulam aplicações vulneráveis - serviços de rede - especialmente preparadas para coletar informações de sondagens recebidas. Já as *honeynets*, por sua vez, é o nome dado a um conjunto de *honeypots* cujo objetivo é potencializar os resultados obtidos por um único *honeypot*.

O conjunto de ataques e informações que chegam até um *honeypot* são úteis para o aprimoramento da segurança de redes. Em geral, as informações coletadas por um *honeypot* não são observadas em outras ferramentas, tal como os IDS's. Para detectar um comportamento malicioso, um IDS necessita de assinaturas de ataques conhecidos e frequentemente falham em detectar novos ataques que ainda não possuem assinaturas. Os *honeypots*, por outro lado, são capazes de identificar vulnerabilidades ou ataques desconhecidos sem a necessidade de assinatura do ataque. Por exemplo, os *honeypots* permitem analisar técnicas - ou até mesmo artefatos - utilizados num ataque a um serviço vulnerável.

Como um *honeypot* não possui nenhum serviço legítimo em produção, toda tentativa

de contato ao sistema é por definição suspeita. Consequentemente, os acessos ao *honeypot* são ataques em potencial o que minimiza drasticamente o número de falsos positivos. Situação essa, que não ocorre em outras ferramentas, como os IDS's que produzem uma alta taxa de falsos positivos. Os *honeypots*, caracterizam-se por atuarem como iscas para os atacantes e permitem capturar o máximo de informações específicas aos serviços emulados. O valor dessas informações coletadas auxilia, de forma complementar, a identificar exceções nos sistemas computacionais. Sendo assim, os *honeypots* provaram ser eficazes para estudar crimes de Internet, como roubo de cartões de crédito (VIRTI et al., 2006), identificação de *botnets*, e coleta de arquivos maliciosos (CERON et al., 2008).

Os *honeypots* podem ser implementados em diferentes sistemas operacionais e disponibilizar diversos serviços vulneráveis. A implementação dos serviços determina o escopo de interação que será fornecido ao atacante (que irá interagir com o *honeypot*). A literatura sugere a classificação dos *honeypots* em dois níveis (MOKUBE; ADAMS, 2007), segundo o grau de interação disponibilizado ao atacante.

Honeypots de baixa interatividade oferecem serviços limitados para o atacante. Os serviços implementados não são efetivamente funcionais mas sim emulados. Nesses sistemas, o atacante tem contato com um ambiente que mascara um sistema operacional real dando ilusão que serviços estão sendo executados. Essas funcionalidades são fornecidas por softwares que emulam serviços simples, tendo em vista o comportamento do atacante e as possíveis ações de sondagem. Por se tratarem de serviços emulados, esse tipo de *honeypot* é considerado mais seguro, afinal o atacante não tem contato com recursos reais (serviços ou sistema operacional).

Honeypots de alta interatividade por outro lado, não emulam nenhum tipo de serviço. Estes *honeypots* permitem com que o atacante interaja com o sistema real, no qual quase nada é restrito. Geralmente esta categoria de *honeypot* é implementada num sistema operacional previamente preparado para monitorar ações do atacante. Por exemplo, um sistema operacional com o núcleo (*kernel*) preparado para monitorar certas chamadas de sistema e com isso traçar ações como teclas digitais e arquivos modificados (HUANG; YANG; AHN, 2009). Pelas suas características intrínsecas, estes *honeypots* apresentam mais riscos de segurança do que os *honeypots* de baixa interatividade. Além disso, os *honeypots* de alta interatividade tendem a ser mais difíceis de gerenciar e implementar, tendo em vista os cuidados extras que devem ser empregados, como por exemplo, uma estrutura de contenção para possíveis comprometimentos (*firewall*, IDS) (SPITZNER, 2002).

Comparativamente, as duas categorias de *honeypots* possuem focos diferentes de atuação: divergem no tipo de informação que se deseja obter dos ataques. Os *honeypots* de baixa interatividade são eficientes para detectar varreduras ou sondagens não autorizadas, como por exemplo, varreduras de *worms* e *bots*. Já, os de alta interatividade são úteis para coletar informações de ataques mais sofisticados, que geralmente envolvem interação humana. Enfim, o tipo de *honeypot* a ser utilizado depende do conjunto de informações que se deseja obter.

Para facilitar a tarefa de implementação e gerenciamento de *honeypots*, diversas ferramentas foram desenvolvidas (PROVOS, 2004) (Nepenthes, 2010) (ZHUGE et al., 2007). Uma ferramenta bastante utilizada para implementar *honeypots* de baixa interatividade é a ferramenta Nepenthes. O Nepenthes permite implementar uma série de serviços vulneráveis e coletar informações sobre as sondagens aos serviços emulados. Por exemplo, é pos-

sível emular vulnerabilidades em diferentes arquiteturas e sistemas operacionais em uma única máquina. Sistemas operacionais Windows e Linux, com respectivos serviços, são facilmente emulados de forma escalável. Experimentos demonstram que o software Nepenthes pode emular mais de 16 000 sistemas vulneráveis em uma única máquina física. Em suma, o software Nepenthes permite a emulação de um conjunto de serviços vulneráveis cada qual com um respectivo endereço IP. Além do mais, o Nepenthes caracteriza-se por ser muito flexível quanto a configuração de novos serviços vulneráveis. Com uma arquitetura modular implementada pelo software, diversas operações podem ser agregadas, como por exemplo: a) agregar novas vulnerabilidades; b) módulo de coleta de binários; c) analisador de *shellcode*; e d) módulo de registro (*logs*).

Destaca-se a funcionalidade de coleta de arquivos binários. Nela o software consegue realizar uma interação com o ataque (*exploit*) e, na maioria das vezes, obter o arquivo binário responsável pela sondagem. Logo, o software Nepenthes torna-se bastante útil para coletar arquivos binários propagados por meios autônomos na rede, como já apresentados em vários trabalhos (NAZARIO; HOLZ, 2008) (BARFORD; BLODGETT, 2007). Além do Nepenthes existem outros softwares como é o caso do *honeyd* que implementa serviços de rede em nível de pilha TCP/IP (PROVOS, 2004), e também softwares mais complexos que implementam *honeypots* de alta interatividade, como é o caso do *Sebek* (HUANG; YANG; AHN, 2009).

3.4 Análise de *Malwares*

A análise de arquivos executáveis, sobretudo arquivos supostamente maliciosos é muito importante para identificar as funcionalidades e características utilizadas pelos *malwares*. O interesse por entender o funcionamento e características de um arquivo é uma preocupação recorrente na comunidade de segurança. A maioria dos produtos de segurança como antivírus e sistemas de detecção de intrusão trabalham com assinaturas - uma sequência características de *bytes* - para identificar um código malicioso (STINSON; MITCHELL, 2008b). Logo, a análise de arquivos maliciosos tende a contribuir com o aprimoramento da segurança dos sistemas como um todo.

Entretanto, o processo de análise de *malwares* é uma tarefa complexa já que os mesmos multiplicam-se e espalham-se muito rapidamente (COOKE; MAO; AND, 2006). A constante evolução dos *malwares* faz com que novas técnicas sejam implementadas com o objetivo de burlar ferramentas de segurança. As características mais recentes que buscam evitar a identificação de *malwares* pelos antivírus são implementadas através de polimorfismo e metamorfismo (CERON; GRANVILLE; TAROUCO, 2009).

Além de analisar um arquivo malicioso em busca de assinaturas, é possível extrair informações a respeito do funcionamento de um arquivo. A análise comportamental consiste em traçar os eventos ou ações desempenhadas por um arquivo binário no sistema operacional no qual é executado. Essa análise do funcionamento de arquivos binários pode ser realizada utilizando duas diferentes abordagens: de forma estática ou dinâmica.

A análise estática consiste em avaliar o funcionamento de um programa sem a execução do mesmo, baseando-se apenas no seu código. As técnicas mais tradicionais consistem em extrair instruções do código ASSEMBLY do programa e inferir conclusões com base na sequência das instruções a serem executadas (KOLTER; MALOOF, 2004). No entanto, a principal deficiência desse método é a possibilidade do código de máquina analisado não refletir o mesmo comportamento que o *malware* apresenta quando executado. Em particular, a análise estática é pouco eficiente para programas que utilizam técnicas

de ofuscação ou polimorfismo, como é o caso do *malware* `Conficker` (KELCHNER, 2010). Tais limitações da análise estática incitaram o surgimento de técnicas complementares, como é o caso da análise dinâmica.

A análise dinâmica, por outro lado, consiste em observar as características funcionais do *malware* através da sua execução num ambiente controlado. Segundo o trabalho de Willems e demais autores (WILLEMS; HOLZ; FREILING, 2007) as principais metodologias de análise dinâmica baseiam-se em: (a) comparação do estado do sistema operacional antes e imediatamente após a execução do arquivo; e (b) monitoramento das ações em tempo de execução. Na primeira abordagem, busca-se fazer uma comparação do sistema operacional completo identificando alterações causadas pelo arquivo binário executado. Como resultado, essa técnica traça uma visão geral das funcionalidades do binário, como arquivos criados, dados removidos, e funções instanciadas. Essa solução, entretanto, não determina mudanças dinâmicas intermediárias ao estado inicial e final da comparação do sistema. Mudanças como a criação de arquivos durante a execução e a deleção de arquivos antes do final do processo são transparentes a essa análise.

Por outro lado, na segunda abordagem, cuja monitoração das ações do *malware* é dada durante a execução, todas as ações são traçadas. Mesmo sendo mais complexa de implementar, a análise de binários durante a execução do mesmo vem se popularizando devido ao bom resultado da técnica perante códigos polimórficos e ofuscados. A principal limitação da análise dinâmica de *malware* é a possibilidade de executar apenas uma amostra de binário de cada vez. Afinal, a execução de outros binários no mesmo ambiente controlado dificulta a distinção das ações de cada *malware*. Recentemente, com os melhoramentos de tecnologias de virtualização de sistemas, a análise dinâmica de *malwares* ganhou outra dimensão. De fato, a facilidade de reconstruir um ambiente virtualizado propiciou o surgimento de ferramentas mais detalhistas e escaláveis para a análise dinâmica, como é o caso dos *sandboxes*.

Ferramentas como CWSandbox (WILLEMS; HOLZ; FREILING, 2007), Norman Sandbox (Norman, 2010) e Anúbis (ISECLAB, 2010) são exemplos de ferramentas automatizadas para análise dinâmica de arquivos binários. Essas ferramentas caracterizam-se por executar um arquivo num ambiente controlado registrando em forma de relatório as ações realizadas pelos *malwares*. Como característica, os *sandboxes* tradicionalmente simulam o sistema operacional Windows, já que a grande maioria de *malwares* existentes é escrita para o mesmo (BARFORD; BLODGETT, 2007). As principais funcionalidades observadas pelos *sandboxes* descrevem características como: arquivos criados ou modificados durante a execução do arquivo; monitoração de acesso ou modificações à chave de registros do sistema; biblioteca dinâmicas carregadas durante a execução; áreas de memória virtual utilizada; processos instanciados; conexões de rede; dados transmitidos, entre outros.

Os relatórios disponibilizados pelos *sandboxes* podem variar bastante devido a particularidades de implementação dos mesmos (WILLEMS; HOLZ; FREILING, 2007). Por exemplo, o Norman Sandbox simula um computador conectado na rede reimplementando partes do núcleo do sistema Windows emulado. Já o Anubis é implementado com base no sistema Qemu (BELLARD, 2005) emulando a arquitetura i386. Além das características inerentes às particularidades do sistema emulado, cada *sandbox* pode variar no que tange ao conjunto de funcionalidades e chamadas do sistema que o mesmo irá avaliar em cada execução do binário analisado.

A grande parte das ferramentas de *sandbox* existentes, mesmo as comerciais, disponi-

biliza uma interface sem restrição de acesso para a avaliação de resultados. Essas ferramentas apresentam um relatório com as ações desempenhas e mapeadas pelo sistema em questão em relação ao arquivo submetido. De fato, algumas ferramentas comerciais como o Norman Sandbox apresentam relatórios pouco descritivos na versão disponível na Web, em contraste à versão comercializada. Já ferramentas como o Anubis e CWSandbox disponibilizam um nível maior de detalhamento. Na sequência, são apresentados exemplos de relatórios de análise obtidos através da submissão de arquivos às ferramentas gratuitamente disponíveis na Web.

A figura 3.4 apresenta um relatório de funcionalidades de um *malware* analisado pela ferramenta Norman Sandbox. Como visto, o relatório é pouco descritivo e apresenta apenas linhas gerais de análise. Assinatura do *malware*, método de compressão do binário, tamanho do arquivo, informações sobre os processos, são apenas alguns exemplos de informações descritas pelo *sandbox*.

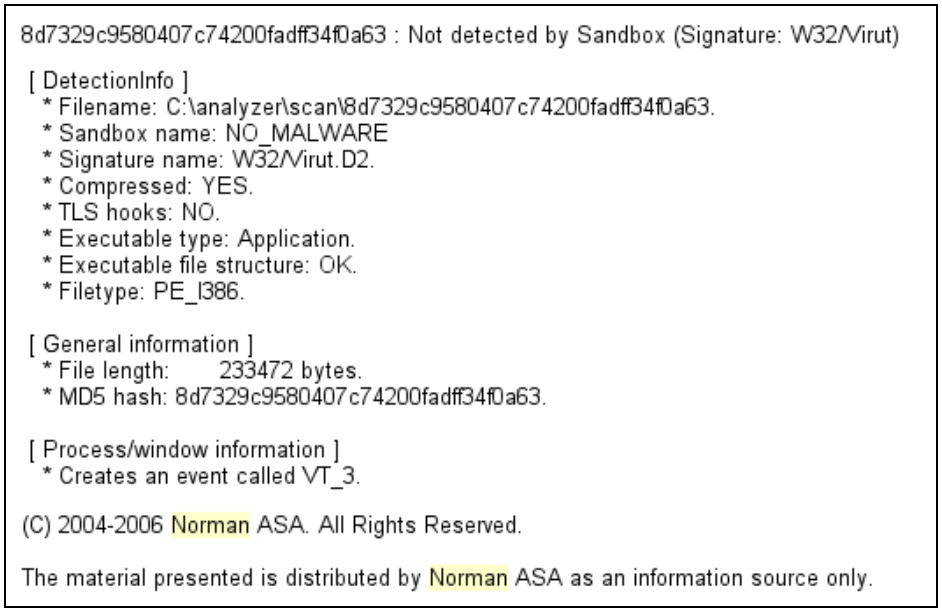
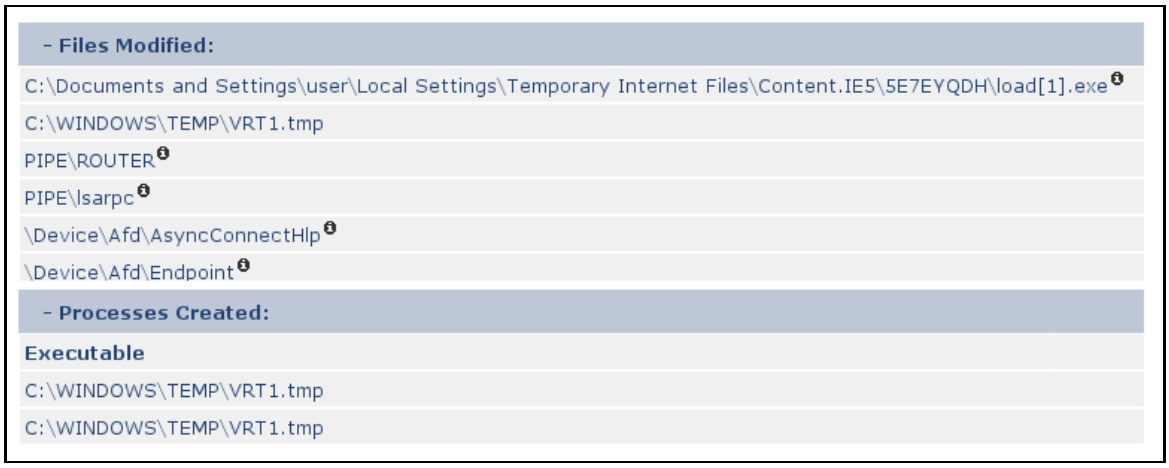


Figura 3.4: Relatório das ações de um *malware* disponibilizado pelo Norman Sandbox.

Por outro lado, o relatório disponibilizado pela ferramenta Anubis é muito mais detalhado. A figura 3.5 apresenta apenas um fragmento do relatório do Anubis, mesmo assim já é possível ter uma noção do nível de detalhamento fornecido pela ferramenta. Nesse relatório, são evidenciadas duas características de execução do binário analisado: na parte superior, todos os arquivos modificados no sistema; e na parte inferior do relatório, os processos instanciados pelo mesmo.



- Files Modified:
C:\Documents and Settings\user\Local Settings\Temporary Internet Files\Content.IE5\5E7EYQDH\load[1].exe
C:\WINDOWS\TEMP\VRT1.tmp
PIPE\ROUTER
PIPE\lsarpc
\Device\Afd\AsyncConnectHlp
\Device\Afd\Endpoint
- Processes Created:
Executable
C:\WINDOWS\TEMP\VRT1.tmp
C:\WINDOWS\TEMP\VRT1.tmp

Figura 3.5: Fragmento de um relatório das ações de um *malware* disponibilizado pelo *sandbox* Anubis.

As diferentes técnicas de análise de arquivos binários descritas anteriormente, em última análise, possuem o mesmo objetivo: lidar com a árdua tarefa de identificar as funcionalidades de um arquivo binário. A utilização de ferramentas para análise tende a contribuir e dar escalabilidade à tarefa de compreender o funcionamento de arquivos maliciosos. As informações coletadas pela análise de *malwares* são fundamentais para melhor entender os vetores de ataque e motivações dos atacantes. Por fim, o conjunto de dados oriundo de análise de arquivos maliciosos viabilizam o desenvolvimento de novas mecanismos e metodologia para aprimorar a segurança dos sistemas computacionais.

4 ARQUITETURA PARA MITIGAÇÃO DE *BOTNETS*

A tarefa de mitigar *botnets* é um processo complexo e extremamente dinâmico. A desarticulação do canal de controle da *botnet* é uma tarefa que não só envolve esforços coordenados entre diferentes gerentes de rede, mas também necessita de técnicas e mecanismos sofisticados para a detecção das mesmas. Dada a dificuldade de uma rápida inibição do controlador da *botnet*, torna-se prudente impedir que os *bots* tenham acesso ao canal de controle, isto é, impossibilitar que máquinas infectadas recebam instruções maliciosas do controlador da *botnet* em questão.

Como já apresentado no capítulo 2, a literatura sobre o assunto apresenta diferentes técnicas de identificação de controladores de *botnet*. É de conhecimento, entretanto, que a detecção de *botnets* é um processo dinâmico e que constantemente deve ser revisitado para contemplar as novas características das *botnets*.

Neste contexto, o presente trabalho busca desenvolver uma arquitetura que auxilie o gerente de segurança a identificar e tomar ações administrativas nas máquinas infectadas (*bots*) em seu próprio domínio administrativo. Para que esse objetivo seja alcançado, foi desenvolvida um arquitetura que compõe diferentes técnicas de mitigação tendo em vista requisitos de flexibilidade e escalabilidade. A flexibilidade, em especial, é um requisito essencial para o sistema, pois o mesmo deve permitir que os componentes da arquitetura sejam modificados para se adequar as novas ameaças e também as especificidades da rede local administrada. Já a escalabilidade deve ser considerada afinal a arquitetura se propõe a identificar e processar um conjunto de assinaturas que o número tende a aumentar constantemente.

Este capítulo descreve uma arquitetura de mitigação de *botnets* compondo técnicas existentes, especialmente ferramentas distribuídas *on-line* para montar uma base de dados que será utilizada como suporte para a identificação de máquinas infectadas. Este capítulo está organizado como segue. Inicialmente, na seção 4.1, é apresentada uma visão geral do funcionamento da arquitetura proposta. Posteriormente são descritos os componentes de cada camada de abstração: na seção 4.2 a camada de *Coleta de Malware*; na seção 4.3 a camada de *Detecção de Botnet*; e, por fim, a camada de *Rastreamento de Botnets* na seção 4.4.

