UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Isabela Liane de Oliveira

Sistema de coleta, análise e detecção de código malicioso baseado no sistema imunológico humano

São José do Rio Preto
2012

Isabela Liane de Oliveira

Sistema de coleta, análise e detecção de código malicioso baseado no sistema imunológico humano

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, Área de Concentração em Sistemas de Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Orientador: Prof. Dr. Adriano Mauro Cansian

São José do Rio Preto
2012

Oliveira, Isabela Liane.
Sistema de coleta, análise e detecção de código malicioso baseado no sistema imunológico humano / Isabela Liane de Oliveira. - São José do Rio Preto : [s.n.], 2012.
87 f. : il. ; 30 cm.

Orientador: Adriano Mauro Cansian
Dissertação (mestrado) - Universidade Estadual Paulista. Instituto de Biociências, Letras e Ciências Exatas.

1. Computação. 2. Redes de computação - Medidas de segurança. 3. Sistemas imunológicos artificiais. 4. Detecção de intrusão. I. Cansian, Adriano Mauro. II. Universidade Estadual Paulista. Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU - 004.7

Ficha catalográfica elaborada pela Biblioteca do IBILCE
Campus de São José do Rio Preto - UNESP

Isabela Liane de Oliveira

Sistema de coleta, análise e detecção de Código malicioso baseado no sistema imunológico humano

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, Área de Concentração em Sistemas de Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Banca Examinadora

Prof. Dr. Adriano Mauro Cansian
UNESP – São José do Rio Preto
Orientador

Prof. Dr. Marcos Antonio Cavenaghi
UNESP – Bauru

Prof. Dr. Rafael Duarte Coelho dos Santos
INPE – São José dos Campos

São José do Rio Preto
26/Março/2012

Dedico este trabalho

Aos meus pais, Eliana e Gilmar, aos meus irmãos Alex e Willian, e ao meu namorado Jefferson, pelo incentivo constante e pelo amor incondicional.

AGRADECIMENTOS

Aos meus pais, irmãos, namorado e Deus por sempre me apoiarem e me incentivarem para chegar até onde cheguei.

Ao meu orientador, Prof. Dr. Adriano Mauro Cansian, pela orientação pessoal e acadêmica, e apoio durante os anos que permaneci no Laboratório ACME!.

Aos meus colegas de laboratório: André, Jorge, Leandro e Maira pela ajuda no projeto e pela enorme amizade que têm comigo.

Aos companheiros de laboratório Adriano, Bruno, Heitor, Jorge, Raphael, Vinicius Galhardi e Vinicius Oliveira.

Ao ex-ACMER André Gregio pelo apoio e discussões sobre o desenvolvimento deste projeto.

Ao Dario Fernandes pelos esclarecimentos sobre o funcionamento da ferramenta BehEMOT.

À FAPESP pelo financiamento deste projeto.

Aos meus amigos e amigas que sempre me acompanharam, ajudando e me dando forças para fazer tudo que necessário, pela amizade e companhia por todos esses anos.

Aos professores do Departamento de Ciências de Computação e Estatística, pelos conhecimentos passados ao longo desses anos.

RESUMO

Os códigos maliciosos (*malware*) podem causar danos graves em sistemas de computação e dados. O mecanismo que o sistema imunológico humano utiliza para proteger e detectar os organismos que ameaçam o corpo humano demonstra ser eficiente e pode ser adaptado para a detecção de *malware* atuantes na Internet. Neste contexto, propõe-se no presente trabalho um sistema que realiza coleta distribuída, análise e detecção de programas maliciosos, sendo a detecção inspirada no sistema imunológico humano. Após a coleta de amostras de *malware* da Internet, as amostras são analisadas de forma dinâmica de modo a proporcionar rastros de execução em nível do sistema operacional e dos fluxos de rede que são usados para criar um modelo comportamental e para gerar uma assinatura de detecção. Essas assinaturas servem como entrada para o detector de *malware* e atuam como anticorpos no processo de detecção de antígenos realizado pelo sistema imunológico humano. Isso permite entender o ataque realizado pelo *malware* e auxilia nos processos de remoção de infecções.

Palavras-chaves: *Malware*, detecção de intrusão, sistemas imunológicos artificiais, análise dinâmica, *honeypots*

ABSTRACT

Malicious programs (malware) can cause severe damages on computer systems and data. The mechanism that the human immune system uses to detect and protect from organisms that threaten the human body is efficient and can be adapted to detect malware attacks. In this context, we propose a system to perform malware distributed collection, analysis and detection, this last inspired by the human immune system. After collecting malware samples from Internet, they are dynamically analyzed so as to provide execution traces at the operating system level and network flows that are used to create a behavioral model and to generate a detection signature. Those signatures serve as input to a malware detector, acting as the antibodies in the antigen detection process performed by immune human system. This allows us to understand the malware attack and aids in the infection removal procedures.

Keywords: Malware, intrusion detection, artificial immune system, dynamic analyze, honeypots

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AIS	*Artificial Immune System*
BehEMOT	*Behavior Evaluation from Malware Observation Tool*
CVDS	*Computer Vírus Detection Immune System*
EMA	*E-mail Malware Analyzer*
ICB	*Immune Collaborative Body*
IEEE	*Institute of Electrical and Electronics Engineer*
IMAP	*Internet Message Access Protocol*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
IPFIX	*IP Flow Information Export*
JSP	*JavaServer Pages*
KDD	*Knowledge Discovery in Databases*
MD5	*Message-Digest algorithm 5*
NSA	*Negative Selection Algorithm*
POP	*Post Office Protocol*
SHA1	*Secure Hash Algorithm 1*
SHA256	*Secure Hash Algorithm 256*
SSL	*Secure Sockets Layer*
SM	*Submit Malware*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
WC	*Web Crawling*
WEKA	*Waikato Environment for Knowledge Analysis*

SUMÁRIO

Capítulo 1 – Introdução

1.1 Considerações Iniciais

Devido ao aumento do uso de redes de computadores aliada a pouca instrução dos usuários, códigos maliciosos (*malware*) se tornaram uma das principais ameaças para usuários de Internet. As principais consequências provocadas por esses programas maliciosos residem no fato de causarem danos para servidores, redes empresariais e computadores pessoais, além de realizarem ataques distribuídos, roubo de informação sensível, dentre outros problemas.

Em meio ao uso da Internet, muitos programas são desenvolvidos sem a preocupação em prover segurança. Esse fato permite que várias vulnerabilidades possam ser exploradas por códigos maliciosos. Dessa forma, os *malware* são espalhados pela Internet em busca de alvos para atacarem. Essa disseminação é rápida devido a popularização da Internet e suas aplicações (e-mail, *web*, *chat*, etc.) e pouca instrução dos usuários.

Código malicioso (*malware*) é um pedaço de código, ou conjunto de instruções, que realizam ações prejudiciais em um sistema de computador uma vez executado. Há várias categorias de *malware*, cada uma age de maneira distinta, individual ou associada a outros, para formar um pacote de *malware*. Seu intuito é comprometer um sistema, tomar o controle dele, atacar outros dispositivos, capturar informações do usuário e usar os

recursos do sistema enquanto se esconde dos mecanismos de segurança ou tenta desabilitar ou subverter esses mecanismos.

Há vários códigos maliciosos espalhados pela Internet. Diariamente, todos os usuários da Internet estão expostos a esses males que buscam por alvos vulneráveis para se disseminarem. Portanto, é necessário que haja programas capazes de detectar e remover os *malware*. Esse comportamento presente na Internet pode ser comparado ao sistema imunológico humano. O ser humano está exposto a várias doenças em seu habitat, por isso, novas barreiras de proteção são desenvolvidas para manter sua saúde. Dentre essas barreiras estão remédios, vacinas, a própria pele humana, dentre outros.

Assim como as doenças, novos *malware* podem surgir ou os já existentes podem evoluir e tornarem-se mais fortes fazendo com que os mecanismos de barreira existentes não sejam eficazes. Assim, as metodologias de combate a esses males devem ser atualizadas constantemente, pois a falta de prevenção pode causar vários problemas.

Dessa forma, prover segurança tornou-se um item essencial em uma empresa, universidade, organização e afins. Este fato proporcionou um aumento de pesquisas nessa área a fim de tentar descobrir novos mecanismos de prevenção que gerem correções rápidas e periódicas.

1.2 Identificação do problema e justificativa

A pesquisa sobre o comportamento de *malware* é um tópico interessante para agências governamentais, universidades e empresas em geral. A análise rápida e eficaz de amostras de *malware* possibilita compreender suas ações e seu comportamento, sendo assim útil para bloquear ataques distribuídos, agir em torno de repostas e ações de contra medidas e, consequentemente, evitar maiores danos.

Segundo o relatório anual de *malware* do PandaLabs (PANDA SECURITY, 2011a), em 2010, os cibercriminosos criaram 34% de todo o *malware* que já existiu e foi classificado pela empresa, ou seja, criaram e

distribuíram um terço de todos os vírus existentes. Em relação ao primeiro trimestre de 2011 (PANDA SECURITY, 2011b), os cibercriminosos criaram cerca de 26% de novas ameaças comparado ao mesmo período de 2010. A análise anual de 2011 mostrou que os tipos de ameaças seguiram as tendências dos anos anteriores, ou seja, *trojans* novamente estão no topo do ranking de novos *malware* que surgiram em 2011. Essa característica deriva do fato de serem o tipo de ameaça preferida dos cibercriminosos para roubar dados bancários, a serem utilizados para fraudes e roubos. Assim como em 2010, o segundo e terceiro lugar foram respectivamente vírus e *worms*. O número de vírus voltou a aumentar em 2009 devido principalmente ao vírus Conficker (RNP, 2009), e ao fato desses novos tipos de vírus possuírem características de *trojan* e capacidade de roubar informações do usuário, como por exemplo, o Sality (MICROSOFT, 2008) e o Virutas (MICROSOFT, 2010). O gráfico ilustrado na Figura 1 exibe a porcentagem de novos *malware* criados em 2011 por tipos de acordo com PandaLabs.

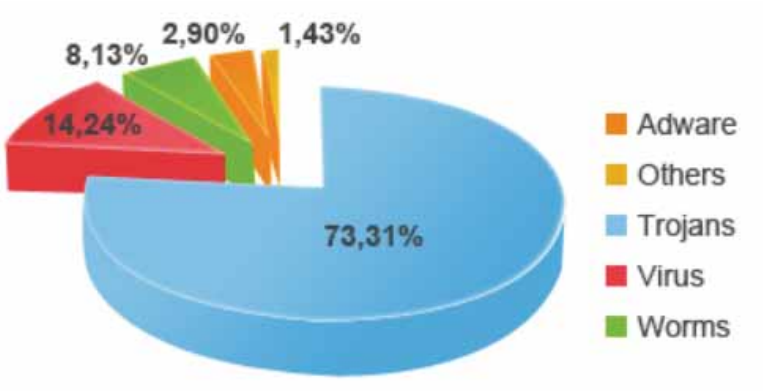

Figura 1 Novos *malware* criados em 2011 (PANDA SECURITY, 2012c).

Uma maneira comum de disseminação de código malicioso ocorre por meio de e-mails, em que os *malware* são enviados como anexos ou acessíveis via *links* no corpo da mensagem. O intuito dessas mensagens é induzir os usuários a executar esses artefatos maliciosos e assim infectar seus sistemas, auxiliando na disseminação destes, buscando e atacando outros computadores, consumindo recursos do sistema ou mesmo roubando informações sensíveis. Segundo o relatório da empresa PandaLabs (PANDA SECURITY, 2011a), em 2010, o spam (mensagens eletrônicas não

solicitadas enviadas em massa para grandes listas de destinatários) manteve a sua posição como uma das principais ameaças, apesar do desmantelamento de grandes *botnets* como a famosa Mariposa ou o Bredolad, que contribuiu para que muitos computadores deixassem de ser utilizados como zumbis para o envio de spam, e consequentemente para a diminuição do tráfego de spam em todo o mundo. Dessa maneira, em 2009, cerca de 95% de todo o tráfego de e-mail foi spam, tendo diminuído para 85% em 2010. Muitos códigos maliciosos são disseminados por meio de anexos ou em *links* contidos no corpo da mensagem do spam.

Como os relatos acima evidenciam, há uma grande variedade de *malware* espalhados pela Internet. Para minimizar os danos causados por tais artefatos é necessário que haja um sistema de resposta rápido e eficiente para tais ameaças.

Vários mecanismos que proveem segurança já são amplamente utilizados para proteger os sistemas computacionais. Dentre esses mecanismos há os antivírus, programas especializados em identificar e remover programas maliciosos conhecidos em computadores pessoais e estações de trabalho; e os *firewalls*, dispositivos dedicados para análise e bloqueio de tráfego não desejado.

No entanto, apesar dos mecanismos de segurança evoluírem, a tarefa de detectar códigos maliciosos torna-se cada vez mais complexa, pois são desenvolvidos para enganar mecanismos de segurança e permanecerem desconhecidos. Segundo Ceron, Granville e Tarouco (CERON, GRANVILLE e TAROUCO, 2009), a nova geração de *malware* possui a capacidade de desabilitar *software* antivírus e ocultar-se no sistema. Além disso, por meio de spams, os códigos maliciosos também conseguem driblar *firewalls* e outros mecanismos de segurança.

Assim, este documento tem como objetivo descrever o estado da arte de análise e detecção de códigos maliciosos além de propor um novo mecanismo, baseado no sistema imunológico humano, para combatê-los por meio de assinaturas de rede e comportamento que sejam capazes de identificar um código malicioso e enviar alertas para os administrador da rede. Através das pesquisas relacionadas é proposta uma arquitetura para

coleta, identificação e resposta a *malware* espalhados na Internet baseado no comportamento do sistema imunológico humano. Esse sistema utiliza as informações obtidas pelas chamadas de sistemas e fluxos de dados gerados pelos *malware* para produzir assinaturas que representam suas características. Essa assinaturas são utilizadas para a detecção de códigos maliciosos em uma rede e em uma máquina.

1.3 Organização da dissertação

Este documento é dividido como se segue: no Capítulo 2 é descrita a fundamentação teórica relativa ao tema, tais como os conceitos sobre os vários tipos de *malware*, sistema imunológico humano, fluxos de rede *NetFlow*, mineração de dados, árvore de decisão e Dionaea. No Capítulo 3 são descritas as pesquisas mais recentes da área, relacionadas à detecção de *malware*. No Capítulo 4 é feita uma breve descrição do sistema de coleta, análise e detecção de código malicioso baseado no sistema imunológico humano. O capítulo 5 ilustra os resultados obtidos no sistema proposto. Por fim, no Capítulo 6, são feitas as considerações finais sobre o tema.

Capítulo 2 – Fundamentação teórica

Este capítulo aborda a fundamentação teórica das tecnologias que compõem o desenvolvimento deste projeto. Na seção 2.1 são descritos os vários tipos de códigos maliciosos e o que eles podem causar no sistema infectado. Na seção 2.2 é abordado o conceito de sistema imunológico humano e como esse conceito se aplica na detecção de códigos maliciosos. Em seguida na seção 2.3 são descritos todos os conceitos relativos ao padrão IPFIX e o protocolo *NetFlow*. A seção 2.4 aborda os conceitos gerais sobre mineração de dados com um foco em árvores de decisão que são utilizadas neste projeto. Por fim, a seção 2.5 contém as principais características da ferramenta Dionaea (sistema utilizado para coletar *malware* espalhados na Internet).

2.1 Códigos Maliciosos

Códigos maliciosos ou *malware* são tipos de programas especificamente desenvolvidos para executar ações maliciosas em um computador (CERT, 2006b). Há inúmeras ações que os *malware* podem realizar em um sistema comprometido e muitas vezes, a vítima não tem conhecimento que tais artefatos estão executando em seu sistema. Algumas dessas ações são (SKOUDIS e ZELTSER, 2004):

- Remover arquivos de configuração que podem deixar o sistema inoperável;
- Tentar se disseminar para outros computadores da rede;
- Monitorar os movimentos de teclado e tela para informar à pessoa mal intencionada;
- Obter informações sobre os costumes do usuário, como por exemplo, páginas acessadas, tempo que permanece conectado, entre outras;
- O atacante pode obter o *desktop* remoto da vítima;
- Roubar arquivos com informações sensíveis com conteúdo pessoal, dados financeiros entre outros;
- Executar outros ataques pelo computador da vítima, como por exemplo, enviar spams, lançar DDoS (Negação de Serviço Distribuído - um atacante utiliza vários computadores para tirar de operação um serviço ou outro computador conectado a Internet (CERT, 2006b)), entre outros;
- Colocar arquivos dentro do sistema infectado, como por exemplo, outros códigos maliciosos, dados roubados, programas piratas, pornografia, entre outros arquivos ilícitos para que outras pessoas possam acessar.

Existem vários tipos de códigos maliciosos. Os tipos mais comuns são:

- ***Vírus*:** Programa ou parte de um programa que se propaga inserindo cópias de si mesmo em outros arquivos/programas do computador (CERT, 2006a). O vírus requer interação humana para se propagar, ou seja, precisa que o programa ou arquivo hospedeiro seja executado para se tornar ativo e se propagar;
- ***Worm*:** Programa capaz de se propagar automaticamente através da rede, enviando cópias de si mesmo de computador para computador (CERT, 2006a). Diferente do vírus, o *worm* não precisa de outros programas para se autorreplicar, ou seja, não embute códigos em outros arquivos e não necessita ser executado para se propagar; ele realiza a propagação utilizando vulnerabilidades existentes nas falhas de programação ou configuração de *software* instalados no computador e, usualmente, não precisa de interação humana;

- ***Keylogger:*** Programa capaz de capturar e armazenar dados digitados por um usuário no teclado de um computador (CERT, 2006a);
- ***Screenlogger:*** Programa capaz de capturar e armazenar a posição do cursor e a tela apresentada no monitor de um computador; ele é uma evolução do *keylogger* já que foi desenvolvido para obter informações que não seriam obtidas pelos *keyloggers*, como por exemplo, senhas clicadas em um teclado virtual;
- ***Cavalo de tróia (trojan horse):*** Programa recebido como um "presente" (por exemplo, cartões virtuais, álbum de fotos, protetor de tela, jogos) que, além de executar funções para o qual foi aparentemente projetado, executa funções maliciosas sem o conhecimento do usuário (CERT, 2006a). Ou seja, um cavalo de tróia é um programa que aparenta ser útil e possuir funcionalidades boas, mas foi desenvolvido para ocultar ações maliciosas (Skoudis e Zeltser, 2004). É normalmente usado por atacantes para roubar dados de usuários como senhas de banco e programas de mensagens instantâneas, além de permitir que o atacante tenha controle do computador;
- ***Bot e Botnets:*** Semelhante ao *worm,* são programas capazes de explorar vulnerabilidades de *software* para se propagarem, com o adicional de prover recursos que possibilitam a comunicação com o atacante (CERT, 2006a). Normalmente o atacante utiliza este tipo de *malware* para desferir ataques a outros computadores de forma distribuída, enviar spams (mensagem eletrônica não solicitada, normalmente não desejada) e *phishings* (fraude eletrônica que utiliza páginas *web* ou e-mails falsos para obter informações de usuários), além de roubar dados da própria máquina infectada;
- ***Backdoors*****:** utilizado para assegurar acesso futuro do invasor à máquina comprometida; assim, o invasor não precisa recorrer aos métodos utilizados na realização da invasão (CERT, 2006a);
- ***Rootkits:*** Conjunto de programas que utilizam mecanismos para esconder e assegurar a presença do invasor no computador

comprometido. Esse tipo de código malicioso possui as mais diversas funcionalidades como: esconder atividades e informações deixadas por um infrator; *backdoors* (funcionalidades escondidas em programas cuja intenção é permitir acesso remoto do atacante a vítima que executou tal aplicativo); remover evidências de registros de sistema; instalar outros tipos de *malware;* varredura para mapear potenciais vulnerabilidades em outros computadores da rede (CERT, 2006a);

- ***Adware* (*Advertising software*):** é um tipo de *software* especificamente projetado para apresentar propagandas, seja através de um navegador, seja através de algum outro programa instalado em um computador (CERT, 2006a);
- ***Spyware*:** refere-se a uma categoria de *software* que tem o objetivo de monitorar atividades de um sistema e enviar as informações coletadas para terceiros (CERT, 2006a); portanto, esse tipo de *malware* compromete a privacidade do usuário;
- ***Dropper*:** Programa designado para "instalar" outros tipos de códigos maliciosos. O *dropper* pode ter esse outro código malicioso dentro de si ou pode ser programado para realizar o *download* desse outro código malicioso;
- ***Security Privacy Risk*:** refere-se a um programa que pode prejudicar a segurança do sistema ao realizar atividades programadas não desejadas pelo usuário ou prejudicar o ambiente privado do usuário (AVIRA, 2012);
- **Código móvel malicioso:** código móvel é um programa leve que é baixado de um sistema remoto e executado localmente, com mínima ou nenhuma intervenção do usuário, portanto, esse tipo de código malicioso utiliza códigos móveis (*Java Script*, *Java Applets, ActiveX Controls*, entre outros) para fazer com que o sistema da vítima faça algo que o usuário não deseja fazer (SKOUDIS e ZELTSER, 2004).

