UNIVERSIDADE CANDIDO MENDES - UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE PRODUÇÃO

André de Azevedo Cunha

APLICAÇÃO DA MINERAÇÃO DE DADOS PARA DESCOBERTA DE PADRÕES NO TRÁFEGO DE REDE E REFINAMENTO DE SUAS REGRAS DE SEGURANÇA NO INSTITUTO FEDERAL FLUMINENSE

CAMPOS DOS GOYTACAZES, RJ.
Junho de 2013.

UNIVERSIDADE CANDIDO MENDES - UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE PRODUÇÃO

André de Azevedo Cunha

APLICAÇÃO DA MINERAÇÃO DE DADOS PARA DESCOBERTA DE PADRÕES NO TRÁFEGO DE REDE E REFINAMENTO DE SUAS REGRAS DE SEGURANÇA NO INSTITUTO FEDERAL FLUMINENSE

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção, da Universidade Candido Mendes – Campos/RJ, para obtenção do grau de MESTRE EM ENGENHARIA DE PRODUÇÃO.

Orientador: Prof.ª Geórgia Regina Rodrigues Gomes, D.Sc.

CAMPOS DOS GOYTACAZES, RJ.
Junho de 2013.

ANDRÉ DE AZEVEDO CUNHA

APLICAÇÃO DA MINERAÇÃO DE DADOS PARA DESCOBERTA DE PADRÕES NO TRÁFEGO DE REDE E REFINAMENTO DE SUAS REGRAS DE SEGURANÇA NO INSTITUTO FEDERAL FLUMINENSE

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção, da Universidade Candido Mendes – Campos/RJ, para obtenção do grau de MESTRE EM ENGENHARIA DE PRODUÇÃO.

Aprovada em 14 de Junho de 2013

BANCA EXAMINADORA

Prof[a]. Geórgia Regina Rodrigues Gomes, D.Sc.

Universidade Candido Mendes

Prof. Dalessandro Soares Vianna, D.Sc.

Universidade Candido Mendes

Universidade Federal Fluminense

Prof. William da Silva Vianna, D.Sc.

Instituto Federal Fluminense

CAMPOS DOS GOYTACAZES, RJ.
2013

Dedico este trabalho primeiramente a minha Noiva Simara, por todo apoio e encorajamento dado durante todo o curso, nos momentos de dificuldade e dúvida.

Familiares pela compreensão em todos os momentos de ausência dedicados a esta dissertação.

AGRADECIMENTOS

Primeiramente a Deus, por me sustentar e conduzir até aqui, sempre dando força, coragem, disposição e sabedoria para prosseguir.

De maneira especial à minha Orientadora Geórgia, que sempre me incentivou e contribuiu para a conclusão deste trabalho com sua vasta experiência.

Ao Instituto Federal Fluminense, que viabilizou através de bolsa a realização do meu Mestrado, e pelo apoio no desenvolvimento do tema da minha dissertação, permitindo a coleta dos dados em sua rede de computadores.

À Universidade Candido Mendes – Campos, pela contribuição dada disponibilizando toda a infra-estrutura necessária para os estudos.

“É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se aos pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem a vitória, nem a derrota.”

(Theodore Roosevelt)

RESUMO

APLICAÇÃO DA MINERAÇÃO DE DADOS PARA DESCOBERTA DE PADRÕES NO TRÁFEGO DE REDE E REFINAMENTO DE SUAS REGRAS DE SEGURANÇA NO INSTITUTO FEDERAL FLUMINENSE

Com o avanço e disseminação dos serviços on-line, as redes de computadores ficaram vulneráveis. Nos dias de hoje, praticamente todos os computadores e dispositivos móveis estão conectados a grande rede mundial, a internet, deixando o maior patrimônio das corporações, a informação, sem a devida proteção. A falta de proteção, geralmente ocorre por descuido do próprio usuário, deixando sua máquina vulnerável, ou por falha de segurança na rede, onde serviços são executados com brechas de segurança já conhecidas. O objetivo deste trabalho é aplicar as técnicas de mineração de dados nos *logs* do tráfego de rede gravados no banco de dados de uma Instituição de Ensino Superior Pública, com o intuito de identificar tráfegos maliciosos que possam deixar a informação confidencial de pesquisas e do histórico dos discentes vulnerável. Com a extração do conhecimento do tráfego não desejado, é possível bloquear este tráfego para aumentar a segurança de toda a rede e otimizar a mesma para o seu devido fim: como plataforma de conhecimento para a comunidade, assim evitando que rede congestione, assim reduzindo o custo para o setor público com contratação de canais de dados, beneficiando os usuários, sejam eles alunos, docentes ou servidores.

O software utilizado para minerar os dados foi o WEKA, e através de algoritmos de classificação e associação foi possível extrair vários conhecimentos até então desconhecidos da equipe de Tecnologia da Informação do Instituto, como por exemplo, os protocolos e serviços mais atacados, a relação da hora do alerta e sua categoria, bem como os países e hosts que mais geraram alertas.

PALAVRAS-CHAVE: Redes de computadores; Segurança; Mineração de dados; descoberta de conhecimento em base de dados; Reconhecimento automático de padrões.

ABSTRACT

APPLICATION OF DATA MINING FOR DISCOVERY OF NETWORK TRAFFIC PATTERNS IN REFINING AND THEIR SAFETY RULES IN FEDERAL INSTITUTE FLUMINENSE.

With the advancement and dissemination of online services, computer networks were vulnerable. Nowadays, almost all computers and mobile devices are connected to world wide web, the internet, leaving the greatest asset of corporations, information, without proper protection. The lack of protection usually occurs by the user's own carelessness, leaving her vulnerable machine, or security hole in the network, where services are executed with security holes known. The objective of this work is to apply data mining techniques on the network traffic logs recorded in the database of a Public Institution of Higher Education, in order to identify malicious traffic that can leave confidential information from the research and history of vulnerable students. With the extraction of knowledge from unwanted traffic, you can block this traffic to enhance security across the network and optimize it to its proper end: as knowledge platform for the community, thus preventing network congest, thus reducing the cost to the public sector with hiring data channels, benefiting users, be they students, teachers or servers.

The software used to mine the data was WEKA, and by means of algorithms for classification and association was possible to extract several hitherto unknown knowledge of the Information Technology staff, for example, the protocols and attacked more services , the relationship of time alarm and category, as well as countries and hosts that generated more alerts.

KEY-WORDS: Computer Networking, Security, Data mining, knowledge discovery in databases, automatic pattern recognition.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

KDD	KNOWLEDGE DATABASE DISCOVERY
DM	DATA MINING
ARFF	ATTRIBUTE RELATION FILE FORMAT
GLP	GENERAL PUBLIC LICENCE
AG	ALGORITMOS GENÉTICOS
LAN	LOCAL AREA NETWORK
MAN	METROPOLITAN AREA NETWORK
CAN	CAMPUS AREA NETWORK
WAN	WIDE AREA NETWORK
IDS	INTRUSION DETECTION SYSTEM
IPS	INTRUSION PREVENTION SYSTEM
CERT.BR	CENTRO DE ESTUDOS, RESPOSTA E TRATAMENTO DE INCIDENTES NO BRASIL
TI	TECNOLOGIA DA INFORMAÇÃO
OSI	OPEN SYSTEMS INTERCONNECTION
SIGE	SISTEMAS INTEGRADOS DE GESTÃO EMPRESARIAL
SIG	SISTEMAS INTEGRADOS DE GESTÃO
ERP	ENTERPRISE RESOURCE PLANNING
SPSS	STATISTICAL PACKAGE FOR THE SOCIAL SCIENCES
CRISP-DM	CROSS INDUSTRY STANDARD PROCESS FOR DATA MINING
SAS	STATISTICAL ANALYSIS SYSTEM
ODM	ORACLE DATA MINING
IBM	INTERNATIONAL BUSINESS MACHINES
KXEN	KNOWLEDGE EXTRACTION ENGINES
DB2	DATABASE 2
HTTP	HYPERTEXT TRANSFER PROTOCOL
HTTPS	HYPERTEXT TRANSFER PROTOCOL SECURE
MAC	MEDIA ACCESS CONTROL
TCP	TRANSMISSION CONTROL PROTOCOL
IP	INTERNET PROTOCOL
IPS	INTERNET PROTOCOL SUITE
FTP	FILE TRANSFER PROTOCOL

UDP	USER DATAGRAM PROTOCOL
IANA	INTERNET ASSIGNED NUMBERS AUTHORITY
VOIP	VOICE OVER INTERNET PROTOCOL
MTU	MAXIMUM TRANSMIT UNIT
TOS	TYPE OF SERVICE
TTL	TIME TO LIVE
ARP	ADDRESS RESOLUTION PROTOCOL
HLEN	HARDWARE LENGTH
PLEN	PROTOCOL LENGTH
ICMP	INTERNET CONTROL MESSAGE PROTOCOL
IGMP	INTERNET GROUP MANAGEMENT PROTOCOL
DNS	DOMAIN NAME SERVER
DOS	DENIAL OF SERVICE

SUMÁRIO

1 INTRODUÇÃO

Atualmente, quase a totalidade das atividades, procedimentos e operações de empresas e instituições, seja ela publica ou privada são feitas eletronicamente, ou ao menos o históricos dessas atividades são posteriormente digitalizados. Segundo a ABNT (2005), como esse volume de informação é cada vez maior, ela se tornou um ativo muito mais valioso que qualquer bem material, imprescindível para o sucesso do negócio em questão.

Segundo Oliveira (2011), a informação é essencial para a sobrevivência das organizações, devendo ser preservada com o intuito de não ser alterada, evitando que pessoas mal intencionadas obtenham acesso a ela. Já Lopes (2012) afirma que a informação assumiu um valor vital para as organizações, que até a pouco tempo atrás tinham o foco basicamente para os bens tangíveis e hoje em dia enxergam a informação como principal ativo.

Além das instituições, devido à convergência de computação e da comunicação, produziu-se uma sociedade que consome mais informação. Pouco tempo atrás, usuários residenciais utilizavam links na casa de poucos Kbps (Kbits por segundo), já atualmente, um usuário residencial pode ter links que passam dos 50Mbps (Megabits por segundo). Assim, o volume de informação mundial hoje é gigantesco, sendo a sua maioria armazenada em bancos de dados. Segundo Frank (2005), nos bancos de dados espalhados ao redor do mundo, existem muitos dados potencialmente importantes, porém ainda desconhecidos ou não relacionados, no entanto, muitos outros padrões presentes nessas bases são banais e irrelevantes.

Além do mais, há uma grande apreensão por parte de vários profissionais e gestores em compreender os dados e em utilizar a informação e conhecimento das bases de dados (COSTA, 2004). Isso provavelmente acontece em decorrência do forte ritmo de geração de dados, o que devido à incapacidade natural do ser humano de explorar, extrair e interpretar estes dados neste ritmo para obter conhecimento dessas bases.

Assim a informática e as tecnologias voltadas para coleta, armazenamento e disponibilização de dados vêm evoluindo muito e disponibilizando técnicas, métodos e ferramentas computacionais automáticas capazes de auxiliar na extração e interpretação de informações úteis contidas em grandes volumes de dados complexos (CARDOSO, 2006), (QUONIAM, 2001).

Uma dessas tecnologias é a aprendizagem de máquina, que provê a base técnica da mineração de dados, sendo usada para extrair conhecimento dos dados armazenados em bancos de dados. Conhecimento este expresso de uma forma compreensível e pode ser utilizado para vários propósitos. (FRANK, 2005).

1.1 REDES E SEGURANÇA

Especificamente na área de segurança de redes de computadores, a análise dos *logs* de acesso e a interpretação dessa informação para que seja possível bloquear qualquer tentativa de acesso não autorizado, ou de tráfego malicioso nesse grande volume de dados é um processo praticamente inviável de ser efetuado sem o auxilio de uma ferramenta de apoio. Ao mesmo passo, diariamente são descobertas novas de brechas de segurança e são criadas ferramentas que possam explora-lás (*malwares*).

Segundo Ulbrich (2003), muitos *bugs* (brechas de segurança) que permitem a ação de criminosos poderiam ser facilmente corrigidos, mas as companhias preferem simplesmente ignorar esses problemas. É o que também mostra uma pesquisa realizada pela *Módulo Security Solutions* no fim de 2002, empresa especialista em segurança. Segundo os dados coletados, a segurança da informação é fator importante para 45% dos executivos, sendo que 16% a consideram crítica e 32% a classificam como vital. Mesmo assim, a falta de conscientização dos executivos (45%) e dos usuários (38%) foram apontadas como os principais obstáculos para a implementação da segurança nas instituições (ULBRICH, 2003).

Outro dado revelado pela pesquisa é extremamente preocupante: 43% das empresas reconheceram ter sofrido ataques nos últimos 12 meses, sendo que 24% dessas ocorrências foram registradas nos últimos seis meses. O mais crítico é que 32% não souberam informar se foram atacadas ou não e, apesar da expectativa de aumento nos problemas de segurança e nos índices de registros de ataques e invasões, a pesquisa mostra que apenas metade das empresas brasileiras (49%) possuem planos de ação formalizados em caso de ataques.

Ainda a percepção de falta de segurança continua sendo o maior obstáculo para o desenvolvimento de negócios digitais em escala global. Pelo menos 66% dos usuários deixam de realizar transações online por este fator (ULBRICH, 2003).

Existem muitos fatores que contribuem para a grande quantidade de ataques virtuais. Por exemplo, existem muitos sites inseguros. Um estudo da Gartner Group estima que 2/3 dos servidores Web no mundo podem ser invadidos de alguma forma. Outro fator que estimula esses ataques é a ampla disponibilidade de ferramentas de ataque na internet. Qualquer pessoa com tempo livre e mesmo com conhecimento técnico médio consegue encontrar informações e os softwares necessários para uma invasão. Mas o principal motivo ainda é a impunidade. Falta uma legislação específica e existem poucos policiais peritos que investigam crimes digitais no Brasil. (ULBRICH, 2003)

Baseado nisso, o número de ataques vem crescendo nos últimos anos. De acordo o relatório de Incidentes reportados em 2012 ao CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes no Brasil), em 2011 foram reportados 399.515 incidentes a esse órgão que trata os incidentes na Internet no Brasil contra apenas 3107 incidentes reportados em 1999, conforme Figura 1 (CERT.BR, 2012).

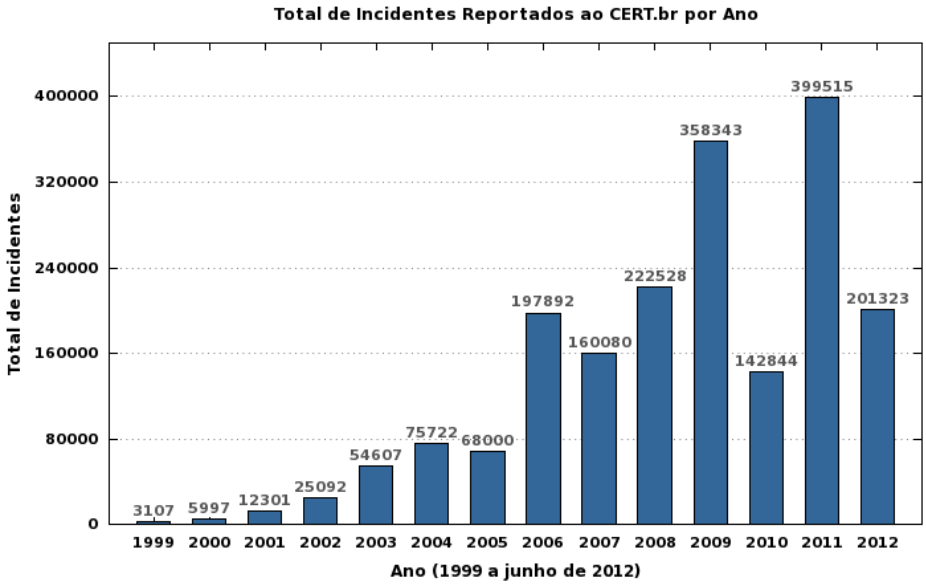

Figura 1: Total de incidentes reportados ao CERT.br por ano.
Fonte: CERT.br, 2012

Para se ter uma idéia da quantidade de ações hackers, um estudo da Universidade da Califórnia mostrou que os hackers tentam realizar mais de 4 mil ataques do tipo *DoS (Denial of Service*) todas as semanas, um número bastante elevado e que mostra que a muito do que se proteger quando se está on-line (Ulbrich, 2003). Já o mesmo relatório da CERT.br também reporta um grande número de incidentes. No período de abril a junho de 2012, foram reportados ao órgão mais de 10.000 incidentes conforme a Figura 2.

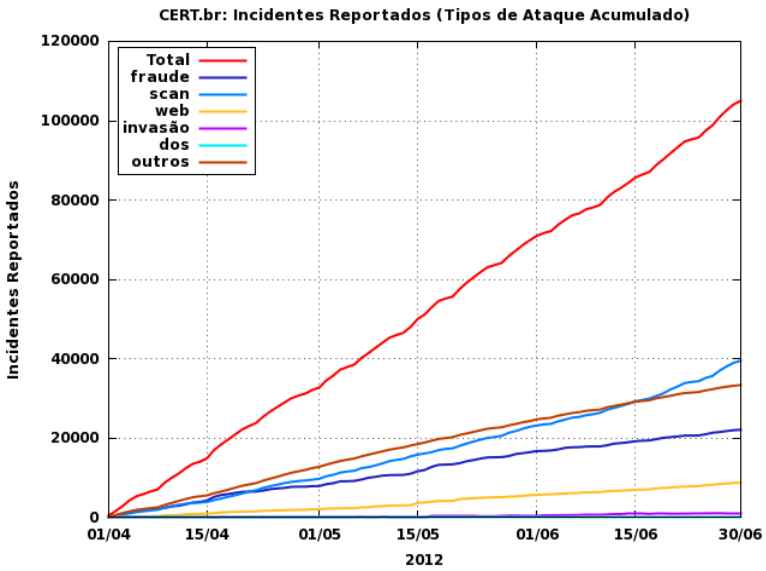

Figura 2: Incidentes reportados ao CERT.br.
Fonte: CERT.br, 2012

1.2 JUSTIFICATIVA

Diante de tantas ameaças, é dever da equipe de tecnologia da informação (T.I.) e dos administradores de rede identificar e proteger a rede e seus sistemas, mesmo com o crescente volume de dados. A Figura 3 ilustra a média de tráfego de rede do Instituto Federal Fluminense (IFF) durante uma semana (*Week* 16), no qual o volume de informações trafegadas ficaram em torno de 20MBit/s.

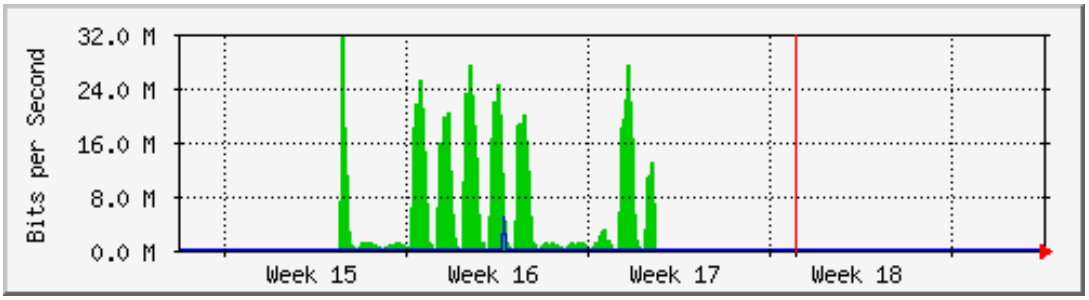

Figura 3: Média de tráfego diário.

Extrair conhecimento útil dessa grande massa de dados, revelando relações ocultas, padrões e gerando regras para correlacionar dados, é uma alternativa estratégica para as instituições. Esta técnica pode ajudar na tomada rápida de decisão ou modificar o planejamento da instituição em longo prazo, com maior confiança do que uma decisão tomada apenas pelo sentimento do gestor, o que é possível através das técnicas de mineração de dados (CARDOSO, 2008).