4.1 Visão Geral da Arquitetura Proposta

A partir dos requisitos de arquitetura partiu-se para a definição de uma solução para a mitigação de *botnets*. A localização de máquinas infectadas na rede é realizada com base em padrões comportamentais das *botnets*. Para isto, a arquitetura proposta automatiza o processo de geração de assinaturas de *botnets* analisando arquivos binários suspeitos e identificando atividades típicas de um *bot*. Todo o processo de análise é automatizado por

um conjunto de ferramentas analisadoras de arquivos denominadas de *sandboxes* (vide seção 3.4).

Os *sandboxes* caracterizam-se por executar um arquivo suspeito (*malware*) num ambiente controlado e analisar o seu comportamento. A resposta dessas ferramentas é constituídas por relatórios que descrevem as funcionalidades mapeadas em diferentes formatos, como por exemplo, TXT, XML, PDF e HTML. Algumas ferramentas disponibilizam, de forma complementar, um arquivo com todas as atividades do arquivo analisado em nível de rede (em formato *pcap*). Os recursos disponibilizados pelas ferramentas de análise são armazenados e processados localmente pelos próximos elementos da arquitetura proposta. Logo após, os sistema faz uma correlação entre as funcionalidades do arquivo analisado e o seu comportamento de rede, buscando identificar um possível canal de controle ou, em última análise, um controlador de *botnet*. Uma vez tendo o padrão comportamental identificado, é realizada uma busca por tais padrões com o objetivo de encontrar máquinas possivelmente infectadas (*bots*). Para cada máquina identificada a arquitetura define regras de bloqueio - regras para ferramentas de segurança (*firewall* ou IDS) - as quais são submetidas para avaliação de um analista de segurança.

Como já comentado, as assinaturas das *botnets* são derivadas da análise de arquivos binários que são coletados pelo sistema. No entanto, o sistema permite que novas assinaturas de *botnets* sejam incorporadas ao sistema. Afinal, é de conhecimento que a coleta de binários é limitada e nem todos os exemplares existentes serão coletados ou até mesmo identificado pelo sistema de análise de *malware*. A importação de assinaturas de *botnets* de outros sistemas ou grupos de pesquisas externos, tal como Shadow Server (Shadowserver, 2010) só tem a corroborar na identificação de máquinas contaminadas. Outra funcionalidade interessante da arquitetura, é fornecer ao gerente de segurança estatísticas referentes a máquinas infectadas detectadas em sua rede local, bem como informações referentes aos controladores de *botnets*.

Para que as funcionalidades acima fossem desempenhadas, foi definida uma arquitetura que envolve três camadas de abstração: *Coleta de Malwares*, *Detecção de Botnets* e *Rastreamento de Botnets*. A figura 4.1 representa a disposição entre as abstrações e a respectiva interface de comunicação entre as mesmas: *acesso criptografado via rede* e *API de consulta e filtros para tráfego de rede*.

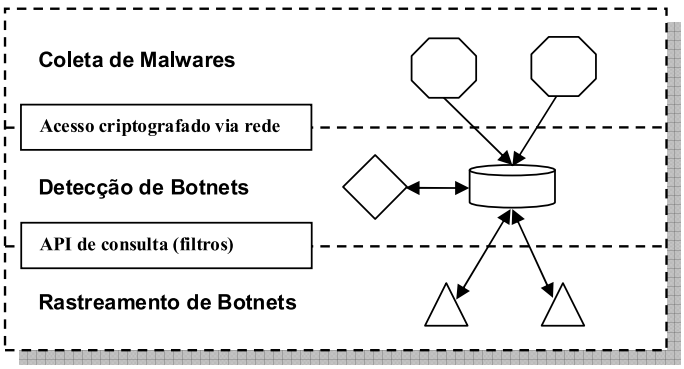


Figura 4.1: Abstrações definidas pela arquitetura proposta.

As camadas são organizadas segundo características funcionais do sistema, sendo que cada uma possui uma função distinta e relaciona-se com as demais via interface definida. A camada *Coleta de Malwares*, por exemplo, é responsável por coletar arquivos binários suspeitos ou maliciosos. Esses arquivos maliciosos ou simplesmente *malwares*

são úteis para a obtenção de mais informações sobre uma sondagem maliciosa, ou seja, sobre as ações que o binário realiza num sistema hospedeiro. Os *bots*, em particular, caracterizam-se por estabelecer uma relação de controle entre a máquina infectada e uma entidade externa de posse do atacante (mais detalhes vide capítulo 3). Logo, os *malwares* classificados como *bots*, podem revelar informações preciosas a respeito do tipo de ataque desempenhado. A disseminação de *malwares*, como já comentado anteriormente, pode ocorre das mais variadas formas: sites maliciosos, e-mails, engenharia social, exploração de vulnerabilidades, entre outras. Sendo assim, essa camada da arquitetura define diferentes componentes para coletar arquivos binários oriundos das diferentes fontes de propagação. Como resultado, é disponibilizado um repositório de arquivos maliciosos que será acessado, via interface criptográfica, pela camada de abstração subsequente.

A camada de *Detecção de Botnets* é a camada mais complexa do sistema. No entanto, a sua principal função é identificar assinaturas típicas de *botnets*. Todo o procedimento de identificação de assinaturas é realizado compondo dados de diferentes fontes de informações externas e internas ao sistema. Inicialmente, os componentes da camada acessam o repositório de *malwares*, disponibilizado pela camada superior, e os submetem para diferentes *sandboxes*. Com a resposta das ferramentas de análise, o sistema desenvolve assinaturas típicas de cada *botnet* que posteriormente serão utilizadas para a identificação de *bots*.

Atuando de forma complementar, a camada de *Rastreamento de Botnets* é responsável por detectar máquinas infectadas num domínio administrativo. O procedimento de identificação utiliza os dados computados na etapa anterior e os mapeia na rede local, mais precisamente, as máquinas infectadas que apresentam o mesmo padrão de comunicação. O mapeamento dos padrões de comunicação pode ser feito de diversas formas e utilizando diferentes ferramentas, como por exemplo, analisadores de tráfego ou exportadores de fluxos de rede. Para isso, cabe aos componentes dessa camada construírem filtros adequados para cada ferramenta de modo a identificar as máquinas comprometidas. Todas as informações mapeadas pelos filtros e ferramentas - *bots* detectados - são armazenadas na base de dados do sistema e posteriormente são utilizadas para uma possível ação mitigatória.

O modelo de camadas, descrito acima, trouxe extensibilidade ao modelo fazendo com que cada abstração possa sofrer mudanças sem afetar o funcionamento da arquitetura como um todo. Para uma visão mais precisa de cada camada, faz-se necessário detalhar as características de cada componente. Por questão de organização, os elementos de cada camada são apresentados seguindo a estrutura de abstração já definida: Camada de Coleta de *Malwares*, Camada de Detecção de *Botnets* e Camada de Rastreamento de *Botnets*.

4.2 Camada de Coleta de *Malwares*

A camada *Coleta de Malware* é responsável por obter os arquivos binários suspeitos e disponibilizá-lo em um repositório. A coleta de *malwares* e uma posterior análise desses arquivos podem revelar informações sensíveis de uma *botnet* que podem auxiliar no processo de identificação das mesmas. A obtenção de arquivos binários tem fundamental importância para o modelo, pois quanto mais arquivos coletados, maior a probabilidade de identificar diferentes assinaturas comportamentais de *botnets*. Considerando esse fato, a solução especifica algumas metodologias para a obtenção de arquivos binários. No entanto, o componente essencial desta camada é o repositório de *malwares* coletados. Os diferentes componentes da camada são ilustrados na figura 4.2.

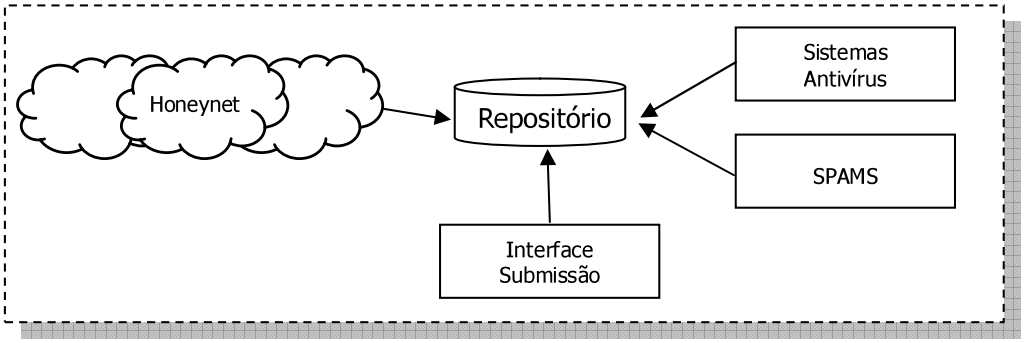


Figura 4.2: Componentes responsáveis pela coleta de arquivos maliciosos.

O componente de coleta **Honeynet** pode ser definido como um conjunto de *honeypots* que emulam serviços de rede com vulnerabilidades. A emulação de serviços vulneráveis, implementados pelos *honeypots*, como já comentado, possibilita que informações detalhadas dos ataques possam ser obtidas, como por exemplo, o *exploit* usado ou ainda o arquivo binário utilizado para comprometer o sistema sondado (BAECHER et al., 2006). Ferramentas como `Honeyd` (PROVOS; HOLZ, 2007) `Nepenthes` automatizam o processo de coleta de informações, permitindo de forma centralizada armazenar dados referentes as sondagens recebidas.

A **Interface para Submissão** constitui um componente que permite a inserção manual de arquivos suspeitos diretamente no sistema. Essa inserção possibilita que usuários credenciados submetam arquivos para análise na arquitetura, ou seja, em busca de informações típicas de uma *botnet*. Do contrário, apenas arquivos coletados de forma automatizada poderiam ser analisados. Para esse propósito, foi definida uma interface Web por onde a submissão de arquivos possa ser realizada. Como resultado, é disponibilizado ao usuário um relatório com todas as funcionalidades encontradas em seus arquivos submetidos.

O componente **SPAMS**, por sua vez, representa uma técnica para coletar arquivos maliciosos no corpo de mensagens de *spams*. Os e-mails indesejados muitas vezes apresentam arquivos maliciosos anexados na mensagem, ou ainda, fornecem uma referência externa (URL) para o mesmo. Criar uma caixa postal de e-mails e analisar o conteúdo das mensagens (URL's, arquivos anexados) é uma técnica eficiente e conhecida pela comunidade de segurança para coletar arquivos binários (CALAIS et al., 2008).

O componente **Sistemas de Antivírus** é responsável por coletar arquivos maliciosos identificados por sistemas de antivírus. Os sistemas de antivírus são úteis para identificar arquivos maliciosos frente as assinaturas pré-existente na sua base de dados. Quando um *malware* é detectado pelo antivírus, o mesmo é eliminado do sistema ou movido para um local específico para quarentena. Sistemas de antivírus corporativos com gerência centralizada, como por exemplo, o `F-secure` (F-Secure, 2010), permitem armazenar arquivos infectados num diretório específico. Desta forma, é possível configurar um sistema de antivírus como mais uma fonte de coleta de arquivos binários maliciosos.

Embora existam diferentes formas para coletar arquivos binários, a arquitetura deste trabalho optou por utilizar formas conhecidas pela comunidade acadêmica e que comprovadamente podem trazer um grande número de *malwares* com relativo esforço de implementação (CERON; GRANVILLE; TAROUCO, 2009). A arquitetura modular permite que novos componentes de coleta sejam agregados e assim aumentando o número de arquivos no sistema. Dessa forma, a composição de todos os componentes coletores culmi-

nam num repositório central de arquivos suspeitos, o qual pode ser acessado pela camada subsequente do sistema, a camada de *Detecção de Botnets*.

4.3 Camada de Detecção de *Botnets*

A camada de detecção de *malwares* pode ser considerada o núcleo do sistema pois é responsável por um grande número de atividades elementares para o funcionamento da solução. A detecção de *botnets*, propriamente dita, é apenas uma das atividades desempenhadas por esta camada. No entanto, outras funcionalidades, tal como análise de *malware*, estatísticas do sistema e coleta de informações sobre *botnets* são de responsabilidade dessa mesma abstração.

É importante relembrar a existência de diferentes metodologias para identificação e mitigação de *botnets*, conforme apresentado no capítulo 2. O presente trabalho, entretanto, utiliza uma técnica que baseia-se em assinaturas, ou ainda no perfil comportamental associado à resposta de um *bot*. Tais assinaturas, ou comportamento dos *bots*, são identificados com base em relatórios de análise de *malware* e por sua vez armazenados numa base de dados local. Além da identificação de padrões comportamentais, os demais componentes atuam no processamento de informações complementares, como por exemplo, filtros para localização de assinaturas das *botnets* encontradas pelo sistema.

Portanto, compreender o funcionamento de cada componente e a inter-relação entre os mesmos é fundamental no processo de entendimento da arquitetura. A figura 4.3 apresenta uma visão geral dos componentes da camada, e na sequência os mesmos são pormenorizados.

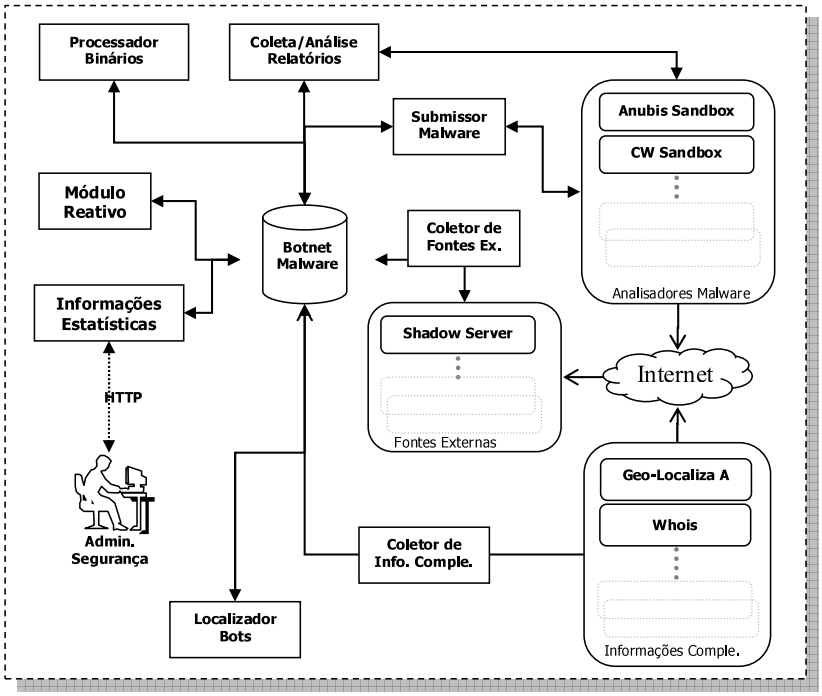


Figura 4.3: Arquitetura da camada de coleta de *malware*.

Todos os componentes representados possuem função específica. No entanto, é possível observar que todos os componentes acessam a base de dados do sistema (*Botnet/-Malware*) de forma direta ou indireta. Por questão de organização, os componentes são descritos segundo o fluxo de execução (ordem que são executados) e por fim a base de

informação é apresentada.

O primeiro componente a ser executado nesta camada é o **Processador de Binários**. A função desse componente é consultar periodicamente o repositório de arquivos suspeitos - arquivos coletados pela camada superior - e armazená-lo na base de dados persistente da arquitetura. Dessa forma, os arquivos coletados por fontes externas - ou até mesmos em outras máquinas - ficam centralizados numa base de dados facilitando o acesso dos outros componentes do sistema aos *malwares*. Adicionalmente, esse componente da arquitetura é responsável por fazer uma verificação em cada arquivo obtido, antes mesmo de inserir na base de dados. Essa verificação consiste em analisar as seguintes características do binário: a) examinar se o arquivo é válido, ou seja, se o mesmo não se encontra corrompido; b) verificar se o arquivo já está presente na base de dados, segundo o seu *hash* criptográfico; e, c) fazer uma pré-classificação no arquivo suspeito utilizando a assinatura de um sistema de antivírus localmente instalado, o que irá facilitar posteriormente a submissão para sistemas de análise de *malware*. Da mesma forma, essa verificação impede que o sistema desperdice recursos de ferramentas externas (analisadores de *malwares*) e também recursos locais (espaço em disco e processamento) evitando a resubimissão dos arquivos obtidos.

O componente **Submissor de *Malware*** tem a finalidade de submeter arquivos suspeitos - já armazenados no banco de dados - para o conjunto de ferramentas e serviços definidos no componente *Analisadores de Malwares*. O processo de submissão propriamente dito pode ocorrer de diferentes formas sendo limitadas às interfaces disponibilizadas pelas ferramentas. Os métodos de submissão mais comuns são baseados em programas específicos (disponibilizados pelas próprias ferramentas), mas também podem ser realizados via *scripts*, interface Web, ou ainda por e-mails. Via de regra, as ferramentas disponibilizam uma interface para submissão massiva (vários *malwares*), a qual pode ser facilmente automatizada.

O componente **Analisadores de *Malware*** especifica um conjunto de ferramentas que analisam arquivos binários de forma automatizada e referem-se basicamente a ferramentas como os *sandboxes* (vide capítulo 2). Essas ferramentas são responsáveis por analisar um arquivo suspeito e, de forma automatizada, registrar o comportamento do arquivo quando em execução no sistema operacional: a) chamadas de sistema, como por exemplo, criação/deleção/acesso de arquivos; b) todo tipo de comunicação gerada pelo *malware* analisado (atividades de rede); c) modificações no registro do sistema; d) acesso a bibliotecas de vínculo dinâmico, entre outras atividades. Tradicionalmente essas ferramentas de análise de arquivos são máquinas virtuais que são preparadas para esse propósito. Algumas ferramentas são disponibilizadas de forma gratuita na rede, tal como as descritas na tabela 4.1. Nesta tabela são listadas as ferramentas e suas respectivas interfaces de submissão de arquivos binários.

As ferramentas (tabela 4.1) diferenciam-se em relação às funcionalidades mapeadas. Algumas ferramentas contemplam um maior número de ações no sistema operacional enquanto outras mapeiam apenas funcionalidades superficiais, tais como criação e remoção de arquivos. Via de regra, as ferramentas comerciais fornecem um grande nível de detalhamento na análise de arquivos, no entanto a versão de avaliação da mesma ferramenta (gratuita) possui limitações. A ferramenta *Norman Sandbox*, por exemplo, é uma ferramenta que apresenta limitações sensíveis na sua versão de avaliação gratuitamente disponível na Web. Por fim, as ferramentas de análise são comprovadamente efetivas para reportar as funcionalidades de *malwares* e também são úteis no processo de identificação de controladores de *botnets* (HOLZ et al., 2008b).

Tabela 4.1: Ferramentas para análise de arquivos binários gratuitamente disponibilizadas na Internet.

Ferramenta	Interface de submissão
Anubis	e-mail, Web, scripts
CWSandbox	e-mail, Web
BitBlaze	Web
Comodo	Web
EUREKA	Web
Joebox	Web, e-mail,scripts
Norman SandBox	Web, e-mail, scripts
ThreatExpert	Web, e-mail, scripts
Xandora	Web

A obtenção da respostas das ferramentas de análise é desempenhada pelo componente **Coleta e Análise de Relatórios**. Além de obter os relatórios dos arquivos submetidos na etapa anterior, esse componente localiza possíveis padrões de comunicação dos *malwares* identificados como *bots*. As seguintes etapas são definidas para o efetivo processamento do componente:

a) Obtenção de relatórios - Cada um dos arquivos submetidos para análise em ferramentas *on-line* produzem como resposta um relatório de funcionalidades. Entretanto, a análise das funcionalidades do arquivo submetido não é imediata porque o processamento de cada arquivo depende das ferramentas e o tempo de resposta pode variar em muitas horas. Logo, o componente é responsável por fazer uma checagem periódica e, assim que o relatório estiver disponível, armazená-lo localmente;

b) Geração de assinaturas - Esta etapa visa identificar padrões de comunicação do canal de controle de uma *botnet*, tendo como base os relatórios das ferramentas de análise. A primeira checagem consiste em verificar se o *malware* analisado possui um controlador, ou seja, se o *malware* em questão é um *bot*. Em caso positivo, o padrão de comunicação é mapeado e armazenado na base de dados. A comunicação com um controlador possui uma série de características. Na listagem 4.1, por exemplo, é ilustrada uma comunicação real observada num arquivo analisado. Nessa listagem, é possível observar características de uma conexão destinada ao endereço IP 83.133.119.206.

```
190.964397  192.168.0.2 83.133.119.206  TCP mtqp > 65520 [PSH, ACK]
   Seq=91 Ack=229 Win=17292 Len=12
```

Listagem 4.1: Informações sobre uma conexão realizada por um *bot*.