Há diferentes análises que podem ser realizadas para obter informações sobre *malware*. Essas análises são classificadas em estática e dinâmica. A análise estática consiste em analisar o *malware* sem executá-lo,

ou seja, coletando o máximo de informações possíveis sobre o conteúdo do arquivo binário através da análise de sua estrutura interna, como cabeçalhos e *strings*, linguagem utilizada, instruções de *assembly* pura, etc. A segunda consiste em executar o *malware* em um ambiente real ou virtual para inspecionar seu comportamento, via técnicas de *debugging*, análises de chamada de sistema, modificações no sistema operacional e tráfego da rede, por exemplo.

2.2 Sistema Imunológico Humano

O sistema imunológico humano é capaz de garantir a sobrevivência de um indivíduo durante toda sua vida, mesmo que ele se depare, a cada dia, com bactérias e vírus potencialmente mortais. Dessa forma, esse sistema biológico provê uma rica fonte de inspiração para a manutenção da segurança computacional (PAULA, 2004).

O sistema imunológico humano (DASGUPTA e NIÑO, 2008) é composto por três camadas: a barreira anatômica, o sistema imunológico inato e o sistema imunológico adaptativo que juntos provêm a defesa do corpo humano. A barreira anatômica é composta pela pele e superfícies de membranas mucosas e compõe a primeira camada de defesa biológica. A pele impede a entrada para dentro do corpo humano da maioria dos patógenos ou agentes patogênicos (microrganismo que causa algum tipo de doença) e também inibe o crescimento de bactérias. No entanto, muitos patógenos entram no organismo por meio das membranas mucosas. Dessa forma, o papel destas membranas é fornecer uma série de mecanismos não específicos que ajudam a evitar tais invasões.

O sistema imunológico inato possui uma natureza congênita e uma capacidade limitada de diferenciar um agente patogênico de outro, reagindo de maneira semelhante contra a maioria dos agentes infecciosos. Esse sistema é composto pelos mecanismos de barreira patogênica e resposta inflamatória. A barreira patogênica é composta por algumas células especializadas em ingerir substâncias estranhas. Esta ingestão possui duas

finalidades: (a) matar o antígeno (substâncias capazes de induzir uma resposta imunológica) e (b) apresentar fragmentos de proteínas do invasor para outras células do sistema imunológico. A resposta inflamatória ativa macrófagos (célula do tecido conjuntivo) a produzir citocinas que provocam a vaso dilatação e o aumento da permeabilidade capilar. Essas mudanças permitem que um grande número de células imunes circulantes sejam recrutadas para o local onde ocorre uma infecção.

O sistema imunológico adaptativo é capaz de identificar especificamente um determinado agente patogênico, permitindo uma resposta bastante eficiente. Outra característica interessante é a sua capacidade de armazenar informações a respeito de um agente infeccioso, de maneira a responder mais vigorosamente em novas exposições a esse agente. Esse sistema produz dois tipos de resposta na presença de um patógeno: imunidade humoral e imunidade celular. A imunidade humoral é realizada por anticorpos presentes no sangue e que são produzidos pelos linfócitos B. A imunidade celular é realizada pelos linfócitos T que são produzidos em resposta a um antígeno. Os linfócitos T citotóxicos (CTLs) também participam da imunidade celular por meio de ataque às células infectadas injetando toxinas que provocam a sua própria destruição.

A Tabela 1 exemplifica esses mecanismos de defesa biológicos e a Figura 2 ilustra a estrutura multicamada do sistema imunológico formado por esses mecanismos de defesa.

Tabela 1 Mecanismos de defesa biológicos (DASGUPTA e NIÑO, 2008).

Mecanismo de defesa não específico		**Mecanismo de defesa específico**
Barreira anatômica	**Sistema imunológico inato**	**Sistema imunológico adaptativo**
Pele, Membranas mucosas, Secreções da pele e membranas mucosas.	Fagocíticos, glóbulos brancos, proteínas antimicrobianas resposta inflamatória.	Linfócitos, Anticorpos.

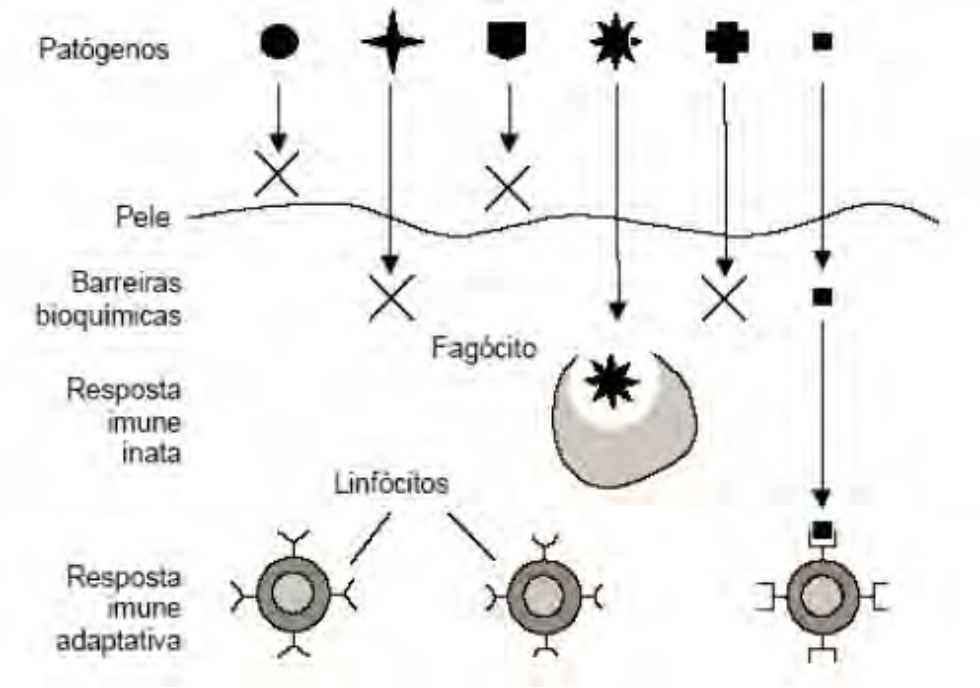

Figura 2 Estrutura multicamadas do sistema imunológico (ALMEIDA, YAMAKAMI e TAKAHASHI, 2007).

O sistema imunológico inato realiza um papel importantíssimo ao prover sinais co-estimuladores para o sistema imunológico adaptativo. Os mecanismos de ambos constituem um sistema de defesa multicamada em que um grande número de células e moléculas agem cooperativamente.

A principal função do sistema imunológico é reconhecer e responder às substâncias chamadas de antígenos, que são substâncias capazes de induzir uma resposta imunológica. Para efetuar essa resposta, o sistema deve executar tarefas de reconhecimento de padrões para diferenciar as células e moléculas do corpo, chamadas *self*, das substâncias estranhas, chamadas *nonself*. Distinguir *self* de *nonself* é uma tarefa complicada visto que são construídas a partir dos mesmos componentes (basicamente proteínas). Além disso, a quantidade de antígenos diferentes que o sistema imunológico deve reconhecer é muito maior que a capacidade do corpo humano de gerar receptores para esses padrões (REIS, 2003). Por isso, uma grande diversidade de receptores são gerados de forma aleatória. No entanto, esse processo pode resultar em receptores que reagem com *self* ao invés de *nonself* causando problemas de autoimunidade em que o sistema

imunológico ataca o próprio corpo. Para solucionar esse problema, as células do sistema imunológico recém criadas passam por um estágio de maturação durante o processo denominado seleção negativa. Dessa forma, qualquer célula cujos receptores reagem com algum *self* durante a maturação, é eliminada. Ou seja, a seleção negativa adere conhecimento sobre o que é normal para o sistema imunológico.

Ainda, o sistema imunológico possui mecanismos de aprendizado e memória para tornar sua proteção mais específica. Esse processo de reconhecimento e resposta é denominado resposta imunológica e é dividido em três fases: (a) fase de detecção, (b) fase de apresentação de antígenos e ativação do sistema adaptativo e (c) fase de eliminação de antígenos.

A analogia entre problemas relacionados com segurança de computadores e o processo biológico foi primeiramente utilizada em 1987 quando Adelman (COHEN, 1987) introduziu o termo “vírus de computador”.

Como pode-se observar, a coleta e identificação de um código malicioso muito se assemelham ao comportamento do sistema imunológico humano. Há várias maneiras de coletar *malware*, como criar um sistema vulnerável e esperar por ataques, ou rastrear páginas web procurando códigos maliciosos armazenados em servidores. No sistema imunológico, as células e produtos do sistema imunológico circulam pelo corpo humano e estão aptas a identificar um antígeno.

2.3 Fluxos de dados (Cisco *NetFlow*)

A Cisco define um fluxo de dados *NetFlow* (também chamado de fluxo de rede) como uma sequência unidirecional de pacotes entre hosts de origem e destino (CLAISE, 2004). Pode-se dizer em resumo que o *NetFlow* provê a sumarização de informações sobre o tráfego de um roteador ou *switch*. Podemos entender um fluxo como uma tupla na qual as seguintes informações aparecem com o mesmo valor: endereço IP de origem e de destino; porta de origem e destino (referente ao protocolo da camada de transporte); valor do campo *Protocol* do datagrama IP; byte *Type of Service*

do datagrama IP; e interface lógica de entrada do datagrama no roteador ou *switch*. Estes campos permitem que um fluxo represente concisamente o tráfego através de um ponto de observação (normalmente um roteador). Todos os datagramas, com um mesmo valor nos campos desta tupla são contados, agrupados e empacotados em um registro de fluxo. Os fluxos mantidos no *Netflow cache* são exportados nas seguintes situações:

- permanece ocioso por mais de 15 segundos;
- sua duração excede 30 minutos;
- uma conexão TCP é encerrada com a *flag* FIN ou RST;
- a tabela de fluxos está cheia;
- o usuário redefine as configurações de fluxo.

Um exemplo deste sistema é mostrado na Figura 3.

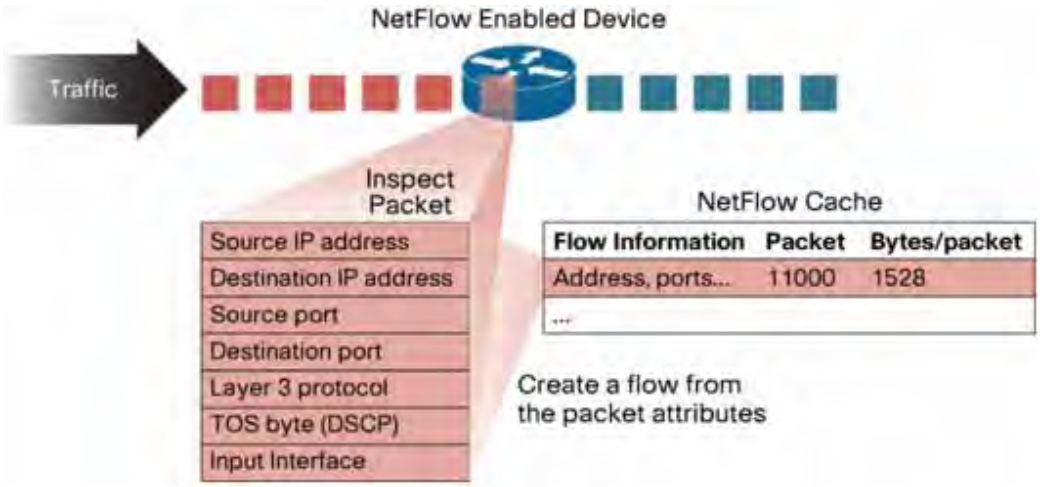

Figura 3 Esquema do padrão Cisco *Netflow* de fluxos de dados (CISCO, 2007).

Uma vez que os fluxos gerados nos equipamentos são exportados e armazenados, passam a constituir uma fonte valiosa de informações. Ou seja, é possível obter detalhes de cada conexão estabelecida por qualquer computador pertencente ao ambiente monitorado, fator este extremamente relevante nas análises de tráfego e de segurança.

Em (CORRÊA, PROTO e CANSIAN, 2008), os autores abordam uma nova metodologia para detecção de eventos em redes de computadores utilizando o protocolo *NetFlow* e o armazenamento de informações em

banco de dados. A arquitetura de armazenamento proposta nesse artigo será utilizada no projeto apresentado neste documento para a detecção de códigos maliciosos. Os dados armazenados e as informações obtidas por meio da análise dinâmica são utilizados para compor o modelo de assinatura proposto nesta dissertação.

2.4 Mineração de dados

Mineração de dados, de maneira geral, consiste em descobrir conhecimentos em bancos de dados (*Knowledge Discovery in Databases* ou KDD) (HAN e KAMBER, 2006). No entanto, esses conhecimentos podem não ser previamente desconhecidos e os dados a serem analisados não precisam necessariamente estar organizados em bancos de dados (GRÉGIO, 2007). O processo de descoberta de conhecimento se resume basicamente em 3 processos:

- **Preparação**: os dados são selecionados (de acordo com sua importância) e preparados (eliminação de ruídos e dados irrelevantes) para servirem de entrada para a próxima etapa;
- **Mineração dos dados**: os dados são processados por alguma técnica de *data mining* e a mineração é realizada com o intuito de obter informações importantes e relevantes;
- **Análise dos resultados**: o resultado da mineração é avaliado, ou seja, os resultados são interpretados para, por exemplo, determinar algum conhecimento adicional e/ou definir a importância dos fatos gerados.

Um sistema de mineração de dados realiza pelo menos uma das seguintes tarefas: descrição de classes, associação, classificação, previsão, agrupamento e análise de série temporal. A descrição completa de cada tarefa pode ser vista em (HAN e KAMBER, 2006).

Nesta dissertação é utilizada a classificação que consiste em analisar um conjunto de dados de treinamento para, em seguida, construir um modelo para cada classe, baseado nas características dos dados. Uma

árvore de decisão ou um conjunto de regras de classificação é gerado por tal processo de classificação, que pode ser usado para entender melhor cada classe no banco de dados e para classificação de futuros dados, ou seja, árvore de decisão é um algoritmo que utiliza os dados de entrada para criar uma estrutura que define regras que classificam os dados de entrada. Essa estrutura é facilmente interpretada por humanos, e recriada automaticamente no evento de novos dados ou fenômenos.

O algoritmo de árvore de decisão escolhido para ser utilizado no detector de *malware* foi o RandomTree. O programa WEKA (*Waikato Environment for Knowledge Analysis)* (WITTEN e FRANK, 2000) possui esse algoritmo, e portanto, a árvore de decisão foi criada pelo programa WEKA para posteriormente ser inserido no detector de *malware*. Essa árvore de decisão recebe como dado de entrada os seguinte campos de uma conexão: Protocolo da camada de transporte, total de *bytes* trafegados e número de porta destino. O resultado da árvore de decisão informa se essa conexão é suspeita (pode ter sido originada por um *malware*) ou normal (não foi originada por uma *malware*). Um exemplo simples de árvore de decisão pode ser visto na Figura 4.

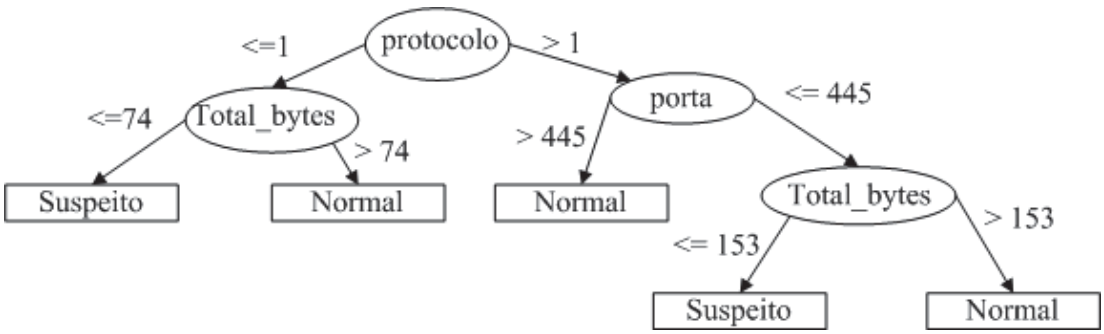

Figura 4 Exemplo de uma árvore de decisão

2.5 Dionaea

Dionaea (DIONAEA, 2012) é uma ferramenta para coletar *malware* que incorpora Python como linguagem de script, utiliza a biblioteca libemu

(LIBEMU, 2012) para detectar *shellcodes*[1], além de apoiar IPv6 (*Internet Protocol version 6*) (DEERING et. al., 1998) e TLS (*Transport Layer Security*) (DIERKS, et. al., 2006).

Desenvolvida inicialmente pelo *Honeynet Project* (HONEYNET PROJECT, 2012) como parte do *Honeynets Summer of Code* durante 2009, Dionaea tende a ser o sucessor da ferramenta Nepenthes (NEPENTHES, 2009) já que foi projetado para sanar as deficiências do Nepenthes. O intuito da Dionaea é montar uma armadilha para *malware* que exploraram as vulnerabilidades expostas pelos serviços em rede. Dessa forma, a ferramenta obtém e armazena uma copia do *malware* que tenta explorar o serviço oferecido.

2.6 Considerações finais

Este capítulo abordou os conceitos e tecnologias que foram utilizadas no desenvolvimento do trabalho. Os conceitos de código malicioso, sistema imunológico humano, fluxo de dados *NetFlow,* mineração de dados e Dionaea formam a base do projeto proposto. O próximo capítulo discute o estado da arte ao abordar vários trabalhos que envolvem os conceitos introduzidos neste capítulo.

[1] *Shellcode* é um trecho de código utilizado na exploração de uma vulnerabilidade de um programa.