Informação e conhecimento são primordiais, estratégicas e imprescindíveis na vida de qualquer instituição, seja qual for sua área de atuação, pois o mercado exige cada vez mais decisões rápidas e bem embasadas, sendo vital para o sucesso das organizações. Por isso, diversas instituições nacionais e internacionais das mais diversas áreas de atuação já adotaram nas suas rotinas a mineração de dados para monitorar suas atividades como: proteção de redes de computadores (MORAIS, 2010), acessos à servidores WEB (CORREIA, 2004), arrecadações, consumo e perfil dos clientes, prevenir fraudes e riscos do mercado (ARAUJO, 2006), evitar evasão escolar (MARTINS, 2011), dentre outras.

Para atender este novo cenário, as organizações vêm utilizando as metodologias da ciência da computação para realizar seus estudos, principalmente a técnica

Knowledge Discovery in Databases (KDD), ou seja, o processo de descoberta de conhecimento das bases de dados, que engloba a mineração de dados, que é a etapa mais importante do KDD (QUONIAM, 2001).

1.3 OBJETIVOS DESTE TRABALHO

O trabalho desenvolvido teve como objetivo utilizar as técnicas de mineração de dados nas informações do tráfego de redes do Instituto Federal Fluminense, tanto o tráfego interno entre seus *campus* quanto o tráfego envolvendo a rede mundial de computadores, a Internet.

Após a extração de conhecimento obtida pelo processo de mineração, foram determinadas as principais vulnerabilidades e tráfegos considerados maliciosos ainda não identificados pela equipe de Tecnologia da Informação do Instituto e foi proposta a devida proteção.

1.3.1 Estrutura do trabalho

Este trabalho foi estruturado em cinco capítulos, sendo o primeiro desta a introdução abordando assuntos gerais que serão mais detalhados nos capítulos seguintes, bem como os objetivos a serem alcançados.

No segundo capítulo faz-se uma fundamentação sobre as redes de computadores, destacando a sua importância nos dias de hoje e descrevendo o seu funcionamento e como agem as ameaças que atacam essas redes, principalmente na rede mundial de computadores, a Internet. Em seguida é detalhado o KDD e a mineração de dados, alguns de seus principais conceitos e técnicas relevantes a este trabalho, bem como suas etapas e algumas aplicações.

O terceiro capítulo aborda toda a metodologia utilizada neste trabalho. Desde a configuração do servidor de coleta dos dados, a efetiva aquisição dos dados através do redirecionamento dos dados, bem como é apresentado o software desenvolvido para enriquecimento dos dados. Por fim são descritos os passos do KDD para preparação,

mineração e analise dos mesmos. Também são apresentados os equipamentos, tecnologias e ferramentas utilizadas para o desenvolvimento do trabalho,

No quarto capítulo relata os resultados alcançados. Onde é de fato aplicado o processo KDD para a extração do conhecimento por meio da mineração de dados. Ele foi estruturado detalhando cada etapa do KDD, desde a preparação dos dados até a efetiva aplicação das técnicas de mineração de dados e posterior extração de conhecimento dos dados do trabalho em questão, sendo aplicada a teoria ilustrada nos capítulos anteriores ao presente trabalho, com o intuito de resolver os problemas propostos nesta dissertação.

O quinto capítulo apresenta as conclusões deste presente trabalho, sendo descritos os conhecimentos extraídos e as melhorias efetuadas na rede do Instituto, bem como as sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO

Neste capitulo, são apresentados tópicos sobre conceitos, padrões e métodos utilizados nesta dissertação. Primeiramente são descritos os conceitos de redes de computadores necessários para um melhor entendimento dos resultados deste trabalho. Estes conceitos abordados serão descritos sucintamente, pois se entende que estes são pré-requisitos para o entendimento do tema apresentado. Em seguida é apresentada a técnica de descoberta de conhecimento em base de dados, descrevendo suas etapas e softwares que aplicam tal técnica.

2.1 REDES DE COMPUTADORES

Rede de computadores pode ser definida como dois ou mais computadores conectados por algum meio através do qual são capazes de compartilhar informações. Existem muitos tipos de redes: LANs (*Local Area Networks*), WANs (*Wide Area Networks*), MAN (*Metropolitan Area Networks*), CAN (*Campus Area Network*), que diferem entre si de acordo com seu alcance (DONAUHE, 2008):

LAN: Uma LAN é uma rede confinada em um espaço limitado, como um prédio ou andar. Ela utiliza tecnologias de pouco alcance, como ethernet, *token ring* e similares. Geralmente uma LAN está sob controle de uma empresa ou entidade que a utiliza. Na Figura 4 é representada uma rede LAN.

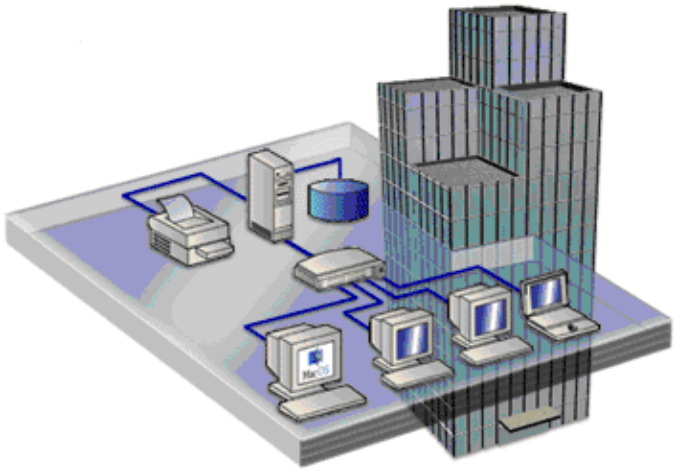

Figura 4: LAN – Local Area Network.
Fonte: Online Help

WAN: Uma WAN, é uma rede utilizada para conectar LANs através de um provedor terceirizado. Um exemplo é a própria Internet ou a interligação de escritórios de uma instituição em diversas cidades através de uma empresa de telecomunicações. Na Figura 5 é apresentada uma rede WAN.

Figura 5: WAN – Wide Area Network.
Fonte: Online Help

CAN: Uma CAN interliga LANs e/ou edifícios de uma instituição em uma área possuída ou controlada por uma entidade. Por exemplo, pode-se citar o campus de uma universidade ou um parque industrial.

MAN: É uma rede que conecta LANs e/ou prédios em uma área geralmente maior que um campus, conforme ilustra a Figura 6. Por exemplo, uma MAN pode ser utilizada para conectar diversos escritórios de uma empresa dentro de uma área metropolitana através de uma empresa de telecomunicações.

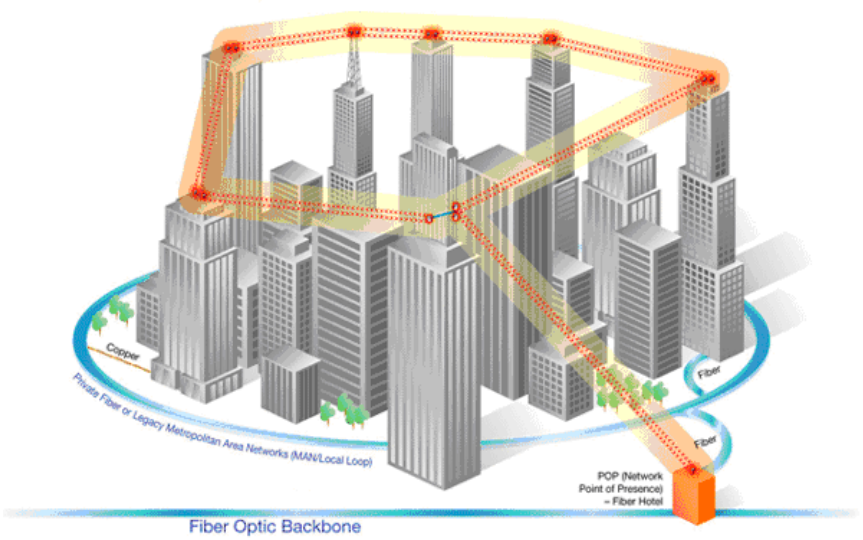

Figura 6: MAN – Metropolitan Area Network.
Fonte: Wimax-industry

Já para Soares, Lemos e Colcher (1995), as redes de computadores surgiram para viabilizar a troca de informações e o compartilhamento de dispositivos, o que permite a integração dos ambientes de trabalho e preserva a independência das várias estações de processamento.

Segundo McQuerry (2008) e Sportack (2004), uma rede consiste em vários dispositivos conectados em um sistema comunicando-se entre si, dispositivos estes como servidores e computadores, compartilhando recursos, tanto físicos quanto lógicos. Mais especificamente, uma rede de computadores que permite a conectividade entre vários sistemas com vários intuitos, como recursos de hardware como impressoras, scanners, sistemas de armazenamento, recursos de software, como por exemplo o Sistemas Integrados de Gestão Empresarial (SIGE ou SIG), em *inglês Enterprise Resource Planning (ERP)* de uma empresa ou recursos de dados, como por exemplo um banco de dados.

Ainda segundo Donauhe (2008), o conceito de conexão é muito importante. É o que distingue uma rede de computadores, com endereçamento, por exemplo, com endereços de IP, de uma “rede peão” ou *sneaker-net*, onde os dados são transferidos fisicamente de um computador para o outro através de uma mídia removível, como um *pen-drive*. Ao inserir a mídia removível do computador destino, não indicação de que os arquivos são provenientes de outro computador, então não existe conexão. Uma conexão então envolve algum tipo de endereçamento ou identificação dos pontos da rede.

Segundo Castelli (2004), existem três características que diferem as redes de computadores:

Topologia: as redes podem assumir diferentes topologias, ou seja, a forma como é feita a interligação dos dispositivos. As mais utilizadas são a topologia estrela, na qual os dispositivos são conectados a um concentrador, e a topologia anel, em que um dispositivo é conectado ao próximo dispositivo, até que essa interligação retorne ao primeiro dispositivo fechando uma espécie de anel.

Protocolo: redes podem seguir diferentes protocolos, o que específica as normas a serem seguidas no envio e recebimento.

Meio físico: existem vários meios na qual uma rede pode ser formada, como via cabos de par-trançado (cabo de rede), fibra óptica, wireless, entre outras menos usuais.

Para que exista uma rede "real", é necessário equipamentos que interliguem os computadores e dispositivos. Comumente em uma rede de computadores são encontrados *Hubs* e/ou *Switchs*.

Segundo Bezerra (2009), um *Hub* é um tipo de concentrador ou um repetidor multiportas, ou seja: recebe a informação em uma porta e distribui para todas as outras, sendo utilizado para realizar a conexão física entre os computadores da rede.

Para Donauhe (2008) um *Hub*, representado na Figura 7 é simplesmente um meio de conectar cabos de Ethernet para que seus sinais possam ser repetidos a todos os cabos conectados ao *Hub.* Por esse motivo, os *hubs* são comumente chamados de repetidores multi portas. É ainda importante destacar que apesar de todo *Hub* ser um tipo de repetidor, nem todo repetidor é um *Hub*. Apesar dos *hubs* serem utilizados há vários anos, eles ainda são utilizados em pequenas redes, e pelo fato dele replicar a informação com destino a uma porta para todas as outras, ele representa um risco a segurança da rede.

Figura 7: HUB.
Fonte Campograndern.com.br, 2012

Já o funcionamento de um *switch* consiste em receber o sinal em um porta, filtrar os campos da mensagem, armazená-los localmente, escolher a porta de destino de acordo com a sua tabela de roteamento interna, encaminhando a transmissão para a porta correspondente (PASCOAL, 2008).

Switch é o termo utilizado para definir qualquer coisa que possa modificar uma seqüência de execução, não importa disciplina ou o que está sendo modificado. Na área de redes, switch normalmente é um switch ethernet. Já para a área de telecomunicações, switch pode ser várias coisas.

Segundo Donauhe (2008), *switch* ethernet, ilustrado na Figura 8 é qualquer dispositivo que transmita quadros baseados nos endereços MAC da camada 2 do modelo OSI (Open Systems Interconnection) utilizando a Ethernet.

Figura 8: Switch Ethernet.
Fonte Campograndern.com.br, 2012

Enquanto os *hubs* repetem todos os quadros em todas as portas, um switch ethernet transmite quadros apenas para as portas para quais elas são destinados. Switchs podem operar tanto na camada 2 (Enlace do modelo OSI) quanto na camada 3 (rede).

Os switches que operam na camada 2 do modelo OSI (*Open Systems Interconnection*) utilizam a tabela ARP (Figura 9) para armazenar os endereços físicos (MAC) dos dispositivos a ele conectados.

ARP Table Entries:

Address	HWtype	HWaddress	Flags Mask	Iface
193.2.1.92	ether	00:11:95:CA:1A:1B	C	eth3
10.1.2.66	ether	00:11:95:CA:1A:1B	C	eth3
10.139.200.3	ether	00:12:17:7D:BE:13	C	br0
129.240.64.3	ether	00:11:95:CA:1A:1B	C	eth3
10.139.200.44	ether	00:12:17:7D:40:F7	C	br0
194.137.39.67	ether	00:11:95:CA:1A:1B	C	eth3
80.190.199.145	ether	00:11:95:CA:1A:1B	C	eth3
129.132.73.145	ether	00:11:95:CA:1A:1B	C	eth3
64.12.162.71	ether	00:11:95:CA:1A:1B	C	eth3
192.168.1.1	ether	00:11:95:CA:1A:1B	C	eth3
134.214.100.6	ether	00:11:95:CA:1A:1B	C	eth3
192.168.222.1	ether	00:FF:BA:B9:D9:A4	C	tap2

Figura 9: Tabela ARP

2.1.1 Modelo OSI

Segundo Tanenbaum (1997), o modelo OSI trata a interconexão de sistemas que podem se comunicar com outros sistemas, definido em um modelo de 7 camadas, conforme ilustra a Figura 10.

Figura 10: Modelo OSI.
Fonte How Stuff Works, 2000

É importante destacar as camadas que os equipamentos de rede atuam no modelo OSI. Hub, por apenas replicar as informações recebidas, apenas entende sinais elétricos, portanto opera na primeira camada do modelo, a camada física.

Já Switches e Roteadores possuem “inteligência” para fazer o encaminhamento dos pacotes recebidos. Switches trabalham na camada de enlace de dados ou camada 2, onde o mesmo de posse dos endereços físicos dos dispositivos conectados a ele, pode encaminhar o tráfego apenas para o destinatário.

E os roteadores trabalham com os endereços IP, sendo assim tem a habilidade de rotear os pacotes entre diferentes redes, e de criar listas de acesso, chamadas de ACL (*Access control lists*).

2.1.2 Modelo TCP

TCP/IP é um modelo (padrão) aberto de protocolos de comunicação, sendo o padrão utilizado nas redes de computadores atuais, especialmente na Internet. Este modelo é amplamente utilizado pois contém todos os mecanismos necessários para suportar todo e qualquer tipo de comunicação de rede (CASTELLI, 2004).

Segundo Sportack, (2004), foi o protocolo de rede que mudou o mundo, revolucionando a forma das pessoas se comunicarem e trocarem informações entre si, seja qual for a finalidade dessa comunicação. A sigla TCP/IP se refere a um conjunto de dois protocolos de comunicação, sendo eles o protocolo de controle de transmissão, o *Transmission Control Protocol* (TCP) e o protocolo de Internet, o *Internet Protocol* (IP). Também é possível encontrar na literatura outras nomenclaturas menos usuais como o *Internet Protocol Suite* (IPS) (HUNT, 2002). A grande popularidade do protocolo TCP/IP, conforme Hunt (2002), pode ser entendida por quatro das suas principais características:

Toda a implementação e todas as normas são abertas, ou seja, independente da arquitetura de hardware ou software do dispositivo em questão, é possível implementar o protocolo TCP/IP. Assim quaisquer destes dispositivos que implementem o protocolo podem estabelecer comunicação entre eles.

Independente de hardware de rede, ou seja, o TCP/IP pode integrar diferentes tipos de rede, sejam sem fio, redes de par metálico, de fibras ópticas, linhas dial-up, redes ethernet, ou seja, qualquer meio físico ou tipo de transmissão;

O modelo TCP/IP implementa um sistema de endereçamento comum, que permite a comunicação entre dispositivos independente do tamanho da rede, seja uma rede com apenas dois computadores (ponto-a-ponto) ou uma rede como a Internet, na qual existem inúmeros dispositivos conectados de diferentes modelos e arquiteturas trocando informações ao mesmo tempo;

Protocolos padronizados, consistentes e amplamente disponíveis para serviços aos usuários. Atualmente, implementações do TCP/IP estão presentes em computadores de todos os portes, aparelhos celulares, *tablets*, *storages*, impressoras, televisores, eletrodomésticos, ou seja o TCP/IP é cada vez mais encontrado em dispositivos que passam a comunicar-se, o que permite novos benefícios à população em geral.

Segundo Hunt (2002), não existe consenso em descrever o TCP/IP em camadas, porém, usualmente na literatura o modelo TCP/IP é assim apresentado, bem como o modelo OSI. Enquanto o modelo OSI se subdivide em sete camadas, o TCP/IP é dividido em apenas quatro.

2.1.2.1 Aplicação – Camada 4

É a camada de interface como usuário, ou seja, nela encontram-se os aplicativos como *browsers* (navegadores de Internet), softwares de bate-papo, clientes de e-mail, clientes ftp (*file transfer protocol*), enfim, qualquer software que utilize comunicação via rede e interaja com o usuário final (HUNT, 2002). Para melhor entendimento de seu funcionamento, é importante ressaltar a existência de duas entidades no processo de comunicação: o emissor e o receptor. Para o envio, os dados do usuário emissor são combinados com os dados do aplicativo, sendo encapsulados com a adição de algumas informações, como a porta de origem. Os dados são passados para a próxima camada do modelo, a camada de transporte. Quando do recebimento, o receptor, na camada de aplicação, remove o cabeçalho da aplicação, efetua o tratamento necessário para finalizar a transação e confirma que o processo foi concluído (TORRES, 2009).

2.1.2.2 Transporte – Camada 3

A camada conhecida como de transporte é uma forma abreviada de Host-to-Host Transport Layer. Os dois protocolos mais importantes e utilizados nas comunicações desta camada são o Transmission Control Protocol (TCP) e o User Datagram Protocol (UDP). A diferença básica entre o TCP e o UDP é que o primeiro tem conexão orientada a serviço, ou seja, o destinatário ao receber os dados da comunicação confirma o recebimento, diferentemente do UDP, no qual não existe nenhuma confirmação durante toda a comunicação. Assim cada protocolo é mais adequado para um tipo de serviço, dependendo da sua criticidade e necessidade de desempenho, sendo o TCP confiável e possui mecanismos de detecção e correção de erros, enquanto o UDP tem menor overhead, ou seja, tem melhor desempenho por não fazer verificação de erros de transmissão, dando acesso direto a um serviço de entrega de datagramas (HUNT, 2002).

2.1.2.2.1 Transmission Control Protocol (TCP)

Segundo Tanenbaum (2003), o protocolo TCP foi concebido para ser um protocolo de transporte confiável, fim a fim, em redes interconectadas, que apresentam diferentes topologias, banda, tipos e tamanho de pacotes, entre outros. O TCP é orientado à conexão, ou seja, cria um circuito virtual, full duplex (capacidade dos envolvidos na transmissão de enviar e receber dados simultaneamente) entre duas aplicações, sendo todos os bytes identificados por uma numeração para que seja possível a retransmissão em caso de falhas, como ilustra a Figura 11.

Figura 11: Protocolo TCP

Geralmente ocorre uma dificuldade em diferenciar as funções dos dois protocolos que formam o modelo TCP/IP. Segundo Dostálek e Kabelová (2006), o IP transmite dados entre dispositivos, enquanto o TCP transfere dados entre as aplicações nestes dispositivos, utilizando para tal a porta na qual o serviço está sendo executado.