A verificação do tráfego de rede permite que mais informações sejam coletadas, como por exemplo, o conteúdo dos datagramas TCP trafegados. O fragmento da listagem 4.2 apresenta o conteúdo de uma conversação realizado por um *bot*, que também foi obtida na análise automatizada do *malware* anteriormente especificado.

```
NICK cucdaseb
USER b020501 . . :-Service Pack 3
JOIN &virtu
:u. PRIVMSG cucdaseb :!get http://updatemania.info/build/setup17.exe
:u. PRIVMSG cucdaseb :!get http://file0129.iwillhavesexygirls.com
   :88/erdown.txt
:u. PRIVMSG cucdaseb :!get http://pozeml.com/oc/box.txt
PING :j.
PONG :j.
JOIN &virtu
PING :j.
PONG :j.
JOIN &virtu
```

Listagem 4.2: Comunicação com o canal de controle de uma *botnet*.

Uma observação mais atenta revela que a comunicação acima utiliza o protocolo IRC, onde comandos de conexão são executados. Logo, as informações da análise de tráfego gerado pelo *malware* auxiliam no desenvolvimento de assinaturas comportamentais que serão posteriormente utilizadas na identificação de *botnets* na rede local.

Embora a análise de *malwares* revele um bom número de controladores de *botnets*, a arquitetura desenvolvida fica limitada à quantidade de *malwares* analisados. Para lidar com essa limitação e aumentar a flexibilidade do sistema, foi desenvolvido o componente **Coletor de Fontes Externas**. Este componente é responsável por buscar em fontes externas padrões comportamentais de controladores de *botnets* já conhecidos. Estas fontes externas podem ser de outros grupos de pesquisas, como o *Emerging Threats* ou até mesmo listas do tipo *blacklists* disponíveis na rede (Shadowserver, 2010). A agregação de novas fontes permite que a base de dados de controladores torne-se mais ampla e consiga mapear um número maior de clientes infectados na rede local. Afinal, com um número maior de assinaturas na base de dados, maior a probabilidade de encontrar clientes infectados.

O componente **Fontes Externas** representa aqui um conjunto de recursos externos utilizados para coletar informações de outras *botnets* detectadas por terceiros. Além de tornarem o sistema mais eficiente, é possível traçar paralelos entre as diferentes fontes de informações externas e até mesmo testar a sua efetividade.

A análise de *malwares* e a importação de assinaturas resultam em assinaturas de *botnets*. Tais assinaturas, em sua maioria, permitem identificar o canal de controle e comando de uma *botnet* específica. As informações presentes na assinatura de uma rede maliciosa, como dados relativos a uma conexão, podem auxiliar no processo mitigatório. Para isso, o componente **Coletor de Informações Complementares** tem a responsabilidade de analisar os dados presentes na assinatura, como endereço IP, e levantar informações relativas. Os serviços acessados pelo **Coletor de Informações Complementares** é definido no componente **Informações Complementares**.

O componente **Informações Complementares** enumera um conjunto de serviços e ferramentas responsáveis por proverem informações complementares sobre controladores de *botnet* mapeados. As fontes de informações definidas pela arquitetura são disponíveis na Internet e podem ser facilmente acessadas via rede. As seguintes classes de serviços *on-line* são definidas:

- *Geo-localização*: as informações de geo-localização visam localizar a latitude, longitude, cidade e país de origem do controlador de *botnet*. Esses dados são relevantes na fase estatística, mais especificamente, na plotagem do mapa global de controla-

dores.

- *Base de informações de roteamento*: informações de roteamento podem ser consultadas em bancos de dados *on-line*. Observatórios como o `CIDR Report` (CIDR, 2010) e do time de segurança `Cymru Team` (Cymru, 2010) são bastante robustos e podem trazer informações mais detalhadas sobre a disponibilidade do controlador.
- *Base Whois*: dados relacionados ao registro do IP podem ser encontrado em bases *Whois*. Por exemplo, informações sobre o contato do responsável, localização e dados de registro podem auxiliar de forma complementar no processo de identificação.

Por questão de consistência, faz-se necessário validar os resultados consultando uma informação em diferentes fontes de dados e verificar se existe alguma discrepância nos resultados. Na ocorrência do mesmo, o componente pode gerar uma exceção ou dar preferência a certa fonte da dados mais confiável.

Por outro lado, o componente **Módulo Reativo** é responsável por impedir as máquinas infectadas (*bots*), detectados pelo sistema, continuem ativas na rede, ou pelo menos não sejam controladas pelos atacantes. Para isto, o componente implementa uma assinatura de mitigação para cada máquina infectada. Essa assinatura de mitigação pode ser desenvolvida para diferentes ferramentas, como por exemplo, regras de filtro de pacotes e/ou assinaturas de sistemas de detecção de intrusão. Como resultado, as assinaturas de mitigação de *botnets* são armazenadas na base de dados da arquitetura permitindo que o gerente de segurança seja munido de possíveis contra-medidas frente as *botnets* encontradas.

De forma complementar, a arquitetura define um componente - denominado **Informações Estatísticas** - que monitora o funcionamento da solução e também ilustra os dados referentes aos processamentos e ações desempenhadas pelo sistema. Esse componente consulta informações na base de dados do sistema e as apresenta para o administrador de segurança por meio de uma interface Web. Devido às características da base de dado, uma grande quantidade de informações pode ser disponibilizada. Desta forma, é possível traçar estatísticas dos arquivos binários coletados, controladores detectados e clientes infectados. A listagem abaixo enumera alguns relatórios que podem facilmente serem obtidos:

- Arquivos binários mais coletados;
- Data com maior número de arquivos binários coletados;
- Tipo de *malware* (assinatura) mais detectado;
- Total de controladores detectados em binários;
- Total de clientes infectados mapeados;
- Data onde foram observados mais clientes infectados;
- Controladores detectados em binários;
- Controladores mais observados em binários coletados;
- Controladores ativos (com acessos de máquinas locais);
- Máquinas infectadas bloqueadas ativas;
- Clientes com maior número de reincidência de infecção.

Além destas expressadas acima, outras informações podem ser derivadas do sistema. Essas informações permitem com que o sistema seja avaliado e também auxiliam na melhoria e no aprimoramento da própria solução.

Outro componente fundamental para arquitetura é o **Localizador de bots**. Esse com-

ponente realiza a consulta dos padrões de comunicação detectados e formata um filtro para que os padrões sejam mapeados na rede local. Esse mapeamento consiste na inspeção de tráfego da rede com o objetivo de localizar máquinas comprometidas que se utilizam do padrão de tráfego previamente traçado. Esse componente atua como um tradutor entre os padrões mapeados e a *Camada Rastreamento de* Botnets. A dinâmica deste componente é discutida com mais detalhes na seção 4.4.

Os componentes descritos acima são essenciais para o funcionamento da arquitetura, mais especificamente para a abstração *Camada de Detecção de* Botnets. Conforme dito anteriormente, essa abstração é peculiar pois apresenta a base de dados do sistema. Na sequência é descrito o modelo de informações definido para armazenar todas as informações da arquitetura proposta. A base de dados é representada pelo componente **Botnet/-Malware** na figura 4.3. O modelo de informações proposto é representado em notação UML na figura 4.4. Esse modelo foi definido por meio de classes que possuem atributos e de relacionamentos com cardinalidade entre as mesmas. O modelo apresentado é formado pelas seguintes classes:

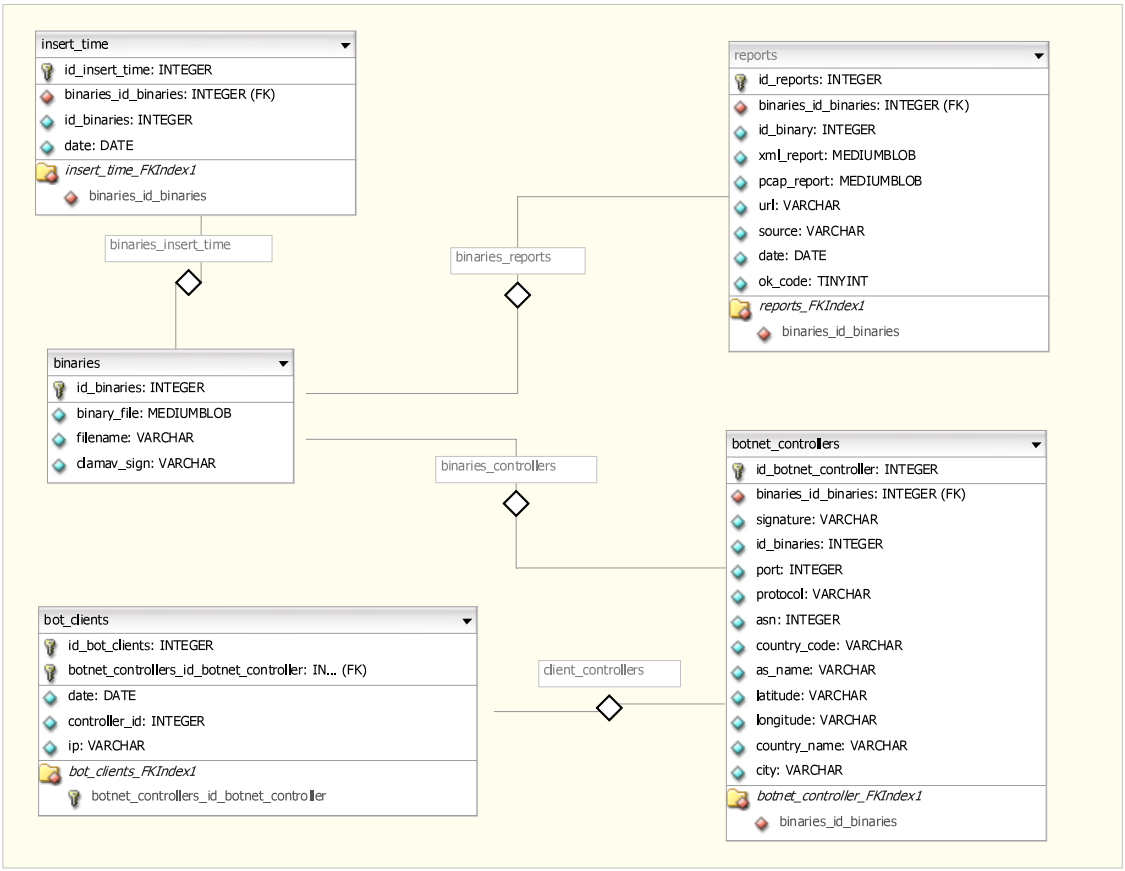


Figura 4.4: Estrutura de informações definidas para a arquitetura proposta.

a) Classe *binaries* - é responsável por armazenar informações referentes aos arquivos binários coletados pelo sistema. Essas informações incluem o próprio arquivo binário em si; nome do arquivo - seu próprio *hash* criptográfico; e uma referência externa que contêm a data na qual o arquivo foi armazenado na base.

b) Classe *insert_time* - é responsável por armazenar informações de data e horário re-

ferente aos arquivos coletados pelo sistema. Por ser um relacionamento externo é possível verificar todas as vezes que um mesmo arquivo - segundo *hash* criptográfico - foi coletado pela arquitetura. Todo o arquivo repetido é armazenado apenas uma vez na base de dados, porém a data e horário é atualizada cada vez que o mesmo é observado na estrutura.

c) Classe *reports* - armazena informações referentes à análise de binários, ou seja, os relatórios disponibilizados pelas ferramentas de análise de arquivos binários. Tais informações descrevem as funcionalidades mapeadas pelo sistema de análise e também um registro com dados de rede no formato *pcap*. Como as ferramentas utilizadas para a análise dos arquivos são *on-line* e não síncronas (o resultado pode demorar para ser disponibilizado), torna-se necessário funcionalmente armazenar uma referência (*URL*) para a instância de processamento. Desta forma é possível verificar a disponibilidade dos relatórios de análise e assim fazer uma cópia de forma persistente.

d) Classe *botnet_controllers* - contém informações relativa os controladores de *botnets* detectados pelos sistemas, segundo análise dos relatórios armazenados na classe *reports*. Além do padrão de comunicação de rede de cada controlador de *botnets*- como por exemplo, protocolo utilizado, endereço IP e porta TCP/UDP - a classe armazena informações complementares do controlador que são utilizadas para apresentar estatísticas dos mesmos. Essas informações estão intrinsecamente relacionadas com o endereço IP do controlador, tal como o nome do sistema autônomo (AS) e dados de geo-localização. Da mesma forma é importante notar que existe uma referência para a classe *binaries*, o que torna possível associar o *malware* com o seu controlador.

e) Classe *bot_clients* - é responsável por manter informações das máquinas infectadas mapeadas na rede local. Essas máquinas infectadas, ou simplesmente *bots*, são mapeadas em fluxos de redes segundo o padrão de comunicação do controlador de rede, previamente armazenado na classe na classe *botnet_controller*. Além da identificação do cliente, endereço IP, também é armazenado a data e hora que o padrão de comunicação foi observado. Do mesmo modo é mantida uma referência ao controlador de *botnet*, ou seja, do padrão de comunicação.

É importante notar a relação estabelecida entre as classes de informações que, de certa forma, possibilita traçar uma visão bastante precisa de um arquivo suspeito. É possível obter as informações referentes à análise dinâmica (arquivo de funcionalidades e padrão comportamental na rede); data e hora que o arquivo binário foi coletado pelo sistema; máquinas infectadas (*bots*) e assinatura do controlador; controlador de rede encontrado num arquivo binário; além de outras informações passíveis de se obter via relacionamento indireto.

A *Camada de Detecção de Botnets* apresenta um grande volume de componentes incluindo a base de informações de toda a arquitetura. Os dados presentes na base de informação da arquitetura também são acessados pela camada responsável pelo rastreamento de máquinas comprometidos.

4.4 Camada de Rastreamento de *Botnets*

O processo de rastreamento de *botnets* visa identificar *bots* na própria rede local. Detectar as máquinas infectadas é o primeiro passo para atenuar os danos causados pelas mesmas. Para isso, essa camada é responsável por observar os padrões comportamentais típicos de cada *botnet* e mapeá-los na rede. A abordagem natural para detectar padrões comportamentais dá-se por meio de ferramentas de análise ou captura de tráfego de rede.

Tais ferramentas permitem que diferentes análises possam ser realizadas na rede para localizar os padrões comportamentais identificados. Sejam elas através da análise de conteúdo dos datagramas ou simplesmente informações do cabeçalho do próprio datagrama.

Ferramentas que utilizam especificações baseadas em conteúdo de datagramas, como por exemplo, sistema de detecção de intrusão conseguem identificar *strings* utilizadas para comunicação com o controlador da *botnet*. A listagem 4.3 ilustra uma assinatura para o IDS *Snort*, na qual busca uma comunicação que apresente a palavra `JOIN` no conteúdo de um datagrama. A palavra *JOIN* faz parte do protocolo IRC é especifica o ingresso a um canal de comunicação. No contexto das *botnets* IRC, isso indica o ingresso de uma máquina no canal IRC do controlador da *botnet* permitindo que a mesma receba comandos remotamente.

```
alert tcp $HOME_NET any -> [195.85.200.11,195.85.200.12]
any (content:'JOIN', msg:" BOTLOG - DROP Known Bot C&C Traffic TCP  -
    BLOCKING";
flags:S; reference:url,www.botlog.org; threshold: type limit, track
by_src, seconds 3600, count 1; classtype:trojan-activity; sid:940006;
    rev:001;
fwsm dst, 30 days;)
```

Listagem 4.3: Regra para detecção de um controlador de rede para o IDS *Snort*.

Por outro lado, ferramentas que analisam informações do cabeçalho conseguem identificar padrões comportamentais, como por exemplo, endereço IP de destino e protocolo utilizado pela comunicação. Ferramentas baseadas em fluxos de rede (*flows*) podem facilmente identificar padrões de comunicação e também dados de cada conexão mapeada pelo sistema coletor (roteador de redes). A listagem abaixo (4.4) descreve uma consulta a base de fluxos previamente armazenada. As primeiras 4 linhas descrevem a sintaxe do comando utilizado para mapear os padrões de rede. No comando, descrito na listagem 4.4, é solicitado para a ferramenta `Nfdump` mapear conexões do arquivo *nfcapd.201002241350*, apresentando informações referentes as conexões de cada IP. A *string* `nfdump filter` descreve o filtro utilizado na listagem (*dst host 200.132.0.131*) selecionando apenas os fluxos com endereço IP destino *200.132.0.131*.

```
1  ** nfdump -M /var/flows/nfsen/profiles-data/live/reitoria:total:vale  -T  -r
2  2010/01/24/nfcapd.201002241350 -n 10 -s ip/flows
3  nfdump filter:
4  dst host 200.132.0.131
5
6  Top 10     IP addr ordered by flows:
7  Date first seen          Duration Proto          IP Addr    Flows  Packets Bytes
8  2010-01-24 13:49:33.829   314.353 any       200.132.X.X       29      134 13103
9  2010-01-24 13:50:54.395   228.669 any        143.54.X.X       20       20  3067
10 2010-01-24 13:53:40.344    27.724 any        143.54.X.X        3       60  5040
11 2010-01-24 13:54:36.154     6.909 any        143.54.X.X        2        2   309
12 2010-01-24 13:52:55.592    15.000 any        143.54.X.X       12       10  1327
13 2010-01-24 13:52:25.592    33.000 any        143.54.X.X       18       11  1227
14 2010-01-24 13:49:33.829   314.353 any        143.54.X.X        2       40  3360
```

Listagem 4.4: Uma consulta por fluxos cujo possuem a assinatura especificada acima.

A listagem obtida como resposta da consulta descrita apresenta informações referentes a protocolo, endereço IP, número de conexões, datagramas, *bytes* entre outros. Esses da-

dos são fundamentalmente importantes para a identificação de máquinas infectadas numa rede local, além de revelar o comportamento realizado pela mesma, como por exemplo, varreduras, envio de *spams*, ou *download* de arquivos.

É importante ressaltar que as ferramentas descritas atuam de forma complementar na tarefa de mapear padrões comportamentais na rede. Dados de uma ferramenta podem ser utilizados como entrada em outra. Por exemplo, a busca de uma *string* num datagrama pode identificar o IP destino do datagrama e esse IP pode ser consultado nos fluxos para identificar todas as conexões realizadas pelo endereço num determinado período de tempo.

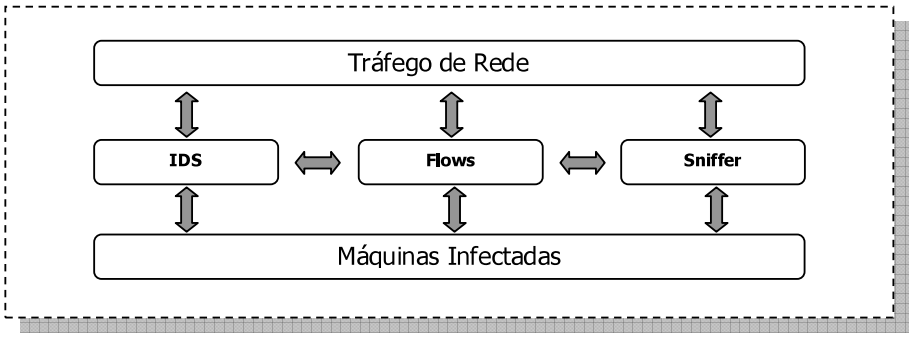


Figura 4.5: Conjunto de ferramentas para mapear padrões de comunicação das *botnet*.