Capítulo 3 – Trabalhos relacionados

Este capítulo aborda o estado da arte na detecção de códigos maliciosos com base no sistema imunológico humano. Outros trabalhos que envolvem análise do tráfego de rede para detecção de *malware* e uso de fluxos *Netflow* para a detecção de intrusão também são abordados. Embora haja vários artigos de detecção de códigos maliciosos, nenhum deles utilizam os conceitos de sistema imunológico humano, fluxo de rede *Netflow* e mineração de dados em um único trabalho. Os trabalhos aqui descritos serviram de apoio para a definição da proposta de trabalho descrita no capítulo 4.

3.1 Detecção de códigos maliciosos e sistema imunológico humano

Em 1994, duas publicações marcaram o início de uma nova área de pesquisa que relaciona os conceitos que envolvem a imunologia com os conceitos de segurança de computadores. Uma dessas publicações foi a de Forrest *et al***.** (FORREST *et al***.**, 1994) em que os autores descrevem uma metodologia para distinguir um usuário legítimo de um usuário não autorizado em um sistema de computador baseado na geração de células T em um sistema imunológico. Na outra publicação, Kephart (KEPHART, 1994) propõe um sistema que desenvolve anticorpos contra vírus de

computador, ou seja, identifica vírus de computador baseado no sistema imunológico biológico. Kephart notou que o sistema imunológico traria algumas propriedades excelentes para um sistema de identificação de vírus descentralizado. Algumas dessas propriedades são:

- Reconhecimento de intrusos conhecidos;
- Eliminação e neutralização de intrusos;
- Capacidade de aprender sobre intrusos previamente desconhecidos;
    - Determinar a ocorrência de intrusos;
    - Aprender a reconhecê-los;
    - Memorizar esse reconhecimento;
- Uso de uma proliferação seletiva e auto-replicação para um reconhecimento e resposta rápidos.

Baseados nessas propriedades, Kephart (KEPHART, 1994) propôs um sistema para detectar intrusos que está ilustrado na Figura 5. O sistema inicia a identificação de um intruso por meio de um processo de detecção baseado em comportamento (Detector de anomalia). Em seguida, o sistema classifica o intruso determinando se este é ou não um intruso conhecido (Verificador de vírus conhecidos). Caso o intruso seja previamente conhecido, ele é eliminado e sinais de alertas podem ser enviados a máquinas vizinhas (representado por linhas tracejadas na Figura 5) na tentativa de impedir a proliferação do intruso. Caso contrário, o sistema obtém um conjunto de *decoys* (programa cuja função é atrair intrusos) para análise. Essa análise compara os *decoys* originais com os infectados pelo intruso. As informações obtidas por essa comparação e as informações sobre como remover o intruso obtidas pelo analisador algorítmico são utilizadas pelo extrator de assinatura para que uma assinatura seja gerada. Por fim, a base de assinaturas é atualizada e o intruso é removido.

A desvantagem desse sistema reside no fato de identificar e remover somente um tipo de *malware*: os vírus. Os outros tipo de código maliciosos não são abordados no artigo.

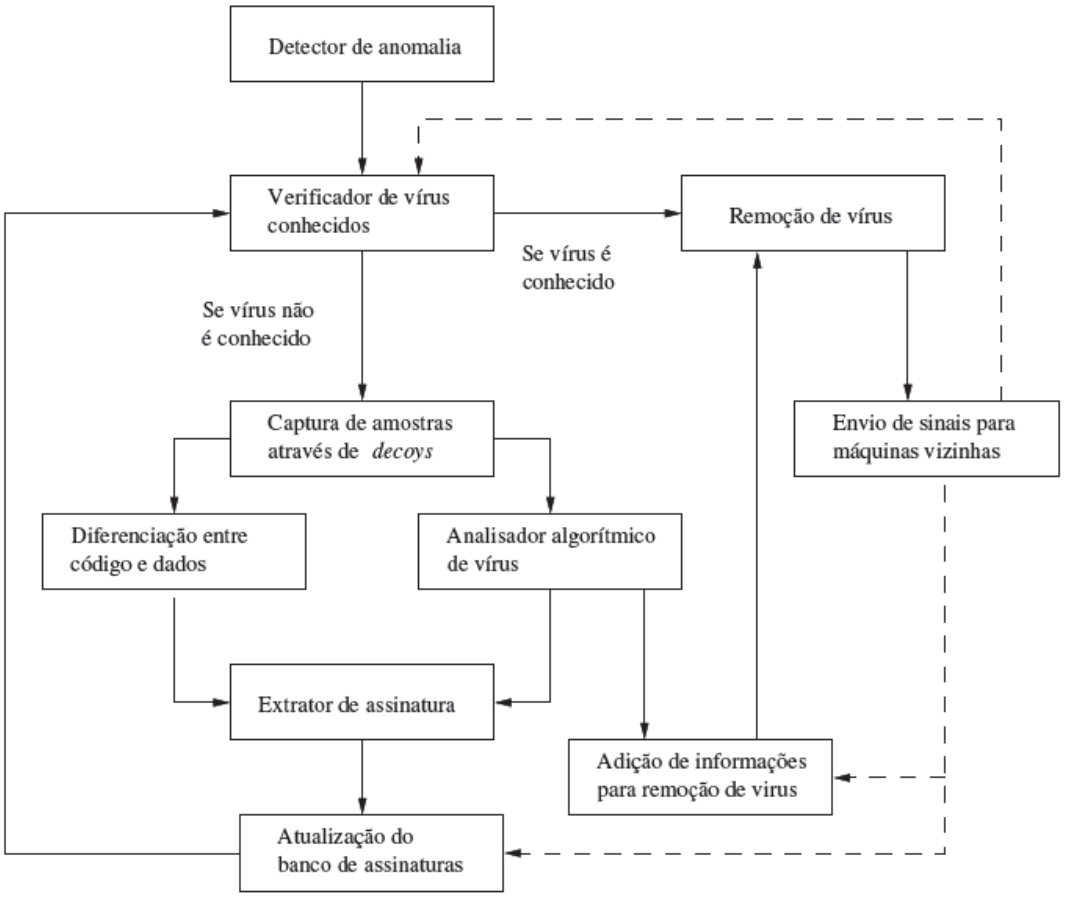

Figura 5 Sistema de detecção de intruso baseado no sistema imunológico proposto por Kephart (PAULA, 2004).

Em 1997, Somayaji, Hofmeyr e Forrest (SOMAYAJI, HOFMEYR e FORREST, 1997) publicaram um novo artigo que discute o sistema imunológico em termos de um conjunto de princípios organizacionais e possíveis arquiteturas de implementação para sistema de computadores imune. Em 2004, Lee, Kim e Hong (LEE, KIM e HONG, 2004), desenvolveram um sistema de assinatura de aprendizagem imune denominado AIVDS (*Artificial Immune based Virus Detection System*). Esse sistema é capaz de detectar vírus desconhecidos, mas possui problemas com a auto-adaptabilidade na detecção de vírus de computador.

A recente pesquisa de Yu Zhang, Tao Li e Renchao Qin (ZHANG, LI e QIN, 2008) apresenta um modelo dinâmico de detecção de vírus de computador capaz de detectar vírus conhecidos e desconhecidos baseado no sistema imunológico biológico. Os autores propõem um novo modelo

para a detecção de vírus denominado CVDIS (*Computer Vírus Detection Immune System*). Esse sistema de aprendizado de assinatura consiste em detectar vírus conhecidos e desconhecidos e variantes de vírus conhecidos e está ilustrado na Figura 6. Para atingir seu objetivo, o CVDIS extrai genes de assinaturas (uma sequência de código pequena e representativa comumente encontrada em vírus) dos vírus e armazena-os na biblioteca de genes de vacina. Em seguida são gerados anticorpos que podem ser classificados em dois grupos: anticorpos específicos e não específicos. Os anticorpos específicos são gerados por meio dos genes de vacina e são utilizados para identificar vírus conhecidos. Os anticorpos não específicos são utilizados para detectar vírus desconhecidos e variações de vírus conhecidos que são criados baseados nos genes de vacina usando mutações, *crossover* e operações de inversão e são para ser anticorpos de memória se a seleção negativa ocorreu. Por fim, os arquivos são classificados como *self* e *nonself* (vírus). Se um vírus desconhecido é identificado por um anticorpo não específico, um gene de vacina será extraído e armazenado na biblioteca de gene de vacina. Quanto maior o número de genes de vacina, melhor será o resultado, a capacidade de autoaprendizagem e a auto adaptabilidade do CVDIS para detectar vírus desconhecidos. Esse modelo analisa e identifica um vírus, mas não elimina o vírus do computador e não possui a funcionalidade de coletar vírus espalhados na Internet como ocorre no sistema proposto neste documento.

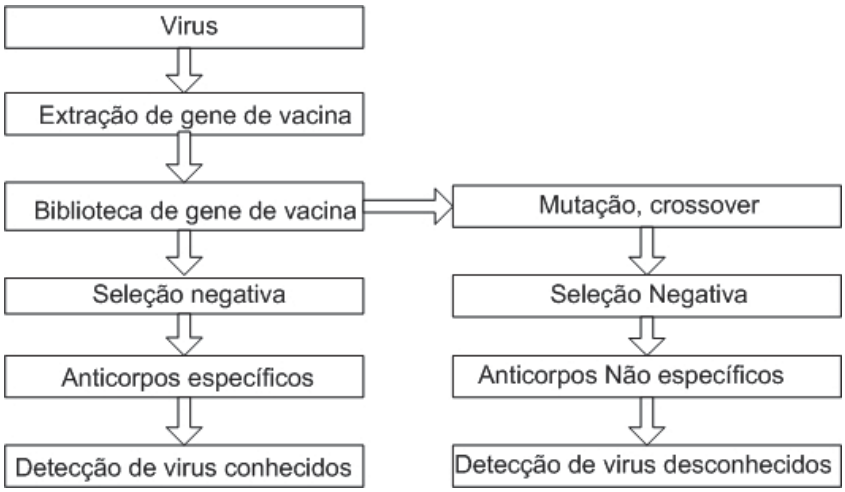

Figura 6 Arquitetura do modelo CVDIS (ZHANG, LI e QIN, 2008).

Em (HE *et. al*., 2010), os autores abordam os problemas de escalabilidade e de cobertura do AIS (*Artificial Immune System*). O AIS utiliza algoritmos NSA (*Negative Selection Algorithm*) para realizar a detecção de códigos maliciosos. O princípio do NSA é que os estados normais são definidos como *self*. Detectores imaturos são treinados por um conjunto *self* conhecido, aqueles que correspondem com um *self* conhecido são deletados e os restantes são analisados por detectores maduros que são capazes de detectar substâncias desconhecidas (*nonself*). No entanto, os detectores maduros podem cobrir uma pequena área o que causa muitos falso-negativos, ou seja, substâncias desconhecidas classificadas erroneamente como substâncias normais. Por isso, é necessário um grande número de detectores e o treinamento desses detectores demanda um longo tempo. Assim, os problemas dessa metodologia reduzem sua eficiência de detecção. Para resolver esse problema, os autores propõem um modelo denominado *Collaborative Artificial Immune System*. Nesse modelo, corpos imunes, independentes em diferentes computadores, são organizados por uma estrutura virtual chamada *Immune Collaborative Body* (ICB) (Figura 7). Corpos imunes podem compartilhar detectores uns com os outros para melhorar a eficiência de detecção. Um módulo de colaboração foi adicionado em cada corpo imune para a comunicação e coordenação. A função do algoritmo imune é executar o algoritmo imune assim como o NSA e o algoritmo de seleção clonal. Esses algoritmos geram detectores maduros que são gerados e armazenados no detector maduro. Os detectores maduros eficientes são armazenados como detectores de memória. Os detectores de memória eficientes são selecionados como detectores colaborativos e espalhados por todo o ICB. Esse modelo se baseia no fato de que certos tipos de *malware* sempre tentam se disseminar em sistemas computacionais similares pois estes tem maior probabilidade de possuir a mesma vulnerabilidade do sistema infectado.

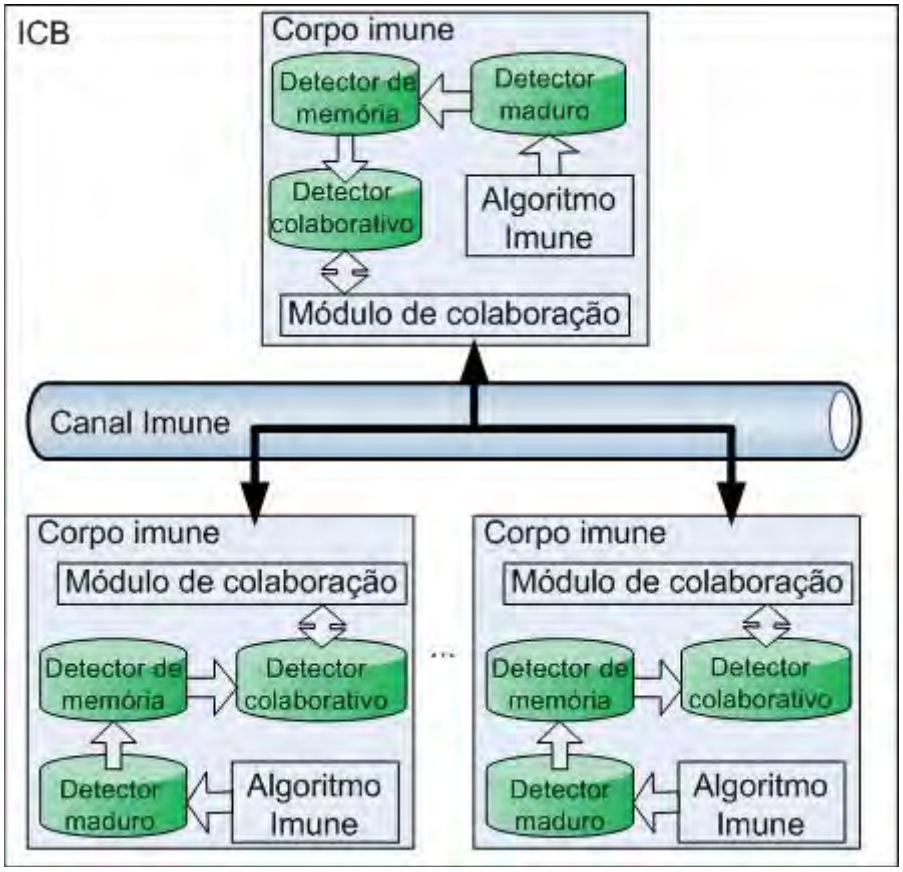

Figura 7 Arquitetura do *Immune Collaborative Body* (ICB) (HE *et. al*., 2010).

3.2 Fluxos de dados e detecção de códigos maliciosos

Em (BIN *et. al*., 2008), os autores propõem um sistema de análise e monitoramento de fluxos baseado em *Netflow* para redes de empresas. Esse sistema é dividido em três módulos:

- Coleta e armazenamento dos fluxos *Netflow*;
- Interface web para visualização dos dados armazenados;
- IDS (Sistema de Detecção de Intrusão) para realizar o monitoramento de tráfego anômalo, por meio de dois algoritmos estatísticos - um baseado em variância de similaridade e outro baseado na distância euclidiana - e a detecção de alguns tipos de *malware* e ataques na rede por meio de um algoritmo de padrões conhecidos baseado nas descrições de eventos utilizando dados dos fluxos.

Esses sistema detecta somente dois tipos de *malware*: Cavalo de Tróia e *worm*. Essa detecção é feita por meio das características presentes nos fluxos e não há um modelo de assinatura definido. O artigo não aborda a maneira como essas características foram obtidas e nem como as características de novos *malware* são inseridas.

Outro trabalho que envolve detecção de códigos maliciosos por meio de fluxos *Netflow* é (HSIAO, CHEN e WU, 2010) em que os autores propõem um sistema para detecção de *malware* disseminados por websites. Um conjunto de dados composto por tráfego web normal e malicioso coletados por um coletor *Netflow* foi utilizado para criar variáveis de agregação espacial e temporal. Esses dados são utilizados por três metodologias de classificação de aprendizagem induzida (*Naïve Bayes*, Árvore de decisão e Máquina de Suporte Vetorial) para a construção de um modelo de previsão que faz a distinção entre tráfego de website normal e malicioso. O modelo de previsão obtido nesse treinamento é utilizado para realizar a detecção de *malware* em tempo real. Os melhores resultados foram obtidos com a árvore de decisão, no entanto, o artigo aborda somente *malware* disseminados por meio de websites.

3.3 Outros trabalhos relacionados

Em (PERDISCI, LEEA e FEAMSTER, 2010), os autores geram assinaturas baseadas nas similaridades encontradas no tráfego de rede HTTP (*Hypertext Transfer Protocol*) gerado por *malware* que utilizam esse protocolo. A metodologia utilizada consiste em encontrar grupos de *malware* que interagem com a web de maneira similar e em seguida extrair um modelo de comportamento de rede em cada grupo para que o modelo possa detectar outras instâncias deste artefato em uma rede monitorada. A grande desvantagem dessa metodologia é o fato de detectar somente *malware* que utilizam o protocolo HTTP. Além disso, as assinaturas são baseadas em dados extraídos do conteúdo dos pacotes o que implica em baixo

desempenho em redes de grande porte uma vez que a análise de *payload* nessas redes demanda muito tempo devido ao grande tráfego da rede.

Por fim, um trabalho que envolve o conceito de fluxo de dados e assinatura de eventos é o trabalho de (CORRÊA, *et al.*, 2009). Esse artigo apresenta um modelo de rastreamento de fluxos baseado em assinaturas focado em detecção de intrusão. O processamento das informações é totalmente voltado a fluxos de rede *NetFlow*. A metodologia utiliza um modelo que cria e reconhece assinaturas classificadas em duas maneiras: abuso e anomalia. As assinaturas são compostas por passos, em que cada evento pode conter um ou mais passos que representem seu comportamento. Utilizando a arquitetura de armazenamento proposta em (CORRÊA, PROTO *et al.*, 2008), as assinaturas são cadastradas por um administrador através de uma interface web, sendo utilizadas por um processador de fluxos que faz o monitoramento do ambiente em tempo real. A arquitetura do sistema proposto pelo artigo está descrita na Figura 8.

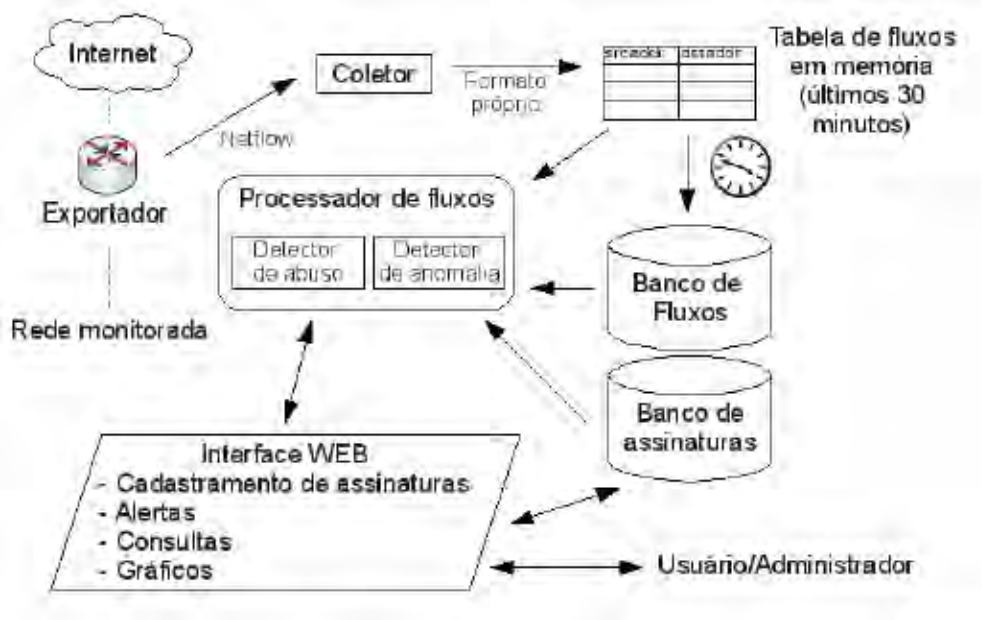

Figura 8 Arquitetura do sistema proposto por (CORRÊA, PROTO *et al.*, 2009).