Todas as aplicações que utilizam o protocolo TCP têm a garantia de que os dados serão entregues. Porém, destacam Dostálek e Kabelová (2006), que essa proteção garantida pelo TCP não oferece proteção contra ataques a rede de dados. A garantia do protocolo é de comunicação fim-a-fim, ou seja, se limita à entrega dos dados a outra ponta. Os pontos de origem e destino na conexão são identificados por um número de porta que, no caso do protocolo TCP, pode variar de 0 a 65535. No momento das transmissões, a aplicação de destino é endereçada por meio do conjunto de dados formado por um endereço IP, um número de porta e o protocolo, sendo assim possível encaminhar os dados (pacotes) para o dispositivo (*host)* em questão. Esse dispositivo, que no momento da recepção dos dados pode estar executar vários aplicativos simultaneamente, utilizará a porta de destino para distinguir para qual aplicação deve ser entregue o pacote TCP (Dostálek; Kabelová, 2006). Conforme Tanenbaum (2003), as portas de 0 até 1.024 são conhecidas e reservadas, com regulação feita pela *Internet Assigned Numbers Authority* (IANA).

2.1.2.2.2 User Datagram Protocol (UDP)

O UDP é um protocolo simples, destinado a aplicações que não exigem confirmação de entrega dos pacotes, ou seja, serviços não confiáveis como, por exemplo, voz sobre IP (VoIP) e para transmissão de vídeos. Ao enviar, o protocolo recebe os dados da camada superior, adiciona o número da porta de destino e calcula o *checksum* (opcional), para que o receptor valide os dados e faça o processo contrário no recebimento. O cabeçalho é bastante pequeno (8 bytes) e o restante são dados. Sem as informações de porta, seria impossível o tráfego de pacotes, pois a camada de transporte, assim como no caso do TCP não saberia distinguir a aplicação a qual pertence tal pacote Carne (2004), Tanenbaum (2003). Segundo Tanenbaum (2003) é importante frisar algumas funcionalidades que o UDP não implementa, como controle de fluxo, controle de erros ou retransmissão de pacotes. A Figura 12 descreve o funcionamento de uma transmissão sob o protocolo UDP.

Figura 12: Protocolo UDP

2.1.2.3 Internet – Camada 2

Segundo Tanenbaum (2003), a camada de Internet, também conhecida como camada de rede, consiste em todos os protocolos utilizados para que os dispositivos de origem e destino se encontrem, independente de sua localização física, já que os datagramas da camada de transporte são analisados para definir a rota que será utilizada. A seguir são apresentados sucintamente quatro protocolos pertencentes a esta camada.

2.1.2.3.1 Internet Protocol (IP)

O protocolo de internet ou *internet protocol* (IP) é o corresponde à camada de rede do modelo OSI, sendo responsável pelo envio de datagramas entre dispositivos, contendo em cada datagrama as informações necessárias (endereços de origem e destino) para que o mesmo seja entregue (DOSTÁLEK; KABELOVÁ, 2006; CARNE, 2004).

Segundo Carne (2004), um datagrama IP é formado pela junção da camada de transporte e o cabeçalho adicionado pelo protocolo IP, que contém os endereços de origem e de destino. Cada tipo de rede define sua unidade máxima de transmissão, o chamado *Maximum Transmit Unit* (MTU), que nas redes de padrão *ethernet*, tem o valor de 1.500 bytes como tamanho padrão ou *default*, sendo que o protocolo IP que fragmenta os pacotes no emissor para adequar-se ao ao MTU da rede e monta os datagramas quando estes são recebidos pelo destinatário. Esse endereço (IP) é único na rede, seja ela de qual tamanho for, desde uma pequena rede local (LAN) até uma grande WAN como a Internet, o que permite aos dispositivos conseguir estabelecer a comunicação, ou seja, que os *hosts* (dispositivos) comuniquem-se entre si.

Na Figura 13 é possível visualizar os campos do cabeçalho IP, que são: Versão, na qual é identificado o uso da versão 4 ou 6 do IP; tipo de serviço ou *type of service* (TOS), campo no qual pode ser utilizado para priorizar o tráfego conforme o tipo de serviço; time to live (TTL), que define o tempo de vida de um pacote, evitando que o mesmo trafegue eternamente na rede; *checksum*, utilizado para a verificação apenas do cabeçalho IP; e finalmente os endereço IP de origem e destino, necessários para a identificação dos hosts envolvidos na troca de informações, entre outros. No total o cabeçalho IP tem 32bits.

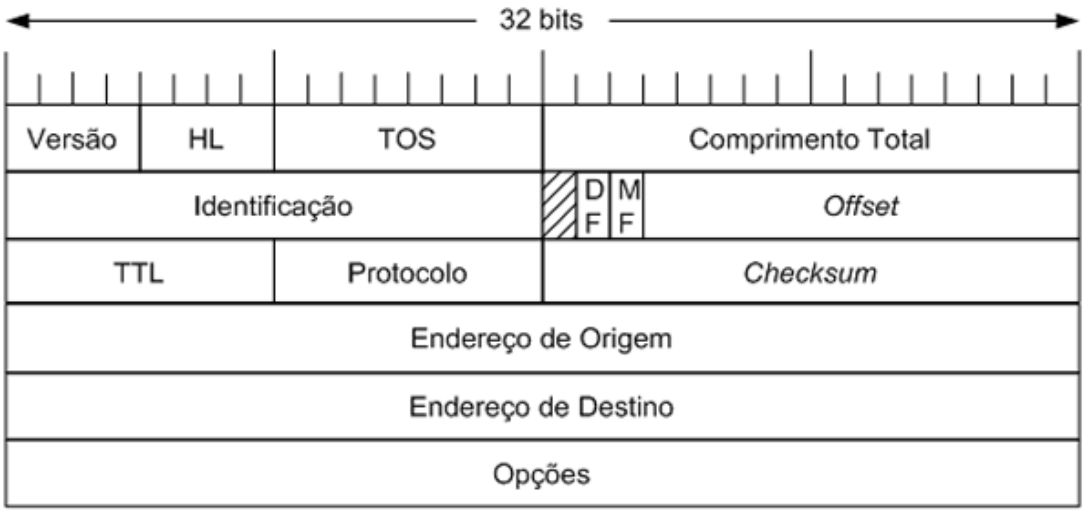

Figura 13: Cabeçalho IP

2.1.2.3.2 Address Resolution Protocol (ARP)

Segundo Held (2003) e Carne (2004), o protocolo de resolução de endereços ou simplesmente ARP, resolve o endereço IP de um nó em seu endereço MAC (*Media Access Control*), que fica gravado fisicamente em um chip na placa de rede. Os dispositivos conectados em uma rede local, para que consigam estabelecer uma comunicação entre si, utilizam do seguinte processo: é criada uma tabela que é gerada a partir de duas mensagens: ARP request, que envia frames via broadcast de nível MAC, e ARP reply, em que o endereço que responde ao IP solicitado responde com seu endereço MAC. De posso dessa tabela que é constituída dos endereços IP e MAC, os dispositivos "conhecem" os *hosts* e os pacotes podem ser enviados na LAN.

A Figura 14 ilustra um pacote ARP. O valor do campo hardware, definido para 1, significa Ethernet. O campo Protocol identifica o endereço do protocolo, em hexadecimal, identificando o valor 0800 o uso de endereços IP. Os campos *hardware length* (HLEN) e *protocol length* (PLEN) definem o tamanho, em bytes, dos endereços a serem utilizados (hardware e protocolo). Operation indica ARP request (1) ou ARP reply (2). Os demais campos referem-se ao protocolo e ao hardware do emissor e ao hardware do destinatário.

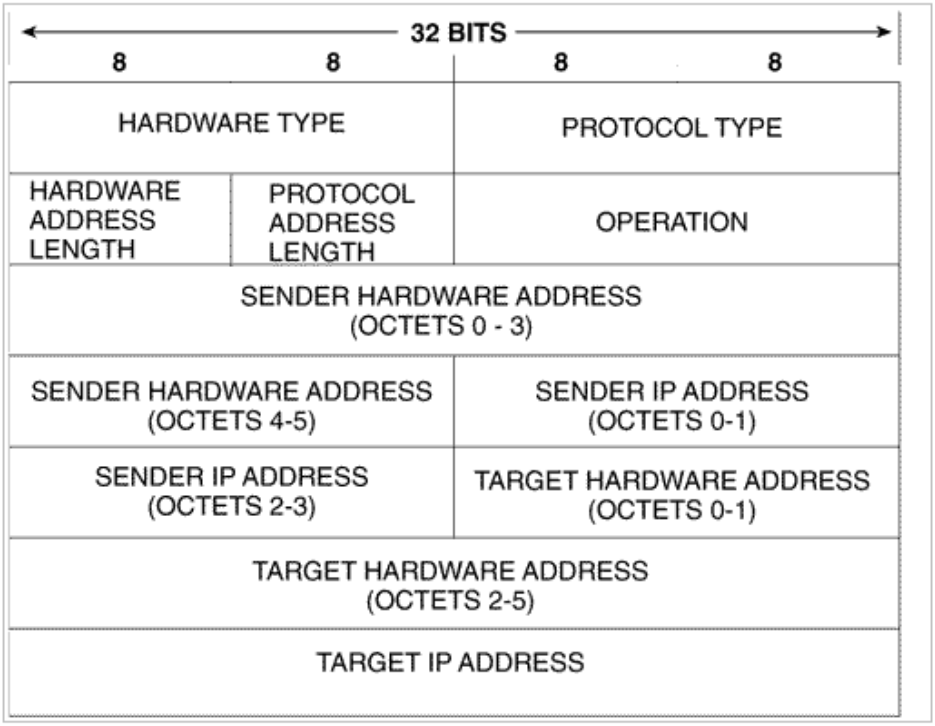

Figura 14: Pacote ARP

2.1.2.3.3 Internet Control Message Protocol (ICMP)

O protocolo ICMP fornece um método para dispositivos em uma rede IP possam trocar informações sobre problemas com a rede que estejam impedindo a troca de pacotes. Partindo da definição que o protocolo IP não é confiável e não garanta a entrega dos pacotes, é necessário que os dispositivos de uma rede tenha uma maneira de informar ao remetente quando a entrega não é possível (HALL, 2000). De acordo com Held (2003), uma mensagem ICMP é formada anexando um cabeçalho IP à mensagem ICMP, sendo cada mensagem formada por quatro campos.

Segundo Tanenbaum (2003), o ICMP também é utilizado por roteadores e quando ocorre algo inesperado ou algum problema de roteamento, o protocolo ICMP é utilizado para reportar os problemas. Este protocolo define 40 mensagens, como “destino não encontrado”, “tempo excedido”, dentre outras. Hall (2000) divide essas mensagens em três famílias: “Query (Reply)” e “Query (Request)”, com quatro mensagens em cada família, e “Error” com cinco. A Tabela 1 relaciona os códigos ICMP e as descrições no caso de utilização do protocolo icmp em computadores.

Tabela 1: Códigos ICMP.

Tipo	Descrição	Utilização
0	Echo Reply	Ping
3	Destination Unreachable	
4	Source Quench	
5	Redirect	
8	Echo Request	Ping
9	Router Advertisement	
10	Router Solicitation	NIS ICMP message type
11	Time Exceeded	
12	Parameter Problem	
13	Timestamp Request	
14	Timestamp Reply	
15	Information Request	
16	Information Reply	NIS ICMP message type
17	Address Mask Request	
18	Address Mask Reply	
30	Traceroute	
31	Datagram Conversion Error	
32	Mobile Host Redirect	
33	IPv6 Where-Are-You	
34	IPv6 I-Am-Here	
35	Mobile Registration Request	
36	Mobile Registration Reply	
37	Domain Name Request	
38	Domain Name Reply	
39	SKIP	
40	Photuris	

2.1.2.3.4 Internet Group Management Protocol (IGMP)

Com a necessidade de transferência de dados de forma simultânea para diversos nós, ou seja, um remetente e vários destinatários em uma rede evidenciou a necessidade de tráfego IP *multicast*, como é o caso, por exemplo, da videoconferência. Para que isso seja possível, o protocolo IGMP envia um único datagrama para os nós locais e o faz o

mesmo por meio de roteadores aos nós localizados em outras redes interessados em receber o tráfego. Para que isso seja possível, o protocolo IGMP oferece um mecanismo para que os *hosts* informem seu interesse em receber o tráfego ou interromper o recebimento deste (CARNE, 2004).

A Figura 15 diferencia o envio de pacotes *unicast* e *multicast*. Enquanto o primeiro trata de comunicações entre um host de origem e um de destino, o segundo encaminha apenas uma cópia da comunicação para todos os destinatários. O protocolo IGMP utiliza somente dois tipos de pacotes: consulta e resposta, cada um contendo algumas informações simples para controle. Sendo assim, é vagamente análogo ao protocolo ICMP (TANENBAUM, 2003).

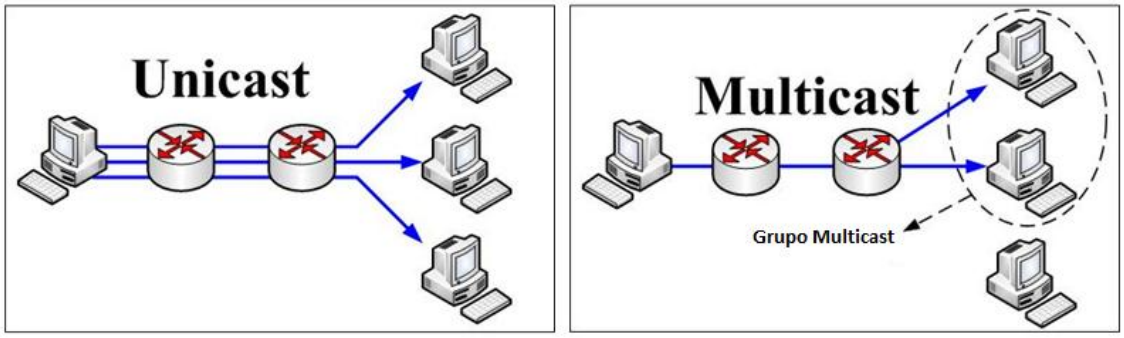

Figura 15: Unicast x Multicast

2.1.2.4 Interface com a rede – Camada 1

A camada de interface de rede é a camada de mais baixo nível no modelo TCP/IP, estando intimamente ligada ao hardware através do *driver* do dispositivo. Os datagramas recebidos por esta camada são convertidos para serem transmitidos utilizando o meio físico disponível, seja sem fio (*wireless*), óptico, cobre, entre outros.

2.1.3 Comparativo entre os modelos (OSI e TCP/IP)

Normalmente existe uma dificuldade em correlacionar as normas ISO OSI e o protocolo TCP/IP. Segundo Dostálek e Kabelová (2006), ambos são divididos em

camadas e em cada camada são definidos os protocolos utilizados. De maneira geral, os modelos são incompatíveis. Semelhante ao modelo OSI da ISO, no modelo TCP/IP os dados são passados para baixo na pilha no momento do envio e são levados até o topo quando do recebimento. (SPORTAK, 2004). Em cada camada são adicionadas informações do protocolo e de controle a fim de garantir a entrega dos dados no destino (HUNT, 2002). Esta adição de informações que ocorre em cada camada é chamada de encapsulamento.

2.1.4 Ameaças

Independente do tipo de rede e de sua utilização, certamente os *hosts* conectados a ela estão sujeitos a ameaças ou a tráfegos maliciosos, por maior que seja a sua proteção. Para que uma máquina (computador) seja infectada, basta que o mesmo tenha contato com um programa mal intencionado localizado na rede, em um *pen-drive* ou em um *e-mail.* (DAMATTO, 2011)

Praticamente todos sabem da existência dessas ameaças ou *malwares*, que são *softwares* que maliciosos que podem causar diversos danos ao computador (DAMATTO,2011). Já para Fuentes (2008), *malware* é um termo geral que pode definir qualquer software malicioso que prejudique o computador. Segundo Manson (1999), *malware* é um programa cuja intenção é causar algum dano, sendo a sua instalação feita sem o conhecimento do usuário.

Os *malwares* dependendo da sua intenção: *adware* (propaganda), vírus e *worms* (infecção), programas espiões (*spyware)* e *trojan (*cavalo de tróia – ameaça oculta em outro software), podem ser definidos de várias formas (DAMATTO, 2011):

Adware é um tipo de software que tem como objetivo exibir propagandas sem a intervenção do usuário. Geralmente esses anúncios são feitos em massa e em momentos diversos, sendo difícil o usuário saber qual software gera tais anúncios.

Vírus é um programa que tem a intenção da replicação. O vírus normalmente tenta se espalhar pela rede de um computador para o outro. O vírus visa causar danos nos dados do usuário e nos software instalados na máquina.

Os Vermes ou *worms*, são um código mal intencionado que também se auto propaga pela rede. A sua ação causa consume dos recursos da rede ou do sistema local

da máquina. Alguns necessitam que o usuário o execute para se propagar, enquanto outros funcionam sem a intervenção do usuário após a infecção sem que o usuário perceba.

Spywares também são conhecidos como software de rastreamento. Os *spywares* geralmente buscam informações pessoais, realizam modificações nas configurações do navegador de Internet, além de degradar o desempenho do sistema e invadir a privacidade do usuário.

Os *trojans* ou cavalos de tróia é um programa que a primeira vista parece ser útil ou inofensivo, porém a sua real intenção é danificar o sistema no qual foi executado. Normalmente essa ameaça se replica via mensagens de email. Sua ação mais comum além de tentar inutilizar o sistema é abrir uma "porta dos fundos" no sistema, pela qual o hacker passa a ter acesso ao sistema e geralmente a utiliza para roubar dados.

Apesar de ser comum se ouvir falar mais da existência de vírus, *adware*, *spyware*, *trojans e fishing*, entre outros, o mais preocupante para os administradores de rede são as vulnerabilidades que programas (softwares e sistemas operacionais) apresentam e é contra estas ameaças que é muito difícil proteger os sistemas e os usuários de uma instituição, como aplicações web, sites, servidores e bases de dados, por exemplo (RIST, 2009)

Segundo a CERT.br (2012), o ataque mais utilizado com 34,65% entre Abril e Junho de 2012 foi o ataque conhecido como *Scan*, que são notificações de varreduras em redes de computadores, com o intuito de identificar quais computadores estão ativos e quais serviços estão sendo disponibilizados por eles. É amplamente utilizado por atacantes para identificar potenciais alvos, pois permite associar possíveis vulnerabilidades aos serviços habilitados em um computador.

Em seguida os ataques de tipo Fraude, com 19,4%, onde segundo o dicionário Houaiss, é "qualquer ato ardiloso, enganoso, de má-fé, com intuito de lesar ou ludibriar outrem, ou de não cumprir determinado dever; logro". Esta categoria engloba as notificações de tentativas de fraudes, ou seja, de incidentes em que ocorre uma tentativa de obter vantagem. Tais dados podem ser visualizados na Figura 16.

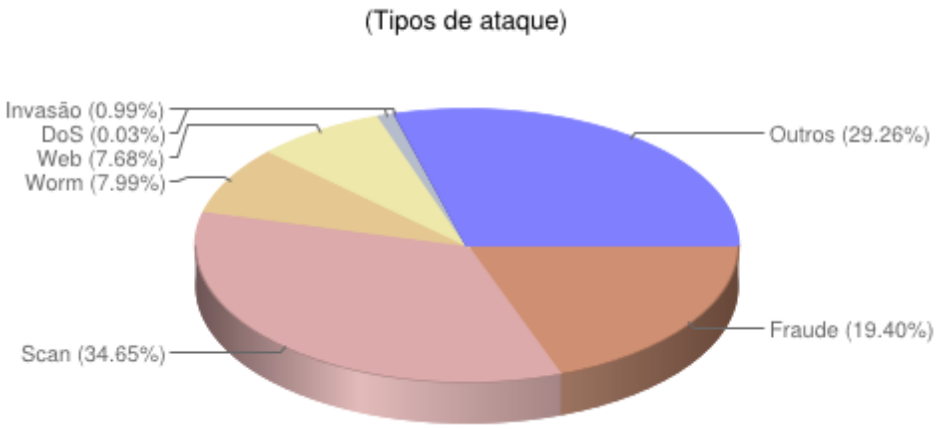

Figura 16: Incidentes reportados (abril a junho/2012).
Fonte CERT.br, 2012

Os *scans* reportados ao CERT.br no período de abril a junho de 2012 teve como destino principal os servidores de email, no qual rodam serviços na porta 25. Logo em seguida como a segunda mais atacada, temos a porta 3389, comumente utilizada para o serviço Terminal Services, ou área de trabalho remota, como ilustra a Figura 17.