O rastreamento de clientes infectados, portanto, vale-se de um conjunto de ferramentas que são configuradas para localizar os padrões comportamentais previamente encontrados. A configuração é particular a cada ferramenta, mas basicamente são compostas de filtros ou assinaturas, conforme descrito acima. O sistema também possibilita que ferramentas correlacionem os resultados, buscando assim, aumentar a eficiência do sistema (vide figura 4.5). Após a identificação dos *bots*, os mesmos são adicionados na base de informação com o seu respectivo horário de coleta, para que posteriormente sejam elaboradas estatísticas e regras de bloqueio restringindo o acesso a essas máquinas.

Os componentes descritos acima permitiram um melhor entendimento da arquitetura proposta como solução deste trabalho. É importante notar que a arquitetura descrita procurou ser robusta o suficiente para assumir novas variações de acordo com as necessidades das *botnets*, permitindo que componentes sejam adicionados e removidos. Por fim, este capítulo buscou descrever os princípios que nortearam a especificação da solução deste trabalho descrevendo em detalhes os componentes utilizados pelo sistema. No próximo capítulo serão apresentados os detalhes de implementação da arquitetura proposta por este trabalho.

5 IMPLEMENTAÇÃO DO PROTÓTIPO

A partir do modelo de arquitetura, apresentado no capítulo anterior, foi implementado um protótipo, que é descrito neste capítulo. O objetivo é discorrer sobre os detalhes da implementação de cada elemento da arquitetura, apresentando particularidades e desafios encontrados na fase de desenvolvimento. Inicialmente este capítulo faz uma breve discussão sobre as ferramentas desenvolvidas. Na sequência o capítulo é estruturado em seções, segundo as camadas de abstrações já definidas: a) Camada de Coleta de *Malwares*; b) Camada de Detecção de *Botnets*; c) Camada de Rastreamento de *Botnets*. Em cada uma das seções são apresentados detalhes técnicos da implementação da mesma.

De forma geral, a arquitetura proposta por este trabalho exigiu que um conjunto de ferramentas fossem implementadas. Os softwares desenvolvidos por este trabalho, tratam-se na maioria de programas *scripts* utilizando a linguagem Perl, PHP e *Shell Script*. A escolha das linguagens foi baseada simplesmente por questões de familiaridade do autor, as mesmas ferramentas poderiam ser implementadas em outras linguagens sem perdas à arquitetura. No decorrer das seções as particularidades de cada ferramenta são descritas. No entanto, a figura 5.1 representa uma abstração dos principais programas desenvolvidos bem como o relacionamento entre os mesmos. Neste modelo, é possível notar a existência de duas bases de dados - **repositório de binários** e **botnets** -, e os softwares auxiliares **sync.sh**, **submit.php**, **processor.pl**, **stats.php**, **find_bot.pl**, **get_report.pl**, **analize.pl**, **sign.pl**.

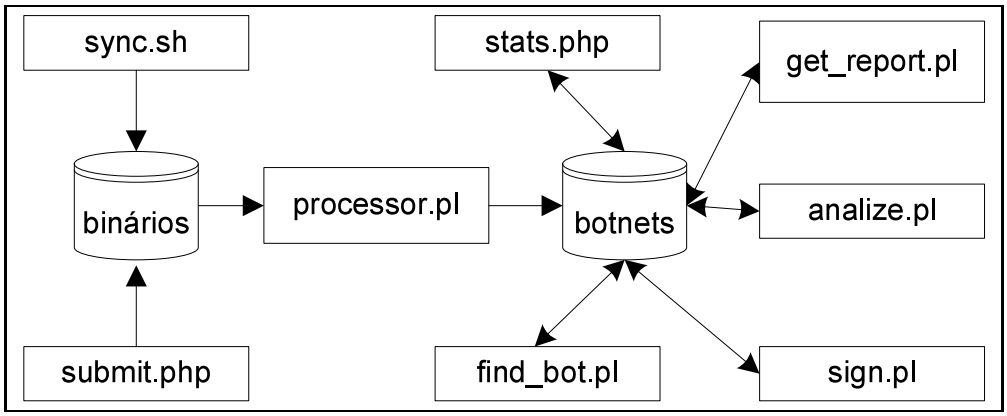


Figura 5.1: Visão Geral: disposição dos programas implementados.

A orquestração desses softwares referenciados permitiram que a solução fosse implementada nos moldes na arquitetura descrita no capítulo 4. O detalhamento dos compo-

nentes são descritos na sequência, tendo em vista a abstração dos mesmos.

5.1 Camada de Coleta de *Malwares*

A *Camada de Coleta de* Malwares, especificada na seção 4.3, define um conjunto de componentes úteis para a obtenção de arquivos suspeitos. Essa implementação, fez o uso de dois componentes: interface para submissão de arquivos via serviço Web, e *honeynet*.

A interface de submissão (ilustrada na figura 5.2) consiste numa página em PHP - submit.php - que implementa um formulário no qual é possível realizar um *upload* do arquivo suspeito e, de forma complementar, inserir uma pequena descrição sobre o binário a ser analisado. Opcionalmente, o usuário pode fornecer um endereço de e-mail e receber informações detalhadas sobre a análise do arquivo submetido. As informações disponibilizadas aos usuários contêm relatórios obtidos pelos *sandboxes* analisados e também a identificação positiva ou negativa de um controlador da *botnet*. Por questões práticas - a já existência de um servidor Web - a interface de submissão foi disponibilizada no *servidor B* conforme figura 6.1.

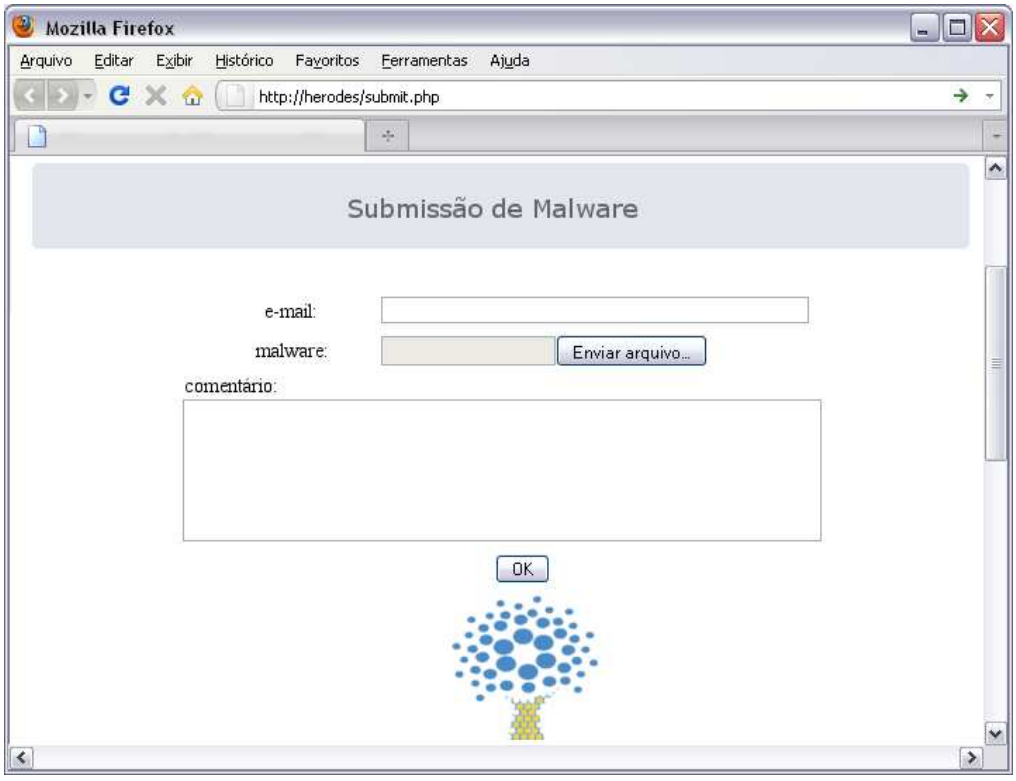


Figura 5.2: Interface de submissão de arquivos binários.

O segundo mecanismo implementado para coleta de arquivos é o componente *honeynet*. A *honeynet* foi implementada num sistema a parte, uma máquina dedicada exclusivamente para emular aplicações vulneráveis, utilizando o software Nepenthes já descrito na seção 3.3. O Nepenthes foi configurado para emular um grande número de vulnerabilidades, descritas na tabela 5.1, e com isto potencializar a quantidade de arquivos maliciosos coletados.

Tabela 5.1: Módulos de vulnerabilidades implementadas pelo *honeypot* Nepenthes (BAECHER et al., 2006).

Módulo	Descrição
vuln-asn1	Vulnerabilidade ASN.1 (MS04-007)
vuln-bagle	*Backdoor* do *worm* Bagle
vuln-dcom	*Buffer Overrun* na interface RPC (MS03-026)
vuln-iis	Vulnerabilidade SSL IIS (MS04-011 e CAN-2004-0120)
vuln-kuang2	Emulação do *backdoor* do *worm* Kuang2
vuln-lsass	Vulnerabilidade LSASS (MS04-011 e CAN-2003-0533)
vuln-msdtc	Vulnerabilidade MSDTC
vuln-msmq	Vulnerabilidade no Message Queuing
vuln-mssql	*Buffer Overrun* no SQL Server 2000 (MS02-039)
vuln-mydoom	*Backdoor* do *worm* myDoom/Novarg
vuln-optix	*Backdoor* do Optix Pro trojan
vuln-pnp	Vulnerabilidade Plug and Play
vuln-sasserftpd	Verme Sasser FTP (OSVDB ID: 6197)
vuln-ssh	Registro de ataques força bruta (SSH)
vuln-sub7	*Backdoor* do *trojan* Sub7
vuln-wins	Vulnerabilidade no WINS (MS04-045)

Por questão de *design*, a ferramenta Nepenthes armazena todos os arquivos coletados para um repositório, no caso, um diretório específico. Os arquivos armazenados são renomeados segundo o seu próprio *hash* criptográfico MD5. A identificação segundo o *hash* criptográfico possibilita que cada arquivo possua uma identificação única em todo o sistema de coleta, garantindo a inexistência de arquivos repetidos no repositório (diretório do *honeypot*). Sendo assim, antes de um novo binário ser armazenado pela ferramenta é verificado se o mesmo já está presente no diretório.

Esse comportamento evita que recursos - principalmente espaço em disco - sejam ocupados desnecessariamente. No entanto, impede que alguns dados sejam observados, como por exemplo, computar a frequência de cada arquivo coletado pelo sistema de coleta. Para atuar com essa limitação - computar arquivos repetidos - optou-se por manter o diretório de coleta da ferramenta (diretório) constantemente vazio. Ou seja, os binários coletados no *honeypot* eram periodicamente transmitidos para uma máquina intermediária para posteriormente serem inseridos na base de dados. Dessa forma, todos os arquivos que chegavam até a ferramenta Nepenthes eram sempre armazenados.

Essa sincronização de repositórios - entre *honeynet* e máquina intermediária - foi efetuada através da ferramenta *rsync*, onde a cada 1 minutos a ferramenta consultava a base de binários do *honeypot* e movia os binários para a máquina intermediária, exemplificado na listagem 5.1.

```
rsync -ave ssh ceron@200.X.X.X:/var/nepenthes/binaries/ /home/honeypot/
   nepenthes/binaries/ --remove-source-files
```

Listagem 5.1: Sincronização de repositórios utilizando a ferramenta *rsync*.

Os dois mecanismos de coleta implementados possibilitam a coleta de diferentes tipos de *malwares*: os arquivos submetidos por usuários credenciados, e arquivos de varreduras

automatizadas. A concatenação dos *malwares* num repositório centralizado foi essencial para a identificação de assinaturas de *botnets*. O processamento dos arquivos coletados é descrito na camada subsequente.

5.2 Camada de Detecção de *Botnets*

A arquitetura da *Camada de Detecção de* Botnets especificada na seção 4.3 define um conjunto de componentes necessários para o processamento de informações dos binários coletados. A referida camada é responsável por um conjunto de atividades elementares ao sistema, como o armazenamento das informações processadas e a geração de assinaturas.

Para o armazenamento de informações referentes aos *malwares* e *botnets* foi utilizado um banco de dados centralizado. A base de dados, caracteriza-se por concentrar e compartilhar as informações processadas pelos demais componentes da arquitetura. Com relação a base de dados, a mesma foi implementada - segundo o modelo de informações descrito na seção 4.3 - utilizando o banco de dados MySQL Server versão 5.1.33. Como o servidor de banco de dados é utilizado por outros componentes da arquitetura, faz-se necessário que o mesmo seja acessível via rede permitindo conexões remotas.

A implementação dos componentes responsáveis pelo processamento de informações se deu por meio de um conjunto de softwares desenvolvidos e executados localmente no *servidor B* (Figura 6.1). O componente *Processador de Binários*, por exemplo, foi implementado pelo *script* em Perl *processor.pl* (Figura 5.1). Esse componente caracteriza-se por inserir os arquivo binários - disponíveis no repositório de arquivos coletados - na base de dados dos sistema. A verificação periódica remove os arquivos do repositório e os armazena no banco de dados na tabela *binaries*, registrando a hora e data de cada arquivo na tabela *insert_time*. Em caso de um arquivo já estar presente na base de dados (tabela *binaries*), apenas as informações de data e horário são atualizadas na base. Esse procedimento possibilitou saber todas os horários que um binário foi inserido na base do sistema, e até mesmo, identificar tendências de certos *malwares*.

```
1  ceron@zarathustra:~$ clamscan -v 0005bdd01ee6441140e393680d4bac2f
2  Scanning 0005bdd01ee6441140e393680d4bac2f
3  bot.exe: Trojan.Poebot-21 FOUND
4
5  ----------- SCAN SUMMARY -----------
6  Known viruses: 720001
7  Engine version: 0.95.3
8  Scanned directories: 0
9  Scanned files: 1
10 Infected files: 1
11 Data scanned: 0.12 MB
12 Data read: 0.12 MB (ratio 1.00:1)
13 Time: 1.367 sec (0 m 1 s)
```

Listagem 5.2: Resultado da varredura do antivírus *clamAV*.

Além de inserir o arquivo binário no sistema e adicionar o horário da coleta, o software também faz uma verificação do arquivo em si. Uma verificação é realizada em duas etapas, cuja a primeira tem o objetivo de descartar arquivos corrompidos ocasionados por algum erro na coleta. Esse procedimento é instanciado pelo próprio *script* utilizando a ferramenta externa *file* disponível na maioria dos sistemas Unix. Já a segunda fase da ve-

rificação visa fazer uma pré-classificação do arquivo binário segundo a assinatura de um sistema de antivírus. Nessa implementação utilizou-se o antivírus `clamAV` que é disponibilizado segundo a licença GPL (Kojm, T., 2010). A listagem 5.2 ilustra a classificação do arquivo *0005bdd01ee6441140e393680d4bac2f*, onde é possível notar inúmeros detalhes da varredura: tempo de processamento para a classificação; tamanho da base de assinaturas; tamanho do arquivo, entre outras. O mais importante aqui é a assinatura do vírus encontrada - *Trojan.Poebot-21* (linha 3) - a qual é armazenada na base de dados (tabela *binaries*).

O componente da arquitetura *Submissor de Malware*, responsável por submeter os arquivos suspeitos para ferramentas de análise *on-line*, foi implementado pelo software *analyze.pl* (Figura 5.1). Esse software é responsável por obter os arquivos binários presentes na base de dados do sistema e submeter para um conjunto de ferramentas. É importante ressaltar que existem diversas ferramentas de análise disponíveis na rede, no entanto esta implementação valeu-se de duas: *CWSandbox* e *Anubis Sandbox*. Essas ferramentas apresentam boa documentação e são extensivamente utilizadas em trabalhos científicos de análise de *malwares* (WURZINGER et al., 2009). O processo de submissão de arquivos pode ser feito de diversas formas, como por exemplo, por interface Web ou por meio de ferramentas disponibilizadas. O software Anubis disponibiliza uma ferramenta própria para a submissão massiva de arquivos, um exemplo de sua utilização é apresentado na listagem 5.3.

```
1 ceron@kilimanjaro:~$ ./submit_to_anubis.py -e ceron@tche.br virus.exe
2 Get the result at
3 http://anubis.iseclab.org/?action=result&
4   task_id=1d45f6fbd52bfa2a449eb514f84890642
5 Successfully submitted 1 analysis subjects.
```

Listagem 5.3: Submissão de arquivos para a ferramentas de análise de *malwares on-line*.

Na referida listagem é possível observar um endereço Web de retorno (linha 3 e 4). Essa URL contém informações sobre a análise do arquivo binário processado pela ferramenta. No entanto, a disponibilidade desse resultado não é imediato, depende do processamento do serviço *sandbox*. Logo, faz-se necessário armazenar essa localização na base de dados (tabela *reports*) para que posteriormente a resposta da análise seja obtida.

O software *analize.pl* utiliza duas interfaces de submissão. Para a submissão ao Anubis Sandbox é utilizada a própria ferramenta disponibilizada (veja listagem 5.3). Já para a ferramenta CWSandbox, a submissão é feita via interface Web, automatizada pelo software *curl* (Stenberg, 2010). Um exemplo é apresentado na listagem 5.4.

```
for i in ‘binaries‘
do
   echo $i
   curl -F "email=ceron@tche.br" -F "upfile=@$i" "http://cwsandbox.
      org/submit.php?action=verify" >> /home/ceron/sandbox.txt
   printf "\n" >> /home/ceron/sandbox.txt
done
```

Listagem 5.4: Submissão de arquivos via interface Web para a ferramenta CWSandbox.

Após o processo de submissão de arquivos para análise, acima descrito, faz-se necessário analisá-los de forma automatizada. O componente da arquitetura *Coleta e Análise de Relatório* busca verificar a disponibilidade dos relatórios e identificar indícios de um canal de controle de *botnet*. Tal componente foi implementado pelo *script get_report.pl* (Figura 5.1) e descreve as funcionalidades do componente em três etapas, conforme pode ser visualizado na figura 5.3: *Obtenção de relatórios, Geração de assinaturas,* e *Processamento de relatório*

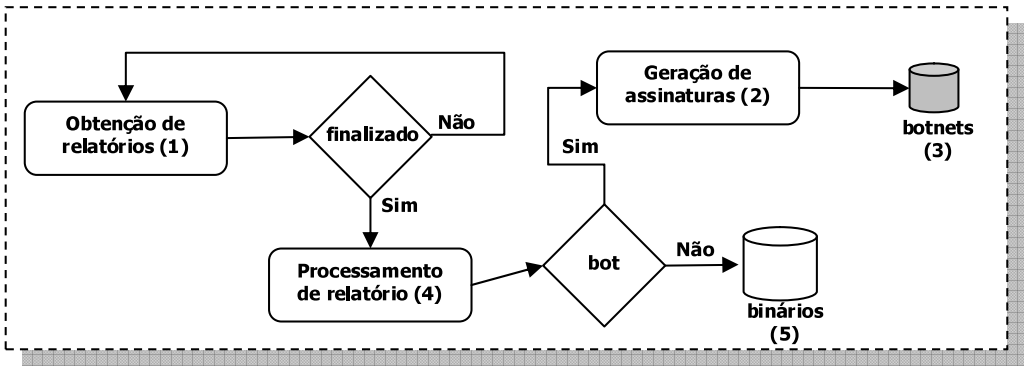


Figura 5.3: Esquemático da coleta e análise de relatórios de funcionalidades de binários.

Obtenção de relatórios - Essa etapa consiste em verificar se o arquivos previamente submetido para análise já foi processado pelo sistema de análise (*sandboxes*). Esse procedimento faz uma verificação periódica - a cada dez minutos - e verifica se o relatório já está disponível na URL especificada (obtida na fase de submissão). Em caso afirmativo, essas informações são salvas em memória e encaminhadas a próxima etapa (Processamento de relatório). Especificamente, cada arquivo analisado produz o seguinte conjunto de recurso:

1- Relatório de funcionalidades do *malware* em formato XML (`CWSandbox`);
2- Relatório de funcionalidades do *malware* em formato XML (`Anubis`);
3- Tráfego de rede gerado pelo *bot* em formato *pcap* (`Anubis`).