3.4 Considerações finais

Este capítulo mostrou os trabalhos relacionados a detecção de código malicioso baseado nos conceitos de sistema imunológico humano, assinaturas e fluxos de rede IPFIX (*NetFlow*). Esses artigos foram utilizados como base para o desenvolvimento da proposta de uma arquitetura para a coleta, análise e detecção de códigos maliciosos que é apresentada e discutida no próximo capítulo. A grande vantagem da arquitetura proposta em relação aos artigos apresentados reside no fato de detectar qualquer tipo de *malware*.

Capítulo 4 – Arquitetura proposta

Este capítulo descreve a proposta de trabalho realizado. A proposta consiste em detectar códigos maliciosos baseando-se no sistema imunológico humano por meio de assinaturas. O capítulo é dividido em seis seções: a primeira descreve de maneira geral a arquitetura desenvolvida para implementação do projeto. A segunda detalha os componentes utilizados no módulo de coleta distribuída de artefatos maliciosos. Em seguida, a terceira seção ilustra o processo de geração de assinaturas e a quarta seção descreve os modelos de assinatura desenvolvidos para armazenar os dados, tanto de rede quanto de comportamento local, que representam um artefato malicioso. A quinta seção descreve o sistema de monitoramento utilizado para realizar a detecção dos *malware.* Por fim, na sexta seção são feitas as considerações sobre o este capítulo.

4.1 Visão geral

Este trabalho pretende agir de maneira similar ao funcionamento do sistema imunológico humano que protege o corpo por meio de um complexo de células e moléculas que possuem funções específicas e trabalham de um modo bastante independente, utilizando ainda, mecanismos de comunicação ou sinalização.

Assim como o corpo humano está exposto a vários antígenos, os sistemas computacionais convivem com várias ameaças presentes na Internet e redes locais. Baseado nesse fato, este projeto consiste em um sistema de coleta, armazenamento e análise de códigos maliciosos para prover conhecimento para o sistema de detecção de artefatos maliciosos. A meta é a detecção de códigos maliciosos que tem causado danos imensuráveis aos sistemas de computadores. Para tanto, esse trabalho possui uma arquitetura composta por vários módulos baseando-se nos conceitos que envolvem o sistema imunológico humano.

A arquitetura que foi desenvolvida para o trabalho (Figura 9) é dividida em três sistemas: sistema de coleta distribuída, sistema de análise de *malware* (BehEMOT - *Behavior Evaluation from Malware Observation Tool*) e sistema de monitoramento.

O sistema de coleta distribuída é formado por dois subsistemas. O primeiro consiste em vários sensores instalados com a ferramenta Dioneae em vários locais. Esses sensores simulam serviços de rede para que os códigos maliciosos espalhados pela Internet possam explorar vulnerabilidades presentes nesses serviços. A ferramenta Dioneae gerencia os códigos maliciosos coletados ao armazenar uma amostra dos *malware* que caem nessas armadilhas de maneira única por meio de seu *hash* MD5 (RIVEST, 1992). O segundo subsistema do sistema de coleta distribuída consiste em uma interface web composta pelas seguintes ferramentas: o *E-mail Malware Analyzer* (EMA) e *Web Crawling* (WC) e *Submit Malware* (SM). O EMA procura por códigos maliciosos que interagem com os usuários por meio da execução de anexos ou do acesso a links enviados por e-mail. Em paralelo a essa atividade, o *Web Crawling* é responsável por rastrear sites específicos na Internet que podem conter códigos maliciosos. O SM analisa se o arquivo submetido é um código malicioso. As amostras de códigos maliciosos coletados pelo sistema de coleta distribuída são posteriormente analisados pela BehEMOT.

O sistema BehEMOT é subdividido em dois módulos: (a) sistema de gerenciamento e (b) análise.

Basicamente, o módulo de gerenciamento é responsável por receber amostras de *malware*, controlar a fila de análises e de dados providos durante a monitoração, para a geração de relatórios sobre a execução do programa malicioso.

No módulo de análise, a análise dinâmica automatizada de um exemplar de *malware* é usada para obter informações sobre seu comportamento de execução, isto é, as interações feitas entre este e o sistema alvo. A análise dinâmica é feita através de BehEMOT (FERNANDES FILHO et. al., 2010), uma ferramenta que monitora as ações de um determinado *malware* no nível do *kernel* do sistema operacional, tais quais a criação, remoção ou modificação de arquivos e chaves de registro, processos criados ou finalizados, operações de mutex[2] e tráfego da rede. A partir deste monitoramento, é gerado um relatório de análise com o comportamento capturado com base nas chamadas de sistema efetuadas, a fim de se obter um perfil da execução do *malware*. O módulo composto pelo sistema BehEMOT foi desenvolvido pelo LAS/IC/Unicamp - Laboratório de Administração e Segurança de Sistemas; Instituto de Computação; Unicamp em conjunto com DSSI/CTI/MCTI - Divisão de Segurança de Sistemas de Informação; Centro de Tecnologia da Informação Renato Archer; Ministério de Ciência e Tecnologia e Inovação.

Ao sistema BehEMOT foi adicionado um modulo para a coleta dos fluxos de dados que caracterizam o tráfego de um *malware*. Esses fluxos passam pelo processo de seleção negativa e uma assinatura de rede é gerada. As chamadas de sistema (*syscalls*) obtidas pela análise dinâmica também passam pelo processo de seleção negativa e uma assinatura de comportamento é extraída.

[2] Operações de mutex ocorrem quando dois ou mais processos ou *threads* precisam acessar um recurso que não pode ser compartilhado ao mesmo tempo

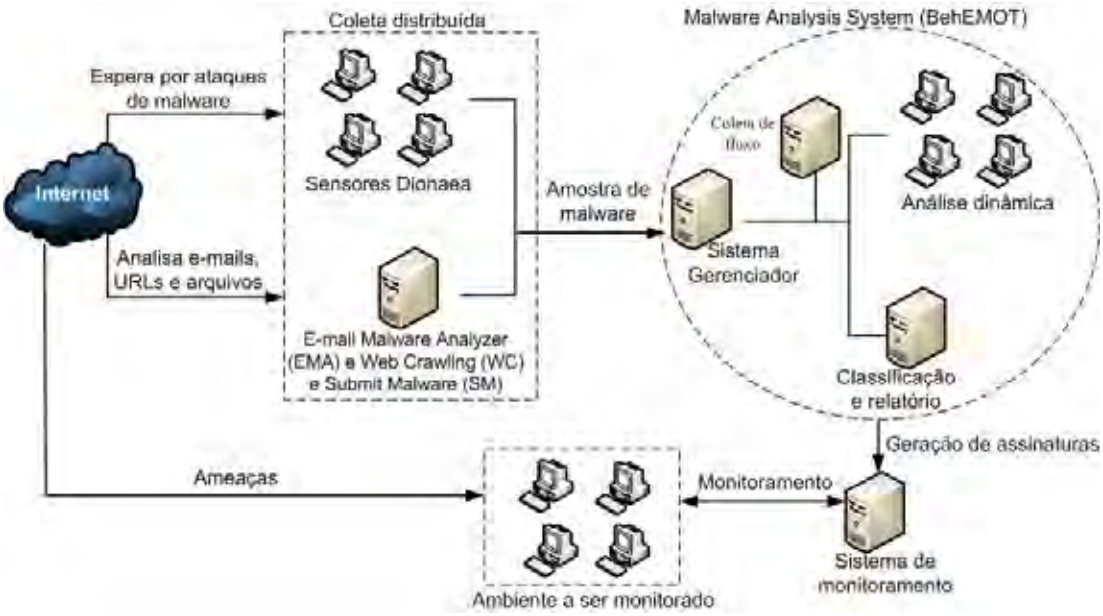

Figura 9 Arquitetura do sistema proposto.

Esses dois tipos de assinaturas são armazenadas em um banco de dados e agirão como anticorpos para o ambiente monitorado. Assim, o sistema de monitoramento usa essas assinaturas para realizar a detecção dos *malware* que ameaçam o ambiente. Caso uma ameaça seja detectada, a resposta imunológica é ativada e um alerta é enviado ao administrador do ambiente monitorado.

4.2 Sistema de coleta distribuída

O sistema de coleta distribuída desenvolvido é subdividido em dois sistemas. O primeiro consiste em vários sensores que emulam serviços de rede vulneráveis, para que os códigos maliciosos provenientes da Internet possam explorá-los. Nestes sensores utilizou-se a ferramenta Dionaea, um *honeypot* que realiza o *download* dos exemplares de *malware* que o atacam (DIONAEA, 2012).

O outro subsistema composto pelo *E-mail Malware Analyzer* (EMA), *Web Crawling* (WC) e *Submit Malware* (SM) analisa arquivos em busca por programas maliciosos, procura por programas maliciosos anexados à mensagens de e-mail e varre *links*, procurando por *malware*. Uma interface

gráfica foi desenvolvida para facilitar a contribuição externa. A Figura 10 ilustra a entrada de dados para a ferramenta EMA. Essa ferramenta é uma evolução da ferramenta descrita em (GRÉGIO, 2009). Assim, para que uma caixa de e-mail seja analisada em busca de *malware* em anexos ou *links* no corpo de e-mail basta informar o e-mail, senha, tipo do servidor (Ex: POP, POPS, IMAP, IMAPS), endereço do servidor de e-mail (Ex: imap.dominio.com) e a profundidade a ser atingida quando um *link* é encontrado no corpo do e-mail. Ao clicar em “Analyze”, a análise é feita e um página é exibida para informar se algum *malware* foi encontrado. O processo de análise está especificado na Figura 11.

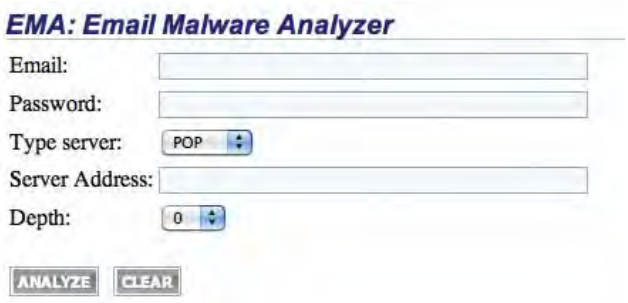

Figura 10 Entrada de dados para o *E-mail Malware Analyzer* (EMA).

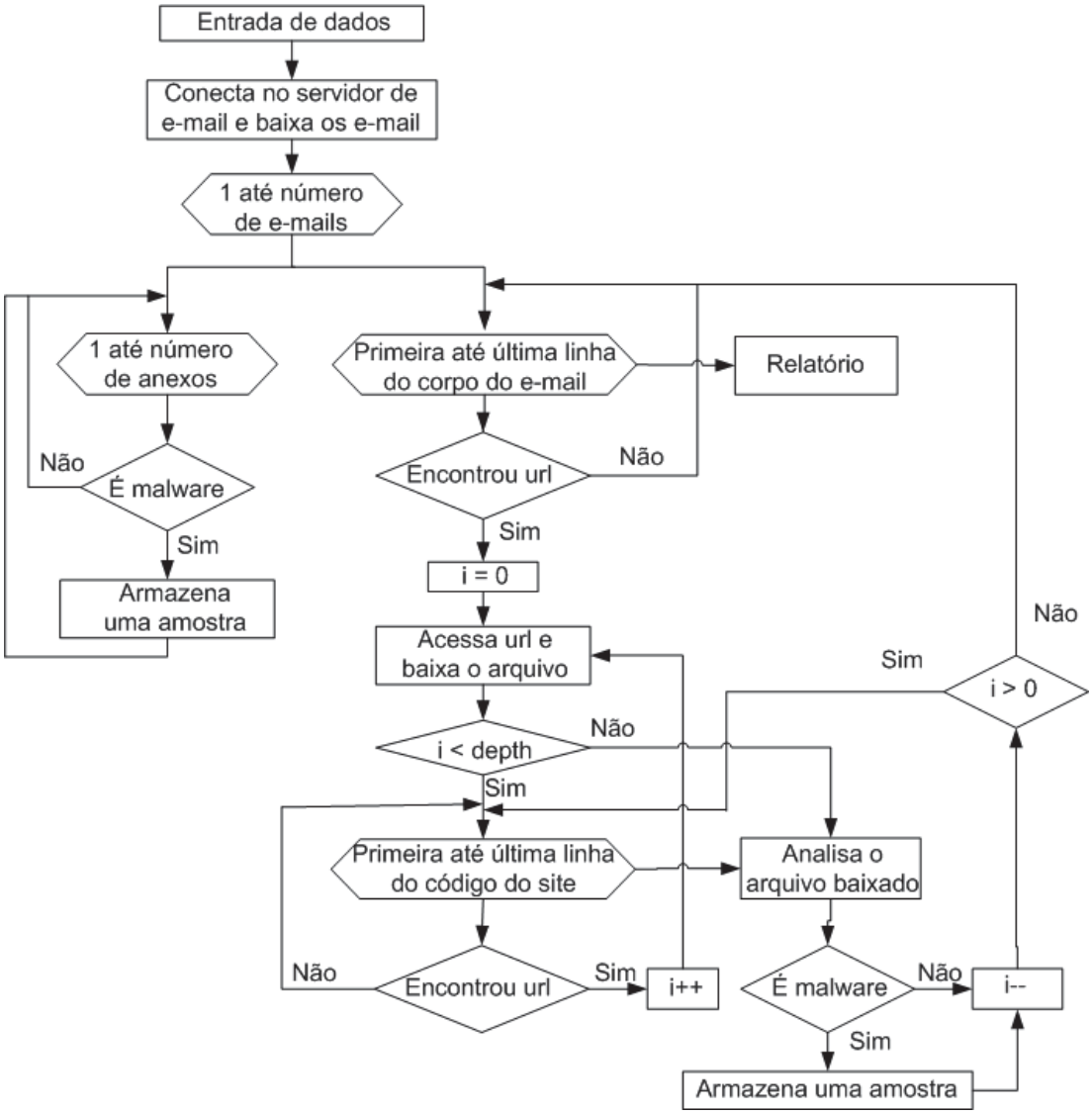

Figura 11 Funcionamento do EMA.

Devido ao fato de um servidor de correio eletrônico poder utilizar os protocolos *Post Office Protocol* (POP) (MYERS, 1996), *Internet Message Access Protocol* (IMAP) (CRISPIN, 2003) ou suas versões de segurança (POPS e IMAPS) para fazer acesso à caixa de entrada de uma conta de e-mail, a ferramenta EMA é capaz de conectar-se ao servidor de correio eletrônico de acordo com o protocolo utilizado.

Com a conexão estabelecida, basta analisar o conteúdo de cada mensagem da caixa de entrada. Sem infringir a privacidade do proprietário da conta de e-mail, EMA faz a separação dos anexos e corpo da mensagem.

Depois dessa segregação, cada anexo passa por um processo de análise. Primeiramente, verifica-se o tipo do arquivo e calcula-se o seu respectivo MD5 (*Message-Digest Algorithm*) (RIVEST, 1992). Em seguida, o arquivo é examinado pelo serviço Virus Total (VIRUS TOTAL, 2012). Esse serviço gratuito e independente possui o intuito de analisar arquivos e URLs suspeitas para identificação de artefatos maliciosos por meio de antivírus. Baseado no resultado obtido por esse serviço, o arquivo é classificado como sendo um código malicioso ou não. Caso seja um código malicioso, uma cópia do arquivo é armazenada de acordo com o seu MD5.

Após a identificação do anexo, EMA concentra-se no conteúdo do corpo da mensagem. Nessa fase do processo, EMA procura por URLs contidas no e-mail. Quando uma URL é encontrada, o próximo passo é acessá-la. O arquivo obtido por meio desse acesso é varrido em busca de URLs, além de ser analisado pelo procedimento citado anteriormente. Esse processo é recursivo, limitado a profundidade definida na entrada de dados Depth, ou seja, quando uma URL conter outra URL, o ciclo de busca termina quando se atinge a n-ésima URL, conforme definido no parâmetro.

Com o término do processo de análise da mensagem, um relatório é emitido contendo as informações extraídas dos arquivos obtidos por meio dos anexos e URLs.

A principal funcionalidade da ferramenta EMA é operar como o módulo de análise de spams enviados para contas de e-mail definidas em busca de códigos maliciosos disseminados por anexos ou URLs, agregando-os a base de dados de *malware* composta pelos códigos maliciosos obtidos pelo sistema de coleta distribuído.

A Figura 12 ilustra a entrada de dados para a ferramenta *Web Crawling*. Para analisar um site pela *Web Crawling* basta informar a URL do site e a profundidade desejada. Ao clicar em “Analyze”, a análise é feita e um página é exibida para informar se algum *malware* foi encontrado. A Figura 13 ilustra o funcionamento da ferramenta *Web Crawling.* O primeiro passo é acessar a URL informada na entrada de dados. O arquivo obtido por meio desse acesso é varrido em busca de URLs. Essa varredura por URL termina quando o parâmetro Depth é atingido. Após analisar o conteúdo do

arquivo acessado, este arquivo é examinado pelo serviço Virus Total (VIRUS TOTAL, 2012). Baseado no resultado obtido por esse serviço, o arquivo é classificado como sendo um código malicioso ou não. Caso seja um código malicioso, uma cópia do arquivo é armazenada de acordo com o seu MD5.

Figura 12 Entrada de dados para o *Web Crawling*.

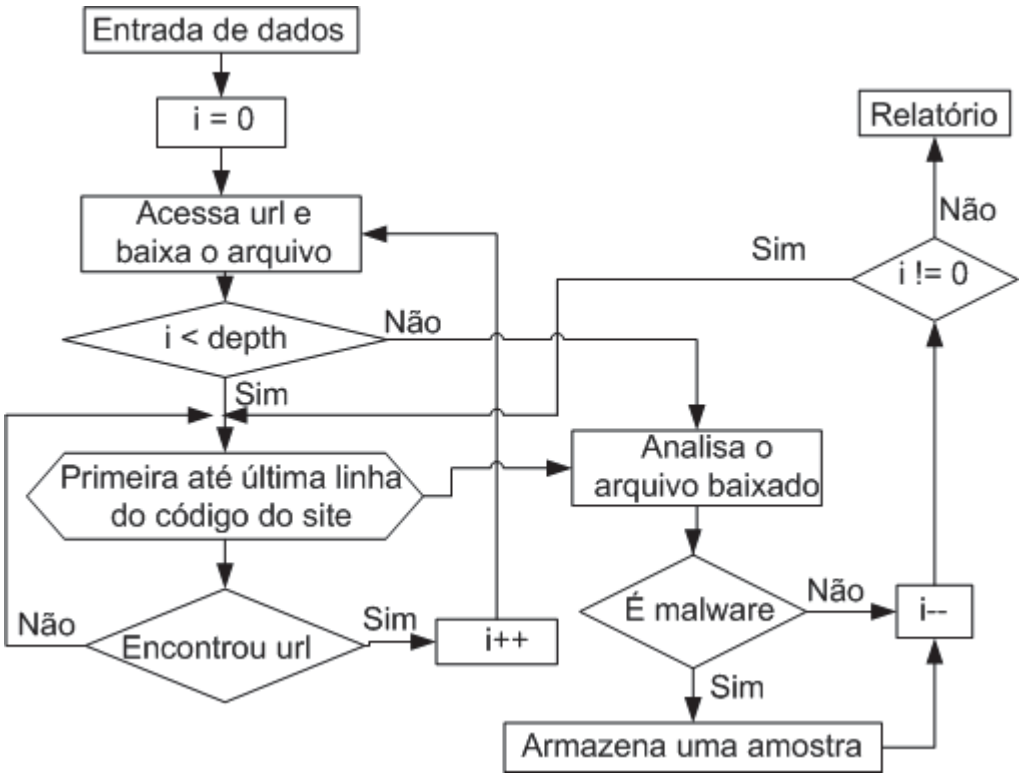

Figura 13 Funcionamento do *Web Crawling*.