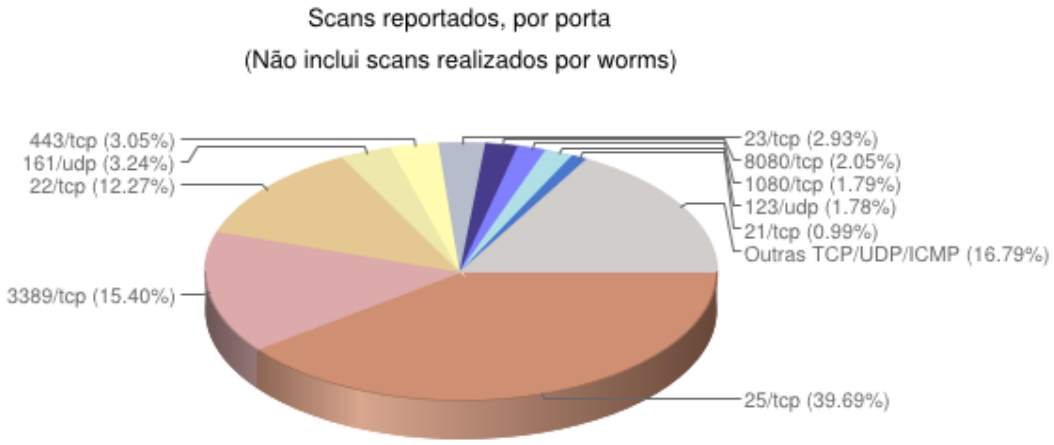

Figura 17: Scans por categoria (abril a junho/2012).
Fonte: CERT.br, 2012

As tentativas de fraude se subdividem em Cavalos de Tróia: Tentativas de fraude com objetivos financeiros envolvendo o uso de cavalos de tróia, Páginas Falsas: Tentativas de fraude com objetivos financeiros envolvendo o uso de páginas falsas, Direitos Autorais: Notificações de eventuais violações de direitos autorais e Outros. O tipo de fraude “Cavalo de tróia foi a mais utilizada segundo a CERT.br entre abril e junho de 2012, com 58,98% dos ataques, conforme Figura 18.

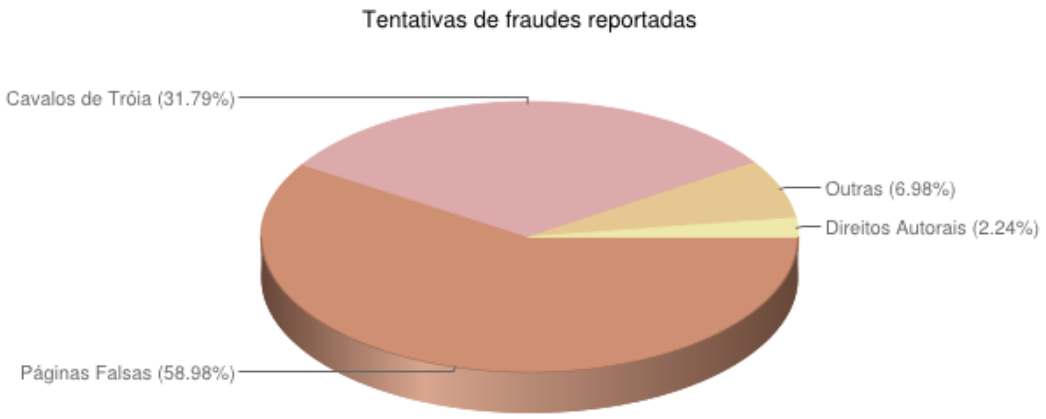

Figura 18: Tentativas de fraude por categoria (abril a junho/2012).
Fonte: CERT.br, 2012

2.1.5 Proteção

Nos últimos dez anos, segundo Long (2006) o número de ataques contra sistemas de informação tem aumentado significativamente. Por conseqüência, tecnologias de informação de segurança, como autenticação e criptografia ganharam muito mais importância.

Enquanto isso, os sistemas de detecção de intrusão (IDS) surgiram como uma nova abordagem para proteger os sistemas e as informações. Um IDS monitora um sistema de informação para gerar alertas de ataques. Uma vez que os ataques são detectados, o IDS gera esses alertas para relatá-los. Os alertas são apresentados a especialistas ou um sistema de conhecimento que os avalia e inicia uma resposta. É uma tarefa desafiadora avaliar os alertas gerados pelos IDS e gerar uma resposta adequada. (LONG, 2006)

Por exemplo, tem sido observado que os IDSs podem gerar milhares de alertas por dia, porém até 99% dos alertas podem ser falso-positivos. Esta grande quantidade de alertas em sua maioria falsas torna difícil a avaliação pelo profissional em identificar as verdadeiras ameaças. (MELLO, 2004), (LONG, 2006)

O Snort é um exemplo típico de software IDS (sistema de detecção de intrusão) e muitos estudos têm sido feitos melhorar o Snort. Wuu (2003) propôs uma modificação ao Snort para torná-lo capaz para detectar ataques seqüenciais. Isto é, enquanto Snort avalia apenas um pacote de cada vez o trabalho propõe tentar identificar seqüências de

pacotes que representam ataques e para aumentar o Snort com regras que disparam quando tais seqüências são detectadas.

Para que isso seja possível, é necessário aplicar técnicas de mineração de dados para extrair assinaturas de ataque e convertê-los em regras do Snort. O Snort é um mecanismo de detecção de intrusão baseado em comportamentos ou regras, que dispara um alerta quando uma série de pacotes de entrada coincide com as assinaturas (regras) (LONG, 2006).

Coit (2001), Sourdis et al. (2004), Liu et al.14 (2004), e Yu et al. (2004) aplicaram técnicas de padrões de correspondência para melhorar a velocidade de detecção do Snort. A precisão da detecção, no entanto, não foi tratada em qualquer uma destes trabalhos. Muitos trabalhos se dedicam a lidar com esse problema, principalmente utilizando técnicas de inteligência artificial. Em particular, Chavan (2004) utiliza redes neurais e um sistema de lógica *Fuzzy* e Shwartz (2002) discute a possibilidade de melhorar a eficácia do Snort através da mudança de funcionamento baseado em regras para o funcionamento baseado em casos.

A utilização de *Clustering* (conjunto) nas regras de detecção de intrusão também foi comprovada com sucesso em outros estudos. Cuppen (2002) aplicou esse modelo em um ambiente de detecção multi-sensor de intrusão, onde os sensores são os IDSs. Este IDS multi-sensor de certa forma, oferece uma visão mais precisa do estado do sistema e, em particular, faz com que seja mais fácil distinguir falsos alertas dos reais. Na medida em que o sistema opera em tempo real, ele pode ser visto como um sistema de agrupamento de tempo real. Essa abordagem parte do pressuposto de que todos os alertas em uma sessão particular da rede pertencem ao mesmo ataque. Assim, a correlação é feita simplesmente olhando para os identificadores de sessão de rede idênticos, como os IP de origem e a porta, IP de destino e a porta.

Já Ning et al. (2004) introduziu uma técnica que constrói cenários de ataque pela correlação dos pré-requisitos básicos para que o ataque ocorra e das conseqüências dos ataques. Com o intuito de facilitar a investigação de grandes conjuntos de alertas, vários softwares de análise interativos foram introduzidas. *Clustering Analysis* um desses softwares, é utilizado sendo aplicado a um conjunto de alertas para encontrar as características comuns.

2.2 KDD E MINERAÇÃO DE DADOS

A descoberta de conhecimento em bases de dados (KDD) pode ser definida como o processo de extração de informação a partir de dados armazenados em uma base de dados, um conhecimento implícito, previamente desconhecido, potencialmente útil e compreensível (GOLDSHMIDT, 2005).

Os termos *knowledge Discovery in databases* (KDD) e *data mining* (mineração de dados) são muitas vezes usados como sinônimos. Na verdade, houve muitos outros nomes dados a este processo de descobrimento de padrões úteis (ocultos) nos dados: extração de conhecimento, descoberta de informações, análise exploratória de dados, colheita de informações e reconhecimento de padrões não supervisionado.

Ao longo dos últimos anos KDD tem sido utilizado para se referir a um processo que consiste em muitos passos, enquanto a mineração de dados é apenas um destes passos. As seguintes definições foram propostas por Usama Fayyad: (DUNHAM, 2003)

Definição de KDD: É o processo de procurar informação útil e padrões nos dados. (Fayyad, 1996)

Definição de *Data Mining*: É o uso de algoritmos para extrair informação e padrões a partir do processo de KDD. (FAYYAD, 1996)

2.2.1 Etapas do KDD

Segundo Martins (2011) o KDD é um processo que busca identificar padrões, associações, modelos ou informações relevantes, que estão ocultos em bases, repositórios, dentre outras formas de armazenamento de dados. Tal processo permite identificar padrões válidos, novos e com alta probabilidade de serem úteis e compreensíveis, envolvendo diversas áreas da ciência como aprendizagem de máquina, banco de dados, estatística, reconhecimento de padrões, entre outras. (MALUCELLI, 2010). Para um melhor entendimento, é apresentado o processo de KDD na Figura 19, que se subdivide em 5 etapas: seleção dos dados, pré-processamento, transformação dos dados, mineração dos dados e interpretação do conhecimento extraído. Cada etapa será melhor descrita a seguir.

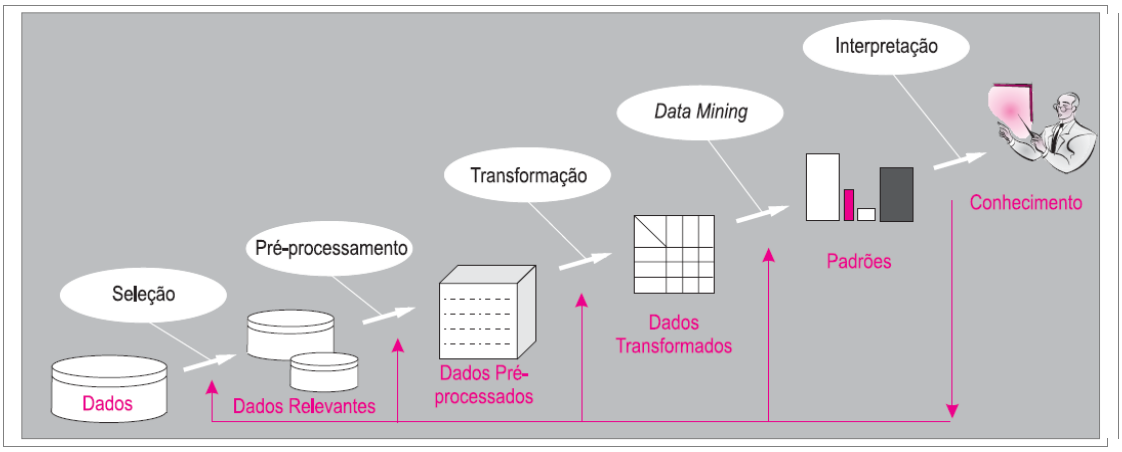

Figura 19: Processo do KDD.
Fonte: Fayaad, 1996

2.2.1.1 Seleção dos dados

De acordo com Dunham (2003), o dado necessário para o processo de mineração de dados pode ser obtido de muitas diferentes e heterogêneas origens. O primeiro passo é obter o dado de várias bases de dados, arquivos e origens não eletrônicas.

2.2.1.2 Pré-processamento

O dado a ser usado no processo pode ter informações incorretas ou faltantes, podendo ser proveniente de múltiplas fontes anômalas, envolvendo diferentes tipos de dados e métricas. Assim, pode ser necessária a execução de diferentes atividades neste momento. Dados incorretos podem ser corrigidos ou removidos, enquanto que os dados faltantes devem ser fornecidos ou estimados (muitas vezes com o uso de ferramentas de mineração de dados) (DUNHAM, 2003)

2.2.1.3 Transformação

Ainda de acordo com Dunham (2003), dados de diferentes origens devem ser convertidos em um formato comum antes serem processados. Alguns dados pode ser encodado ou transformado em um formato mais usual. A redução dos dados deve ser considerada para reduzir o número de possíveis valores que o dado em questão pode assumir.

2.2.1.4 Mineração de dados (D*ata mining*)

O termo Mineração de Dados (DM) surge inicialmente, como um sinônimo de KDD, mas é unicamente uma das etapas da descoberta de conhecimento em bases de dados, do processo global chamado de KDD (QUONIAM, 2001). O conhecimento que pode ser obtido através da mineração de dados tem se mostrado bastante útil em diversas áreas de pesquisa e de atuação, como informática, medicina, ensino público e privado, finanças, comércio, marketing, telecomunicações, meteorologia, agropecuária, bioinformáticas, entre outras (JONES, 2000).

A mineração de dados não é um processo simples e trivial; consiste em identificar, nos dados os padrões válidos, novos, potencialmente úteis e compreensíveis, envolvendo métodos estatísticos, ferramentas de visualização e técnicas de inteligência artificial (FAYYAD, 1996). Portanto, o processo do KDD utiliza conceitos de base de dados, estatística, ferramentas de visualização e técnicas de inteligência artificial, dividindo-se nas etapas de seleção, pré-processamento, transformação, mineração e interpretação/avaliação dos resultados (FAYYAD, 1996).

Dentre essas etapas, a mais relevante é a mineração de dados, foco de vários estudos em diversas áreas de conhecimento (WICKERT, 2007), que comprovam que a extração informação a partir de um conjunto de dados, e posteriormente a transformação dessa informação em conhecimento, é imprescindível para o processo de tomada de decisão.

Na mineração de dados, o dado é armazenado eletronicamente e a busca é automatizada, porém este processo não é relativamente novo. Economistas, estatísticos,

meteorologistas e engenheiros trabalham com a idéia que padrões nos dados podem ser procurados automaticamente, identificados e validados, sendo usados para previsões. (FRANK, 2005)

Com isso, o processo de mineração de dados consiste em analisar dados já presentes em bancos de dados. Por exemplo, uma base de dados das compras dos clientes de algum estabelecimento, juntamente com o perfil desses clientes, é a chave de um problema. Correlacionar essas informações analisando o comportamento desses clientes trás uma vantagem estratégica, na qual a empresa pode oferecer um determinado produto somente a um grupo especifico de clientes que realmente se interessam por aquela oferta. (FRANK, 2005)

Para tanto, a mineração de dados ou *data mining* (DM) possui várias etapas: a definição do problema; a seleção de todos os dados e a posterior preparação destes dados, o que inclui o pré-processamento e reformatação dos dados (retirada de caracteres e linhas em branco, por exemplo) e análise dos resultados obtidos do processo de DM (CARDOSO, 2008). A descoberta do conhecimento deve apresentar as seguintes características: ser eficiente, genérica (possibilidade de ser aplicável a vários tipos de dados) e flexível (facilmente modificável) (STEINER, 2006). O processo de desenvolvimento de DM envolve vários procedimentos, métodos e algoritmos que possibilitam a extração de novos conhecimentos (CARDOSO, 2008). Entre as tarefas de DM, é possível destacar algumas que são as mais utilizadas: associação, classificação, regressão, clusterização e sumarização (GOLDSHMIDT, 2005).

2.2.2 Tarefas de mineração de dados

Na mineração de dados, são definidas as tarefas e os algoritmos que são utilizados de acordo com os objetivos do estudo, com o objetivo de obter uma resposta para o problema em questão (MATOS, 2006). Os algoritmos de extração de padrões podem ser agrupados em atividades preditivas e descritivas.

As duas principais tarefas para predição são a classificação e a regressão. Na tarefa de classificação o objetivo é descobrir uma função que agrupe um conjunto de dados de acordo com variáveis pré-definidas, denominadas classes. Essas funções também podem ser aplicadas em novos dados, de forma de prever a classe na quais

esses registros se enquadram. Dentre os algoritmos de classificação, pode-se destacar os mais utilizados: Redes Neurais, Back-Propagation, Classificadores Bayesianos e Algoritmos Genéticos (PEREIRA, 2008).

Já na tarefa de regressão, consiste na busca por funções lineares ou não, sendo que a variável a ser predita é um atributo numérico (contínuo) presente em banco de dados com valores reais (PEREIRA, 2001). Para implementar a tarefa de regressão, é utilizado métodos de estatística e de redes neurais.

Na tarefa de clusterização, que é usada para separar os registros de uma base de dados em subconjuntos, de forma que os elementos de um subgrupo compartilhem características em comum, com o objetivo que cada tenham a máxima similaridade dentro do grupo e mínima similaridade com os dados dos outros grupos. Diferindo da tarefa de classificação, onde as variáveis são pré-definidas, a clusterização necessita identificar automaticamente os grupos de dados, para posteriormente agrupá-los.(ZHU, 1999). Os algoritmos mais utilizados nessa tarefa são os K-Means, KModes, K-Protopypes, K-Medoids, Kohonem, dentre outros (SCARPEL, 2007).

A associação consiste em identificar e descrever associações entre variáveis nos mesmo registros ou associações entre registro diferentes que ocorram simultaneamente, de forma freqüente em uma base de dados (CARDOSO, 2008). Também é realizada comumente a procura de associações entre registros durante um intervalo de tempo (ABBOTT, 2006). Para tanto, os algoritmos Apriori e GSP (Generalized Sequential Patterns - Padrão Seqüencial Geral), dentre outros, são alguns utilizados na tarefa de descoberta de associações.

Já a sumarização procura identificar e indicar características comuns entre um conjunto de dados. Essa tarefa é realizada nos agrupamentos obtidos na tarefa anteriormente na clusterização, sendo a Lógica Indutiva e Algoritmos Genéticos exemplos de tecnologias que podem implementar a sumarização (GOLDSHMIDT, 2005).

2.2.3 Métodos de mineração de dados

Os métodos são tecnologias existentes, independente do contexto mineração de dados, uma vez que, aplicados na KDD, produzem bons resultados em diversas áreas, transformando dados em conhecimento útil e favorecendo as atividades e tomadas de decisão não somente baseada no sentimento do gestor (MARTINS, 2011). Os principais métodos utilizados para tal fim são: Rede Neurais, Árvore de Decisão, Algoritmos Genéticos (AGs), Lógica Nebulosa (Fuzzy logic) e Estatística (MEIRA, 2008).

A Rede Neural Artificial (RNA) é uma técnica que simula o funcionamento do cérebro humano, através de um modelo matemático para reconhecimento de imagens e sons, com capacidade de aprendizado, generalização, associação e abstração. As redes neurais é construída por sistemas paralelos distribuídos compostos por unidades simples de processamento (neurônios) (GOLDSHMIDT, 2005).

Os neurônios compõem uma ou mais camadas interligadas por um grande número de conexões (sinapses); na maioria dos modelos, essas conexões estão associadas a pesos, os quais, após o processo de aprendizagem, armazenam o conhecimento adquirido pela rede (TARAPANOFF, 2000).

As RNAs são utilizadas com sucesso para modelar relações envolvendo séries temporais complexas em várias áreas do conhecimento (TARAPANOFF, 2000). Uma grande vantagem das redes neurais sobre outros métodos é que elas não requerem informação detalhada sobre os processos físicos do sistema a ser modelado, abrindo a possibilidade de este ser descrito explicitamente na forma matemática (modelo de entrada-saída) e ainda por ser robusta e ter uma alta taxa de aprendizado (ABBOTT,2006). Após várias execuções (iterações), a RNA aprende padrões, procura relacionamentos e constrói modelos automaticamente.