Processamento de relatório - Essa etapa busca identificar, tendo como base os recursos acima, os *malwares* que possuem um canal de controle, ou seja, são *bot*. O relatório da ferramenta CWSandbox (item 1) disponibiliza de forma direta informações sobre um possível canal de controle e comando, se o mesmo for identificado na análise. A listagem 5.5 exemplifica o fato.

A listagem 5.5 exibe um fragmento do relatório de uma análise realizada pelo *CWSandbox*. Na linha 10 é possível verificar as conexões de saída realizadas pelo *malware* utilizando o protocolo IRC, com comandos ou instruções típicas de controladores de *botnets* (linha 16). Esse padrão de conexão caracteriza um possível canal de controle. Uma vez que o *malware* seja caracterizado como *bot*, os relatórios são mantidos em memória e encaminhados para a etapa seguinte. Em caso contrário, ou seja, não foi identificado um canal de controle no arquivo analisado, o processamento é interrompido e todos os recursos são armazenados na base de dados. Com isto, os relatórios dos arquivos não *bots* podem ser reprocessados futuramente com diferente heurísticas, sem a necessidade de um reprocessamento do arquivo.

```
<winsock_section>
  <connections_unknown>
     <connection connectionestablished="0" socket="0">
       <gethostbyname requested_host="proxim.ircgalaxy.pl"
  resulting_addr="85.11.143.208"/>
     </connection>
  </connections_unknown>

  <connections_outgoing>
    <connection transportprotocol="TCP" remoteaddr="85.11.143.208"
  remoteport="65520"
  protocol="IRC" connectionestablished="1" socket="1984">
      <irc_data username="d020501" hostname="." servername="."
 realname="-Service" nick="cjrjoyip" non_rfc_conform="1">
 <channel name="&virtu"/>
         <privmsg_deleted value=":* PRIVMSG cjrjoyip :!get
 http://dl2.teenpassage.com/~grander/unpr.exe"/>
      </irc_data>
    </connection>
  </connections_outgoing>
</winsock_section>
```

Listagem 5.5: Análise de um *bot* demonstrando o seu canal de controle.

Geração de assinaturas: Conforme apresentado no capítulo 2 existem diferentes estudos e metodologias para identificar assinaturas de *botnets*. Esse trabalho, entretanto, implementa um método que é embasado na composição de diferentes relatórios obtidos da análise de *malwares* em *sandboxes* virtuais. Alguns *sandboxes*, como o CWSandbox, possuem heurísticas próprias que permitem identificar certas *botnets*. Logo, o que a implementação deste trabalho faz para a geração de assinaturas é utilizar-se destas informações e correlacioná-las. De forma pragmática, este trabalho coleta informações de *botnets* identificadas pelo CWSandbox e verifica se o mesmo comportamento (atividade de rede) é refletida nos relatórios do *sandbox* Anubis. Caso o comportamento seja correlacionado em ambas as análises, uma assinatura é automaticamente gerada e armazenada no sistema. A listagem 5.6 exemplifica uma assinatura gerada dinamicamente segundo heurística implementada.

```
<xml>
<botnet_signature>
  <id>0234</id>
  <remoteaddr>85.11.143.208</remoteaddr>
  <remoteport>65520</remoteport>
  <protocol>TCP</protocol>
  <date>Ter Jul 27 15:57:43 UTC 2010</date>
  <comment>automatically generated</comment>
</botnet_signature>
</xml>
```

Listagem 5.6: Assinatura de uma *botnet* gerada automaticamente com base à análise de relatórios de funcionalidades de *malwares*.

Além do processo de geração de assinaturas recém descrito, outras informações sobre os controladores são manejadas. O componente *Coletor de Informações Complementares*, implementado pelo software *get_report.pl* (Figura 5.1, popula o banco de dados com

informações extras a respeito dos controladores de *botnet* encontrados. Essas informações são relativas ao endereço IP do controlador, como por exemplo: latitude, longitude, sistema autônomo. Para informações de geo-localização foram utilizadas as seguintes bases de informações *on-line*: a) IPinfoDB (IPInfoDB, 2010); b) geoLocation (GeoIP, 2010); c) Cymru Internet Security Research Team (Cymru, 2010). Por questão de precisão, o endereço IP de cada controlador de *botnet* é consultado nas duas primeiras classes de informações geográficas, as quais contem informações mais detalhadas (país, estado e cidade). Caso haja uma incongruência nas respostas, cidades diferentes, por exemplo, uma terceira fonte é consultada (Cymru). Se ainda assim não houver consenso a informação não é armazenada no banco de dados. Para fins de ilustração, a listagem 5.7 apresenta a resposta de uma consulta a ferramenta de geo-localização IPinfoDB.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Response>
3    <Ip>74.125.45.100</Ip>
4    <Status>OK</Status>
5    <CountryCode>BR</CountryCode>
6    <CountryName>Brazil</CountryName>
7    <RegionCode>23</RegionCode>
8    <RegionName>Rio Grande do Sul</RegionName>
9    <City>Porto Alegre</City>
10   <ZipPostalCode></ZipPostalCode>
11   <Latitude>-30.033</Latitude>
12   <Longitude>51.2</Longitude>
13 </Response>
```

Listagem 5.7: Resultado da consulta de informações geográficas a base IPinfoDB.

A importação de assinaturas de *botnets* definidas pelo componente *Coletor de Fontes Externas* foi implementada pelo software *sign.pl* (Figura 5.1). Entretanto, atualmente não se tem conhecimento de uma base de informações (assinaturas) de *botnets* pública. Algo que mais se aproxima de uma lista de assinaturas são as *blacklists* do grupo de pesquisa *Shadow Server*. Esse grupo disponibiliza diariamente uma lista de controladores de rede, por meio do site *Emerging Threats*. (Matt Jonkman, 2010). Logo, esse componente se concentra em obter essa lista diariamente e convertê-las para o formato da arquitetura para que posteriormente as mesmas sejam identificadas na rede. Infelizmente a lista não é muito precisa, contém apenas o endereço IP do controlador, sem informações de porta ou protocolo. Mesmo assim, são informações valiosas para a identificação de *botnets* e são traduzidas para o modelo e assinatura do sistema.

O componente *Estatística* é implementado por meio de uma página Web utilizando a linguagem PHP *stats.php* (Figura 5.1). Essa página faz a indexação de diversos relatórios que dinamicamente são construídos com base nas informações do banco de dados. Além dos relatórios descritivos, foi possível também traçar mapas da localização geográfica dos controladores. Os mapas são criados utilizando a ferramenta `plot-latlong` do `CAIDA` (CAIDA, 2010). Essa ferramenta permite plotar pontos em mapas geométricos tendo como base um par de latitude e longitude. Dessa forma, optou-se por plotar um mapa com informações de controladores mapeados diariamente. O conjunto desses mapas possibilitou a criação de uma animação que serve para entender a dinâmica dos controladores. A Figura 5.4 apresenta a interface Web utilizada para apresentar algumas estatísticas do sistema, já o mapa com a localização dos controladores é apresentada na Figura 6.6.

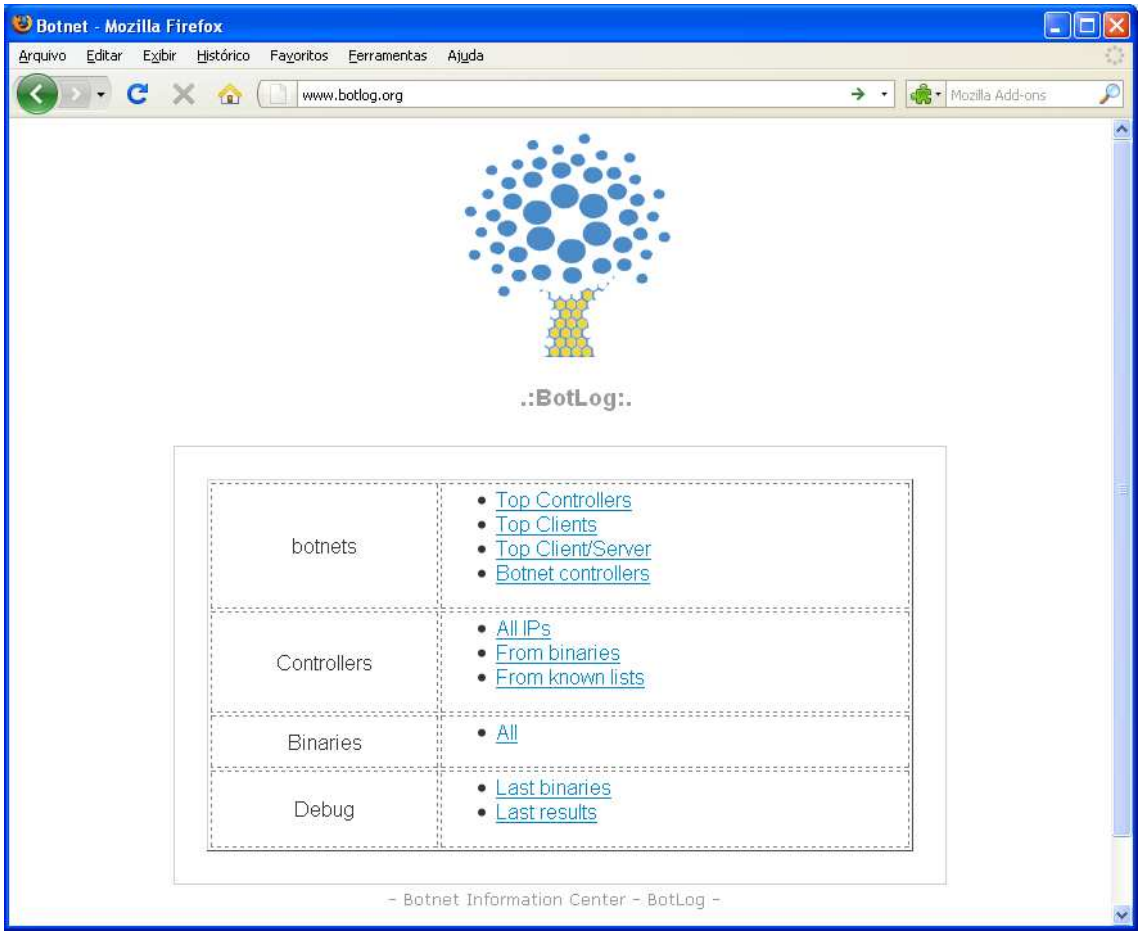


Figura 5.4: Interface de estatísticas.

O componente *Módulo Reativo* implementado por *sign.pl* (Figura 5.1) converte cada assinatura armazenada no banco dos sistema para uma regra de bloqueio. A regra de bloqueio consiste numa série de comandos que podem ser executados na *firewall* para impedir que a máquina suspeita de estar infectada tenha acesso a rede (internamente ou externamente). Por fim, esse comportamento impede que a máquina comprometida possa ser utilizada para propagar *malwares* na rede local ou ser utilizada como intermediária para ataques na Internet. A listagem 5.8 apresenta o formato de uma assinatura de bloqueio para o filtro de pacotes *Iptables*.

```
1  iptables -A security -s 143.54.224.30 -j DROP
2  iptables -A security -d 143.54.224.30 -j DROP
3  iptables -A PREROUTING -s 143.54.224.30 -p tcp -m physdev
4     --physdev-in eth0 -m        tcp --dport 80 -j DNAT
5 --to-destination 143.54.1.38:80
```

Listagem 5.8: Regra para bloqueio de uma máquina comprometida no filtro de pacotes *Iptables*.

A listagem 5.8 ilustra 3 regras ou instruções para o filtro de pacotes. Na primeira linha é feito o bloqueio para todas as conexões oriundas da máquina suspeita 143.54.224.30. Na segunda, o oposto, é bloqueado todas as conexões destinadas a máquina 143.54.224.30. A terceira linha faz um redirecionamento de todas as conexões TCP para uma página de aviso (figura 5.5), a qual informa o usuário o motivo do bloqueio, instruções e informa-

ções para contato. Todo esse procedimento tem como objetivo restringir uma máquina infectada o mais rápido possível, tão logo detectado pelo sistema.

Figura 5.5: Informações disponibilizada para um usuário que possivelmente possua a sua máquina infectada por um *bot*.

O componente *Localizador de Bots* é responsável por obter as assinaturas de *botnets* e mapeá-los para ferramentas de monitoramento. Esse componente atua como uma interface para a *Camada de Rastreamento de Botnet* mapeando as assinaturas da base de dados para as ferramentas de localização de tráfego. A próxima seção descreve com maiores detalhes como é realizado o acesso as assinaturas do sistema.

5.3 Camada de Rastreamento de *Botnets*

A Camada de Rastreamento de *Botnets*, conforme descrita na seção 4.4, é responsável por identificar máquinas comprometidas por *botnets*. Todo esse procedimento é realizado com base as assinaturas de tráfego identificadas pela arquitetura desenvolvida.

Na descrição da arquitetura são exemplificadas diferentes formas e ferramentas para identificar assinaturas de rede analisando o tráfego local. Esta implementação, no entanto, desenvolveu a localização de máquinas através da análise de fluxos de rede. Isso só foi possível por já haver uma estrutura funcional no local da implantação da solução. Sendo assim, todos os padrões comportamentais de tráfego supostamente classificado como *botnet* eram mapeados para a ferramentas de análise de fluxos. Na figura 5.6 é apresentado uma visão geral da arquitetura utilizada para a coleta e análise de fluxos.

Na arquitetura de fluxos utilizada (Figura 5.6) é possível observar quatro elementos fundamentais da estrutura de coleta.

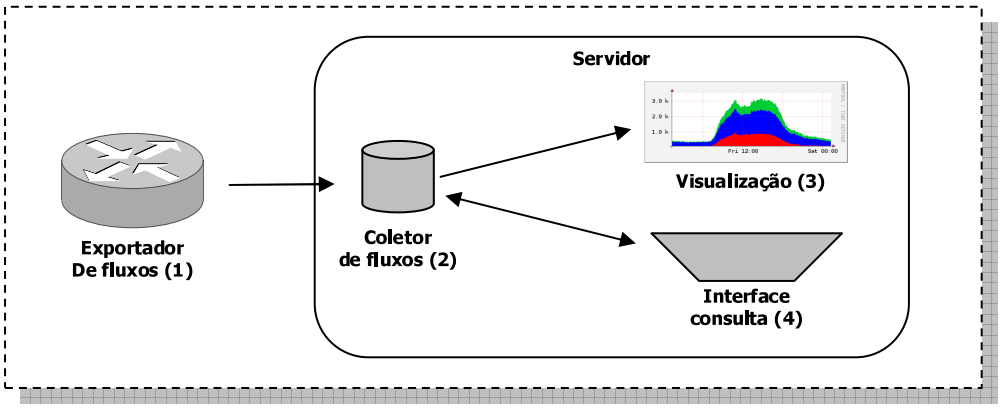


Figura 5.6: Arquitetura de fluxos utilizada na implementação.

O *Exportador de Fluxos* (figura 5.6, item 1) responsável por coletar informações do tráfego da rede foi implementado pelo roteador de borda da universidade, mais especificamente um roteador Cisco Catalyst 6509-E (IOS 12.2(33)S XI2a) com suporte ao protocolo `Netflow` versão 5 ativado. Todos os dados ou fluxos obtidos pelo roteador foram constantemente enviados para o servidor (descrito no capítulo 6.1). Nesse servidor, a entidade *Coletor de fluxos* (Figura 5.6, item 2) tinha como funcionalidade receber os fluxos do roteador e armazenar localmente de forma persistente. O *Coletor de fluxos* foi implementado pela ferramenta *Nfcapd* que permite o recebimento de fluxos via rede através do encapsulamento UDP.

Além do armazenamento de fluxos, o servidor disponibiliza uma interface de visualização de dados (Figura 5.6, item 3). A visualização é implementada pela ferramenta Nfsen, a qual analisa os dados armazenados pelo *Coletor de fluxos* e automaticamente gera gráficos com características temporais da matriz de tráfego de rede. Complementar a análise gráfica, é interessante que uma interface seja disponibilizada na qual possa ser realizada consultas na base de fluxos armazenada. Para isso, utilizou-se o conjunto de ferramentas disponibilizado pelo pacote Nfdump. Por meio do Nfdump é possível consultar informações dos fluxos armazenados de forma dinâmica, aplicando filtros e correlações segundo características desejáveis. A listagem 5.9 exemplifica uma consulta a base de fluxos armazenada utilizando a sintaxe da própria ferramenta Nfdump.

```
nfdump -r nfcapd.2010 "tcp and (src port > 1024 and dst port 80)"
```

Listagem 5.9: Consulta a base de fluxos utilizando a ferramenta Nfdump.

A sintaxe do comando Nfdump acima referida descreve o processamento de um arquivo de fluxos através da aplicação de um filtro. O arquivo analisado (*nfcapd.2010*) é listado segundo o filtro `tcp and ( src port > 1024 and dst port 80 )`. O filtro, que segue a sintaxe *pcap*, identifica fluxos TCP com porta origem acima de 1024 e com porta destino 80. Além de consultas simples a interface permite consultas mais complexas que podem auxiliar no processo de rastreamento de tráfego suspeito numa rede monitorada.

Essa interface disponibilizada pelo Nfdump permitiu a identificação de assinaturas de *botnets* desenvolvidas pela arquitetura proposta. Para isto, foi necessário converter as assinaturas na base de dados (listagem 5.6) para a sintaxe do Nfdump. Após a criação de filtros uma consulta a base de fluxos era realizada. Como resultado eram listadas todas as

conexões que apresentavam o padrão requisitado, ou seja, da assinatura da *botnet*. Esse rastreamento por assinaturas foi realizado diariamente nos fluxos com todas as *assinaturas vigentes* na base de dados.

O conceito *assinatura vigente* é definido por esse trabalho como assinaturas de *botnets* possivelmente ativas. Identificar o tempo de vida de *botnet* é uma tarefa delicada. Por exemplo, quando se identifica uma assinatura tal como acesso a um endereço comprometido não se sabe se o mesmo continua comprometido. As assinaturas da arquitetura possuem diferentes características, dependendo da fonte de onde as mesmas foram constituídas. Para as assinaturas importadas diariamente, o próprio grupo de pesquisa (Shadow Server) se encarrega de atualizar e remover assinaturas não mais ativas. Já para as assinaturas identificadas na análise de *malwares* torna-se necessário estipular um tempo de vida, já que não se pode testar se a *botnet* identificada continua ativa.

Sendo assim, a arquitetura definiu a seguinte lógica para determinar uma assinatura ativa: as assinaturas importadas são válidas somente no dia da importação, afinal as mesmas são atualizadas diariamente. Já as assinaturas de *malwares* são consideradas ativas numa janela de tempo definida em 2 dias. Logo, todo acesso a uma assinatura de *malware* é válida acessos do dia anterior e no dia da coleta (identificação). Sendo assim, todos os padrões vigentes são consultados na base de fluxos diariamente numa janela temporal que pode variar de 24 a 48 horas, conforme descrito acima. A identificação de acessos define uma máquina como comprometida.

Devido ao grande volume de assinaturas a serem consultadas na base de fluxos - via interface Nfdump - foi necessário limitar a consulta em 10 assinaturas por requisição. Esse fracionamento foi necessário devido ao tempo de consulta, que devido ao grande número de informações sobrecarrega o servidor de fluxos. A utilização de filtros menores implica em realizar diversas consultas a base de dado, no entanto isso não se torna um problema, afinal a consulta não é onerosa em relação a consumo de banda. A listagem 5.10 ilustra o processo de consulta por assinaturas realizado diariamente pela arquitetura em busca de máquinas comprometidas.