A Figura 14 ilustra o relatório que a interface web desenvolvida (subsistema do sistema de coleta distribuída) exibe quando um *malware* é detectado por uma das seguinte ferramentas: EMA, *Web Crawling* ou *pelo Submit Malware*.

```
Suspicious      Backdoor.IRC.Kelebek.h
file(s):
MD5:            f4f05ef10b6f6d285c124d3ad738b29b
Type:           PE32 executable for MS Windows (GUI) Intel
                80386 32-bit
                nProtect:       Backdoor/W32.Agent.729455
                CAT-
                QuickHeal:      Trojan.Flood.l
                McAfee:         IRC/Flood.dt.dr
                K7AntiVirus:    Riskware
                NOD32:          IRC/Kelebek.H
                F-Prot:         W32/Malware!2384
                Symantec:       Trojan.Dropper
                Norman:         W32/Suspicious_Gen2.XQNP
                TrendMicro-
                HouseCall:      BKDR_KELEBEK.U
                Avast:          Win32:Trojan-gen
                eSafe:          Win32.IRC.Kelebek.h
                Kaspersky:      Backdoor.IRC.Kelebek.h
Detection:      BitDefender:    Backdoor.Irc.Kelebek.H
                Emsisoft:       Backdoor.IRC.Kelebek!IK
                Comodo:         Backdoor.Win32.IRC.~B
                F-Secure:       Backdoor.Irc.Kelebek.H
                DrWeb:          Trojan.MulDrop3.6237
                VIPRE:          Trojan.Win32.Generic!BT
                AntiVir:        DR/PSW.Kelebek.H
                TrendMicro:     BKDR_KELEBEK.U
                McAfee-GW-
                Edition:        IRC/Flood.dt.dr
                Sophos:         Troj/Zapchas-CT
                eTrust-Vet:     Win32/IRCFlood.AQ
                Microsoft:      Trojan:Win32/Flood.L
                GData:          Backdoor.Irc.Kelebek.H
                Commtouch:      W32/Malware!2384
                VBA32:          TrojanDropper.VBS.Dummytag.a
                PCTools:        Backdoor.IRCBot
                Rising:         Trojan.Win32.Generic.122B54D0
                Ikarus:         Backdoor.IRC.Kelebek
                Fortinet:       IRC/FLOOD.DT!tr.bdr
                AVG:            IRC/BackDoor.Flood
                Panda:          Trj/Dropper.WF
Packer:         Install Stub 32-bit
```

Figura 14 Exemplo de relatório exibido pelo sistema de coleta distribuída ao detectar *malware* pela interface web desenvolvida.

Todos os códigos maliciosos coletados pelo sistema distribuído (sensores Dioneae, EMA, WC e SM) são armazenados e posteriormente analisados pelo sistema BehEMOT.

4.3 Sistema de geração de assinaturas

O sistema de geração de assinatura é ilustrado na Figura 15 que mostra uma analogia entre o processo de geração de assinatura e o sistema imunológico humano. Para gerar uma assinatura, uma amostra de *malware* é executado três vezes no sistema BehEMOT. O *malware* é como um antígeno para o sistema imunológico humano. Assim, a execução do código malicioso consiste em adquirir informações sobre o programa que causa danos para um sistema computacional. Dessa forma, para cada execução, o tráfego de saída do computador sob a forma de fluxos de dados e as chamadas de sistemas provocados pelo *malware* são armazenadas para serem analisados na próxima etapa.

Após a etapa de coleta de informações, os dados armazenados passam por um processo de seleção negativa em que os dados irrelevantes são descartados. Ao final desse processo de maturação, uma assinatura de rede e outra de comportamento que caracteriza um artefato malicioso são geradas e armazenadas em um banco de dados. Esse processo de aprendizado adere memória ao sistema uma vez que futuras ocorrências desse artefato em um ambiente monitorado poderão ser detectadas.

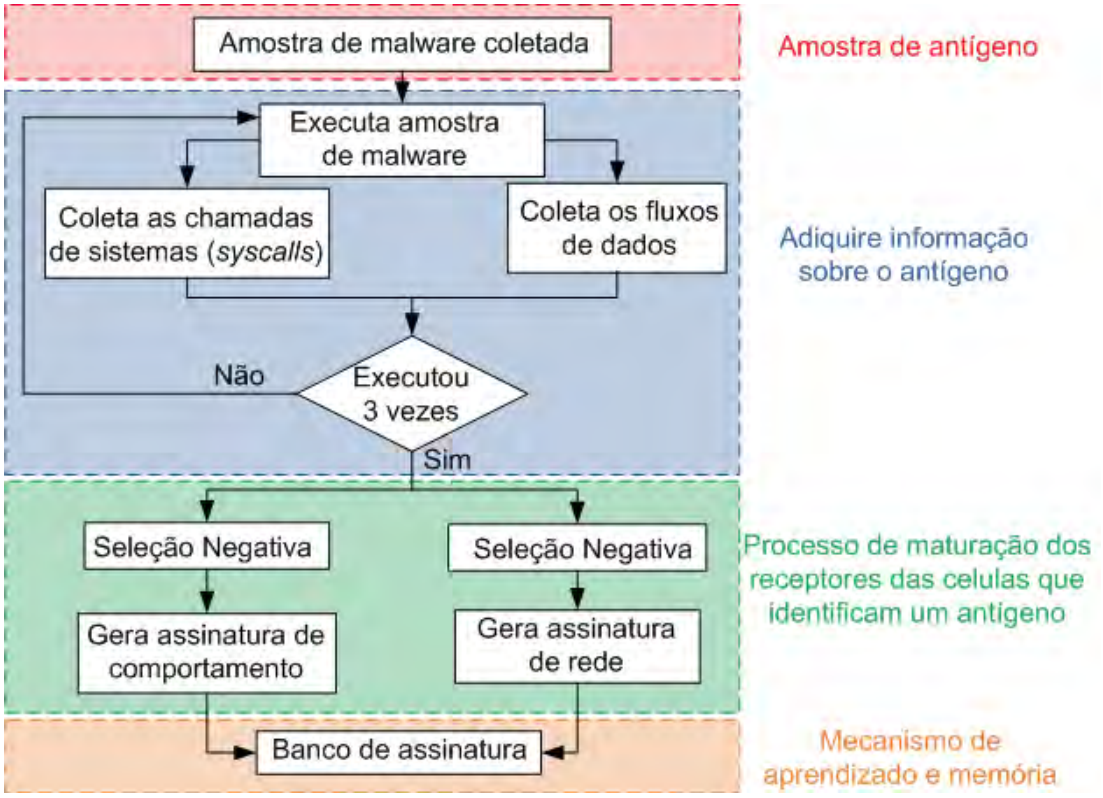

Figura 15 Geração de assinatura.

4.4 Os modelos de assinaturas

Como dito na seção 4.3 dois tipos de assinaturas foram desenvolvidas: comportamento e rede. No entanto, algumas informações são comuns a ambos modelos. Esses campos são utilizados para identificação do *malware* e geração de relatórios mais completos sobre o código malicioso que ameaça o ambiente monitorado.

Abaixo estão as informações gerais sobre o *malware*:

- Identificador único;
- Nome;
- Tipo do binário;
- Tamanho do binário;
- Data da criação da assinatura;
- Classificação de alguns antivírus;
- *Hashes* MD5 (RIVEST, 1992), SHA1 (EASTLAKE e JONES, 2001) e SHA256 (EASTLAKE e HANSEN, 2006);

- Packer[3];
- Local onde uma cópia do binário está armazenado;

4.4.1 Modelo baseado no comportamento local

O modelo de assinaturas de comportamento é criado a partir das informações produzidas pela análise dinâmica dos exemplares de *malware.* Um perfil comportamental de *malware* é pré-processado de forma que, para cada atividade capturada seja gerada uma tupla com os seguintes campos:

- Ação (Ex: abrir, ler, criar, remover);
- Tipo da ação (Ex: arquivo, processo, registrador, rede, mutex);
- Origem da ação (executor);
- Destino da ação (alvo);
- Parâmetro da ação, se existente.

Um *malware* pode gerar uma ou mais ações, ou seja, uma assinatura é composta por uma ou mais tuplas que descrevem as ações do artefato malicioso dentro de uma máquina. As *syscalls* também monitoram as ações referente ao tráfego de rede gerado pelo computador. No entanto, o modelo de assinatura de comportamento não irá considerar essas ações de rede visto que há um modelo de assinatura separado para armazenar as características do tráfego de rede.

Um exemplo de assinatura de comportamento é apresentado na Tabela 2.

[3] *Packers* são *software que utilizam técnicas de compressão de dados para reduzir o tamanho de um arquivo. Essencialmente, é uma técnica de ocultação, em que o arquivo comprimido é formado pela combinação dos dados comprimidos e o código de descompressão.*

Tabela 2 Fragmento de assinatura de comportamento de um exemplar de *malware*.

Action	Action Type	Action Source	Action Target	Parameters
WRITE	REGISTRY	%HOMEPATH%\desktop\MALWARE.exe	\registry\machine\software\microsoft\windows\currentversion\internet settings\cache\paths\path	
[...]	[...]	[...]	[...]	[…]
CREATE	FILE	%HOMEPATH%\desktop\MALWARE.exe	%PATH%\nt09.exe	
[...]	[...]	[...]	[...]	[…]
WRITE	REGISTRY	%PATH%\nt09.exe	\registry\user\{RID}\software\microsoft\windows\currentversion\explorer\ shell folders\cookies	

4.4.2 Modelo baseado no tráfego de rede

O modelo de assinatura de rede foi criado baseado nos dados obtidos pelos fluxos de dados de uma máquina. Como dito na seção 4.1, foi desenvolvido um módulo adicional ao BehEMOT para coletar os fluxos. Um fluxo gerado pelo *malware* passa por um pré-processamento e uma tupla é gerada. Esta tupla é composta pelas seguintes informações extraída pelo tráfego de saída da máquina infectada:

- Total de bytes;
- Número da porta de destino;
- Protocolo da camada de transporte utilizado.

A assinatura de rede também é composta por uma ou várias tuplas extraídas dos fluxos coletados. Um exemplo de assinatura de rede é apresentado na Tabela 3

Tabela 3 Fragmento de assinatura de rede de um exemplar de *malware*.

Total Bytes	Destination Port	Protocol
61	53	UDP
2136	80	TCP
[...]	[...]	[...]
106	4244	TCP

4.5 Sistema de monitoramento

A Figura 16 ilustra a maneira como um ambiente é monitorado. O monitoramento é dividido em dois módulos: módulo de detecção de rede e módulo de detecção de comportamento.

O módulo de detecção de rede contém um servidor de coleta de fluxo que recebe os fluxos do ambiente monitorado que foram exportados por um equipamento que exporta fluxo *Netflow* v5 localizado na borda do ambiente monitorado. O agente de rede, responsável por realizar a detecção de uma *malware* baseado no tráfego de rede, analisa os fluxos no servidor de coleta de fluxo e compara as assinaturas armazenadas no banco de assinaturas. Se houver uma detecção, um relatório é enviado ao administrador do ambiente monitorado, caso contrário, o agente de rede continua a sua monitoria.

O módulo de detecção de comportamento possui um servidor de coleta das *syscalls* geradas pelo ambiente monitorado. O agente de comportamento, responsável por realizar a detecção de um *malware* baseado nas chamadas de sistemas (*syscalls*), analisa as *syscalls* no

servidor de coleta de *syscalls* e compara as assinaturas armazenadas no banco de assinaturas. Se houver uma detecção, um relatório é enviado ao administrador do ambiente monitorado, caso contrário, o agente de comportamento continua seu processo de monitoramento.

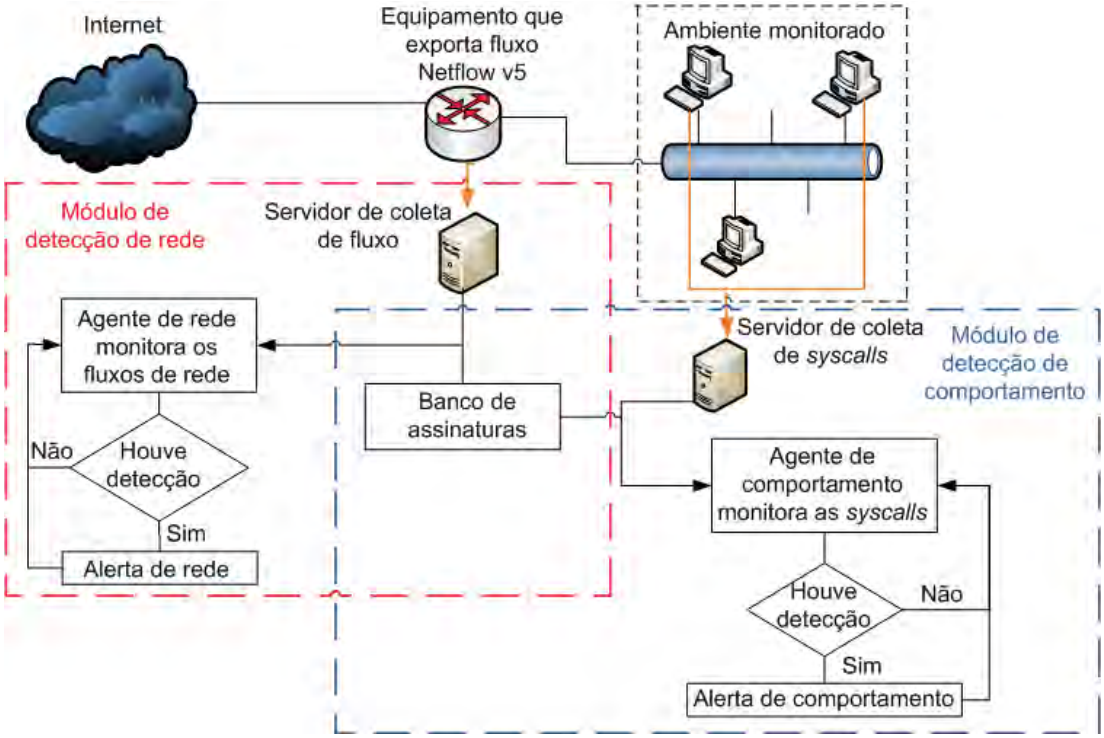

Figura 16 Sistema de monitoramento.

Os agentes de rede e de comportamento são como os anticorpos que ficam circulando pelo corpo humano em busca de alguma substância estranha. As assinaturas de rede e de comportamento são como os receptores que identificam a substância estranha. Dessa forma, quando um *malware* é detectado, uma resposta inflamatória é ativada e um relatório sobre o *malware* é enviado ao administrador do ambiente monitorado.

4.5.1 Agente de comportamento

O agente de comportamento consiste em analisar as chamadas de sistema feitas em um computador e comparar com as assinaturas de comportamento armazenadas em um banco de dados. Caso hajam correspondência, é enviada uma mensagem de alerta ao administrador do ambiente informando a detecção da execução de um determinado *malware*. Estas chamadas de sistema capturadas também passam por uma seleção negativa, na qual é feito um pré-processamento de modo que somente aquelas relevantes, do ponto de vista da segurança do sistema, são comparadas com uma assinatura.

O alerta de detecção é composto por um relatório que é enviado ao administrador do ambiente por meio de correio eletrônico. O e-mail de alerta contém informações gerais sobre o *malware* detectado e como removê-lo.

4.5.2 Agente de rede

O agente de rede foi desenvolvido para analisar os fluxos da rede do ambiente monitorado e comparar com as assinaturas de rede armazenadas no banco. O processo é o mesmo mencionado anteriormente, culminando no envio de uma mensagem de alerta.

Devido à alta quantidade de fluxos que uma rede pode gerar, optou-se por utilizar uma árvore de decisão dentro do agente de rede para classificar os fluxos como normais (não foram originados por um *malware*) ou suspeitos (podem ter sido originados por um *malware*). Dessa forma, somente os fluxos considerados suspeitos são comparados com as assinaturas. A Figura 17 ilustra o processo de criação da árvore de decisão utilizada pelo agente de rede. Foram utilizados tanto fluxos normais quanto dados suspeitos (assinaturas de rede) como entrada para a criação da árvore de decisão. Os fluxos normais passaram por um processamento em que os fluxos foram transformados em tuplas iguais ao de uma assinatura de rede. Em seguida, o programa WEKA realizou a mineração de dados com os dados de entrada

e criou uma árvore de decisão. Posteriormente essa árvore de decisão foi inserida no agente de rede.

Figura 17 Processo de criação da árvore de decisão utilizada pelo agente de rede.

Utilizou-se 299760 tuplas sendo aproximadamente 0,32% de tuplas suspeitas e 99,68% de dados normais. Em uma máquina com 4GB de memória RAM e processador Intel Core 2 Duo com velocidade 2.26 GHz, o tempo dispendido para a construção da árvore de decisão foi de aproximadamente 1,97 segundos e o tempo para testar o modelo foi de aproximadamente 1,24 segundos. O tamanho da árvore obtida é 285 e pode ser vista no Anexo A. O resultado do treinamento mostrou que aproximadamente 99,988% dos dados de entrada foram classificados corretamente e aproximadamente 0,012% foram falsos positivos, ou seja, dados normais classificados erroneamente como suspeitos. Esses dados evidenciam os benefícios do uso da árvore de decisão já que sua taxa de classificação é eficiente. Com isso, o detector não perderá tempo analisando dados normais o que implica em uma detecção em curto prazo.

A Figura 18 ilustra a matriz de confusão resultante da árvore de decisão criada. Essa matriz contém informações que permitem compreender o resultado do algoritmo RandomTree. O ‘a’ são as instâncias normais e o ‘b’

são as instâncias suspeitas. A primeira linha da matriz indica que das 298809 instâncias normais 298773 foram corretamente classificadas como normais e 36 foram erroneamente classificadas como suspeitas. A segunda linha indica que todas as 951 instâncias suspeitas foram corretamente classificadas.

a	b	Classificado como
298773	36	a = normal
0	951	b = suspeito

Figura 18 Matriz de confusão da árvore de decisão criada.