A Árvore de Decisão é um modelo representado graficamente por nós e galhos, parecido com uma árvore, porém deve ser analisada do topo para as folhas. As árvores de decisão também são chamadas de árvores de classificação ou de regressão (GOLDSHMIDT, 2005) . Para melhor entendimento, é apresentado na Figura 20 um exemplo de uma árvore de decisão (MARTINS, 2011)

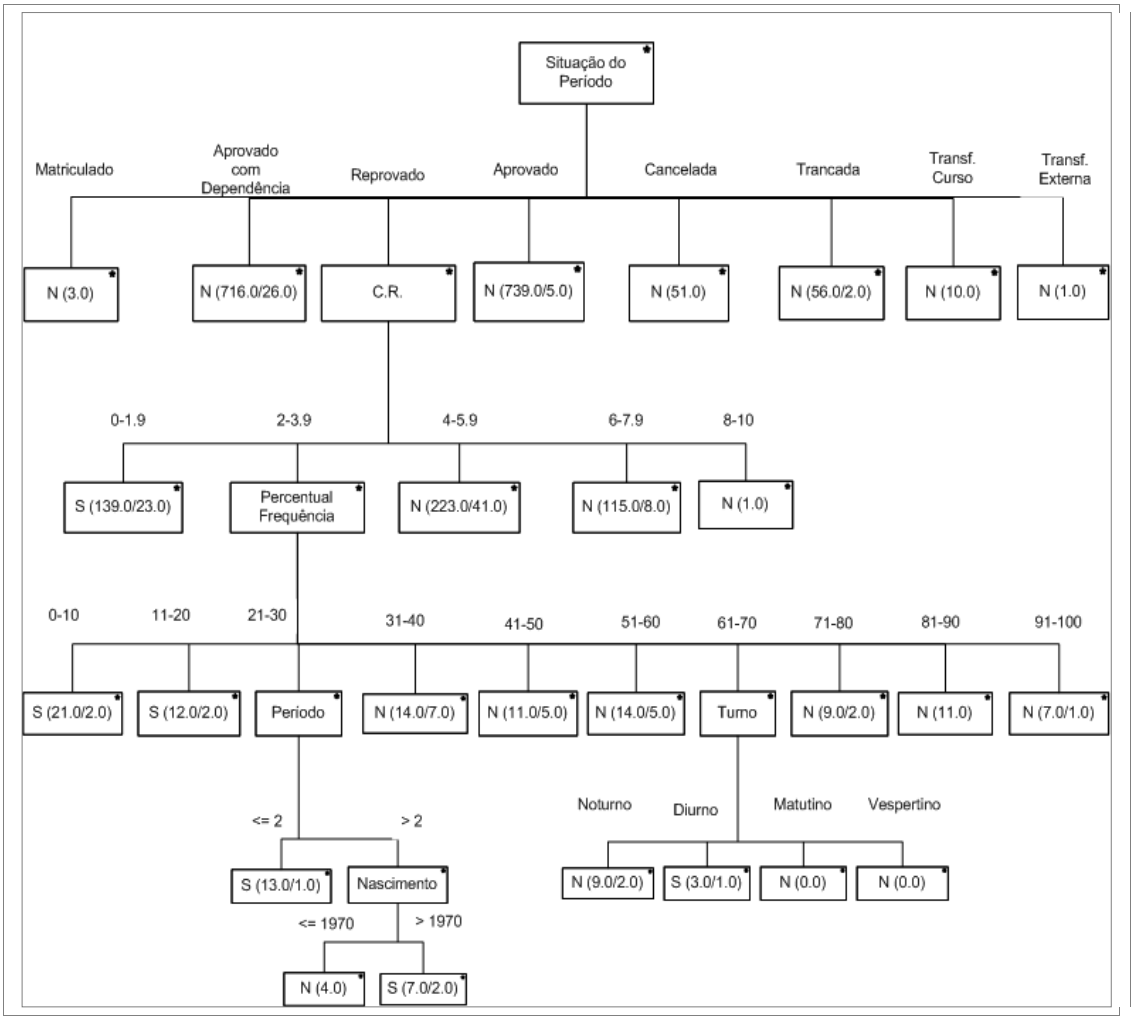

Figura 20: Exemplo de árvore de decisão.
Fonte: Martins, 2011

O intuito de uma Árvore de Decisão é produzir um modelo para facilitar a compreensão de quais variáveis e interações conduzem o problema estudado. Algumas pesquisas recentes têm utilizado a Árvore de Decisão para predizer e obter conhecimento (MEIRA, 2008).

Por outro lado, os Algoritmos Genéticos (AG) formulam estratégias de otimização inspiradas nos princípios observados na evolução natural e na genética, para solução dos problemas. Os AGs utilizam operadores de seleção, cruzamento e mutação para desenvolver sucessivas gerações de soluções – chamado de gerações. Com a evolução do algoritmo, somente as gerações com maior poder de previsão (mais adequadas ao problema) sobrevivem, até convergirem numa solução ideal ou ótima (CARDOSO, 2008).

As Redes Bayesianas, segundo ABBOTT (2006) surgiram recentemente como uma poderosa tecnologia de mineração de dados, fornecendo representações gráficas de

distribuições probabilísticas baseadas na contagem da ocorrência dos dados num determinado conjunto, ou seja, representam graficamente os relacionamentos entre as variáveis (ou dados) em questão.

2.2.4 Interpretação do conhecimento

Como os resultados da mineração de dados são apresentados para os usuários é extremamente importante porque a utilidade dos resultados é completamente dependente disso. Várias formas de visualização e interfaces estão disponíveis e são usadas nessa etapa (DUNHAM, 2003).

2.2.5 Ferramentas

Uma das aplicações mais utilizadas no processo de mineração de dados é o WEKA. O WEKA foi desenvolvido na Universidade de Waikato na Nova Zelândia, e a abreviatura WEKA significa *Waikato Environment for Knowledge Analysis*. Além da ferramenta, WEKA, também é uma ave que não voa com uma natureza inquisitiva encontrada apenas nas ilhas da Nova Zelândia.

Segundo FRANK (2005), o projeto WEKA é baseado em uma coleção organizada de algoritmos de aprendizado de máquina e ferramentas de pré-processamento de dados. A ferramenta provê suporte extensivo para todo o processo de mineração de dados, incluindo a preparação dos dados, avaliação estatística dos sistemas de aprendizagem e visualização gráfica dos dados de entrada e do resultado da aprendizagem. Bem como a ferramenta possui uma grande variedade de algoritmos de aprendizagem, incluindo uma ampla variedade de ferramentas de pré-processamento. Este kit diversificado e abrangente é acessado através de uma interface gráfica para que seus usuários possam comparar diferentes métodos e identificar aqueles que são mais adequadas para o problema em questão.

Além da interface gráfica a sua execução pode ser feita via linha de comando. Na interface gráfica da aplicação, muito mais conveniente para o usuário, é escrito na

linguagem de programação Java e distribuída sob os termos da Licença Pública Geral GNU. Ele roda em qualquer plataforma e foi testado no Linux, Windows e sistemas operacionais Macintosh. Ele fornece uma interface única para muitos algoritmos diferentes de aprendizagem, juntamente com métodos de pré e pós-processamento, e também é utilizado para avaliar o resultado da aprendizagem de qualquer conjunto de dados (FRANK,2005). A figura 21 apresenta a interface gráfica inicial do WEKA, e cada uma das suas interfaces de trabalho que serão descritas abaixo.

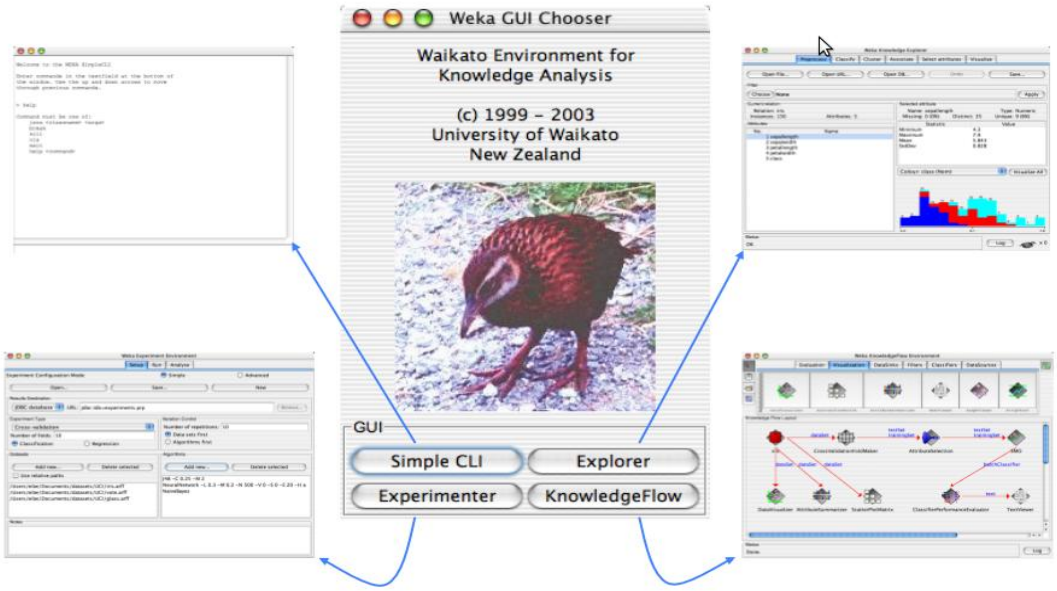

Figura 21: Interface do Software WEKA.
Fonte: Frank, 2005

Segundo Frank (2005), a maneira mais fácil de utilizar o WEKA é via sua interface gráfica, chamada *Explorer*. Ela dá acesso a todas as facilidades usando o menu de seleção e o preenchimento de formulários. Por exemplo, é possível ler um conjunto de dados de um arquivo ARFF (ou de uma planilha) e criar uma árvore de decisão a partir desses dados, porém árvores de decisão são só o início. Existem muitos outros algoritmos para serem utilizados, e a interface *Explorer* auxilia a utilização deles.

Por trás da interface *Explorer,* representada na Figura 22, e das interfaces *Knowledge Flow* e *Experimenter* existe a funcionalidade básica do WEKA, a interface *Command-line* ou linha de comando. De acordo com o desenvolvedor da ferramenta, Frank (2005), a interface de linha de comando do WEKA consiste em um painel simples com uma linha na parte inferior, similar a interface de linha de comando dos sistemas operacionais.

Alternativamente pode-se utilizar o WEKA também através das interfaces de linha de comando dos sistemas operacionais, bastando para isso para executar as classes weka.jar diretamente. Neste caso é preciso primeiro definir a variável de ambiente *CLASSPATH* como explicado no arquivo leia-me (*README)* do Weka.

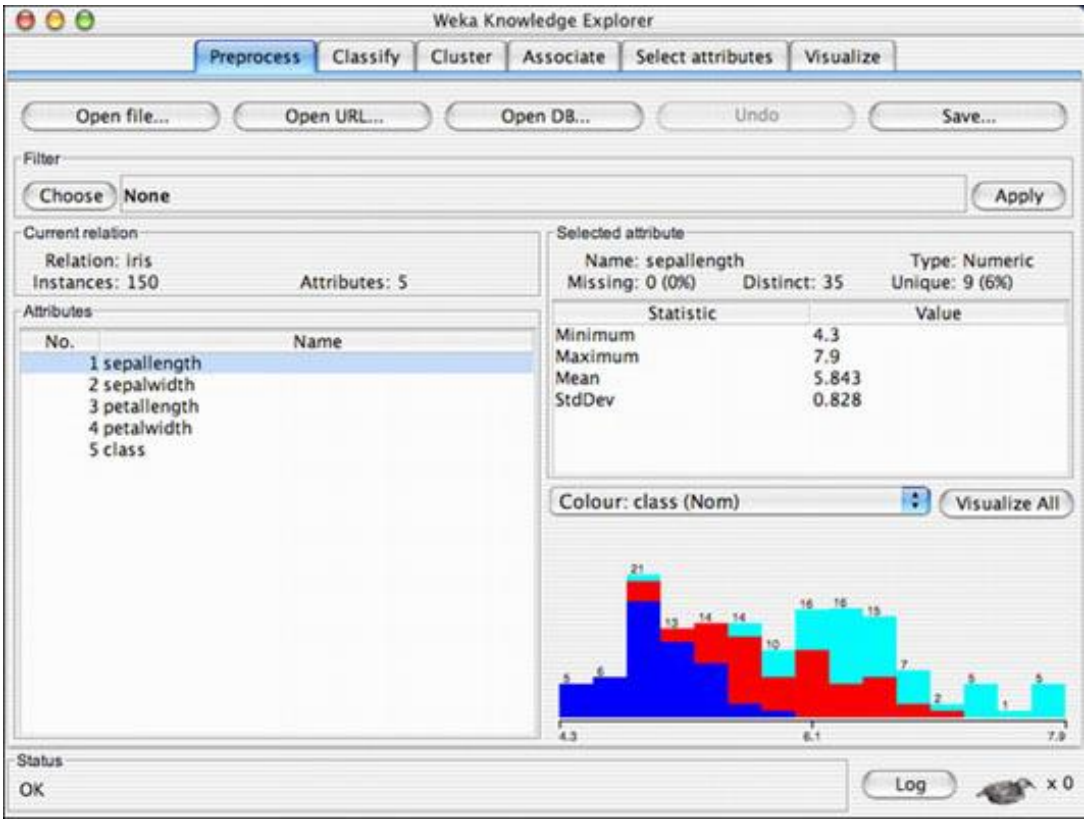

Figura 22: Interface Explorer do WEKA.
Fonte Frank, 2005

Além do WEKA, segundo Camilo (2009), existem diversas ferramentas que foram desenvolvidas com o intuito de tornar a mineração de dados uma tarefa menos técnica e mais acessível a profissionais de várias áreas. Neste sentido, são apresentadas algumas outras ferramentas mais utilizadas atualmente.

Clementine: É uma das ferramentas líderes de mercado, foi desenvolvida pela SPSS. O Clementine suporta o processo CRISP-DM, além de possuir outras facilidades (VASCONCELOS, 2004).

SAS *Enterprise Miner Suíte*: Ferramenta desenvolvida pela empresa SAS. É uma das ferramentas mais conhecidas para mineração. Possui módulos para trabalhar em todas as etapas do processo de mineração (SAS, 2012).

SAS *Text Miner*: Ferramenta da SAS para mineração de textos (SAS, 2012).

***Oracle Data Mining* (ODM):** É uma ferramenta para a Mineração de Dados desenvolvida pela Oracle para o uso em seu banco de dados ORACLE (ORACLE, 2012).

KXEN *Analytic Framework*: Ferramenta de Mineração de Dados comercial que utiliza conceitos do Professor Vladimir Vapnik como Minimização de Risco Estruturada (Structured Risk Minimization ou SRM) e outros (KXEN, 2012).

IBM *Intelligent Miner*: Ferramenta de mineração da IBM para a mineração de dados no banco de dados DB2 (IBM, 2012).

Pimiento: Ferramenta livre para mineração de textos (PIMIENTO,2012).

MDR: Ferramenta livre em Java para detecção de interações entre atributos utilizando o método da multifactor dimensionality reduction (MDR) (DARTMOUTH, 2012).

LingPipe: Ferramenta de mineração livre voltada para análise linguística (ALIAS-I, 2012).

KNIME: Plataforma de mineração de dados aberta, que implementa o paradigma de pipelining de dados (KNIME, 2012).

3 METODOLOGIA

3.1 EQUIPAMENTOS E SOFTWARES DE COLETA DOS DADOS

Para que a coleta dos dados seja possível, foi utilizado um servidor de análise, para qual foi direcionado todo o tráfego de rede do Instituto. Nesse servidor foi instalado o sistema operacional Linux Debian 6.0.5 x64 (64 bits) e as seguintes ferramentas: um sistema de detecção de intrusos ou *intrusion prevention system* (IDS) – Snort 2.9.3; Baynard 2.1.10 (faz a interface do software IDS com o banco de dados); Banco de dados Mysql 14.04; e o *Basic Analysis and Security Engine* – BASE 1.4.5 (interface web para o IDS).

3.1.1 Instalação do servidor de análise

Como mídia de instalação foi utilizada a imagem disponível no site http://www.debian.org/distrib/netinst. Após download da imagem, a mesma foi gravada em um CD-R.

3.1.1.1 Instalação do sistema operacional e software básico

Para a instalação do servidor utilizado para o projeto são necessárias duas placas de rede, uma para a gerência do equipamento e outra para a coleta dos dados. A interface de rede eth0 foi utilizada como interface de gerência e a interface eth1 como interface de coleta. Foram escolhidas as opções padrões da instalação, utilizando a opção chamada de “instalação mínima”, conforme Figura 23.

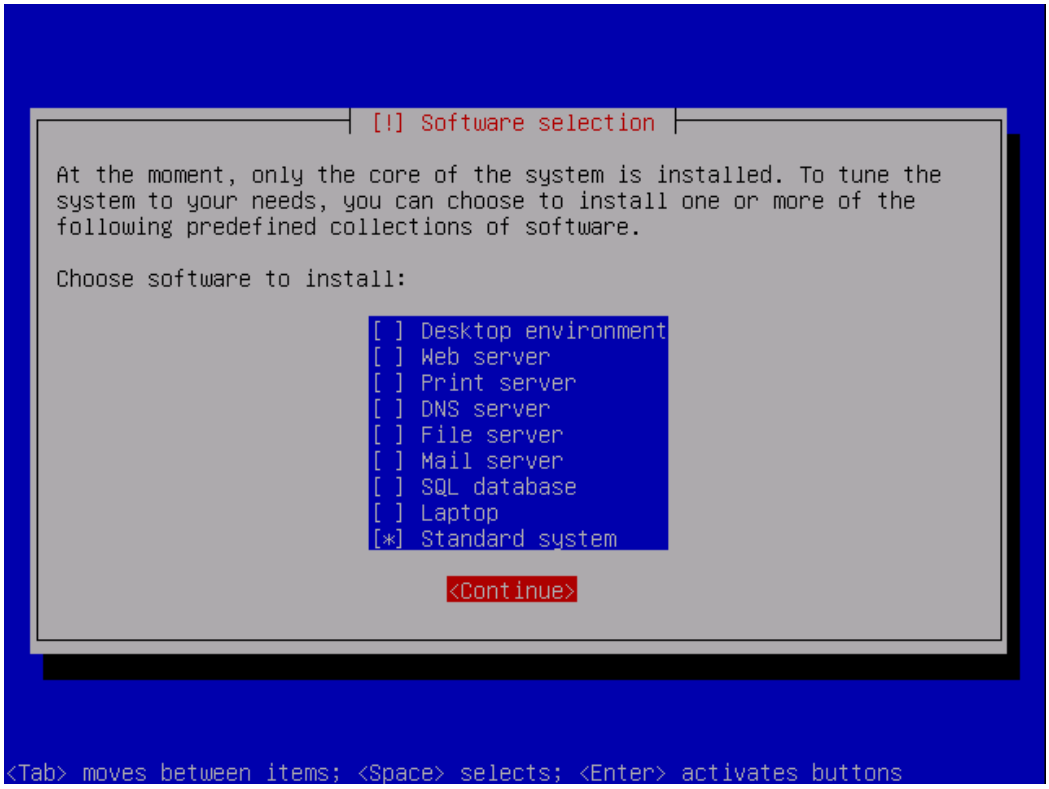

Figura 23: Instalação mínima do Debian.