```
ssh ceron@flows.ufrgs.br '/usr/local/scripts/controllers_reports.
   perl daily "dst ip 61.195.154.6 or dst ip 67.43.236.66 or dst ip
    211.179.172.219 or dst ip 218.61.22.10 or dst ip 67.43.232.36
   or dst ip 92.237.69.206 or dst ip 67.43.236.67 or dst ip
   210.112.170.142 or dst ip 83.68.16.6 or dst ip 190.34.152.18"|
   sort | uniq '
```

Listagem 5.10: Filtro para a ferramenta *Nfdump* automaticamente gerado para rastreamento de acesso a possíveis controladores.

É importante notar que o procedimento de consulta aos padrões é realizado no servidor remoto valendo-se de uma interface criptográfica. Esse comando é gerado automaticamente pelo componente *Localizador de Bots* e implementado pelo software *find_bot.pl* (figura 5.1). O próprio software *find_bot.pl* identifica o tipo de assinatura na base de dados (assinaturas geradas com base na análise de *malware* ou importada) e especifica a janela de análise ou seja a vigência da assinatura.

Na listagem 5.10 é apresentado o *script /usr/local/scripts/controllers_reports.perl* que é executado na servidor remoto (servidor onde os fluxos são armazenados). Esse *script* é responsável por calcular a janela de tempo tendo em base o horário de consulta e formatar a resposta da consulta. O resultado da execução do comando realizado segundo a interface descrita são automaticamente armazenado na base de dados.

Essas informações são compostas pela data da identificação, endereço do controlador e cliente possivelmente infectado. A listagem 5.11 apresenta uma visão de como tudo isto é armazenado na base de dados, mais especificamente na tabela *bot_client*. Note que os dados foram anonimizados.

```
mysql> select * from bot_client limit 10;
+------------+--------------+--------------+
| data       | controller    | client      |
+------------+--------------+--------------+
| 2009-09-18 | 66.198.80.67 | 143.54.X.36  |
| 2009-09-18 | 66.198.80.67 | 143.54.X.34  |
| 2009-09-18 | 66.198.80.67 | 143.54.X.31  |
| 2009-09-18 | 66.198.80.67 | 143.54.X.30  |
| 2009-09-18 | 66.198.80.67 | 143.54.X.3   |
| 2009-09-18 | 66.198.80.67 | 143.54.X.25  |
| 2009-09-18 | 66.198.80.67 | 143.54.X.2   |
| 2009-09-18 | 66.198.80.67 | 143.54.X.13  |
| 2009-09-18 | 66.198.80.67 | 143.54.X.125 |
| 2009-09-18 | 66.198.80.67 | 143.54.X.120 |
+------------+--------------+--------------+
10 rows in set (0.84 sec)
```

Listagem 5.11: Visão da tabela *bot_clients* populada pelo componente de rastreamento.

Por fim, os dados descritos acima possibilitam identificar as máquinas infectadas na rede através da monitoração de fluxos. Dessa forma é possível construir uma base de informações robusta o suficiente para melhor entender as ameaças representadas pelas *botnets*. A descrição dos resultados derivados da implementação são discutidos no próximo capítulo.

6 AVALIAÇÃO EXPERIMENTAL

Neste capítulo é apresentada a avaliação da solução proposta quando aplicada a detecção e mitigação de *botnets*. Para isso, utilizou-se da implementação descrita no capítulo 5 para coletar informações numa rede de produção. Os dados obtidos com essa implantação são discutidos no decorrer do capítulo e servirão de base para avaliar a eficiência do sistema. Este capítulo está estruturado da seguinte maneira: na seção 6.1 é descrito o cenário de implantação onde a arquitetura foi implementada; na seção 6.2 é realizado uma avaliação funcional dos *malwares* coletados pelo sistema; e na seção 6.3 é feito uma discussão relativa as *botnets* mapeadas pela própria arquitetura. Por fim, o capítulo é encerrado relatando as considerações finais sobre o experimento.

6.1 Cenário de Implantação

A arquitetura desenvolvida por este trabalho foi implementada utilizando um conjunto de servidores dedicados. Dessa forma, a implementação das camadas de abstrações da arquitetura pôde ser distribuída entre os servidores observando critérios de eficiência e otimização de recursos disponíveis. A figura 6.1 apresenta o cenários utilizado, e na sequência o mesmo é descrito.

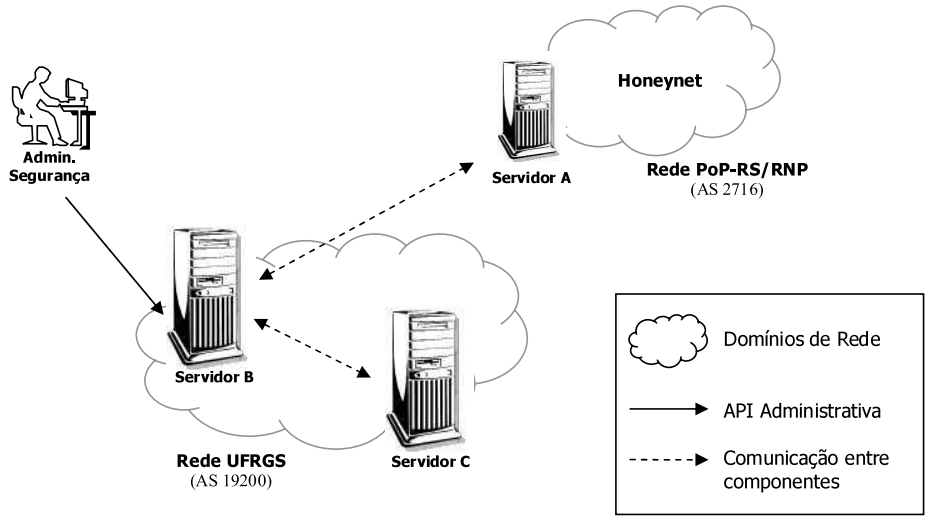


Figura 6.1: Cenário de implementação.

No cenário (figura 6.1) são apresentados três servidores utilizados para avaliação da solução. É importante notar que os servidores estão localizados em domínios adminis-

trativos diferentes: o *servidor A* está sediado na Rede Nacional de Pesquisa (Sistema Autônomo 2716); já o *servidor B* e *servidor C*, localizam-se na rede da Universidade Federal do Rio Grande do Sul (Sistema Autônomo 19200). Além dos servidores, o cenário apresenta um Administrador de Segurança o qual interage com o sistema disparando ações por meio de uma interface HTML já descrita.

Com relação as características dos servidores, nota-se uma certa heterogeneidade entre a configuração das máquinas. Na sequência, portanto, as características de cada servidor são apresentadas:

servidor A: Sistema operacional OpenBSD 4.1, utilizando software Nepenthes para emular 254 máquinas virtuais vulneráveis. Como recurso físico a máquina conta com 40 Gigabyte de disco rígido, memória 1G e processador AMD Athlon(tm) XP 2000+. O *hardware* se mostrou mais do que suficiente, já que esta máquina é apenas responsável pela coleta dos dados, uma vez que os arquivos coletados são removidos periodicamente (veja seção 5.1).

servidor B: Sistema operacional FreeBSD 7.1, servidor de banco de dados Mysql Server 5.1.33, servidor Web Apache 2.1 com suporte a linguagem PHP (versão 5). A máquina conta com 40 Gigabytes de disco rígido, memória 2G e processador Intel(R) Xeon(TM) MP CPU 2.00GHz. Durante a implementação constatou-se que o *hardware* da máquina foi muito modesto, causando lentidão no processamento de atividades e falta de espaço de armazenamento.

servidor C: Sistema operacional Linux Kernel 2.6.8, utilizando o coletor de fluxos Nfcapd e Nfdump como ferramenta de processamento e fluxos. A máquina conta com 180 Gigabytes de disco rígido, memória 4G e processador Intel(R) Xeon(TM) CPU 3.20GHz. Dada a capacidade do disco foi possível armazenar informações de fluxos referente aos últimos dois meses. Os fluxos de borda exportados da rede UFRGS - em média 200 Mbits de tráfego por segundo - equivalem a aproximadamente 2.4 Gigabytes por dia de armazenamento no formato Nfdump.

Durante o experimento alguns problemas foram encontrados. O principal deles foi o tamanho do banco de dados, que por armazenar os arquivos binários tomou proporções não estimadas. O tamanho do banco de dados ficou em torno de 15 Gigabytes, o que forçou a substituição do *servidor B* para uma máquina mais robusta (configuração descrita acima). Outro problema encontrado foi o alto custo computacional para processar informações armazenadas na base de dados. Por apresentar muitas entradas nas tabelas de dados, o processamento de estatísticas desenvolvido em PHP, em particular era bastante demorado. Como melhoria, o servidor Web idealmente deveria ser deslocado para uma máquina dedicada liberando mais recursos para o processamento de estatísticas. No entanto, por falta de recursos físicos isso não foi possível nesta implementação.

Na sequência são discutidos os resultados coletados pela arquitetura implementada no cenário descrito acima (figura 6.1). Toda a avaliação dos resultados tem como base o período entre 18 de Agosto de 2009 a 20 de Março de 2010, período no qual a solução foi utilizada na rede da Universidade Federal do Rio Grande do Sul.

6.2 Avaliação dos *Malwares*

Os arquivos maliciosos, como já comentado anteriormente, foram obtidos pelos componentes implementados na *Camada de Coleta* da arquitetura. Embora a implementação da arquitetura tenha desenvolvido uma interface Web para submissão de *malwares*, o

grande volume de arquivos maliciosos coletados refere-se a arquivos coletados via *honeynet* (componente *Honeynet*).

O *servidor A*, responsável pela coleta de arquivos binários, sofreu uma grande quantidade de sondagens ao sistema emulado (*honeynet*). Boa parte dos ataques identificados foi ocasionada por varreduras automatizadas ou vermes que se propagam de forma autonômica em busca de aplicações vulneráveis. Ataques mais elaborados (intervenção humana) não foram observados, até mesmo porque o tipo de *honeypot* utilizados - *honeypot* de baixa interatividade - não contempla esse tipo de ataque, afinal os serviços são emulados de forma limitada.

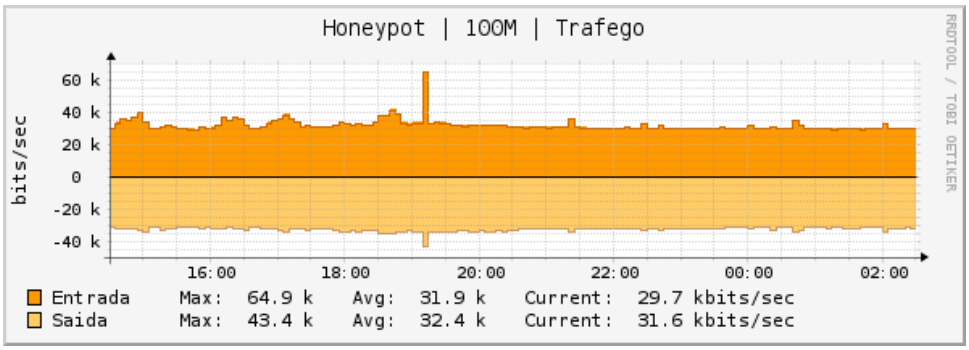


Figura 6.2: Volume de tráfego observado no *honeypot* (servidor A).

A figura 6.2 apresenta informações sobre o volume de tráfego de rede (em *bits*) observado no servidor de coleta (*servidor A*). É importante notar que todo o tráfego recebido, representado na figura pela legenda `Entrada`, corresponde ao tráfego malicioso coletado. Os dados identificado por `Saída` representa o tráfego dos ataques - respostas do servidor - mas também o tráfego ocasionado pelo *script* de sincronização de repositórios. Isto justifica o aspecto simétrico do gráfico: para cada binário coletado (Entrada) o mesmo é transferido para o *servidor B* (Saída). Esse volume de tráfego coletado resultou na coleta de 64.169 arquivos maliciosos, sendo 61521 arquivos únicos segundo o seu *hash* criptográfico MD5. Com uma análise mais profunda em relação à natureza dos *malwares*, foi possível identificar diferentes famílias de *malwares*, num total de 45 classificações únicas (sumarizadas na tabela 6.1).

A tabela 6.1 apresenta as 12 famílias de *malwares* mais detectadas tendo como base assinatura do sistema de antivírus *clamAV* (Kojm, T., 2010). As assinaturas mais frequentes correspondem a *malwares* polimórficos, isto é, *malwares* que modificam o seu conteúdo a cada replicação (via encriptação ou encapsulamento). O *Worm.Allaple* caracteriza-se por se auto-replicar no sistema infectado e propagar-se por meio de arquivos HTML. Além da replicação, o Allaple implementa a funcionalidade de negação de serviço (DoS) do tipo SYN Flood (WILLEMS; HOLZ; FREILING, 2007). Também da família dos *malwares* polimórficos, o *worm W32.Virut* possui um comportamento diferente. Esse *malware* utiliza a técnica chamada *Entry Point Obscuring* para infectar a área de memória de arquivos com extensões .exe (executáveis) e .scr (*screensaver*) (SZöR; FERRIE, 2001).

A terceira classificação mais encontrada refere-se aos *malwares* sem assinaturas, representados pela palavra *OK*. Isso significa que o antivírus não encontrou nenhuma assinatura para o arquivo no momento da sua análise (mais especificamente, no momento que o *malware* é inserido na base de dados do sistema). A ausência de assinaturas pode representar que o *malware* ainda não possui vacina conhecida, ou ainda, trata-se de um *malware* que explora vulnerabilidades do tipo *zero day* (COOKE; MAO; AND, 2006).

Tabela 6.1: Assinaturas mais frequentes dos *malwares* coletados.

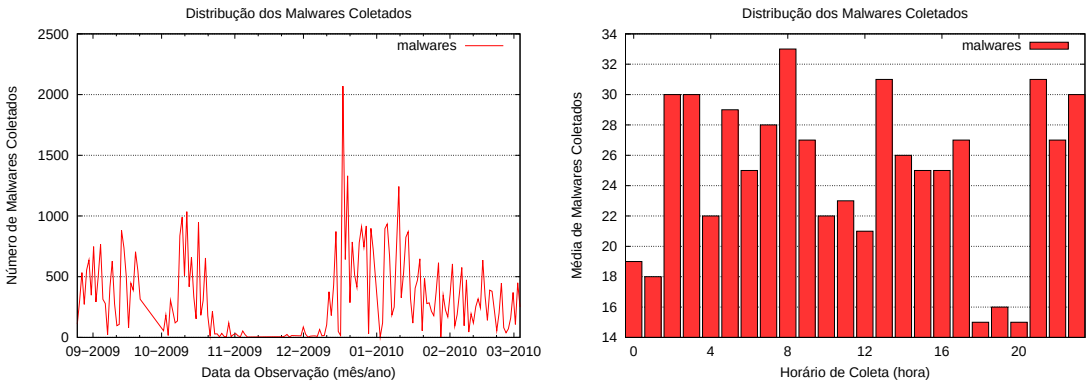
Quantidade	Assinatura de *malwares*.
59886	Worm.Allaple
3753	W32.Virut
266	OK
59	Trojan.VanBot
54	Worm.Vesser.A
27	Trojan.SdBot
23	Trojan.Small
15	Trojan.Agent
12	Worm.Palevo
12	Trojan.KillAV
10	Worm.Sasser.B
10	Trojan.Buzus

Para entender melhor esse tipo de *malware* não classificado, buscou-se reanalisar todos os *malwares* não identificados pelo antivírus no momento da coleta. Para isso, todos os *malwares* com assinatura *OK* foram reanalisados após 30 dias de coleta.

Como resultado dessa análise, 52% dos *malwares* foram classificados nas mais diversas famílias de *malwares*. Isso significa, que mesmo após 30 dias do *malware* ser coletado, quase a metade *malwares* dos não identificados pelo antivírus continuam sem identificação.

As demais famílias de *malwares* apresentadas na tabela 6.1 são observadas em menor frequência. Como pode ser observado, existe uma grande variedade de família de *malwares* sendo, na sua maioria, diferentes tipos de *worms* e *trojans*.

Além da avaliação das famílias dos *malwares* coletados, a avaliação de *malwares* buscou analisar a incidência dos arquivos coletados no sistema. Isso é, se existe uma padrão de propagação de *malwares* observados na rede monitorada.



(a) Distribuição segundo data.

(b) Distribuição segundo horário.

Figura 6.3: *Malwares* coletados na fase de avaliação experimental.

Na figura 6.3(a) é possível observar o número total de binários coletados pelo sistema segundo sua distribuição temporal. Fica evidente na figura uma oscilação no número de

coleta de arquivos, no entanto o número médio de binários é relativamente estável sendo 364. No gráfico em questão tivemos exceções nos meses de Novembro e Dezembro onde o sistema teve problemas físicos de *hardware* (disco) e não pode coletar arquivos binários. Outra exceção foi observado em Dezembro, onde um número maior de novos *malwares* foi coletado pela arquitetura. Curiosamente em Dezembro, a Microsoft publicou 6 boletins de segurança que abordam ao todo 12 vulnerabilidades em produtos da empresa. A exploração destas vulnerabilidades permite desde a execução remota de código até a negação de serviço (DoS) (Microsoft, 2010). Logo, pressupõem-se que o aumento no volume de *malwares* tem uma relação direta com as novas vulnerabilidades encontradas. Novas vulnerabilidades fazem com que os atacantes desenvolvam novos binários incorporando explorações para as novas vulnerabilidades observadas.