Tanto o detector de comportamento quanto o detector de rede enviam ao administrador do ambiente um alerta composto por um relatório com informações gerais sobre o *malware* identificado, conforme mencionado anteriormente. Esse alerta é enviado por meio de correio eletrônico. O e-mail de alerta contém informações gerais sobre o *malware* detectado. No detector de rede, o endereço IP da máquina suspeita também é enviado no relatório. A Figura 19 ilustra um exemplo resumido de relatório de alerta enviado.

```
################## ALERTA DE DETECÇÃO ##################

ID: 44
IP: 192.168.0.2
Nome: 0a367bfd7f4ddfd24db6409126ece748
Data de criação: 2011-12-24
Tipo do arquivo: PE32 executable for MS Windows (GUI) Intel
80386 32-bit
MD5: 0a367bfd7f4ddfd24db6409126ece748
SHA1: a4ab9f3ded2e5d8590b5775668ee5b9328d18c9d
SHA256:      7e25127f9582285d72866ca7c926f1df9728e7762fb8d4529
e4c569d77a1ab9e
Packer: Nenhum packer encontrado
Classificação por antivírus:
nProtect: Worm/W32.Kolab.406528
[ ... ]
AVG: Dropper.Generic.BNPG
SUPERAntiSpyware: N/I
Para remover o malware:
1. Inicio o computador em modo de segurança para fechar todos
os processos em execução.
2. Lembre-se de fazer backup do sistema antes de realizar
qualquer mudança para futura restauração se necessário.
3. Remova os seguintes arquivos:
- %PATH%\nt09.exe
[ ... ]
4. Abra o Editor de registro e remova as seguintes entradas de
registro:
-\registry\user\{RID}\software\microsoft\windows\
currentversion\explorer\shell folders\cookies
[ ... ]
-      \registry\machine\software\microsoft\windows\currentver
sion\internet settings\cache\paths\path1\cachepath
5. Há casos em que o malware se esconde dentro do arquivo de
sistema WIN.INI e strings "run=" e "load=". Verifique
cuidadosamente os passos executados para que o malware seja
eliminado completamente.
Observação: Alguns caminhos de arquivos e/ou registros podem
mudar.
```

Figura 19 Exemplo de relatório de alerta do detector de rede.

4.6 Considerações finais

Este capítulo descreveu como o projeto foi desenvolvido e estruturado. Essa descrição permitiu compreender (a) como os artefatos são coletados, (b) como os dados são coletados e armazenados em forma de assinaturas, (c) como é realizada a detecção de um *malware* e (d) como as tecnologias

descritas no capítulo 2 são utilizadas no projeto. A grande vantagem dos modelos de assinaturas propostos é que um *malware* pode ser detectado mesmo se ele for alterado por meio de técnicas de ofuscação de código. O próximo capítulo apresenta os resultados obtidos nesses ambiente.

Capítulo 5 – Resultados

Este capítulo descreve os resultados obtidos com o ambiente descrito no capítulo 4. O capítulo é dividido em quatro seções: a primeira contém os resultados obtidos pelo sistema de coleta distribuída. A segunda seção descreve alguns dados estatísticos referentes aos *malware* que possuem assinatura. Em seguida, a terceira seção descreve alguns resultado obtidos em um ambiente simulado. Por fim, na quarta seção são feitas as considerações sobre os resultados obtidos.

5.1 Resultados obtidos pelo sistema de coleta distribuída

O sistema de coleta distribuída desenvolvido foi implantado em duas redes distintas, de instituições separadas geograficamente. Os códigos maliciosos coletados por estes sensores são armazenados para, posteriormente, serem enviados ao sistema de análise dinâmica. A quantidade média de exemplares não repetidos de *malware* coletados por mês, no período entre 01 de março de 2011 até 31 de dezembro de 2011 em um dos sensores foi de aproximadamente 51. O segundo sensor teve como quantidade média aproximadamente 20 exemplares não repetidos coletados por mês, entre 01 de março e 31 de dezembro de 2011. A seguir são mostrado resultados referentes às assinaturas geradas.

5.2 Assinaturas geradas

Com os modelos de assinaturas definidos, os exemplares de *malware* coletados pelo sistema distribuído foram submetidos gradativamente ao sistema de geração de assinaturas. Todas as assinaturas geradas são armazenadas em um banco de dados, para uso posterior na detecção pelo sistema de monitoramento.

Todos os 687 exemplares de *malware* que produziram uma assinatura foram submetidos ao sistema Virus Total (VIRUS TOTAL, 2012). Esse serviço gratuito e independente, desenvolvido pela Hispasec Sistemas, tem o intuito é analisar arquivos e URLs suspeitas por meio de diversos antivírus, na tentativa de identificar programas maliciosos conhecidos. A Figura 20 ilustra o total de *malware* detectados por alguns dos antivírus disponíveis. Nos casos em que o Virus Total não apresentou resultado do antivírus referenciado considerou-se que o antivírus não detectou o *malware*.

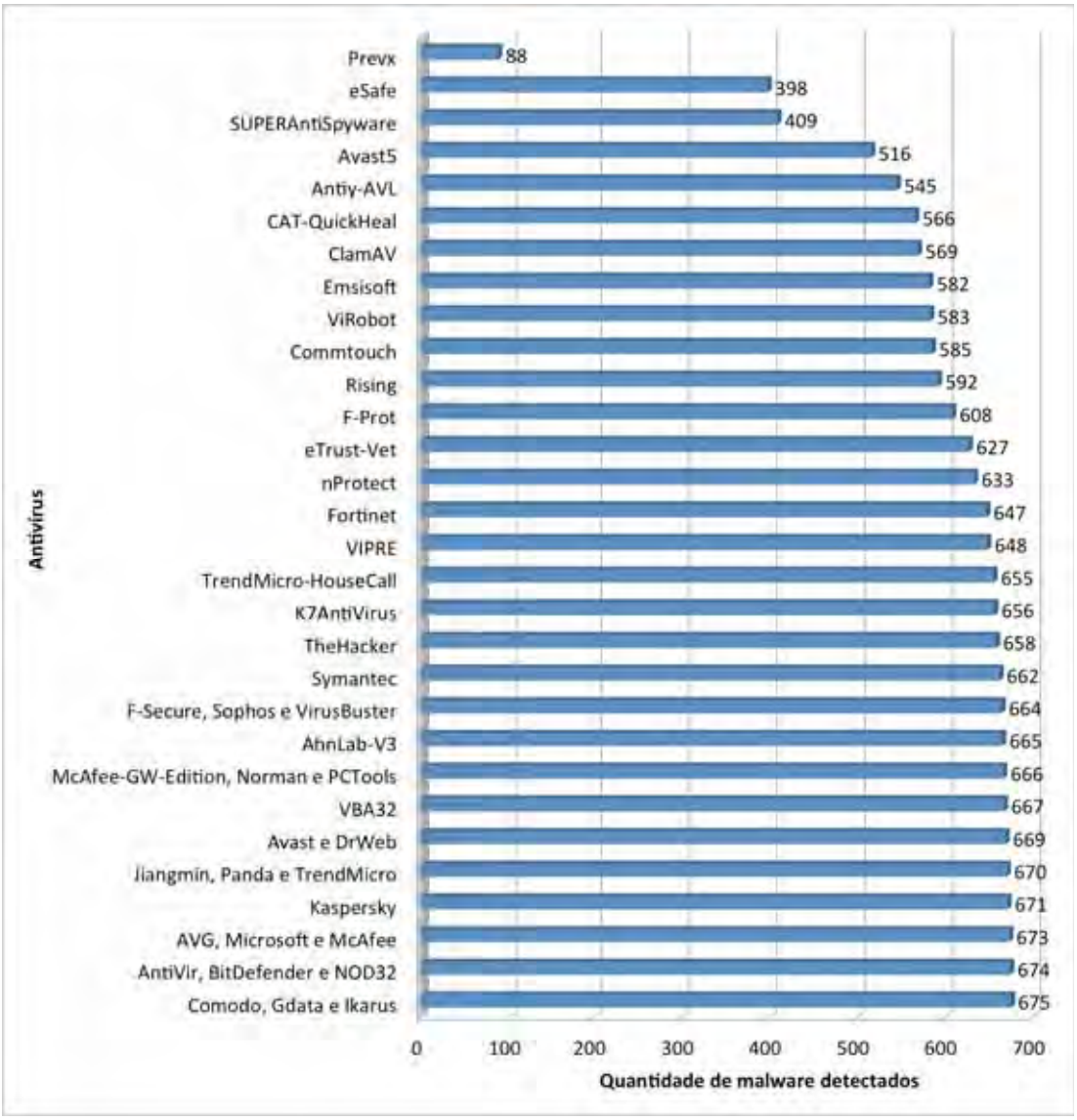

Figura 20 Gráfico comparativo do resultado do vírus total.

Como pode-se observar na Figura 20, os antivírus com maior porcentagem de acerto foram Ikarus, Gdata e Comodo (98.25%) e o antivírus com pior taxa de acerto foi o Prevx, com 12,81%. Além disso, somente 23 *malware* foram detectados por todos os antivírus. A Figura 21 ilustra a porcentagem de *malware* de acordo com o seu tipo baseado na classificação obtida pelo antivírus AntiVir para os *malware* com assinatura. Esse antivírus foi escolhido pois está presente em quase todos os relatórios providos pelo Virus Total. Nota-se que as classes de *malware* que mais aparecem são *Worm*, Cavalo de Troia (*Trojan Horse*) e Vírus.

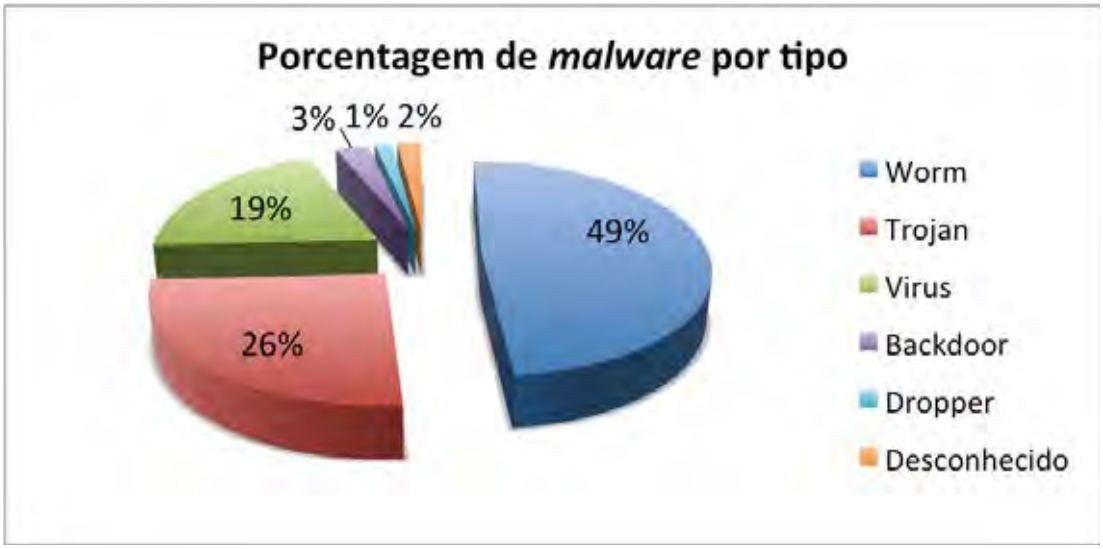

Figura 21 Porcentagem de *malware* por tipo.

Todos os *malware* com assinatura também foram submetidos ao sistema Anubis (ANUBIS, 2009). Esse sistema analisa o comportamento de um determinado executável Windows e gera um relatório com as ações e informações sobre o arquivo analisado. Este relatório contém uma classificação do risco para cada ação provocada pelo *malware*. Exemplos de ações que recebem uma classificação são: mudanças na configurações do *Firewall*, arquivos modificados, atividades realizados nos registradores, etc. Baseado nesta classificação, os *malware* foram classificados em risco baixo, médio, alto e desconhecido. Os *malware* que o sistema Anubis não conseguiu analisar foram classificados com risco desconhecido. A Figura 22 ilustra a porcentagem de *malware* por risco. Note que a maioria dos *malware* possui alto risco.

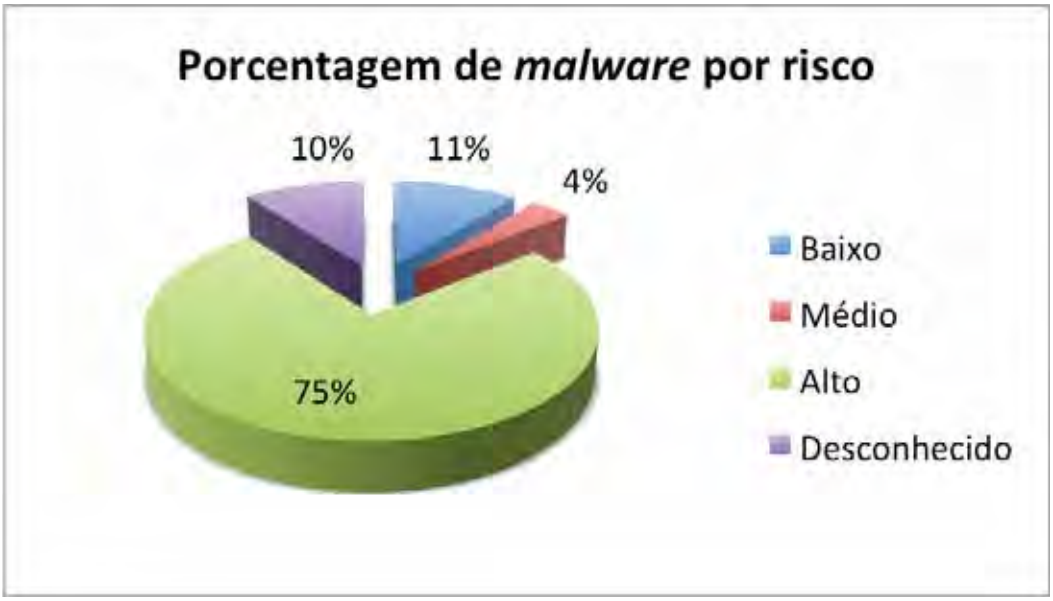

Figura 22 Porcentagem de *malware* por risco.

Com as informações obtidas pelos sistemas Anubis e Virus Total foi possível relacionar a quantidade de *malware* de acordo com seu risco que cada antivírus detectou (Tabela 4). Note que os antivírus Ikarus, BitDefender, NOD32 detectaram todos os *malware* com risco baixo, médio e alto. O antivírus SUPERAntiSpyware teve baixa taxa de detecção para os *malware* com risco baixo (5,4%), médio (12,5%) e desconhecido (26,76%).

Tabela 4 Quantidade de *malware* detectado por antivírus de acordo com o risco.

Antivírus	**Baixo**	**Médio**	**Alto**	**Desconhecido**
Comodo	73	24	518	60
GData	73	24	518	60
Ikarus	74	24	518	59
AntiVir	74	24	517	59
BitDefender	74	24	518	58
NOD32	74	24	518	58
AVG	73	23	518	59
Microsoft	74	24	517	58
McAfee	73	24	516	60

Antivírus	Baixo	Médio	Alto	Desconhecido
Kaspersky	74	24	515	58
Jiangmin	73	24	514	59
Panda	73	24	514	59
TrendMicro	73	24	515	58
Avast	74	24	513	58
DrWeb	73	23	513	60
VBA32	73	24	513	57
McAfee-GW-Edition	74	24	510	58
Norman	71	24	513	58
PCTools	74	24	514	54
AhnLab-V3	71	24	515	55
F-Secure	71	24	514	55
Sophos	74	24	509	57
VirusBuster	74	24	510	58
Symantec	73	24	509	56
TheHacker	69	24	506	59
K7AntiVirus	71	24	508	53
TrendMicro-HouseCall	71	23	504	57
VIPRE	71	23	498	56
Fortinet	68	22	504	53
nProtect	72	19	490	52
eTrust-Vet:	70	23	493	41
F-Prot	62	20	490	36
Rising	69	21	464	38
Commtouch	62	18	471	34
ViRobot	49	17	481	36
Emsisoft	40	17	478	47
ClamAV	46	17	470	36
CAT-QuickHeal	73	23	415	55

Antivírus	Baixo	Médio	Alto	Desconhecido
Antiy-AVL	58	22	413	52
Avast5	71	22	371	52
SUPERAntiSpyware	4	3	383	19
eSafe	33	14	319	32
Prevx	9	1	72	6
Total	74	24	518	71

5.3 Ambiente simulado

Um ambiente monitorado simulado foi criado para validar a abordagem proposta e os resultados promissores, a fim de demonstrar que tanto o agente de comportamento quanto o de rede, são eficientes na detecção de um ataque de *malware* ao emitir alertas para amostras de *malware* cujo comportamento padrão correspondesse a uma assinatura armazenada no banco de dados.

Os agentes tanto de rede quanto de comportamento analisam os fluxos de dados e as chamadas de sistema, respectivamente, a cada minuto. Os dados analisados são os que ocorreram nos últimos 30 minutos.

Os testes realizados demonstraram que tanto o detector de comportamento quanto o detector de rede são eficientes na detecção de um artefato malicioso visto que emitiram alertas para os exemplares de *malware* que ainda estão ativos e cujo comportamento corresponde a uma assinatura presente no banco de dados.

Dessa forma, nos casos em que a detecção foi realizada com sucesso, os agentes enviaram um e-mail de alerta para notificar o administrador da rede qual(is) o(s) possível(is) *malware* está(ão) gerando ações maléficas em sua rede.

5.4 Considerações finais

Esse capítulo relatou alguns dados estatísticos referente ao sistema de coleta distribuído e dados estatísticos providos pelo serviço Virus Total referentes às assinaturas geradas. Além disso, apresentou alguns testes realizados em um ambiente simulado que foi monitorado. No próximo capítulo algumas considerações sobre o desenvolvimento do projeto e sobre trabalhos futuros são feitas.

Capítulo 6 – Conclusão

A ameaça de ataques à sistemas computacionais por meio de programas maliciosos, aliada a uma imensa quantidade de variantes de *malware* que surgem diariamente, faz com que novos métodos de proteção tenham que ser desenvolvidos. Assim, foi proposto um sistema de coleta, análise e detecção de *malware* que obtém os fluxos de rede e os comportamentos de um programa malicioso (chamada de sistema) durante a execução deste no sistema operacional. Estes dois tipos de informações são utilizados na confecção de assinaturas específicas, as quais são armazenadas em um banco de dados.

O sistema proposto inspira-se no sistema imunológico humano, cuja principal vantagem é a adaptabilidade no tratamento de novos problemas. A fim de popular um conhecimento inicial sobre ataques, foram analisados centenas de exemplares de *malware*, dos quais foram extraídas as assinaturas de comportamento e de rede. O conhecimento gerado e armazenado neste banco é utilizado na detecção de ataques em ambientes reais. Para isso, desenvolveu-se também agentes que monitoram as atividades de rede e do computador em um ambiente que se deseje proteger, os quais enviam alertas por meio de mensagens.

A Tabela 5 sumariza a analogia entre o sistema imunológico humano e o sistema proposto, sendo explicada em mais detalhes em seguida:

Tabela 5 Analogia entre o sistema imunológico humano e o sistema proposto.