Após a instalação completa, foi necessário atualizar os repositórios do Debian para proceder a instalação dos pacotes necessários, conforme Figura 24:

apt-get update

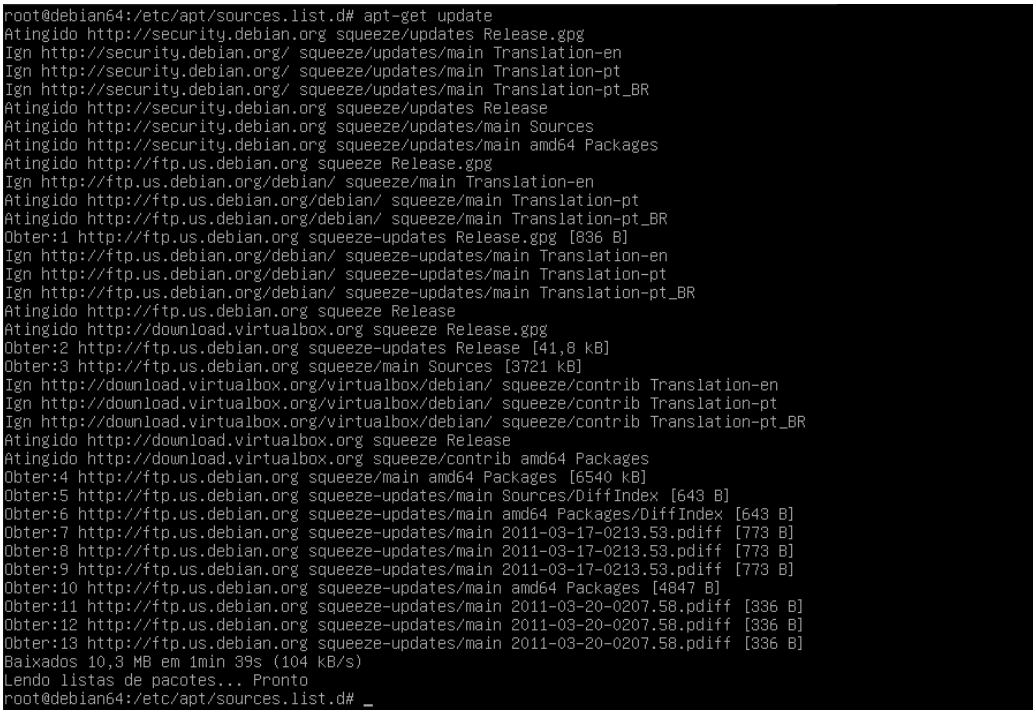

Figura 24: Atualização dos repositórios.

Após a atualização dos repositórios foi instalado o servidor SSH para acesso ao servidor:

apt-get -y install ssh

Devido aos repositórios do Debian não possuir as últimas versões de muitos pacotes, foi adicionado ao source.list do debian o repositório dotbed.org:

vi /etc/apt/sources.list

Adicionadas as seguintes linhas:

deb http://packages.dotdeb.org squeeze all

deb-src http://packages.dotdeb.org squeeze all

Instalada a chave GnuPG do repositório dotdeb:

cd /usr/src && wget http://www.dotdeb.org/dotdeb.gpg

cat dotdeb.gpg | apt-key add –

Após o repositório ser adicionado, foi feita a instalação dos pacotes necessários. Durante a instalação o apt pede a senha usuário root para o servidor mysql, conforme a Figura 25.

apt-get update && apt-get -y install apache2 apache2-doc autoconf automake bison ca-certificates ethtool flex g++ gcc gcc-4.4 libapache2-modphp5 libcrypt-ssleay-perl libmysqlclient-dev libnet1 libnet1-dev libpcre3 libpcre3-dev libphp-adodb libssl-dev libtool libwww-perl make mysqlclient mysql-common mysql-server ntp php5-cli php5-gd php5-mysql php-pear sendmail sysstat vim

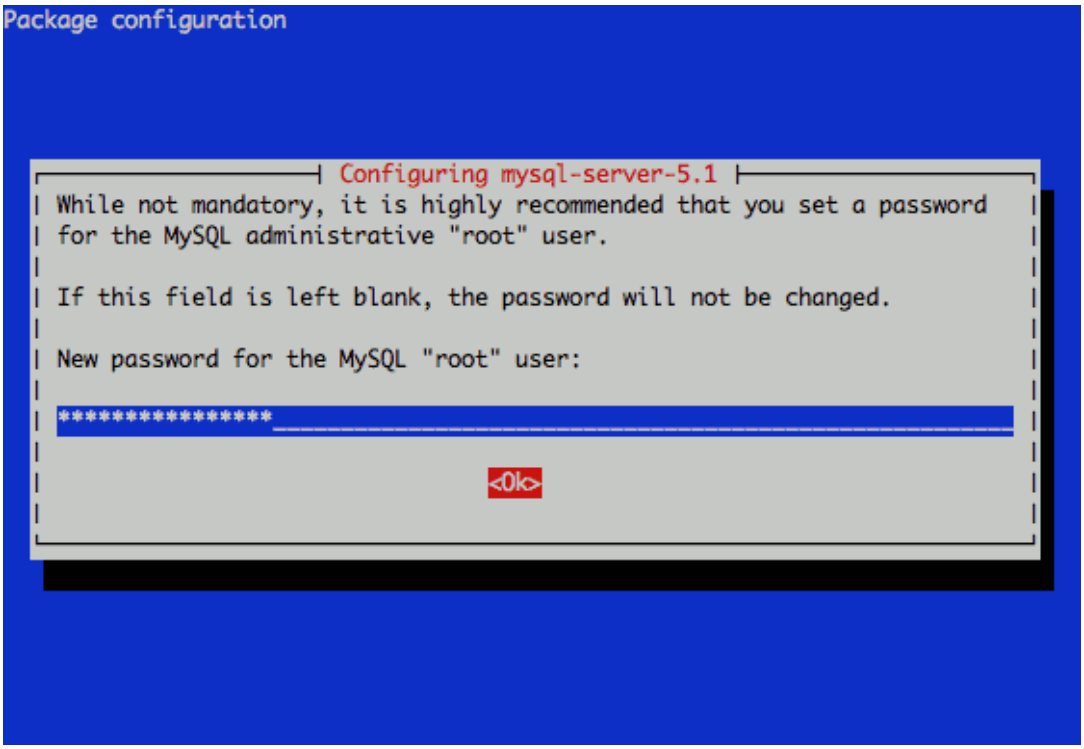

Figura 25: Senha para o servidor MySQL.

Após a instalação dos pacotes, é necessário desabilitar os parâmetros "*Large Receive Offload*" e *"Generic Receive Offload"* da placa de rede responsável pela coleta dos dados:

ethtool -K eth1 gro off

ethtool -K eth1 lro off

3.1.1.2 Instalação dos pré-requisitos do snort

Para o correto funcionamento do Snort, é necessário a instalação de alguns softwares e bibliotecas, como o libpcap, libdnet e o daq:

Instalação do libpcap

```
# cd /usr/src && wget http://www.tcpdump.org/release/libpcap-1.3.0.tar.gz
# tar -zxf libpcap-1.3.0.tar.gz && cd libpcap-1.3.0
# ./configure --prefix=/usr --enable-shared && make && make install
```

Instalação do libdnet:

```
# cd /usr/src && wget http://libdnet.googlecode.com/files/libdnet-1.12.tgz
# tar -zxf libdnet-1.12.tgz && cd libdnet-1.12
# ./configure --prefix=/usr --enable-shared && make && make install
```

Instalação do daq:

```
# cd /usr/src && wget http://www.snort.org/dl/snort-current/daq-1.1.1.tar.gz
# tar -zxf daq-1.1.1.tar.gz && cd daq-1.1.1
# ./configure && make && make install
```

Atualização da biblioteca compartilhada:

```
# echo >> /etc/ld.so.conf /usr/lib
# echo >> /etc/ld.so.conf /usr/local/lib && ldconfig
```

3.1.1.3 Instalação e configuração do snort

A instalação do IDS Snort foi realizada segundo os comandos abaixo:

Download do snort

```
# cd /usr/src
# wget http://labs.snort.org/snort/2930/snort.2930.conf -O snort.conf
```

wget http://www.snort.org/dl/snort-current/snort-2.9.3.tar.gz -O snort-2.9.3.tar.gz

Logo após o download, foi descompactada a instalação:

tar -zxf snort-2.9.3.tar.gz && cd snort-2.9.3

Em seguida é feita a instalação:

./configure --enable-sourcefire && make && make install

mkdir /etc/snort /etc/snort/rules /var/log/snort /var/log/barnyard2 /usr/local/lib/snort_dynamicrules

touch /etc/snort/rules/white_list.rules /etc/snort/rules/black_list.rules

groupadd snort && useradd -g snort snort

chown snort:snort /var/log/snort /var/log/barnyard2

cp /usr/src/snort-2.9.3/etc/*.conf* /etc/snort

cp /usr/src/snort-2.9.3/etc/*.map /etc/snort

cp /usr/src/snort.conf /etc/snort

Após a instalação ser realizada, é necessário a alteração de alguns parâmetros no arquivo de configuração do Snort, conforme a Figura 26:

vi /etc/snort/snort.conf

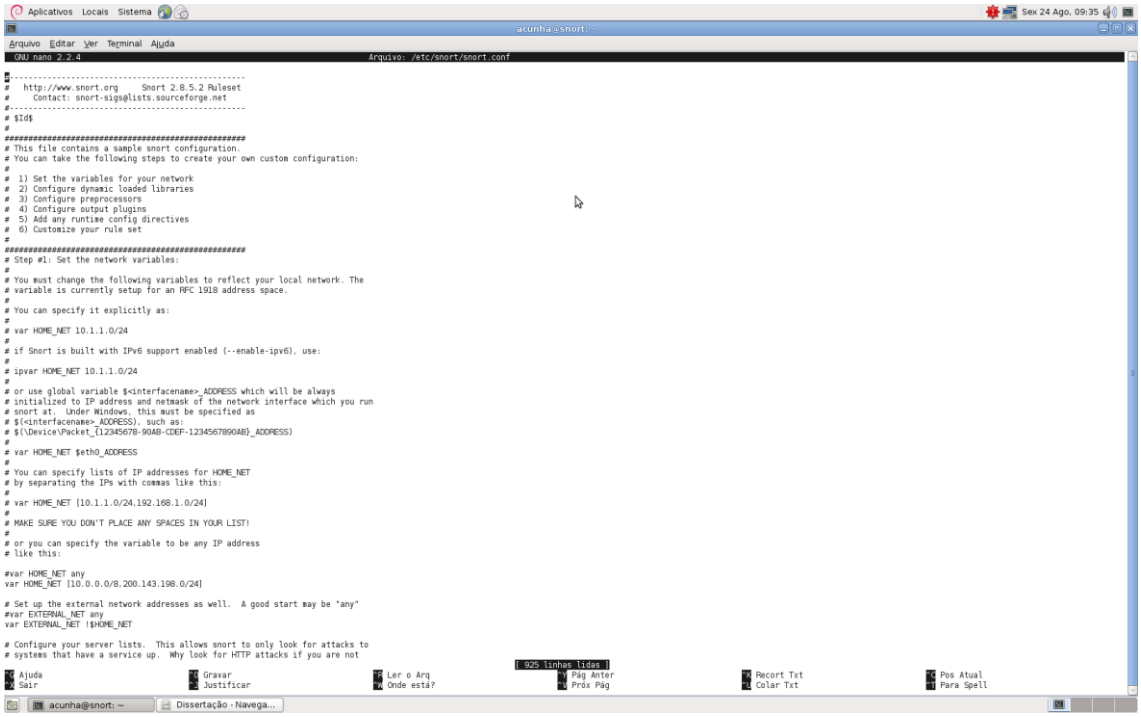

Figura 26: Arquivo snort.conf

Foram alteradas as seguintes linhas para o conteúdo abaixo:

Linha #45 - ipvar HOME_NET 10.0.0.0/8 //definida a classe de IP utilizada na rede interna.

Linha #48 - ipvar EXTERNAL_NET !$HOME_NET // Considera qualquer IP diferente do definido em "HOME_NET" como externo.

Linha #104 - var RULE_PATH ./rules //diretório com as regras do snort

Linha #113 - var WHITE_LIST_PATH ./rules

Linha #114 - var BLACK_LIST_PATH ./rules

Linha #297 – Adicionar esse conteúdo após "decompress_depth 65535" max_gzip_mem 104857600

Linha #538 – Adicionar a linha: output unified2: filename snort.log, limit 128

3.1.1.4 Instalação e configuração do baynard

Após a instalação do Snort, foi feito o download, instalação e configuração do Baynard, que faz a interface entre o Snort e o banco de dados. Ele é o processo responsável pela captura dos alertas do Snort e a gravação destes no banco de dados.

Download do Baynard:

```
# cd /usr/src
# wget https://nodeload.github.com/firnsy/barnyard2/tarball/master
```

Instalação do Baynard:

```
# tar -zxf master && cd firnsy-barnyard2-*
# autoreconf -fvi -I ./m4 && ./configure --with-mysql && make && make install
# mv /usr/local/etc/barnyard2.conf /etc/snort
# cp schemas/create_mysql /usr/src
```

Após a instalação é necessária a configuração do acesso ao banco de dados:

```
# vi /etc/snort/barnyard2.conf
```

Linha #215 Alterar para **output alert_fast**

E no fim do arquivo adicionar a seguinte linha:

output database: log, mysql, user=snort password=<*mypassword*> dbname=snort host=localhost

3.1.1.5 Configuração do servidor mysql

Com o Snort e o Baynard instalados e configurados é necessária a criação da base de dados onde os alertas do Snort serão gravados:

Primeiramente é feito o acesso ao banco de dados:

 # mysql -u root -p #Utilizar a senha definida durante a instalação do servidor MySQL.

Comando para criar a base "snort":

mysql> create database snort;

Com a base criada, é criado o usuário snort com privilégios para criar, selecionar, inserir, deletar e atualizar registros na base snort:

mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost;

E em seguida é definida a senha do usuário snort

mysql> SET PASSWORD FOR snort@localhost=PASSWORD('mypassword');

Logo após é feito o acesso a base snort e criadas as tabelas com o script do baynard, conforme Figura 27. Todo conteúdo do script está disponível no Apêndice D.

mysql> use snort;

mysql> source /usr/src/create_mysql -

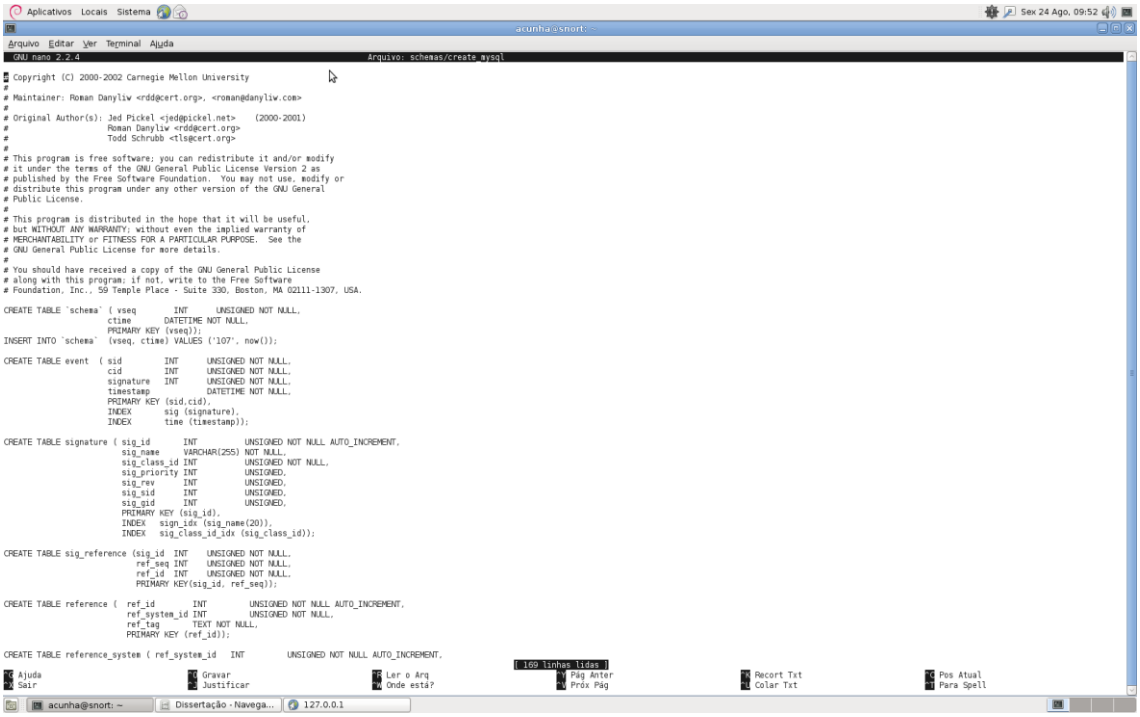

Figura 27: Script de criação das tabelas no banco MySQL.

Para visualizar se as tabelas foram criadas é possível utilizar o comando "*show tables*", conforme apresentado na Figura 28.

mysql> show tables;

mysql> exit

```
mysql> show tables;
+------------------+
| Tables_in_snort  |
+------------------+
| acid_ag          |
| acid_ag_alert    |
| acid_event       |
| acid_ip_cache    |
| base_roles       |
| base_users       |
| data             |
| detail           |
| encoding         |
| event            |
| icmphdr          |
| iphdr            |
| opt              |
| reference        |
| reference_system |
| schema           |
| sensor           |
| sig_class        |
| sig_reference    |
| signature        |
| tcphdr           |
| udphdr           |
+------------------+
22 rows in set (0.00 sec)

mysql>
```

Figura 28: Tabelas criadas no banco MySQL.