Por outro lado, no gráfico representado na figura 6.3(b) é ilustrado a distribuição média dos *malwares* segundo o horário de coleta (GMT-3). Através do mesmo, é possível constatar um volume menor de binários coletados no período noturno (por volta das 20 horas). A avaliação do experimento não pode precisar o motivo desse comportamento, no entanto outros trabalhos que analisam coleta de *malwares* tiveram resultados diferenciados (BACHER et al., 2005). Uma possível explicação desse comportamento pode ser a característica da rede onde a *honeynet* foi emulada. Por ser uma rede acadêmica, a maior quantidade de tráfego é observada em períodos diurnos. Período esse onde um maior número de máquinas estão ligadas e, se infectadas, realizando varreduras e propagando *malwares*.

Os *bots* por sua vez, correspondem a uma categoria específica de *malwares*, como já descrito anteriormente. No processo de classificação dos *malwares* foram identificados um conjunto expressivo de *bots*. As próximas seções apresentaram com maior detalhamento uma avaliação expressiva dos *bots* e suas respectivas redes maliciosas.

6.3 Avaliação das *Botnets*

Todos os arquivos coletados foram submetidos para análise funcional, com exceção das variações polimórficas. Os *malwares* polimórficos ou metamórficos não apresentam diferenças significativas em relação as funcionalidades, tanto é que alguns analisadores de *malwares* não processam tais arquivos, como é o caso do *CWSandbox*. A análise dos relatório de arquivos *pcap*, metodologia descrita no capítulo 5, possibilitou a identificação de 428 arquivos maliciosos do tipo *bots*. Embora o número pareça ser pequeno, é uma quantidade considerável, visto que a maioria dos *malwares* coletados são oriundos da *honeynet* ou seja, de 254 *hosts* emulando um pequeno conjunto de aplicações. Na sequência são apresentados os resultados das *botnets* identificadas. Por questão de organização, inicialmente é descrito informações a respeito dos controladores de *botnets*, e na sequência, dados relativos as máquinas comprometidas identificadas.

6.3.1 Controladores de *Botnets*

Como já comentado, os controladores identificados pela arquitetura desenvolvida é resultado de análise de arquivos binários coletados e também da importação de outras fontes de informação, nesse caso a lista Shadow Server obtida via Emerging Threats.

Os controladores obtidos via análise de *malwares* apresentaram um conjunto de informações detalhadas, fato esse não encontrado em listas de controladores importadas ao sistema, as quais disponibilizam um conjunto limitado de informações sobre os controladores. Na heurística desenvolvida, por outro lado, foi possível determinar o endereço IP

do controlador, protocolo, e também a porta utilizada para a conexão.

Na sequência é feita uma avaliação desses controladores - obtidos através da análise de *malwares* - tendo em vista características de conexões. Na tabela 6.2 são sumarizados os controladores mais frequentes identificados nos *malwares* analisados. É importante notar que o último octeto do IP foi anonimizado.

Tabela 6.2: Controladores mais frequentes dos *malwares* coletados.

Acessos	Possível controlador de *botnet*
392	83.68.16.X
7	83.68.16.X
6	125.214.65.X
5	74.63.78.X
4	69.65.19.X
4	69.64.147.X
4	141.152.124.X
2	128.130.56.X
2	103.72.47.X
2	103.72.47.X

Como pode ser observado na tabela 6.2, existe um controlador (83.68.16.X) que se destaca quanto ao número de acessos. Provavelmente esse IP, corresponde a um *pool* de endereços, uma espécie de balanceador de carga que faz a distribuição de acessos entre outros controladores. Outra possibilidade é que esse endereço pertença a um servidor dedicado apenas para controlar *botnet* utilizado em sistemas de locação em *data-centers*. Essa constatação só tem a colaborar com a tese observada pelos autores do trabalho (SORANAMAGESWARI; MEENA, 2010), no qual os autores identificam criminosos que constituem redes (até mesmo sistemas autônomos) exclusivamente para realizar atividades ilícitas.

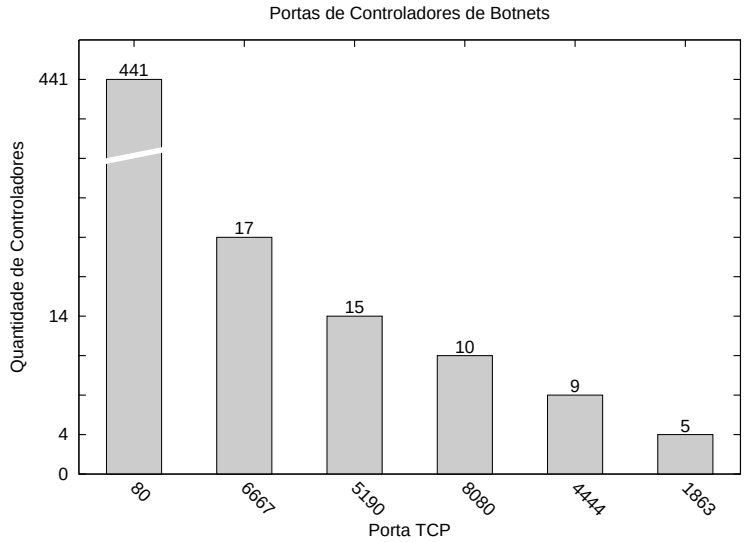


Figura 6.4: Portas mais utilizadas pelos controladores de *botnets*.

Ainda a respeito dos controladores identificados pela análise de *malwares*, foi feito

um estudo em relação as portas mais utilizadas nos controladores das redes maliciosas. A figura 6.4 apresenta um gráfico com as portas mais frequentes encontrada pela análise nos *sandboxes*.

Como representado, a porta mais frequente nos *bots* analisados é a porta 80/TCP, seguidas pelas portas 6667/TCP, 5190/TCP, 8080/TCP, 4444/TCP, 1863/TCP respectivamente. A porta 80/TCP, em particular, é utilizada pela grande parte dos controladores para estabelecer o canal de comunicação da *botnet*. Por ser uma porta utilizada para comunicação Web, a porta 80/TCP tipicamente não contem filtros ou restrição de acesso externo. Dessa forma, o envio e o recebimento de informações entre o controlador da rede e a máquina comprometida ficam facilitados.

De forma complementar, foi realizado uma avaliação semelhante nos controladores importados através de fontes externas. Como a lista de controladores importada pelo sistema - disponibilizado pelo grupo Shadow Server - apresentava apenas o endereço de controladores de *botnets*, a avaliação das portas utilizadas foi suprimida. Mesmo assim, foi possível identificar informações relevantes no contexto do trabalho. Foram importados para o sistema um total de 310244 controladores de rede, que logo após resultaram em assinaturas de detecção de *botnets*. Nesse montante, foi observada uma média de 2234 controladores diários, com desvio padrão de 809.87. Uma melhor visualização das informações é ilustrada no gráfico da figura 6.5.

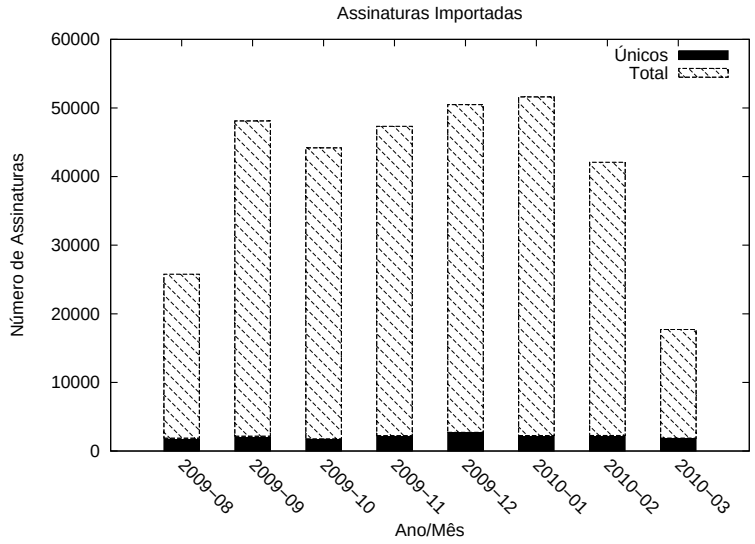


Figura 6.5: Assinaturas importadas pela arquitetura via *Emerging Threats*.

Na figura 6.5 nota-se uma estabilidade no número de controladores importados. Com exceção dos meses de Agosto (2009-08) e Março (2010-03), onde os dados coletados representam um mês incompleto devido ao período de avaliação, apresentado anteriormente (seção 6.1). Nota-se uma grande quantidade de IPs importados, no entanto esse número diminui expressivamente quando se exclui os IPs repetidos - no total 6% (média) correspondem a IPs únicos. Esse fato ilustra uma grande diversidade de IPs utilizados como controladores de *botnets*, isto é, existem diferentes máquinas sendo utilizadas para controlar redes maliciosas. Situação diferente da encontrada pela análise de *malwares*, onde a maioria dos *bots* utiliza um pequeno número de endereços (controladores).

Outra análise realizada diz respeito ao tempo que um controlador permanece ativo.

Ou seja, buscou-se verificar o número de dias consecutivos que um controlador permanecia presente nas listas diárias importadas para o sistema. A presença de um controlador dia após dia significa que o servidor (IP) continua comprometido e sendo utilizado para atividades maliciosas. Sendo assim, após uma mineração das informações constatou-se que: a) do total de 310244 IPs, obtidos no período da avaliação, 1.5% ficaram ativos por mais de 1 dia consecutivo; b) o número médio de dias que esses controladores ficaram ativos de forma consecutiva é aproximadamente 56 dias. Essas informações sugerem que os controladores de *botnets* - quando detectados - são rapidamente desinfetados pelos seus responsáveis. Apenas 1.5% desses IPs continuam infectados por mais de 1 dia seguido. No entanto essas máquinas parecem estar abandonadas na rede e ficam por um longo período infectado (média de 56 dias).

Após ter apresentado uma avaliação sobre os controladores oriundos das diferentes fontes (análise de *malwares* e importação), na sequência, é levantado informações tendo como base todos os dados sumarizados no sistema.

A compilação de todos os IPs dos controladores obtidos, tornou possível identificar - ou ao menos estimar - a localização geográfica dos controladores de *botnets*. Com isso, deseja-se identificar onde a maioria dos atacantes (*botmasters*) sediam suas máquinas para controlar uma rede maliciosa de máquinas comprometidas. Para isso, foi feito o uso de ferramentas de geo-localização *online* que, com base a um IP, disponibilizam uma localização geográfica aproximada. A figura 6.6 apresenta a dispersão geográfica dos controladores de *botnet* detectados através das assinaturas geradas pela arquitetura no dia 28 de Agosto de 2009.

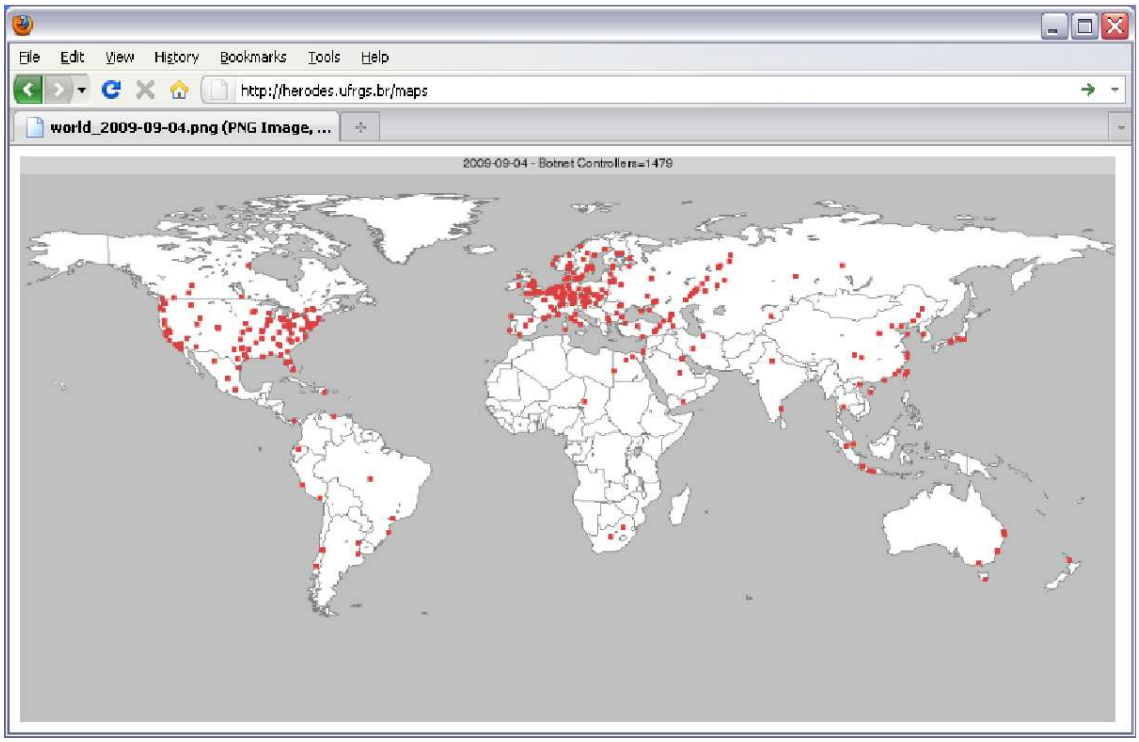
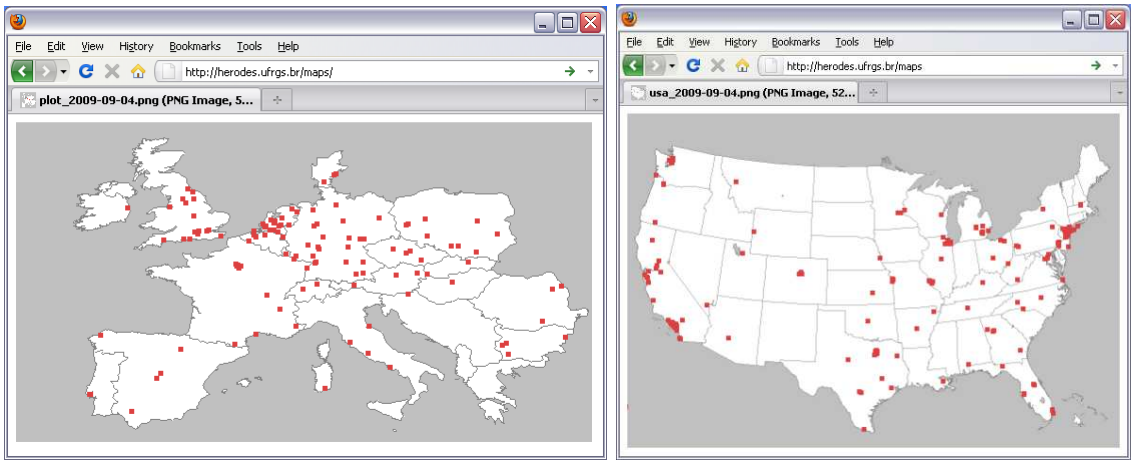


Figura 6.6: Dispersão geográfica dos controladores de *botnets*.

Mapeando os controladores diariamente foi possível desenvolver um mapa diário da localização dos controladores, e com esse conjunto de mapas diários uma animação temporal da localização dos controladores. A análise temporal da dispersão dos controladores (animação) apontou fatos interessantes : a) a maioria dos controladores está localizados

em países desenvolvidos, possivelmente em empresas de hospedagens (*co-location*); b) há pouca variabilidade, ou seja, os controladores migram de endereçamento mas permanecem numa mesma região geográfica.

A maior concentração dos controladores identificados localiza-se no continente Europeu e nos Estados Unidos da América. Na figura 6.7 as regiões com maior número de controladores são evidenciadas. É importante notar que o referido gráfico foi desenvolvido com os mesmos dados apresentados na figura 6.6.



(a) Controladores sediados na Europa. (b) Controladores sediados nos Estados Unidos.

Figura 6.7: Visão dos controladores de *botnets* de forma ampliada.

Essa regionalização dos controladores pode indicar que os controladores de *botnets* podem ser comandados grupos especializados. Estudos recentes apontam redes maliciosas operadas por *gangs* com propósitos específicos, como por exemplo, roubo de informações bancárias (SORANAMAGESWARI; MEENA, 2010).

Além dos controladores de rede, as máquinas comprometidas possuem um papel fundamental para o entendimento das ações maliciosas realizados pelos *botmasters*. Na sequência, os clientes detectados pela arquitetura são avaliados.

6.3.2 *Bots*

Os *bots* representam máquinas comprometidas recebendo instruções de um controlador de *botnets*. Essas máquinas foram localizadas na rede com base nas assinaturas desenvolvidas pela arquitetura e mapeadas com auxílio de ferramentas de segurança (fluxos e filtros de pacotes).

No período em que arquitetura esteve em avaliação foram identificados um total de 4.149 máquinas (IPs) comprometidas. Como já descrito na seção 5.3, os número total de clientes não representa acessos repetidos de um cliente no mesmo dia. A figura 6.8 ilustra o número de controladores identificados mensalmente. É importante notar no gráfico que existem duas categorias ilustradas em cada mês representado. A primeira (*Total*) representa o número total de clientes identificados, já a categoria (*Únicos*) descreve a quantidade de clientes sem repetição. Com isso é possível constatar que a maioria dos clientes de *botnets* identificados na UFRGS não são reincidentes. Dada a premissa que a UFRGS utilizada endereçamento fixo nas suas máquinas é possível concluir que o processo de desinfecção das máquinas está sendo eficaz, bem como a educação de segurança passada ao usuário.

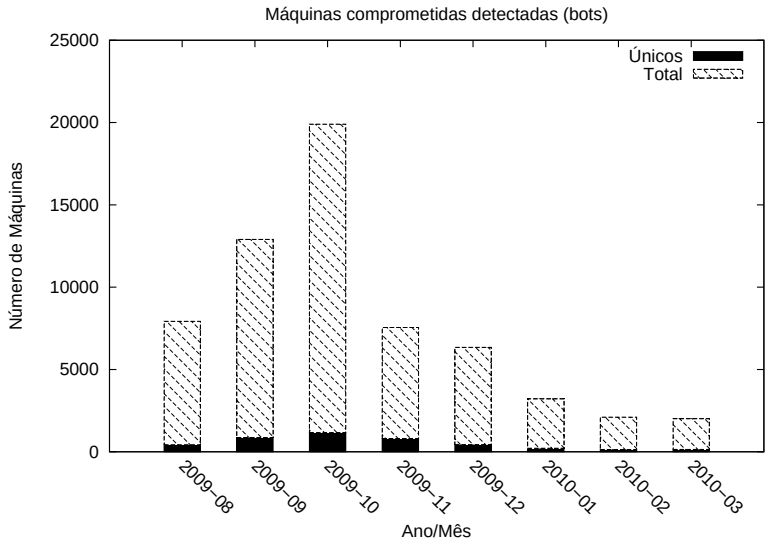


Figura 6.8: Máquinas comprometidas detectadas pelo mapeamento de assinaturas de rede.

Durante a avaliação, notou-se um pequeno número de falsos positivos - aproximadamente 1% (48 máquinas) - , ou seja, máquinas classificadas como infectadas indevidamente. Boa parte dos falsos positivos estava relacionados com os controladores importados. Como já comentado, nas listas importadas a precisão é pequena, afinal as listas utilizadas (Shadow Server obtida via Emerging Threats) disponibiliza apenas o endereço IP de controladores, sem informações de portas e protocolos (TCP ou UDP). Além da falta de precisão das listas, outros fatores levaram a identificação erronia de máquinas comprometidas. Por exemplo, os servidores de *proxy*, DNS, e tradutores de nomes (NAT) eram constantemente classificados como comprometidos quando um dos seus clientes estavam infectados.

Além da identificação de *botnets* o sistema implementou mecanismos que auxiliam o gerente de segurança a restringir acesso das máquinas comprometidas. Esse mecanismo funciona como uma proteção, afinal as máquinas infectadas continuam ativas, mesmo sem acesso ao controlador da rede maliciosas. De forma automatizada, um cliente de *botnet* pode realizar atividades pré-agendadas, sem que uma ordem direta do *botmaster*. A listagem 6.1 ilustra o comportamento típico de um máquina comprometida com acesso irrestrito a rede.