Sistema Imunológico Humano	**Sistema detector de *malware* proposto**
Antígenos	*Malware*
Seleção negativa	Geração de assinatura
Receptores das células que identificam um antígeno	Assinaturas de comportamento e de rede
Células que identificam um antígeno	Agentes de rede e comportamento
Resposta imunológica	Relatório de detecção
Mecanismos de aprendizado e memória	Banco de dados de assinatura

- Os *malware* que ameaçam os sistemas computacionais são com os antígenos que atacam o ser humano. Para se defender desses antígenos, o sistema imunológico humano realiza uma seleção negativa para diferenciar as substâncias *self* de *nonself*. O resultado desse processo são receptores que identificam os antígenos. Essa seleção negativa inspirou o processo de geração de assinaturas que armazenam as características de um *malware*. Portanto, as assinaturas funcionam com os receptores para identificar um código malicioso;
- Para identificar um antígeno, células circulam pelo corpo humano para identificar qualquer substância estranha. Essa é a função dos agentes de rede e de comportamento que analisam os dados de um sistema computacional para detectar *malware*;
- Caso o sistema imunológico detecte algum antígeno, uma resposta imunológica é acionada para realizar a identificação e remoção do antígeno. O mesmo ocorre com o sistema proposto. Os agentes de rede e comportamento geram um relatório de detecção com informações sobre o *malware* identificado e como removê-lo;

- Por fim, o sistema imunológico humano possui mecanismos de aprendizado e memória que ajudam na identificação de futuras ocorrências de tais antígenos. Essa característica é realizada pelo banco de dados de assinaturas no sistema proposto. Esse banco possibilita a detecção de novos ataques de *malware*.

6.1 Problemas encontrados durante o desenvolvimento da pesquisa

Alguns problemas foram encontrados durante o desenvolvimento dessa pesquisa. A principal dificuldade foi encontrar um modelo de rede que fosse eficiente para a detecção de códigos maliciosos visto que esses programas mudam algumas características de seu tráfego como por exemplo, endereço IP de destino, para que a sua detecção seja mais complexa. Esse problema foi solucionado após várias análises de tráfego de rede de diferentes artefatos.

Outro problema encontrado foi o tempo dispendido com as três execuções de um artefato no BehEMOT para que os dados extraídos pudessem ser utilizados na geração de uma assinatura. A solução encontrada para esse problema foi realizar as análises em paralelo.

A princípio, o agente de rede utilizaria rede neural do tipo MLP (*Multi Layer Perceptron*) (HAYKIN, 2001) para realizar a segregação dos fluxos entre normais e suspeitos. O algoritmo utilizado para o treinamento foi *Backpropagation* (WILKINSON, MIGHELL e GOODMAN, 1989). No primeiro teste realizado foram utilizados 299760 dados de entrada sendo 99,68% dados normais e 0,32% dados suspeitos. Os dados normais foram obtidos durante 30 minutos de fluxos de um dia normal. No primeiro e segundo teste a arquitetura da rede neural MLP utilizada foi “3:2:2”, isto é, 3 entradas, 2 neurônios na camada escondida e 2 neurônios de saída que indicam as classes. A topologia da rede neural utilizada está ilustrada na Figura 23.

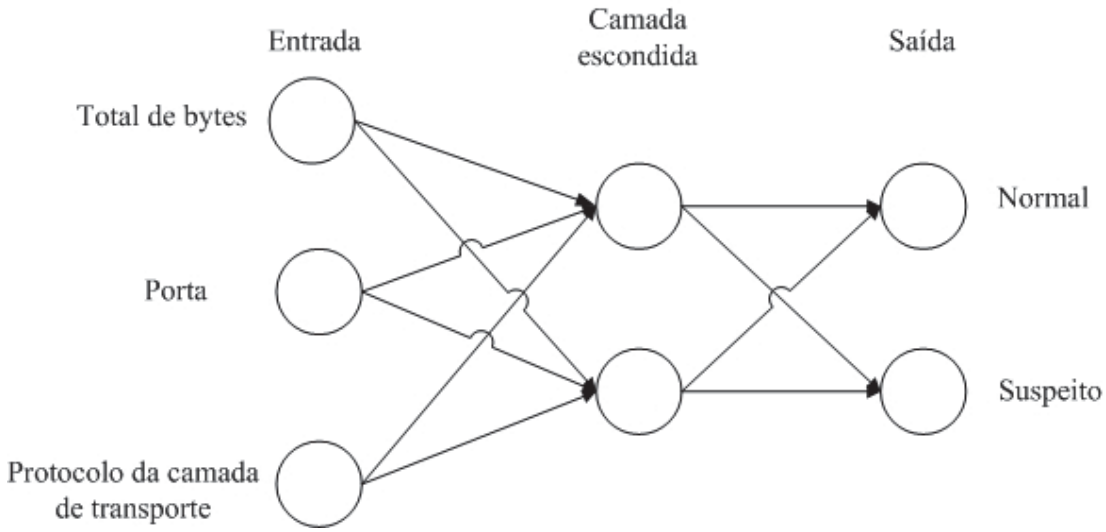

Figura 23 Arquitetura da rede MPL utilizada no primeiro e segundo teste.

Foi utilizado o programa WEKA (*Waikato Environment for Knowledge Analysis*) [WF00] para criar e fazer os testes com a rede neural MLP. Em uma máquina com 4GB de memória RAM e processador Intel Core 2 Duo com velocidade 2.26 GHz, o tempo dispendido pelo WEKA para construir a rede foi de aproximadamente 207,17 segundos e o tempo para testar o modelo foi de aproximadamente 1,52 segundos. Os resultados obtidos mostraram que 99,6827% foram classificados corretamente. No entanto, como ilustra a matriz de confusão (Figura 24) todos os dados suspeitos foram erroneamente classificados como normais (falso negativo).

a	b	Classificado como
298809	0	a = normal
951	0	b = suspeito

Figura 24 Matriz de confusão do primeiro teste.

Com o mesmo conjunto de entrada outros testes foram realizados variando o número de neurônios na camada escondida. Os resultados obtidos foram similares ao teste com 2 neurônios na camada escondida.

Como a rede estava viciada devida ao alto número de dados normais um segundo teste foi realizado. Nesse segundo teste, foram utilizados 2819 dados de entrada sendo que 66,27% dos dados são normais e 33,73%

suspeitos. Os dados normais foram obtidos durante 1 minuto de fluxos de um dia normal. Em uma máquina com 4GB de memória RAM e processador Intel Core 2 Duo com velocidade 2.26 GHz, o tempo dispendido pelo WEKA para construir a rede foi de aproximadamente 3,9 segundos e o tempo para testar o modelo foi de aproximadamente 0,13 segundos. Os resultados obtidos mostraram que 92,84% dos dados foram classificados corretamente e 7,16% dos dados foram classificados incorretamente. A Figura 25 ilustra a matriz de confusão desse teste. Na sua primeira linha é possível notar que 171 dados de entrada foram falsos positivo, ou seja, 171 dados normais foram erroneamente classificados como suspeitos. Na segunda linha é possível notar que 31 dados de entrada foram falsos negativo, ou seja, 31 dados suspeitos foram erroneamente classificados como normais.

a	b	Classificado como
1697	171	a = normal
31	920	b = suspeito

Figura 25 Matriz de confusão do segundo teste

Um terceiro teste foi executado para tentar melhorar ainda mais a rede neural. Nesse terceiro teste havia somente dois dados de entrada (total de bytes e porta), ou seja, a arquitetura da rede neural MLP utilizada foi “2:2:2”, isto é, 2 entradas, 2 neurônios na camada escondida e 2 neurônios de saída que indicam as classes (Figura 26). Como entrada de dados foram utilizados os mesmos dados do segundo teste sem os dados referentes ao protocolo da camada de transporte. O tempo dispendido para construção e teste do modelo e os resultados obtidos foram similares aos resultados do segundo teste como pode ser notado pela matriz de confusão ilustrada na Figura 27.

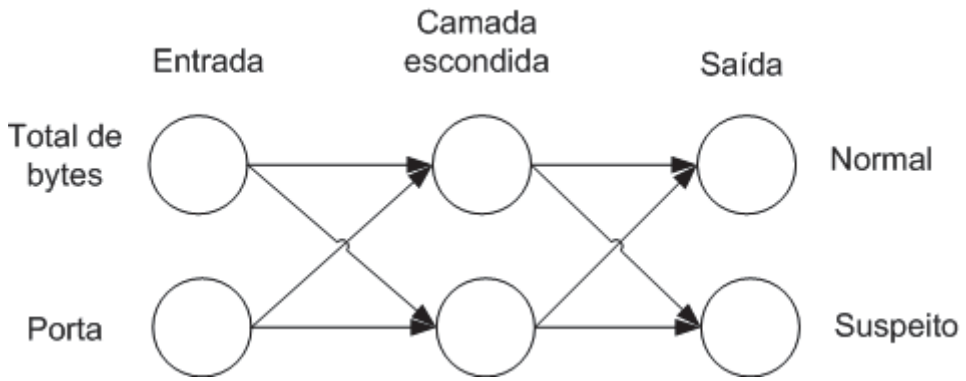

Figura 26 Arquitetura da rede MPL utilizada no terceiro teste.

a	b	Classificado como
1661	188	a = normal
28	923	b = suspeito

Figura 27 Matriz de confusão do terceiro teste

No segundo e terceiro testes se obteve melhores resultados que no primeiro teste visto que houve pouco falso positivo. No entanto, na metodologia desenvolvida, o detector de rede compara somente os dados classificados como suspeitos com as assinaturas e para emitir um alerta é necessário que todos os passos de uma assinatura ocorram. Caso um dado suspeito seja erroneamente classificado como normal o detector não irá compará-lo com as assinaturas e o *malware* que possuir em sua assinatura esse dado suspeito não poderá ser detectado. Tendo em vista esse problema, alguns testes foram realizados com mineração de dados e observou-se que a árvore de decisão com o algoritmo RandomTree solucionava esse problema.

Um problema em relação a árvore de decisão é a necessidade de periodicamente refazer a etapa de treinamento para atualizar a árvore de decisão. Para minimizar esse problema, scripts foram desenvolvidos de forma a automatizar esse processo.

6.2 Trabalhos futuros e publicações

Alguns trabalhos futuros são:

- Uma das limitações do projeto desenvolvido é a plataforma de análise, uma vez que é restrita a sistemas operacionais Windows XP. Portanto, como trabalho futuro, pode-se generalizar o processo de detecção de *malware* para abranger *malware* cujo alvos são outras plataformas.
- Criar um sistema colaborativo para as assinaturas. O intuito é adicionar mais um módulo no sistema web desenvolvido de forma que outras pessoas também possam adicionar assinaturas no banco de dados.
- Desenvolver um módulo de distribuição de assinaturas de maneira que todos as pessoas que utilizem o sistema proposto tenham acesso as assinaturas atualizadas e o mais rápido possível.
- Aplicar a metodologia apresentada nesse trabalho em um ambiente real, de médio ou grande porte.

Por fim, este trabalho produziu como resultado o artigo publicado na Conferência Ibero-Americana WWW/Internet 2011 (IADIS) (OLIVEIRA, GRÉGIO e CANSIAN, 2011). O artigo foi premiado como um dos melhores artigos na conferência. Além disso, o artigo intitulado "*A Malware Detection System Inspired on the Human Immune System*" foi aceito para publicação no *12th International Conference on Computational Science and Applications* (ICCSA 2012).

Referências Bibliográficas

ALMEIDA, T. A.; YAMAKAMI, A.; TAKAHASHI, M. T. Sistema imunológico artificial para resolver o problema da árvore geradora mínima com parâmetros fuzzy. **Pesquisa Operacional** vol.27 no.1 Rio de Janeiro Jan./Apr. 2007.

ANUBIS. About Anubis. 2009. Disponível em: <http://anubis.iseclab.org/?action=about>. Acesso em: 09 Abr. 2012.

AVIRA. Virus Glossary. 2012. Disponível em: <http://www.avira.com/en/support-about-malware>. Acesso em: 02 Fev. 2012.

BIN, L. et al. A NetFlow based flow analysis and monitoring system in enterprise networks. **Computer Networks**, v. 52, n. 5, p. 1074-1092, 2008. ISSN 1389-1286.

CERON, J. M.; GRANVILLE, L.; TAROUCO, L. Taxonomia de Malwares: Uma Avaliação dos Malwares Automaticamente Propagados na Rede. **IX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)** - Campinas – SP 2009, pp. 43-56.

CERT. Cartilha de Segurança para a Internet - Parte VIII: Códigos Maliciosos (Malware). 2006a. Disponível em: < http://cartilha.cert.br/download/cartilha-08-malware.pdf >. Acesso em: 02 Fev. 2012.

CERT. Cartilha de Segurança para Internet - Part I: Conceitos de Segurança. 2006b. Disponível em: <http://cartilha.cert.br/download/cartilha-01-conceitos.pdf >. Acesso em: 02 Fev. 2012.

CISCO. Introduction to Cisco IOS NetFlow - A Technical Overview. 2007. Disponível em: < http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.pdf>. Acesso em: 02 Fev. 2012.

CRISPIN, M. RFC3501 – Internet Message Access Protocol – Version 4, 2003. Disponível em: <http://www.faqs.org/rfcs/rfc3501.html>. Acesso em : 02 Fev. 2012.

CLAISE, B. RFC 3954: Cisco Systems NetFlow Services Export Version 9. 2004. Disponível em: < http://www.ietf.org/rfc/rfc3954.txt >. Acesso em: 02 Fev. 2012.

COHEN, F. **Computer Viruses**. Computers & Security, 6:22-35, 1987.

CORRÊA, J. L. et al. Detectando eventos em redes utilizando um modelo de rastreamento de fluxos baseado em assinaturas. **IX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. Campinas - SP**:** 189 a 202 p. 2009.

CORRÊA, J. L.; PROTO, A.; CANSIAN, A. M. Modelo de armazenamento de fluxos de rede para análises de tráfego e de segurança. **VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)**. Gramado - RS 2008, v. 8. pp. 73-86.

DASGUPTA, D; NIÑO, L. F. **Immunological Computation – Theory and Applications**. Taylor & Francis Group, 2008. 296. ISBN:1420065459

DEERING, S. et. al. RFC 2460: Internet Protocol, Version 6 (IPv6) Specification. 1998. Disponível em: <http://tools.ietf.org/html/rfc2460>. Acesso em: 02 de Fev. de 2012.

DIERKS, T. et. al. RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1. 2006. Disponível em: <http://tools.ietf.org/html/rfc4346>. Acesso em: 02 de Fev. de 2012.

DIONAEA. Dionaea catches bugs. 2012. Disponível em: <http://dionaea.carnivore.it/>. Acesso em: 02 de Fev. de 2012.

EASTLAKE, D. e HANSEN, T. RFC: 4634: RFC: 3174: US Secure Hash Algorithm 1 (SHA and HMAC-SHA). 2006 Disponível em: < http://tools.ietf.org/html/ rfc4634>. Acesso em: 02 Fev. 2012.

EASTLAKE, D. e JONES, P. RFC: 3174: US Secure Hash Algorithm 1 (SHA1). 2001. Disponível em: < http://tools.ietf.org/html/rfc3174>. Acesso em: 02 Fev. 2012.

FERNANDES FILHO, D. S. F. et. al. Análise Comportamental de Código Malicioso através da Monitoração de Chamadas de Sistema e Tráfego de Rede**.** In: **X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)**. Fortaleza - CE 2010, pp. 311-324.

FORREST, S. et al**.** Self-nonself Discrimination in a Computer. In **Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy**, Los Alamos, CA, 1994. IEEE Computer Society Press, pp. 202 – 212.

GRÉGIO, A. R. A. et al. Malware distributed collection and pre-classification system using honeypot technology, In: **Proceedings of the SPIE**, Volume 7344, pp. 73440B-73440B-8. 2009.

GRÉGIO, A. R. A. 2007. **Aplicação de técnicas de Data Minings para a análise de logs de tráfego TCP/IP**. In: Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada. São José dos Campos.

HAN, J e KAMBER, M. 2006**. Data Mining Concepts and Techniques**. Morgan Kaufmann Publishers. 550. ISBN 1-55860-901-6.

HAYKIN, S. **Redes Neurais Princípios e Prática**. Editora Bookman, 2ª edição, Porto Alegre – RS, 2001.

HE, Y. et. al. 2010. **A Model of Collaborative Artificial Immune System.** In: 2nd International Asia Conference on Informatics in Control, Automation and Robotics. Volume 3, 101-104 p.

HONEYNET PROJECT. About The Honeynet Project. 2012. Disponível em: < http://honeynet.org/about>. Acesso em: 02 Fev. 2012.

HSIAO, H. W.; CHEN, D. N. e WU, T. J. Detecting Hiding Malicious Website Using Network Traffic Mining Approach. In: **2nd International Conference on Education Technology and Computer (ICETC)**, Volume 5, pp. V5-276 - V5-280. 2010.

KEPHART, J. O. A Biologically Inspired Immune System for Computers. In **Artificial Life IV**, 1994. MIT Press. pp. 130-139.

LEE, H. KIM, W. e HONG, M. Biologically inspired computer virus detection system , **BioADIT 2004, LNCS 3141**, 2004, pp. 153-165.

LIBEMU. Libemu – x86 Shellcode Emulation. 2012. Disponível em: < http://libemu.carnivore.it/>. Acesso em: 02 Fev. 2012.

MICROSOFT. Virus:Win32/Sality. 2008. Disponível em: < http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Virus%3aWin32%2fSality >. Acesso em: 10 Fev. 2012

MICROSOFT. Virus:Win32/Virut.E. 2010. Disponível em: < http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Virus%3AWin32%2FVirut.E >. Acesso em: 10 Fev. 2012.

MYERS, J. RFC1939 - Post Office Protocol - Version 3, 1996. Disponível em: < http://tools.ietf.org/html/rfc1939 >. Acesso em: 02 Fev. 2012.

NEPENTHES. Nepenthes – Finest Collection. 2009. Disponível em: < http://nepenthes.carnivore.it/documentation>. Acesso em: 02 de Fev. de 2012.

OLIVEIRA, I. L. GRÉGIO, A. R. E CANSIAN, A. M. Sistema de coleta, análise e detecção de código malicioso baseado no sistema imunológico

humano. In: **Conferência IADIS Ibero-Americana WWW/Internet 2011 (IADIS)**, Rio de Janeiro.

PANDA SECURITY. Annual Report PandaLabs 2010. 2011a. Disponível em: <http://press.pandasecurity.com/wp-content/uploads/2010/05/PandaLabs-Annual-Report-2010.pdf>. Acesso em: 02 Fev. 2012.

PANDA SECURITY. Quarterly Report PandaLabs January–March 2011. 2011b. Disponível em: <http://press.pandasecurity.com/wp-content/uploads/2011/04/PandaL abs-Report-Q1-2011.pdf>. Acesso em: 02 Fev. 2012.

PANDA SECURITY. PandaLabs Annual Report 2011. 2011c. Disponível em: <http://press.pandasecurity.com/wp-content/uploads/2012/01/Annual-Report-PandaL abs-2011.pdf>. Acesso em: 02 Fev. 2011.

PAULA, F. S. **Uma arquitetura de segurança computacional inspirada no sistema imunológico**. 2004. In: Tese de Doutorado (Doutor em Ciência da Computação) – UNICAMP – Instituto de Computação.

PERDISCI, R, LEEA, W., e FEAMSTER, N. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. 2010. In: **NSDI'10 Proceedings of the 7th USENIX conference on Networked systems design and implementation**. pp. 391-404.

REIS, M. A. **Forense computacional e sua aplicação em segurança imunológica**. 2003. In: Dissertação de Mestrado (Mestre em Ciência da Computação) – UNICAMP – Instituto de Computação.

RIVEST, R. RFC: 1321: The MD5 Message-Digest Algorithm. 1992. Disponível em: < http://tools.ietf.org/html/rfc1321 >. Acesso em: 02 Fev. 2012.

RNP. Como identificar e remover o *malware* Conficker (Downadup ou Kido). 2009. Disponível em: <http://www.rnp.br/cais/alertas/2009/cais-alr-2009-2a.html>. Acesso em: 10 Fev. 2012.