3.1.1.6 Configuração do servidor web

São necessárias poucas configurações no servidor de páginas WEB, conforme abaixo:

Habilitar o servidor WEB seguro (HTTPS):

cp /etc/apache2/sites-available/default-ssl /etc/apache2/sites-enabled

a2enmod ssl

Alterar configuração do PHP

vi /etc/php5/apache2/php.ini

Linha #521 – Alterar linha - error_reporting = E_ALL & ~E_NOTICE

pear config-set preferred_state alpha && pear channel-update pear.php.net && pear install --alldeps Image_Color Image_Canvas Image_Graph

Feitas as configurações é necessário reiniciar o servidor WEB

/etc/init.d/apache2 restart

3.1.1.7 Instalação e configuração do base

Com todo o servidor IDS instalado, foi feita a instalação da interface WEB para o Snort. O BASE é útil para acompanhar o funcionamento do sistema, e monitorar os alertas. O procedimento de instalação é descrito abaixo:

Download

cd /usr/src && wget http://sourceforge.net/projects/secureideas/files/BASE/base-1.4.5/base-1.4.5.tar.gz

Descompactação

tar -zxf base-1.4.5.tar.gz && cp -r base-1.4.5 /var/www/base

Definidas as permissões para o arquivo

 # chmod 777 /var/www/base (apenas para o processo de instalação)

Foi aberto o browser no endereço: https://localhost/base e escolhida a opção “Continue”, conforme Figura 29:

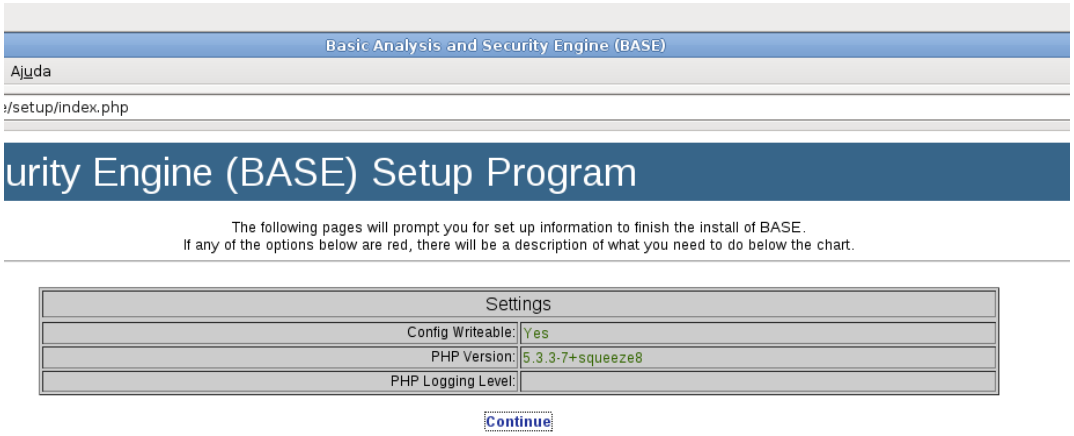

Figura 29: Instalação do Base (1)

No próximo passo, foram selecionadas as seguintes opções: “English”, na opção Diretório do adodb configurado: ‘/usr/share/php/adodb’ e Selecianado “Continue”; Para que seja possível a conexão do BASE com o banco de dados foram utilizados os seguintes parâmetros, conforme a Figura 30:

Database Name: snort

Database Host: localhost

Database Port: (deixar em branco)

Database User Name: snort

Database Password: mypassword

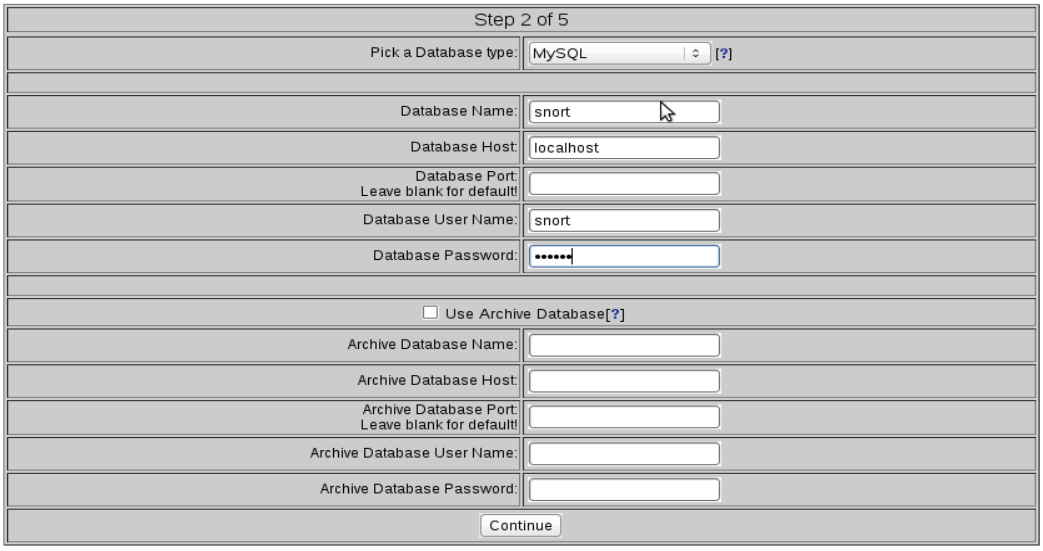

Figura 30: Instalação do Base (2).

Após a definição dos parâmetros, foi selecionada a opção de submeter os dados. Na página seguinte foi selecionada a opção "create baseag" que atualiza o banco de dados do snort para suportar o BASE. Feito isso o BASE está instalado e foi selecionada a opção “Step 5” para logar no sistema e verificar o funcionamento, conforme Figura 31.

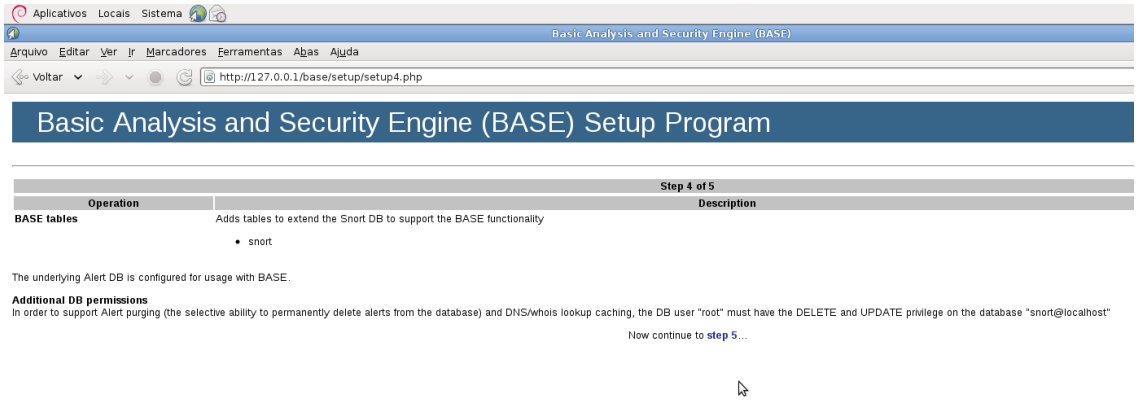

Figura 31: Instalação do Base (3)

3.1.1.8 Inicialização do sistema

Para facilitar a inicialização do IDS, foi criado um script apresentado no Apêndice C com os comandos necessários para iniciar o snort e o baynard, e o mesmo

foi configurado para ser executado no momento da inicialização do servidor segundo os comandos a seguir:

vi /etc/init.d/snortbarn

Adicionado conteúdo do Apêndice C, e após tornado o arquivo executável:

chmod +x /etc/init.d/snortbarn

Adicionando o script na inicialização do servidor:

insserv -f -v snortbarn

3.2 REDIRECIONAMENTO DOS DADOS

O direcionamento dos dados a serem monitorados foi estabelecido da seguinte forma: foi configurado em um switch 3com 4500 a função de espelhamento de porta (*port mirroring).* Este switch interliga o roteador do provedor de acesso à Internet ao firewall da Instituição, portanto toda informação a ser monitorada passa pelo equipamento em questão, conforme apresentado na Figura 32. No mesmo switch foi configurada outra Vlan, para conexão dos hosts internos da rede. Essas portas também foram espelhadas, e assim foi possível avaliar tanto o tráfego interno da rede quanto o externo (com destino à Internet).

Figura 32: Estrutura do espelhamento de tráfego.
Fonte Comutadores.com, 2012

Para ativar a função de *port-mirroring* no switch, foram executados os seguintes comandos abaixo, neste exemplo apenas a porta ethernet 1 foi monitorada:

#

mirroring-group 1 local

! Criando o Grupo 1 de portas para o Espelhamento

#

interface Ethernet1/0/3

stp disable

! desabilitando o Spanning-Tree da porta para não interferir na coleta

mirroring-group 1 monitor-port

!Configurando a porta para monitorar o tráfego da porta mirroring (no exemplo a porta Ethernet1/0/1)

#

interface Ethernet1/0/1

mirroring-group 1 mirroring-port both

! Configurando a Porta de origem que terá seu tráfego copiado no sentido inbound (entrada) e outbound (saída); comando both

#

3.3 SOFTWARE DE CONSULTA WHOIS

O IDS instalado no servidor de análise de *logs* inspeciona todo o tráfego e baseado nas assinaturas das ameaças conhecidas registra no banco de dados toda comunicação suspeita. Para enriquecer a informação disponível, foi desenvolvido um software na linguagem PHP, disponível no Apêndice A, o qual de posse do endereço IP de origem e destino das comunicações armazenadas no banco de dados, efetua a consulta uma consulta *whois*, e armazena no banco de dados o país e o domínio a qual esses IPs pertencem.

3.4 DESCOBERTA DE CONHECIMENTO

Com os todos os dados armazenados em banco de dados, foi possível a extração dos mesmos para a posterior análise no software Weka. A seguir são descritas as etapas da técnica de descoberta de conhecimento em banco de dados, desde a definição do domínio até a extração do conhecimento.

A mineração de dados de fato é uma das etapas do KDD, sendo necessária a definição dos dados a serem tratados, a seleção e a preparação dos mesmos previamente.

3.4.1 Etapa de definição e compreensão do domínio

Primeiramente, foram definidos os dados que seriam relevantes. Analisando a estrutura do banco de dados gerado pelo IDS Snort e pela interface Web BASE, além das informações disponibilizadas pelo sistema de consulta *whois*, foi considerado que seriam necessários os dados do horário do alerta, qual o alerta e sua categoria, protocolo e portas envolvidas na comunicação, além das portas de origem e destino e os domínios e países envolvidos na comunicação.

3.4.2 Etapa de seleção dos dados

Com a posse desse histórico do tráfego, foi feita a mineração de dados no software WEKA dos seguintes atributos:

- Data
- Hora
- Assinatura da Ameaça

- Categoria da ameaça
- IP Origem
- IP Destino
- Protocolo
- Porta Origem
- Porta Destino
- Domínio IP origem
- País IP origem
- Domínio IP destino
- País IP destino

Com os atributos definidos, foi executado o select para extração dos dados, conforme código abaixo. Os dados selecionados foram considerados os necessários para a execução da análise no Weka.

```
select SUBSTR(timestamp,1,4) as ano, SUBSTR(timestamp,6,2) as mes, SUBSTR(timestamp,9,2) as dia, SUBSTR(timestamp,12,2) as hora, SUBSTR(timestamp,15,2) as minuto, sig_name, sig_class_name, INET_NTOA(ip_src), INET_NTOA(ip_dst), ip_proto, layer4_sport, layer4_dport, country_src, owner_src, country_dst, owner_dst
FROM acid_event e, sig_class sc
WHERE e.sig_class_id=sc.sig_class_id and country_dst is not null and e.sig_class_id=<CODIGO>
ORDER by TCP, UDP, ICMP limit 100000;
```

3.4.3 Etapa de limpeza, preparação e seleção de atributos

Com os dados selecionados via *select* e exportados via arquivo separado por vírgulas (.csv), conforme Figura 33. Devido ao grande volume de dados (41GB), foi limitado ao máximo de 100.000 alertas por assinatura, pois o hardware disponível para executar o processo do KDD não possuía memória RAM suficiente para efetuar o processo. Assim, após vários testes foi definido o número máximo de 100.00 alertas por

categoria, onde foi possível ter efetuar tal processo sem afetar a analise, já que apenas 2 categorias de alertas extrapolaram este limite. Ainda tal limitação permitiu diminuir o tempo de processamento dos classificadores do Weka. Assim, os dados foram importados no Weka, gerando um arquivo arff, tornando os dados prontos para a etapa de Mineração de Dados e Análise dos Resultados.

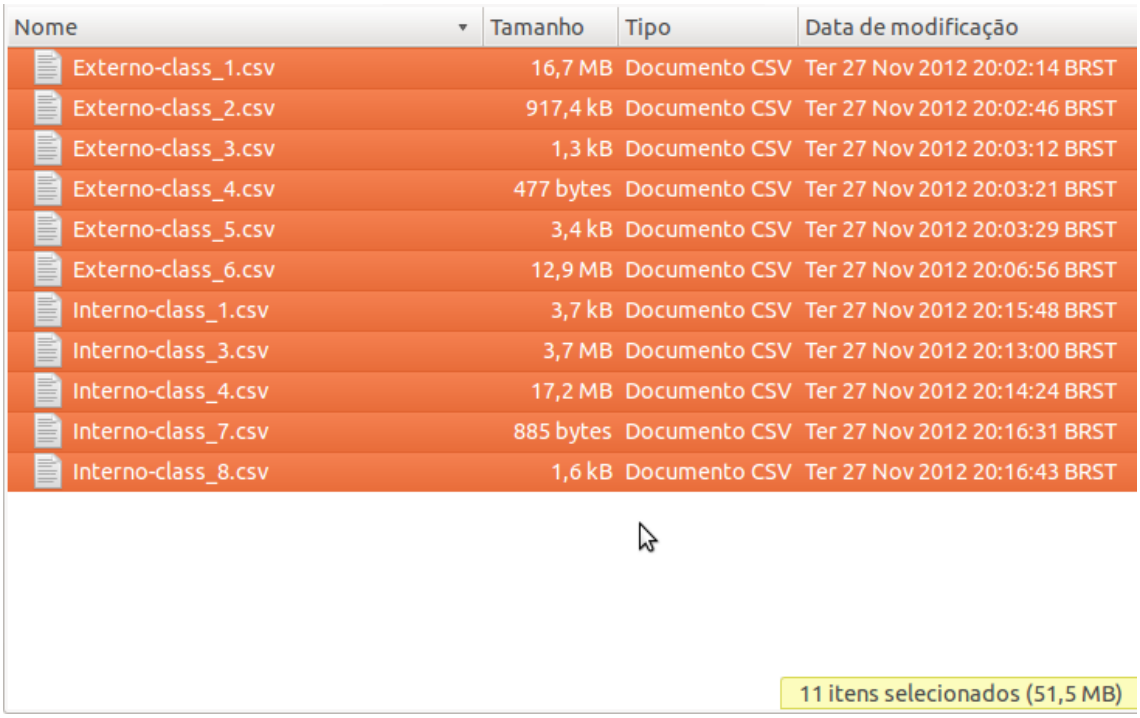

Figura 33: Arquivos .csv exportados do banco de dados.

3.4.4 Etapas de mineração dos dados e extração do conhecimento

Na etapa de mineração de dados, utilizando o software WEKA, foram utilizadas a tarefas de classificação e associação, sendo selecionados os algoritmos *Apriori, ZeroR, J48 e DecisionTable*. Os resultados gerados por tais algoritmos, e a posterior análise destes resultados, são descritos no próximo capítulo.

4 RESULTADOS

Após todo o período de coleta de dados, iniciado em Dezembro de 2011 e finalizado apenas em Novembro de 2012, passando durante este tempo por ajustes nos filtros de coleta, foram armazenados no banco de dados de alertas o total de 41GB (*Gigabytes).* Devido ao alto volume dos dados, foi necessária que a extração fosse particionada em 5, sendo que em cada arquivo foram extraídos agrupados os alertas de por categoria, conforme a Figura 34.

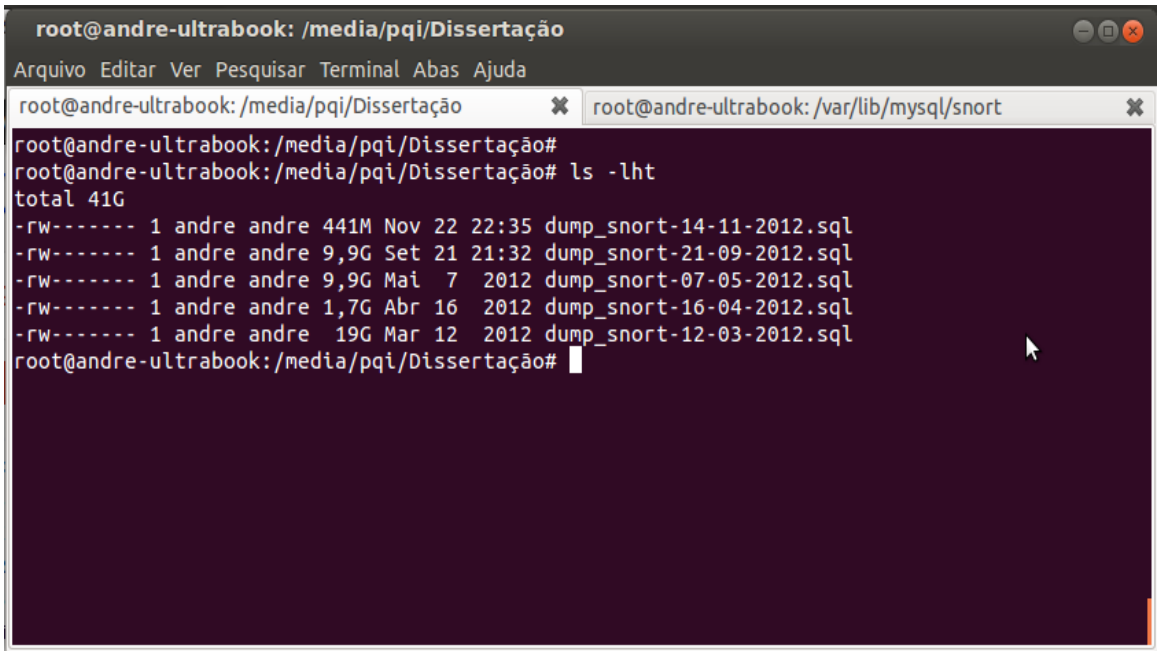

Figura 34: Tamanho total dos dados armazenados no banco de dados.

Para um melhor entendimento, os resultados foram divididos em dois tipos de alertas:

Externos: alertas coletados na interface de ligação da rede ao roteador de saída para o roteador, após a passagem pelo firewall do instituto;

Internos: alertas coletados nas outras portas do switch de coleta, em comunicação intra-rede, ou seja, dentro da rede do IFF ou com destino à Internet antes da passagem do pacote pelo firewall da instituição.

4.1 ALERTAS DE TRÁFEGOS EXTERNOS

Os alertas gerados sobre as comunicações externas, estão todos contidos em 5 categorias, sendo elas em ordem decrescente de alertas gerados:

- *Attempted-dos* ou tentativa de negação de serviço, com 261.009 alertas, onde um *host* gera uma grande quantidade de conexões para tentar deixar o *host* atacado indisponível para outros clientes.
- *Misc-activity* ou atividades diversas, dentre elas tráfego ICMP bloqueado ou com destino inalcançável, com 4.902 alertas.
- *Bad-unknown*, onde são categorizados os alertas de pacotes mal formatados, como por exemplo com a mesma origem e destino, com 24 alertas.
- Sucessfull-admin ou administração com sucesso, onde são armazenadas as comunicações que obtiveram sucesso em obter algum arquivo que tenha informações sensíveis, como um arquivo de senhas, com 8 alertas.
- *Attempted-admin* ou tentativa de administração, onde diferentemente da categoria anterior, o tráfego não obteve o arquivo requisitado, com 2 alertas.

Os dados da quantidade de alertas por categoria foram obtidos via *query* SQL feito no banco de dados, conforme a figura 35.

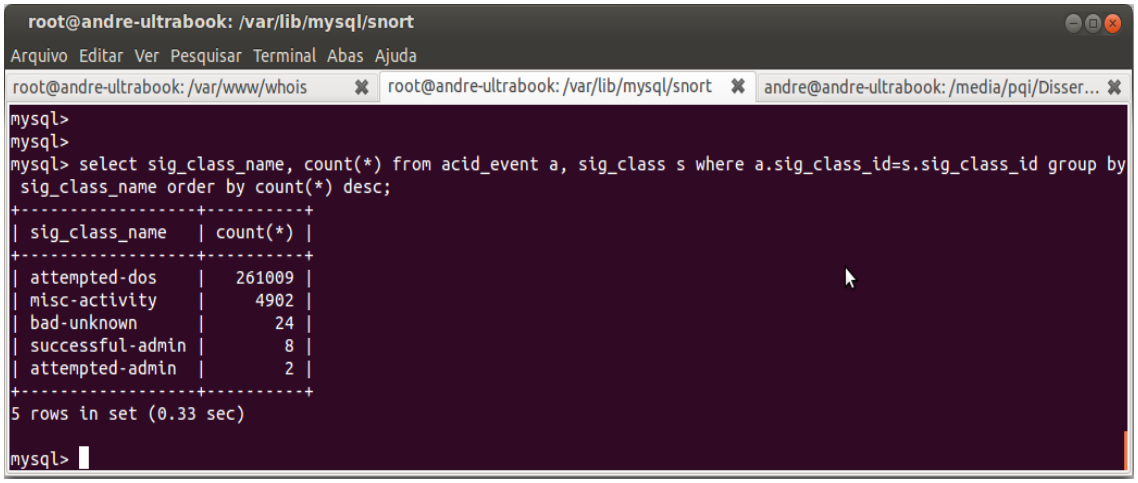

Figura 35: Categoria dos ataques externos.

Os dados sobre as assinaturas que estão englobadas nas 5 categorias apresentadas na Figura 35 foram obtidos via *query* SQL conforme apresentado na Figura 36.