```
1  2010-03-03 21:24:34.558 ICMP 143.54.63.142:0 -> 143.54.168.139:0.0
2  2010-03-03 21:24:34.558 ICMP 143.54.63.142:0 -> 143.54.168.138:0.0
3  2010-03-03 21:24:34.558 ICMP 143.54.63.142:0 -> 143.54.168.137:0.0
4  2010-03-03 21:24:34.558 ICMP 143.54.63.142:0 -> 143.54.168.136:0.0
5  2010-03-03 21:24:34.558 ICMP 143.54.63.142:0 -> 143.54.168.142:0.0
6  2010-03-03 21:24:34.558 ICMP 143.54.63.142:0 -> 143.54.168.143:0.0
7  2010-03-03 21:24:34.558 ICMP 143.54.63.142:0 -> 143.54.168.140:0.0
8  2010-03-03 21:24:34.558 ICMP 143.54.63.142:0 -> 143.54.168.144:0.0
9  2010-03-03 21:24:34.559 ICMP 143.54.63.142:0 -> 143.54.168.145:0.0
10 2010-03-03 21:24:34.559 ICMP 143.54.63.142:0 -> 143.54.168.141:0.0
11 2010-03-03 21:24:39.559 ICMP 143.54.63.142:0 -> 143.54.168.151:0.0
12 2010-03-03 21:24:39.559 ICMP 143.54.63.142:0 -> 143.54.168.150:0.0
13 2010-03-03 21:24:39.559 ICMP 143.54.63.142:0 -> 143.54.168.149:0.0
```

Listagem 6.1: Máquina comprometida realizando varredura na rede local.

Na referida listagem (listagem 6.1) é apresentado as conexões (obtidas através de fluxos) da máquina 143.54.63.142 realizando uma varredura sequencial do tipo ICMP na rede 143.54.168.0/24. Esse comportamento ilustra uma máquina comprometida no processo de enumeração de novos alvos em potencial. O componente *Módulo Reativo* da arquitetura implementa mecanismos para impedir que esse comportamento seja propagado na rede monitorada.

Observando o acesso das máquinas comprometidas, foi possível identificar algumas tendências comportamentais. Do total de 4149 clientes comprometido 1763 acessaram mais de um controlador (mais de uma assinatura da base de dados). Isso é, mais de 42.5% dos clientes infectados acessaram outros controladores. E analisando apenas as máquinas que acessaram mais de um controlador, constatou-se que em média foram realizados 10 acessos a diferentes controladores. Isso sugere que quase a metade dos *bots* detectados trocaram de servidor de controle, ou pelo menos, havia uma renovação do endereçamento do controlador. Isso pode representar uma preocupação dos atacantes a respeito da localização do servidor, ou ainda, a utilização de técnicas de resiliência, como apontado por Thorsten em (THORSTEN et al., 2008).

A avaliação dos clientes também tentou verifica qual foi o tempo médio que um cliente continuava ativo na rede (verificando os registros da *firewall*). Com essa informação seria possível verificar quanto tempo a máquina comprometida demorava a ser desinfetada pelo usuário, ou pela equipe da própria universidade. Segundo a análise, observou-se que a que 32.8% das máquinas identificadas ficaram ativas por mais de um 1 consecutivo. E nessas máquinas - identificadas e ativas por mais de 1 dia - em média as mesmas ficaram 5 dias consecutivos realizando atividade maliciosa.

A identificação rápida dos clientes de *botnets*, implementado pela arquitetura, contribuiu para a segurança da rede como um todo. Isolando a máquina comprometida faz com que outras máquinas vizinhas não sejam afetadas, ou ainda, que dados sensíveis do *bot* sejam enviados ao *botmaster*.

De forma conclusiva, a avaliação da arquitetura apresentou resultados animadores tendo em vista o tamanho do experimento. A quantidade de máquinas detectadas na rede local e o baixo número de falso positivo sugerem que a arquitetura pode ser expandida (quanto ao número de assinaturas) com resultados satisfatórios.

7 CONCLUSÕES

Nesta dissertação foi definida uma arquitetura para uma ferramenta de mitigação e detecção de *botnets* baseada em assinatura de rede de máquinas comprometidas por *bots*. Através da orquestração de softwares e serviços *on-line* foi possível automatizar o processo de detecção e mitigação de *botnets*. O conjunto de informações definido pela arquitetura e observado na implementação demonstrou ser adequado para a interpretação de particularidades das *botnets*. As informações armazenadas na base de dados da arquitetura, em especial, permitem que similaridades entre diferentes tipos de *botnets* sejam traçadas.

A definição da arquitetura, sobretudo, possibilitou que uma ferramenta fosse implementada e validada no contexto de uma rede acadêmica - Universidade Federal do Rio Grande do Sul - por um período de 8 meses. O estágio atual de implementação já permite que a mesma possa ser utilizada por pesquisadores e administradores de rede com interesse de combater e estudar o comportamento de *botnets*. Algumas características interessantes e comuns a respeito das diferentes *botnets* puderam observadas durante a avaliação da arquitetura:

- Poucos controladores de *botnet* são responsáveis por um grande número de *bots*. Ou seja, poucos controladores - super controladores - são responsáveis pelo controle de um grande número de máquinas comprometidas;
- A maioria dos controladores de *botnets*, quando identificados, são rapidamente desabilitados pelos seus responsáveis. No entanto, um pequeno número de controladores (1,5%) ficam por um longo período coordenando atividades maliciosas (em média 56 dias);
- A comunicação do controlador de *botnet* com os seus clientes se dá por meio de portas de difícil controle, tal como a porta 80/TCP (HTTP). Essas portas dificilmente são bloqueadas no filtro de pacotes das instituições e permitem o livre fluxo de informações. Isso demonstra uma preocupação dos atacantes em tornar o tráfego de controle de *botnet* mais difícil de ser detectado;
- Os *bots* acessam diferentes controladores de rede (IP). Esse comportamento é típico de técnicas que buscam incrementar a resiliência das *botnets* utilizando, por exemplo, balanceamento de rede, ou ainda, técnicas mais sofisticadas como `fast-flux` (NAZARIO; HOLZ, 2008);
- O fluxo de *malwares* coletados pelo sistema é relativamente constante. As oscilações no número de *malwares* coletados coincide com novas vulnerabilidades identificadas em serviços de rede. Logo, as novas vulnerabilidades tendem a ser rapidamente incorporadas nos arquivos maliciosos autonomicamente propagados pela rede;

- Não foi observada uma migração sazonal dos controladores entre os diferentes países ou continentes. A maior concentração dos controladores detectados são localizados na América do Norte e Europa Ocidental, não havendo muita variação entre os países.

A avaliação preliminar da ferramenta desenvolvida em termos de identificação de assinaturas e mapeamento de máquinas comprometidas apresentou resultados satisfatórios. As máquinas comprometidas foram rapidamente identificadas e restringidas de acessar a rede por meio de filtros de acesso. Os responsáveis pela máquina comprometida eram notificados por e-mail e por meio de uma Web informativa. Adicionalmente, a base de dados definida pela arquitetura apresentou informações que possibilitam entender melhor características e similaridades das diferentes *botnets* detectadas (CERON; GRANVILLE; TAROUCO, 2009).

Apesar do observado acima, ainda existem melhorias que podem ser realizadas na arquitetura. Como trabalho futuro, destaca-se o desenvolvimento e avaliação de novas metodologia de geração de assinatura de *malwares*. Tais metodologias podem descrever novos algoritmos ou heurísticas para identificar assinaturas de *bots*, tendo como base a análise de arquivos binários já realizada e presenta na base de dados. Dessa forma, *malwares* não identificados como *bots* podem futuramente ser reclassificados com novos algoritmos a serem desenvolvidos. Adicionalmente, novas extensões podem ser incorporadas a arquitetura, como por exemplo, módulos que contemplem novos serviços ou metodologias de análise de arquivos maliciosos. Seria, igualmente, interessante desenvolver uma linguagem unificada para descrever a avaliação de arquivos maliciosos de diferentes fontes de análise. Com isso, as funcionalidades mapeadas por meio de diversos serviços *on-line*, por exemplo, poderiam ser unificadas num único arquivo com formatação padronizada. Dessa forma, o processamento de informações oriundas de análise de *malwares* poderiam ser avaliadas de forma mais otimizada por mecanismos e algoritmos.

Por fim, a arquitetura pode ser aprimorada com a incorporação de novos módulos funcionais ao sistema. Módulos específicos podem ser aprimorados sem afetar o funcionamento da arquitetura auxiliando no processo extremamente dinâmico de detecção e mitigação de *botnets*.

REFERÊNCIAS

BACHER, P. et al. **Know your Enemy**:tracking botnets. Disponível em: `http://www.honeynet.org/papers/bots`. Acesso em Março de 2010.

BAECHER, P. et al. The nepenthes platform: an efficient approach to collect malware. In: IN PROCEEDINGS OF THE 9 TH INTERNATIONAL SYMPOSIUM ON RECENT ADVANCES IN INTRUSION DETECTION (RAID. **Anais...** Springer, 2006. p.165–184.

BARFORD, P.; BLODGETT, M. Toward botnet mesocosms. In: HOTBOTS'07: PROCEEDINGS OF THE FIRST CONFERENCE ON FIRST WORKSHOP ON HOT TOPICS IN UNDERSTANDING BOTNETS, Berkeley, CA, USA. **Anais...** USENIX Association, 2007.

BELLARD, F. QEMU, a fast and portable dynamic translator. In: ATEC '05: PROCEEDINGS OF THE ANNUAL CONFERENCE ON USENIX ANNUAL TECHNICAL CONFERENCE, Berkeley, CA, USA. **Anais...** USENIX Association, 2005. p.41–41.

BINKLEY, J.; SINGH, S. An Algorithm for Anomaly-based Botnet Detection. In: IN PROCEEDINGS OF USENIX STEPS TO REDUCING UNWANTED TRAFFIC ON THE INTERNET WORKSHOP (SRUTI. **Anais...** [S.l.: s.n.], 2006. p.43–48.

CAIDA. **The Cooperative Association for Internet Data Analysis**. Disponível em: `http://www.caida.org/`. Acesso em: Março de 2010.

CALAIS, P. et al. A Campaign-based Characterization of Spamming Strategies. In: CEAS. **Anais...** [S.l.: s.n.], 2008.

CERON, J.; GRANVILLE, L. Z.; TAROUCO, L. Taxonomia de Malwares: uma avaliação dos malwares automaticamente propagados na rede. In: SBSEG 2009 - ARTIGOS COMPLETOS/FULL PAPERS. **Anais...** [S.l.: s.n.], 2009.

CERON, J. M. et al. Vulnerabilidades em Aplicações Web: uma análise baseada nos dados coletados em honeypot. In: VIII SIMPóSIO BRASILEIRO EM SEGURANçA DA INFORMAçãO E DE SISTEMAS COMPUTACIONAIS (SBSEG 2008). **Anais...** [S.l.: s.n.], 2008.

CHOI, H. et al. Botnet Detection by Monitoring Group Activities in DNS Traffic. In: CIT '07: PROCEEDINGS OF THE 7TH IEEE INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY, Washington, DC, USA. **Anais...** IEEE Computer Society, 2007. p.715–720.

CIDR. **Classless Inter-Domain Routing Report**. Disponível em: http://www.cidr-report.org/. Acesso em: Março de 2010.

Cisco. **Cisco NetFlow - Cisco Systems**. Disponível em: http://www.cisco.com/en/US/products/ps6601/products.html. Acesso em: Junho de 2010.

COOKE, E.; MAO, Z. M.; AND, F. J. Hotspots: the root causes of non-uniformity in self-propagating malware. In: IMC '06: 6TH ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT. **Anais...** [S.l.: s.n.], 2006.

COVA, M.; KRUEGEL, C.; VIGNA, G. There is No Free Phish:an analysis of .free. and live phishing kits. In: WOOT'08: PROCEEDINGS OF THE 2ND USENIX WORKSHOP ON OFFENSIVE TECHNOLOGIES, San Jose, CA, USA. **Anais...** USENIX Association, 2008.

Cymru. **Internet Security Research and Insight - Team Cymru**. Disponível em: http://www.cymru.org/. Acesso em: Março de 2010.

DAGON, D. Botnet Detection and Response, The Network is the Infection. In: OARC WORKSHOP. **Anais...** [S.l.: s.n.], 2005.

DASWANI, N.; STOPPELMAN, M. The anatomy of Clickbot.A. In: HOTBOTS'07: PROCEEDINGS OF THE FIRST CONFERENCE ON FIRST WORKSHOP ON HOT TOPICS IN UNDERSTANDING BOTNETS, Berkeley, CA, USA. **Anais...** USENIX Association, 2007. p.11–11.

DIETZ, T. et al. **Definitions of Managed Objects for IP Flow Information Export**. 2010.

EHRLICH, W. K. et al. Detection of Spam Hosts and Spam Bots Using Network Flow Traffic Modeling. In: USENIX WORKSHOP ON LARGE SCALE EXPLOITS AND EMERGENT THREATS (LEET'10), San Jose, USA. **Anais...** USENIX, 2010.

F-Secure. **F-Secure – Antivirus Software**. Disponível em: http://www.f-secure.org/. Acesso em: Março de 2010.

GeoIP. **GeoIP API - Location from IP**. Disponível em: http://www.geoipapi.com/. Acesso em: Junho de 2010.

GOEBEL, J.; HOLZ, T. Rishi: identify bot contaminated hosts by irc nickname evaluation. In: HOTBOTS'07: PROCEEDINGS OF THE FIRST CONFERENCE ON FIRST WORKSHOP ON HOT TOPICS IN UNDERSTANDING BOTNETS, Berkeley, CA, USA. **Anais...** USENIX Association, 2007.

GRIZZARD, J. B. et al. Peer-to-peer botnets: overview and case study. In: HOTBOTS'07: PROCEEDINGS OF THE FIRST CONFERENCE ON FIRST WORKSHOP ON HOT TOPICS IN UNDERSTANDING BOTNETS, Berkeley, CA, USA. **Anais...** USENIX Association, 2007. p.1.

GU, G. et al. BotHunter: detecting malware infection through ids-driven dialog correlation. In: USENIX SECURITY SYMPOSIUM (SECURITY'07), 16. **Anais...** [S.l.: s.n.], 2007.

GU, G. et al. BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: SS'08: PROCEEDINGS OF THE 17TH CONFERENCE ON SECURITY SYMPOSIUM, Berkeley, CA, USA. **Anais...** USENIX Association, 2008. p.139–154.

GU, G.; ZHANG, J.; LEE, W. BotSniffer: detecting botnet command and control channels in network traffic. In: ANNUAL NETWORK AND DISTRIBUTED SYSTEM SECURITY SYMPOSIUM (NDSS'08), 15. **Anais...** [S.l.: s.n.], 2008.

HOLZ, T. et al. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In: LEET'08: PROCEEDINGS OF THE 1ST USENIX WORKSHOP ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS, Berkeley, CA, USA. **Anais...** USENIX Association, 2008. p.1–9.

HOLZ, T. et al. Measurements and Mitigation of Peer-to-Peer-based Botnets: a case study on storm worm. In: USENIX SECURITY SYMPOSIUM, 17., San Jose, CA. **Anais...** [S.l.: s.n.], 2008.

HUANG, P.-S.; YANG, C.-H.; AHN, T.-N. Design and implementation of a distributed early warning system combined with intrusion detection system and honeypot. In: ICHIT '09: PROCEEDINGS OF THE 2009 INTERNATIONAL CONFERENCE ON HYBRID INFORMATION TECHNOLOGY, New York, NY, USA. **Anais...** ACM, 2009. p.232–238.

IPInfoDB. **Free IP address geolocation tools - IPInfoDB**. Disponível em: `http://ipinfodb.com/`. Acesso em: Junho de 2010.

ISECLAB. **Anubis**: analyzing unknown binaries. Disponível em: `http://anubis.iseclab.org/`. Acesso em: Junho de 2010.

JOHN, J. P. et al. Studying spamming botnets using Botlab. In: NSDI'09: PROCEEDINGS OF THE 6TH USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION, Berkeley, CA, USA. **Anais...** USENIX Association, 2009. p.291–306.

KARASARIDIS, A.; REXROAD, B.; HOEFLIN, D. Wide-scale botnet detection and characterization. In: HOTBOTS'07: PROCEEDINGS OF THE FIRST CONFERENCE ON FIRST WORKSHOP ON HOT TOPICS IN UNDERSTANDING BOTNETS, Berkeley, CA, USA. **Anais...** USENIX Association, 2007.

KELCHNER, T. The (in)consistent naming of malcode. **Computer Fraud & Security**, [S.l.], v.2010, n.2, p.5–7, February 2010.

Kojm, T. **ClamAV Antivirus**. Disponível em: `http://www.clamav.net/`. Acesso em: Março de 2010.

KOLTER, J. Z.; MALOOF, M. A. Learning to detect malicious executables in the wild. In: KDD '04: PROCEEDINGS OF THE TENTH ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, New York, NY, USA. **Anais...** ACM, 2004. p.470–478.

KREIBICH, C. et al. Spamcraft: an inside look at spam campaign orchestration. In: SECOND USENIX WORKSHOP ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS (LEET), Boston, USA. **Anais...** [S.l.: s.n.], 2009.

LAUINGER, T. et al. Honeybot, Your Man in the Middle for Automated Social Engineering. In: USENIX WORKSHOP ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS (LEET), 3., San Jose, CA. **Anais...** [S.l.: s.n.], 2010.

Matt Jonkman. **Emerging Threats - Signatures** . Disponível em: `http://www.emergingthreats.net/`. Acesso em: Março de 2010.

Microsoft. **Microsoft Security Bulletin Summary for December 2009**. Disponível em: `http://www.microsoft.com/technet/security/ms09-dec.mspx`. Acesso em: Março de 2010.

MOKUBE, I.; ADAMS, M. Honeypots: concepts, approaches, and challenges. In: ACM-SE 45: PROCEEDINGS OF THE 45TH ANNUAL SOUTHEAST REGIONAL CONFERENCE, New York, NY, USA. **Anais...** ACM, 2007. p.321–326.

NAZARIO, J.; HOLZ, T. As the Net Churns: fast-flux botnet observations. In: INTERNATIONAL CONFERENCE ON MALICIOUS AND UNWANTED SOFTWARE MALWARE'08, 3. **Anais...** [S.l.: s.n.], 2008.

Nepenthes. **Nepenthes - finest collection** . Disponível em: `http://nepenthes.carnivore.it/`. Acesso em: Junho 2010.

NFDUMP. **nfdump - NetFlow Dump.** Disponível em: `http://nfdump.sourceforge.net/`. Acesso em: Junho de 2010.

Norman. **Norman Sandbox Information Center**. Disponível em: `http://www.norman.com/`. Acesso em: Junho de 2010.

PROVOS, N. A virtual honeypot framework. In: SSYM'04: PROCEEDINGS OF THE 13TH CONFERENCE ON USENIX SECURITY SYMPOSIUM, Berkeley, CA, USA. **Anais...** USENIX Association, 2004. p.1–1.

PROVOS, N.; HOLZ, T. **Virtual honeypots**: from botnet tracking to intrusion detection. [S.l.]: Addison-Wesley Professional, 2007.

RAJAB, M. A. et al. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In: HOTBOTS'07: PROCEEDINGS OF THE FIRST CONFERENCE ON FIRST WORKSHOP ON HOT TOPICS IN UNDERSTANDING BOTNETS, Berkeley, CA, USA. **Anais...** USENIX Association, 2007.

ROMIG, S. The OSU Flow-tools Package and CISCO NetFlow Logs. In: LISA '00: PROCEEDINGS OF THE 14TH USENIX CONFERENCE ON SYSTEM ADMINISTRATION, Berkeley, CA, USA. **Anais...** USENIX Association, 2000. p.291–304.

SCHOOF, R.; KONING, R. Detecting peer-to-peer botnets. In: SYSTEM AND NETWORK ENGINEERING, UNIVERSITY OF AMSTERDAM. **Anais...** [S.l.: s.n.], 2007.

Shadowserver. **The Shadowserver Foundation**. Disponível em: http://www.shadowserver.org/. Acesso em: Março de 2010.

SORANAMAGESWARI, M.; MEENA, D. C. Histogram based Image Spam Detection using Back propagation Neural Networks. **Global Journal of Computer Science and Technology**, [S.l.], v.9, n.5, 2010.

SPITZNER, L. **Honeypots**: tracking hackers. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

Stenberg. **Curl - transfer a URL** . Disponível em: http://curl.haxx.se/. Acesso em: Junho de 2010.

STINSON, E.; MITCHELL, J. C. Towards Systematic Evaluation of the Evadability of Bot/Botnet Detection Methods. In: WOOT'08: PROCEEDINGS OF THE 2ND USENIX WORKSHOP ON OFFENSIVE TECHNOLOGIES, San Jose, CA, USA. **Anais...** USENIX Association, 2008.

STINSON, E.; MITCHELL, J. C. Towards Systematic Evaluation of the Evadability of Bot/Botnet Detection Methods. In: WOOT'08: PROCEEDINGS OF THE 2ND USENIX WORKSHOP ON OFFENSIVE TECHNOLOGIES, San Jose, CA, USA. **Anais...** USENIX Association, 2008.

STOCK, B. et al. Walowdac - Analysis of a Peer-to-Peer Botnet. In: EUROPEAN CONFERENCE ON COMPUTER NETWORK DEFENSE (EC2ND), 5. **Anais...** [S.l.: s.n.], 2009.

STRAYER, W. T. et al. Botnet Detection Based on Network Behavior. In: BOTNET DETECTION: COUNTERING THE LARGEST SECURITY THREAT. **Anais...** Springer-Verlag, 2007.

SZöR, P.; FERRIE, P. Hunting for metamorphic. In: IN VIRUS BULLETIN CONFERENCE. **Anais...** [S.l.: s.n.], 2001. p.123–144.

THORSTEN, H. et al. Measuring and Detecting Fast-Flux Service Networks. In: NETWORK & DISTRIBUTED SYSTEM SECURITY SYMPOSIUM (NDSS), 15. **Anais...** [S.l.: s.n.], 2008.

VIEIRA, A. C. G. **Pesquisa sobre o uso das tecnologias da informacao e da comunicacao no Brasil 2007**. [S.l.]: Comite Gestor da Internet no Brasil, 2008.

VIRTI, E. S. et al. Honeypots as a security mechanism. In: IEEE/IST WORKSHOP ON MONITORING, ATTACK DETECTION AND MITIGATION (MONAM 2006), Tubingen, Alemanha. **Anais...** IEEE, 2006.

WANG, M.; LI, B.; LI, Z. sFlow: towards resource-efficient and agile service federation in service overlay networks. In: ICDCS '04: PROCEEDINGS OF THE 24TH INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS (ICDCS'04), Washington, DC, USA. **Anais...** IEEE Computer Society, 2004. p.628–635.

WILLEMS, C.; HOLZ, T.; FREILING, F. Toward Automated Dynamic Malware Analysis Using CWSandbox. **IEEE Security and Privacy**, Piscataway, NJ, USA, v.5, n.2, p.32–39, 2007.

WURZINGER, P. et al. Automatically Generating Models for Botnet Detection TR-iSecLab-0609-001. In: LECTURE NOTES IN COMPUTER SCIENCE. **Anais…** [S.l.: s.n.], 2009. p.108–125.

ZHUGE, J. et al. Collecting autonomous spreading malware using high-interaction honeypots. In: ICICS'07: PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE ON INFORMATION AND COMMUNICATIONS SECURITY, Berlin, Heidelberg. **Anais…** Springer-Verlag, 2007. p.438–451.