RUMELHART, D. E.; HINTON, G. E. E WILLIAMS, R. J. **Learning representations by back-propagating erros**. Nature, 323:533-536, 1986.

RUMELHART, D. E. e MCCLELLAND, J. L. **Parallel Distributed Processing**, volume 1: Foundations. The MIT Press, 1986.

SKOUDIS, E.; ZELTSER, L. **Malware Fighting Malicious Code**. Prentice Hall PTR, 2004. 647. ISBN 0-13-101405-6.

SOMAYAJI, A. HOFMEYR, S. A. e FORREST, S. **Principles of a Computer Immune System**. In Proceedings of the Second New Security Paradigms Workshop, 1997, pp. 75-82.

VIRUS TOTAL. Virus Total. 2012. Disponível em: http://www.virustotal.com/about.html. Acesso em: 02 Fev. 2012.

ZHANG, Y. LI, T. e QIN, T. A Dynamic Immunity-based Model for Computer Virus Detection. In **International Symposiums on Information Processing (ISIP),** 2008, pp. 515 - 519.

WILKINSON, T. S.; MIGHELL, D. A. e GOODMAN, J. W. **Backpropagation and its application to handwritten signature verification**. In Book: Advances in neural information processing systems 1, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1989.

WITTEN, I. H.; FRANK, E. 2000. **Data mining: practical machine learning tools and techniques with java implementations**. New York, NY, USA: Morgan Kaufmann Publishers. 629.

Anexo A – Árvore de decisão utilizada no trabalho

Abaixo está a árvore de decisão criada e utilizada no agente de rede. A classificação ‘0’ indica normal e a ‘1’ suspeito.

```
port < 447
|   length_packet < 144.5
|   |   port < 52.5
|   |   |   port < 2.5 : 0 (94/0)
|   |   |   port >= 2.5
|   |   |   |   port < 4 : 1 (1/0)
|   |   |   |   port >= 4 : 0 (25/0)
|   |   port >= 52.5
|   |   |   length_packet < 96.5
|   |   |   |   length_packet < 95.5
|   |   |   |   |   length_packet < 75.5
|   |   |   |   |   |   port < 66.5
|   |   |   |   |   |   |   length_packet < 55.5
|   |   |   |   |   |   |   |   length_packet < 51.5
|   |   |   |   |   |   |   |   |   length_packet < 45 : 0 (1/0)
|   |   |   |   |   |   |   |   |   length_packet >= 45 : 1 (5/1)
|   |   |   |   |   |   |   |   length_packet >= 51.5 : 0 (3/0)
|   |   |   |   |   |   |   length_packet >= 55.5
|   |   |   |   |   |   |   |   length_packet < 62.5
|   |   |   |   |   |   |   |   |   length_packet < 60.5
|   |   |   |   |   |   |   |   |   |   length_packet < 58.5
|   |   |   |   |   |   |   |   |   |   |   length_packet < 57.5 : 1 (2/0)
|   |   |   |   |   |   |   |   |   |   |   length_packet >= 57.5 : 1 (12/1)
|   |   |   |   |   |   |   |   |   |   length_packet >= 58.5
|   |   |   |   |   |   |   |   |   |   |   length_packet < 59.5 : 0 (1/0)
|   |   |   |   |   |   |   |   |   |   |   length_packet >= 59.5 : 1 (4/1)
|   |   |   |   |   |   |   |   |   length_packet >= 60.5
|   |   |   |   |   |   |   |   |   |   length_packet < 61.5 : 1 (127/1)
```

```
|  |  |  |  |  |  |  |  |  length_packet >= 61.5 : 1 (45/1)
|  |  |  |  |  |  |  |  length_packet >= 62.5
|  |  |  |  |  |  |  |  |  length_packet < 73.5
|  |  |  |  |  |  |  |  |  |  length_packet < 72.5
|  |  |  |  |  |  |  |  |  |  |  length_packet < 69.5
|  |  |  |  |  |  |  |  |  |  |  |  length_packet < 67.5
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 64.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 63.5 : 1 (5/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 63.5 : 1 (6/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 64.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 65.5 : 1 (10/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 65.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 66.5 : 1 (6/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 66.5 : 1 (8/1)
|  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 67.5
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 68.5 : 0 (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 68.5 : 1 (3/1)
|  |  |  |  |  |  |  |  |  |  |  length_packet >= 69.5 : 1 (3/0)
|  |  |  |  |  |  |  |  |  |  length_packet >= 72.5 : 0 (1/0)
|  |  |  |  |  |  |  |  |  length_packet >= 73.5
|  |  |  |  |  |  |  |  |  |  length_packet < 74.5 : 1 (46/1)
|  |  |  |  |  |  |  |  |  |  length_packet >= 74.5 : 1 (1/0)
|  |  |  |  |  |  port >= 66.5
|  |  |  |  |  |  |  length_packet < 49.5
|  |  |  |  |  |  |  |  prot < 11.5
|  |  |  |  |  |  |  |  |  length_packet < 40.5
|  |  |  |  |  |  |  |  |  |  port < 90 : 1 (95/1)
|  |  |  |  |  |  |  |  |  |  port >= 90
|  |  |  |  |  |  |  |  |  |  |  port < 409 : 0 (2/0)
|  |  |  |  |  |  |  |  |  |  |  port >= 409 : 1 (48/1)
|  |  |  |  |  |  |  |  |  length_packet >= 40.5
|  |  |  |  |  |  |  |  |  |  length_packet < 46.5 : 0 (6/0)
|  |  |  |  |  |  |  |  |  |  length_packet >= 46.5
|  |  |  |  |  |  |  |  |  |  |  port < 109.5 : 0 (1/0)
|  |  |  |  |  |  |  |  |  |  |  port >= 109.5
|  |  |  |  |  |  |  |  |  |  |  |  port < 291 : 1 (33/1)
|  |  |  |  |  |  |  |  |  |  |  |  port >= 291
|  |  |  |  |  |  |  |  |  |  |  |  |  port < 444 : 0 (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  port >= 444 : 1 (31/1)
|  |  |  |  |  |  |  |  prot >= 11.5 : 0 (9/0)
|  |  |  |  |  |  |  length_packet >= 49.5 : 0 (25/0)
|  |  |  |  |  length_packet >= 75.5
|  |  |  |  |  |  length_packet < 80.5
|  |  |  |  |  |  |  prot < 11.5
|  |  |  |  |  |  |  |  port < 66.5 : 0 (1/0)
|  |  |  |  |  |  |  |  port >= 66.5
|  |  |  |  |  |  |  |  |  port < 261.5 : 1 (3/1)
|  |  |  |  |  |  |  |  |  port >= 261.5 : 0 (1/0)
|  |  |  |  |  |  |  prot >= 11.5 : 0 (10/0)
```

```
|   |   |   |   |   |   length_packet >= 80.5 : 0 (31/0)
|   |   |   |   length_packet >= 95.5
|   |   |   |   |   port < 109.5 : 0 (2/0)
|   |   |   |   |   port >= 109.5
|   |   |   |   |   |   port < 291 : 1 (95/1)
|   |   |   |   |   |   port >= 291
|   |   |   |   |   |   |   port < 444 : 0 (1/0)
|   |   |   |   |   |   |   port >= 444 : 1 (79/1)
|   |   |   length_packet >= 96.5
|   |   |   |   length_packet < 143.5
|   |   |   |   |   length_packet < 127.5 : 0 (74/0)
|   |   |   |   |   length_packet >= 127.5
|   |   |   |   |   |   length_packet < 128.5
|   |   |   |   |   |   |   port < 150
|   |   |   |   |   |   |   |   port < 66.5 : 1 (1/0)
|   |   |   |   |   |   |   |   port >= 66.5
|   |   |   |   |   |   |   |   |   port < 109.5 : 0 (1/0)
|   |   |   |   |   |   |   |   |   port >= 109.5 : 1 (5/1)
|   |   |   |   |   |   |   port >= 150 : 0 (2/0)
|   |   |   |   |   |   length_packet >= 128.5 : 0 (33/0)
|   |   |   |   length_packet >= 143.5
|   |   |   |   |   port < 66.5 : 0 (1/0)
|   |   |   |   |   port >= 66.5
|   |   |   |   |   |   port < 80.5 : 1 (8/1)
|   |   |   |   |   |   port >= 80.5
|   |   |   |   |   |   |   port < 291
|   |   |   |   |   |   |   |   port < 110 : 1 (1/0)
|   |   |   |   |   |   |   |   port >= 110 : 1 (70/1)
|   |   |   |   |   |   |   port >= 291
|   |   |   |   |   |   |   |   port < 444 : 0 (1/0)
|   |   |   |   |   |   |   |   port >= 444 : 1 (50/1)
|   length_packet >= 144.5
|   |   length_packet < 2160.5
|   |   |   length_packet < 2095.5
|   |   |   |   length_packet < 1200.5
|   |   |   |   |   length_packet < 1199.5
|   |   |   |   |   |   port < 80.5
|   |   |   |   |   |   |   prot < 11.5
|   |   |   |   |   |   |   |   length_packet < 347.5
|   |   |   |   |   |   |   |   |   length_packet < 346
|   |   |   |   |   |   |   |   |   |   length_packet < 194
|   |   |   |   |   |   |   |   |   |   |   length_packet < 182 : 0 (22/0)
|   |   |   |   |   |   |   |   |   |   |   length_packet >= 182
|   |   |   |   |   |   |   |   |   |   |   |   length_packet < 183.5 : 1 (1/0)
|   |   |   |   |   |   |   |   |   |   |   |   length_packet >= 183.5
|   |   |   |   |   |   |   |   |   |   |   |   |   port < 1.5 : 0 (1/0)
|   |   |   |   |   |   |   |   |   |   |   |   |   port >= 1.5
|   |   |   |   |   |   |   |   |   |   |   |   |   |   length_packet < 191.5
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   port < 41.5 : 1 (1/0)
```

```
|  |  |  |  |  |  |  |  |  |  |  |  |  port >= 41.5 : 0 (3/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 191.5 : 1 (1/0)
|  |  |  |  |  |  |  |  |  |  length_packet >= 194
|  |  |  |  |  |  |  |  |  |  length_packet < 314.5 : 0 (85/0)
|  |  |  |  |  |  |  |  |  |  length_packet >= 314.5
|  |  |  |  |  |  |  |  |  |  |  length_packet < 315.5
|  |  |  |  |  |  |  |  |  |  |  |  port < 66.5 : 0 (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  port >= 66.5 : 1 (1/0)
|  |  |  |  |  |  |  |  |  |  |  length_packet >= 315.5 : 0 (26/0)
|  |  |  |  |  |  |  |  |  length_packet >= 346 : 1 (29/1)
|  |  |  |  |  |  |  |  length_packet >= 347.5
|  |  |  |  |  |  |  |  |  length_packet < 926.5
|  |  |  |  |  |  |  |  |  |  length_packet < 925.5
|  |  |  |  |  |  |  |  |  |  |  length_packet < 675.5
|  |  |  |  |  |  |  |  |  |  |  |  length_packet < 552.5
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 422.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 421
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 384.5 : 0 (43/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 384.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 385.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  port < 52.5 : 0 (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  port >= 52.5 : 1 (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 385.5 : 0 (39/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 421 : 1 (4/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 422.5 : 0 (143/0)
|  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 552.5
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 555.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  port < 52.5 : 0 (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  port >= 52.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 554.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 553.5 : 1 (3/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 553.5 : 0 (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 554.5 : 1 (6/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 555.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 674.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 589.5 : 0 (35/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 589.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 590.5 : 1 (5/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 590.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 641.5 : 0 (59/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 641.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet < 642.5 : 1 (5/1)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 642.5 : 0
(33/0)
|  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 674.5 : 1 (3/1)
|  |  |  |  |  |  |  |  |  |  length_packet >= 675.5 : 0 (254/0)
|  |  |  |  |  |  |  |  |  length_packet >= 925.5 : 1 (12/1)
|  |  |  |  |  |  |  |  length_packet >= 926.5 : 0 (275/0)
|  |  |  |  |  |  prot >= 11.5
```

```
|   |   |   |   |   |   |   |   length_packet < 192.5
|   |   |   |   |   |   |   |   |   length_packet < 190.5 : 0 (76/0)
|   |   |   |   |   |   |   |   |   length_packet >= 190.5 : 1 (1/0)
|   |   |   |   |   |   |   |   length_packet >= 192.5 : 0 (510/0)
|   |   |   |   |   |   port >= 80.5
|   |   |   |   |   |   |   length_packet < 282
|   |   |   |   |   |   |   |   length_packet < 279.5 : 0 (86/0)
|   |   |   |   |   |   |   |   length_packet >= 279.5 : 1 (1/0)
|   |   |   |   |   |   |   length_packet >= 282 : 0 (777/0)
|   |   |   |   |   length_packet >= 1199.5
|   |   |   |   |   |   port < 40 : 0 (1/0)
|   |   |   |   |   |   port >= 40
|   |   |   |   |   |   |   port < 261.5 : 1 (11/1)
|   |   |   |   |   |   |   port >= 261.5 : 0 (1/0)
|   |   |   |   length_packet >= 1200.5
|   |   |   |   |   length_packet < 1397.5
|   |   |   |   |   |   length_packet < 1396.5 : 0 (423/0)
|   |   |   |   |   |   length_packet >= 1396.5
|   |   |   |   |   |   |   port < 261.5 : 0 (1/0)
|   |   |   |   |   |   |   port >= 261.5 : 1 (1/0)
|   |   |   |   |   length_packet >= 1397.5 : 0 (1401/0)
|   |   |   length_packet >= 2095.5
|   |   |   |   port < 261.5
|   |   |   |   |   length_packet < 2136.5
|   |   |   |   |   |   length_packet < 2135.5
|   |   |   |   |   |   |   length_packet < 2096.5 : 1 (13/1)
|   |   |   |   |   |   |   length_packet >= 2096.5
|   |   |   |   |   |   |   |   length_packet < 2119.5 : 0 (25/0)
|   |   |   |   |   |   |   |   length_packet >= 2119.5
|   |   |   |   |   |   |   |   |   length_packet < 2120.5 : 1 (5/1)
|   |   |   |   |   |   |   |   |   length_packet >= 2120.5 : 0 (15/0)
|   |   |   |   |   |   length_packet >= 2135.5 : 1 (47/1)
|   |   |   |   |   length_packet >= 2136.5
|   |   |   |   |   |   length_packet < 2159.5 : 0 (24/0)
|   |   |   |   |   |   length_packet >= 2159.5 : 1 (4/1)
|   |   |   |   port >= 261.5 : 0 (55/0)
|   |   length_packet >= 2160.5 : 0 (10443/0)
port >= 447
|   port < 8718.5
|   |   length_packet < 483.5
|   |   |   length_packet < 482.5
|   |   |   |   prot < 11.5
|   |   |   |   |   port < 6666
|   |   |   |   |   |   length_packet < 143.5
|   |   |   |   |   |   |   port < 3317 : 0 (3869/0)
|   |   |   |   |   |   |   port >= 3317
|   |   |   |   |   |   |   |   port < 3325 : 1 (1/0)
|   |   |   |   |   |   |   |   port >= 3325
|   |   |   |   |   |   |   |   |   length_packet < 105 : 0 (1397/0)
```

```
|  |  |  |  |  |  |  |  |  length_packet >= 105
|  |  |  |  |  |  |  |  |  |  length_packet < 106.5
|  |  |  |  |  |  |  |  |  |  |  port < 4436
|  |  |  |  |  |  |  |  |  |  |  |  port < 4219 : 0 (2/0)
|  |  |  |  |  |  |  |  |  |  |  |  port >= 4219 : 1 (1/0)
|  |  |  |  |  |  |  |  |  |  |  port >= 4436 : 0 (7/0)
|  |  |  |  |  |  |  |  |  |  length_packet >= 106.5 : 0 (499/0)
|  |  |  |  |  |  length_packet >= 143.5
|  |  |  |  |  |  |  length_packet < 145
|  |  |  |  |  |  |  |  port < 5900.5
|  |  |  |  |  |  |  |  |  port < 1865
|  |  |  |  |  |  |  |  |  |  port < 1855.5
|  |  |  |  |  |  |  |  |  |  |  port < 1115.5
|  |  |  |  |  |  |  |  |  |  |  |  port < 1099.5 : 0 (4/0)
|  |  |  |  |  |  |  |  |  |  |  |  port >= 1099.5 : 1 (1/0)
|  |  |  |  |  |  |  |  |  |  |  port >= 1115.5 : 0 (49/0)
|  |  |  |  |  |  |  |  |  |  port >= 1855.5 : 1 (3/0)
|  |  |  |  |  |  |  |  |  port >= 1865 : 0 (66/0)
|  |  |  |  |  |  |  |  port >= 5900.5
|  |  |  |  |  |  |  |  |  port < 5967.5
|  |  |  |  |  |  |  |  |  |  port < 5918.5 : 1 (1/0)
|  |  |  |  |  |  |  |  |  |  port >= 5918.5
|  |  |  |  |  |  |  |  |  |  |  port < 5940.5 : 0 (2/0)
|  |  |  |  |  |  |  |  |  |  |  port >= 5940.5 : 1 (1/0)
|  |  |  |  |  |  |  |  |  port >= 5967.5 : 0 (7/0)
|  |  |  |  |  |  length_packet >= 145
|  |  |  |  |  |  |  |  port < 3317 : 0 (3267/0)
|  |  |  |  |  |  |  |  port >= 3317
|  |  |  |  |  |  |  |  |  port < 3323 : 1 (4/0)
|  |  |  |  |  |  |  |  |  port >= 3323 : 0 (1281/0)
|  |  |  |  |  port >= 6666
|  |  |  |  |  |  port < 6667.5
|  |  |  |  |  |  |  length_packet < 194.5 : 0 (1/0)
|  |  |  |  |  |  |  length_packet >= 194.5 : 1 (4/0)
|  |  |  |  |  |  port >= 6667.5
|  |  |  |  |  |  |  port < 8711
|  |  |  |  |  |  |  |  port < 7199
|  |  |  |  |  |  |  |  |  port < 7169
|  |  |  |  |  |  |  |  |  |  port < 6942 : 0 (134/0)
|  |  |  |  |  |  |  |  |  |  port >= 6942
|  |  |  |  |  |  |  |  |  |  |  port < 6950
|  |  |  |  |  |  |  |  |  |  |  |  length_packet < 97 : 0 (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  length_packet >= 97 : 1 (1/0)
|  |  |  |  |  |  |  |  |  |  |  port >= 6950 : 0 (21/0)
|  |  |  |  |  |  |  |  |  port >= 7169 : 1 (2/0)
|  |  |  |  |  |  |  |  port >= 7199 : 0 (232/0)
|  |  |  |  |  |  |  port >= 8711 : 1 (1/0)
|  |  |  |  prot >= 11.5
|  |  |  |  |  length_packet < 35.5
```

```
|   |   |   |   |   length_packet < 33 : 0 (306/0)
|   |   |   |   |   length_packet >= 33 : 1 (4/0)
|   |   |   |   length_packet >= 35.5 : 0 (16268/0)
|   |   |   length_packet >= 482.5
|   |   |   |   prot < 11.5 : 0 (6/0)
|   |   |   |   prot >= 11.5
|   |   |   |   |   port < 1736.5 : 0 (1/0)
|   |   |   |   |   port >= 1736.5
|   |   |   |   |   |   port < 4799.5 : 1 (5/0)
|   |   |   |   |   |   port >= 4799.5 : 0 (1/0)
|   |   length_packet >= 483.5 : 0 (15739/0)
|   port >= 8718.5 : 0 (240425/0)
```