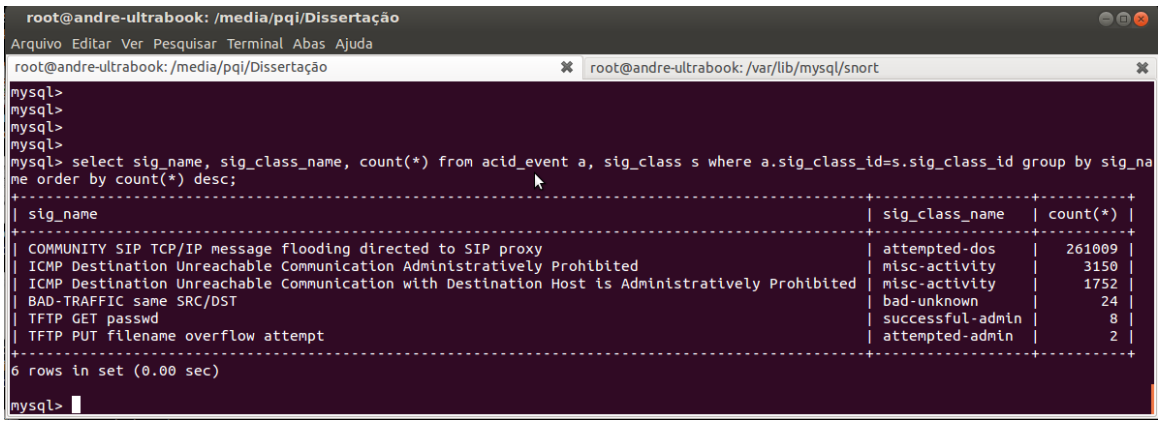

Figura 36: Assinaturas dos ataques externos.

Ao todo foram gerados 6 tipos de alertas diferentes, tendo como destaque as assinaturas “COMMUNITY SIP TCP/IP message flooding directed to SIP proxy” com 261.009 alertas, onde são identificadas tentativas de ataque negação de serviço, conhecidos como DoS. As assinaturas “ICMP Destination Unreachable Communication Administratively Prohibited”, onde são identificados os tráfegos icmp bloqueados pela origem e “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” são considerados irrelevantes. Os alertas “TFTP GET passwd” e “TFTP PUT filename overflow attempt” apesar de com poucos alertas, respectivamente 8 e 2 indicam a tentativa de acessar ou substituir informações sensíveis como por exemplo arquivos de senha.

Após a extração dos dados via *query* SQL, os mesmos foram exportados no formato de arquivo de texto separado por vírgula (.csv), possibilitando a importação no Weka. No Weka tais dados foram transformados em arquivo arff, possibilitando analisar os alertas armazenados na etapa chamada pré-processamento. Os mesmos foram analisados de acordo com o horário do tráfego, onde é possível identificar que grande parte desses alertas foram gerados entre 06:00h e 07:00h, horário de Brasília (GMT-03), conforme mostra a Figura 37:

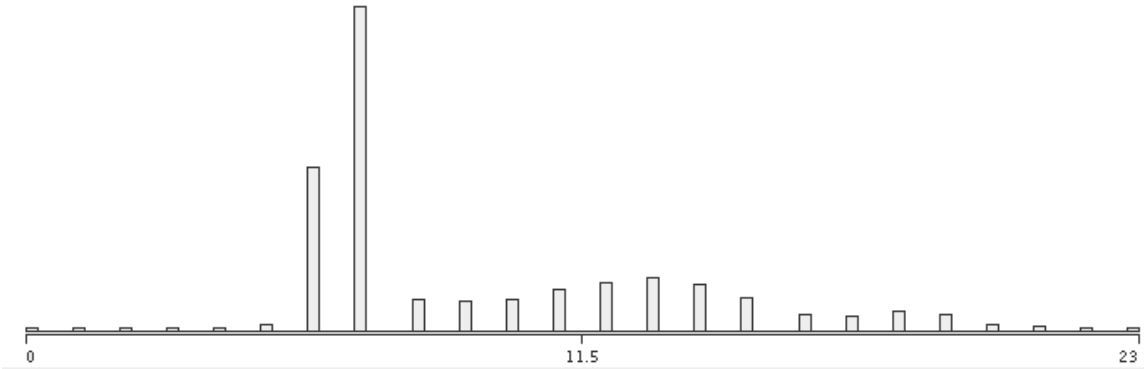

Figura 37: Assinaturas por hora.

Também foi analisado os alertas quanto ao protocolo da comunicação. O protocolo sob o código 255, reservado pela IANA (*Internet Assigned Numbers Authority*, em português, Autoridade para Atribuição de Números da Internet) foi o protocolo com mais alertas, devido as grande volume de *portscan`s* identificados. Em seguida o protocolo TCP, código 6 com 93.956 alertas foi o segundo protocolo com mais alertas. Em menor volume foram identificados alertas dos protocolos UDP, ICMP e ESP com 5.158, 4.981 e 843 alertas respectivamente, conforme apresentado na Figura 38:

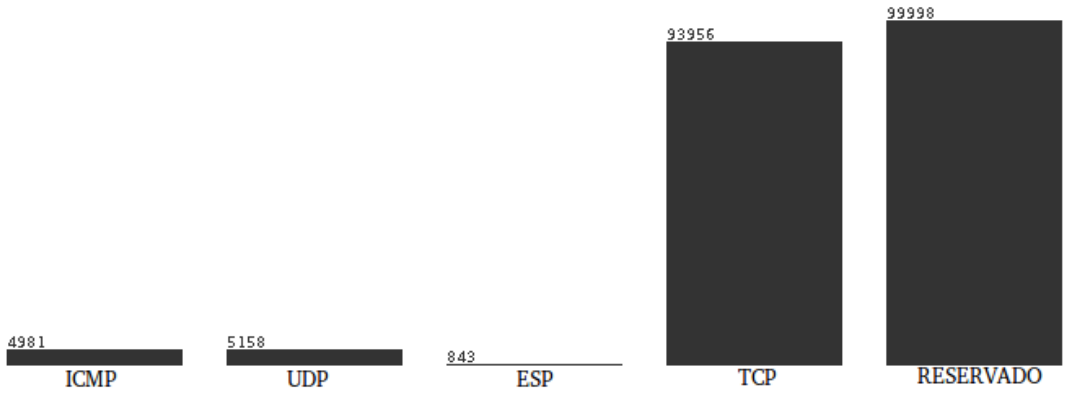

Figura 38: Assinaturas por protocolo.

Analisando a porta de destino dos alertas, é possível identificar que foram atacadas principalmente: a porta 80 TCP com 36.444 alertas, a porta 514 UDP com 6.780 alertas, a porta 443 TCP com 4.980 alertas, a porta 53 com 1.072 alertas e a porta 2257 com 850 alertas, conforme apresentado na Figura 39. Observando as portas, é possível identificar que os serviços HTTP e HTTPS (Web), DNS, POP3 (E-mail) estão entre os mais atacados.

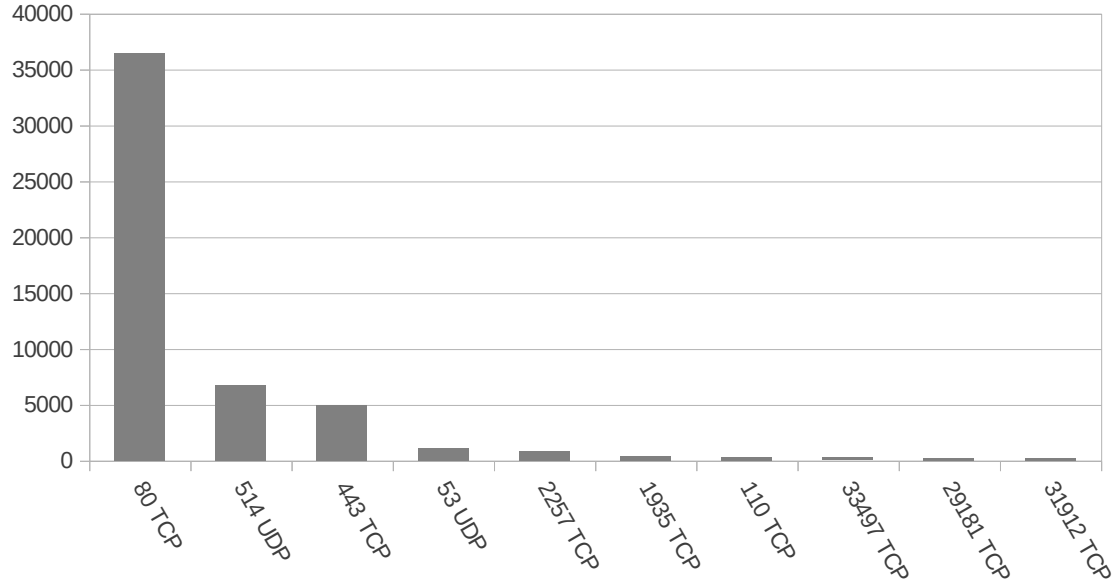

Figura 39: Assinaturas por porta.

A partir de uma lista de 16 atributos, incluídos na etapa de identificação dos atributos foram classificados 13 como essenciais e 3 atributos como irrelevantes, como ano, mês e minuto. Com isso, permitiu-se aumentar a taxa de acerto dos classificadores e diminuir o número de regras geradas, assim como apresentar regras de mais fácil compreensão devido ao valor do atributo na descrição ser mais objetivo.

Na etapa seguinte do processo, que consiste na mineração de dados propriamente dita, selecionou-se os algoritmos classificadores dentre os oferecidos pelo WEKA: o *Apriori*, um classificador que gera regras de associação, *DecisionTable* que regras de classificação e o J48, baseado em árvore de decisão.

As regras formuladas pelo algoritmo *apriori* foram expressas no seguinte formato:

Antecedente: SE <condição>
$\longrightarrow$
Conseqüente: ENTÃO <conclusão>

Na Tabela 2, são apresentadas as regras geradas, que ao todo foram 10. Todas as regras tiveram confiança igual a 1, ou seja, 100%.

Tabela 2: Regras geradas pelo algoritmo apriori.

Se	Então
Owner_dst=iff.edu.br	country_dst=BR
country_src=US	country_dst=BR
country_src=US e owner_dst=iff.edu.br	country_dst=BR
sig_class_name=attempted-dos	sig_name=COMMUNITY SIP TCP/IP message flooding directed to SIP proxy
sig_name=COMMUNITY SIP TCP/IP message flooding directed to SIP proxy	sig_class_name=attempted-dos
protocol=Reserved	sig_class_name=port_scan
sig_name=(portscan) Open Port	sig_class_name=port_scan
sig_name=(portscan) Open Port	protocol=Reserved
sig_name=(portscan) Open Port e protocol=Reserved	sig_class_name=port_scan
sig_name=(portscan) Open Port e sig_class_name=port_scan	protocol=Reserved

É possível observar que nos alertas com destino no domínio do Instituto Federal Fluminense, o país de destino é o próprio Brasil. Quando o tráfego é originado nos Estados Unidos (US) o seu destino é o Brasil (BR). Em seguida tem-se uma regra que envolve as anteriores, caso país de origem seja os Estados Unidos e o domínio de destino seja o iff.edu.br, o país de destino será o Brasil. Após tem-se a regra que diz que se o alerta seja da assinatura "*COMMUNITY SIP TCP/IP message flooding directed to SIP proxy*", este é um alerta de tentativa de D.O.S. (*denial of service*, ou negação de serviço). As regras seguintes relacionam o protocolo reservado, que caso seja utilizado o alerta é da categoria de *port scan* ou scanneamento de portas.

Com base nos resultados do algoritmo *DecisionTable*, foi feita uma analise dos países de origem nas comunicações em que foram gerados alertas armazenados no banco de dados. Essa classificação obteve 86,857% das instâncias corretamente classificadas, que comprova a eficiência do classificador para a tarefa. Na Tabela 3 constam os 10 primeiros resultados gerados pelo algoritmo, tendo o destaque os países

Brasil (BR), Estados Unidos (US) e Japão (JP) como os maiores gerados das comunicações com alerta de segurança.

Tabela 3: Regras geradas pelo algoritmo DecisionTable – País de origem.

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0,869	0,089	0,874	0,869	0,871	0,889	BR
	0,913	0,149	0,865	0,913	0,888	0,909	US
	0,763	0	0,991	0,763	0,862	0,925	JP
	0,784	0,001	0,56	0,784	0,653	0,999	VG
	0,834	0	0,983	0,834	0,903	0,968	DE
	0,253	0,002	0,544	0,253	0,345	0,945	A1
	0,656	0	0,994	0,656	0,79	0,914	GB
	0,73	0	0,996	0,73	0,843	0,976	NL
	0,348	0,003	0,645	0,348	0,452	0,958	CA
	0,244	0	0,938	0,244	0,387	0,704	IL
Weighted Avg.	0,869	0,113	0,866	0,869	0,864	0,903	

Os países BR e US tiveram respectivamente 86,9% e 91,3% das amostras corretamente classificadas, conforme indica o valor "*Recall*". Ainda é possível observar que o país DE teve a média ponderada entre a precisão e a sensibilidade igual a 90,3% (F-Measure = 0,903), o que identifica esses países como maiores geradores de alertas.

De posse dos resultados do algoritmo J48 é possível visualizar que as assinaturas dos alertas foram relacionadas com suas as categorias, gerando a seguinte árvore de decisão, representada na Figura 40, com 99,9% das instâncias corretamente classificadas:

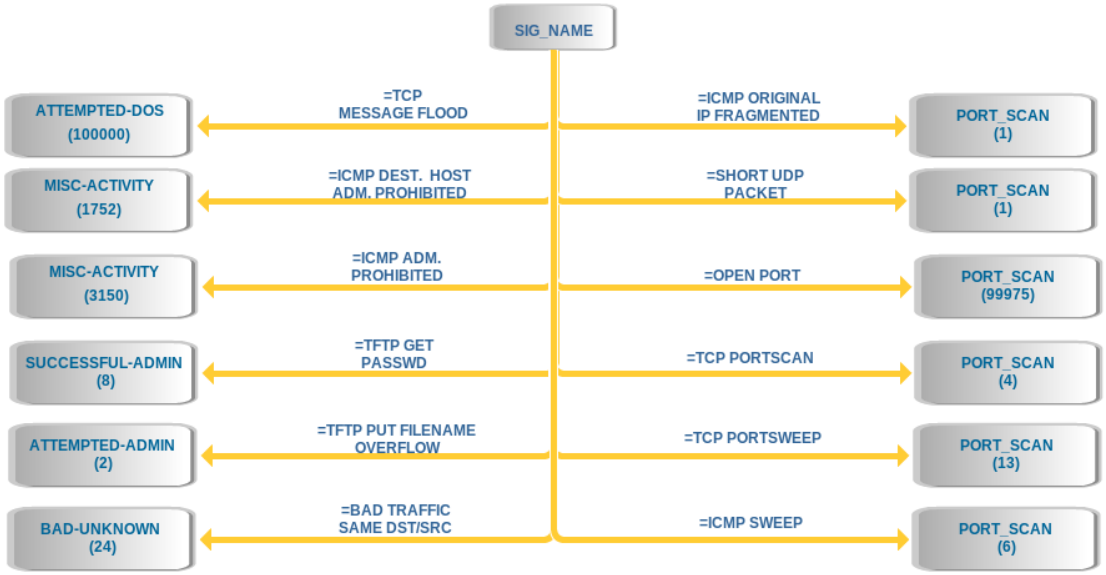

Figura 40: Árvore de decisão gerada pelo algoritmo J48.

Ainda de acordo com a Figura 40, é possível visualizar a quantidade de alertas por categorias, tendo destaque a assinatura "*Attempted-DoS*" com 100.000 alertas, sendo a única assinatura da categoria "*TCP Message Flood*".

Com o intuito de extrair um melhor conhecimento, foi gerada uma nova árvore através do classificador J48, utilizando os atributos hora, sig_name (assinatura do alerta), sig_class_name (categoria do alerta), protocol (protocolo), country_src (país de origem), country_dst (país de destino), owner_dst (domínio de destino) e owner_scr (domínio do origem), gerando com 99.62% das instancias corretamente classificadas a seguinte árvore, que relaciona os alertas com os países de origem e a hora do alerta, onde é possível observar que por exemplo que se o alerta tem origem nos Estados Unidos (US) e o tráfego acontece antes das 5 horas, o tipo do alerta em 82 das 707 vezes é o "*ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited*", conforme a Figura 41 que ilustra parte da árvore de decisão, devido ao seu tamanho, com 97 folhas. A árvore completa pode ser visualizada no apêndice E.

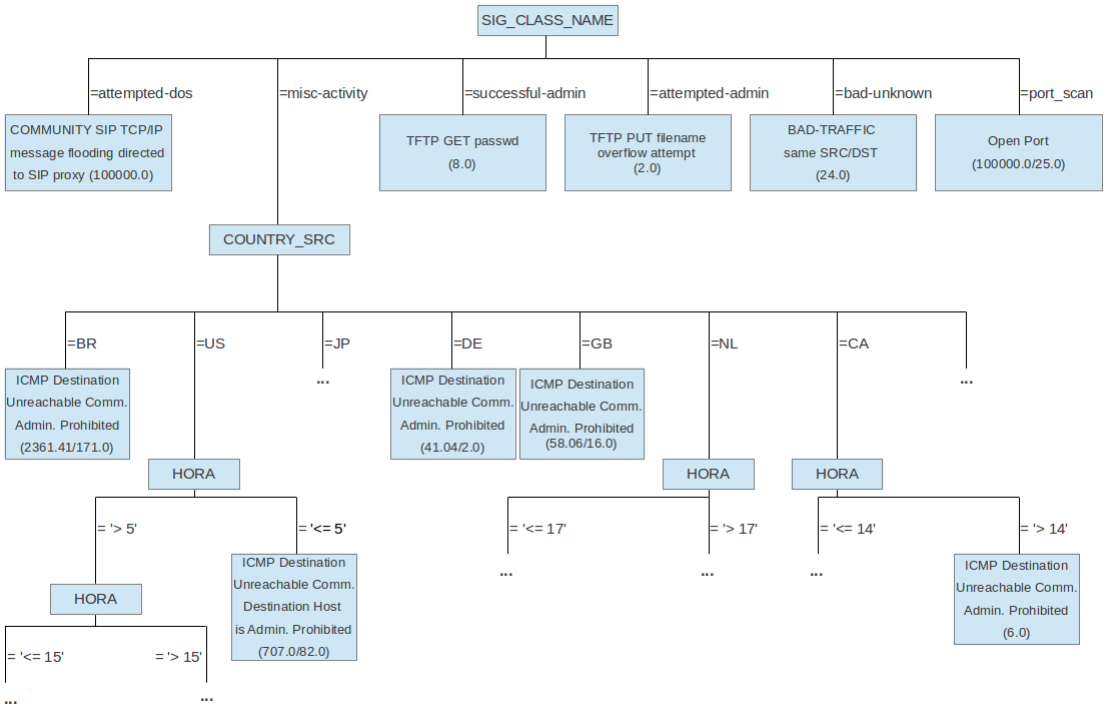

Figura 41: Árvore de Decisão – Dados externos.

De posse dos resultados do tráfego externo, foi possível visualizar que os serviços HTTP e DNS estão entre os mais atacados, devendo uma atenção pela equipe

de gerência de redes. Por serem serviços com grande quantidade de acessos, tais resultados eram esperados. Ainda foi possível fazer a descoberta do horário de maior incidência dos ataques, sendo estes compreendidos em grande parte no período da manhã. Já quanto aos países, Brasil, Estados Unidos e Japão reúnem grande parte dos alertas.

4.2 ALERTAS DE TRÁFEGOS INTERNOS

Os alertas gerados sobre as comunicações internas, também estão todos contidos em 5 categorias, porém diferentemente dos alertas externos, com a categoria de atividade de trojan, típica de computadores de uso pessoal infectados e outras categorias como a policy-violation na qual estão as assinaturas de tráfego de spywares. Em seguida são apresentados as categorias dos alertas armazenados, sendo elas em ordem crescente de alertas:

- *Trojan-activity* ou atividade de trojan, com 762.342 alertas, onde um *host* está infectado por um programa malicioso (trojan) que é disseminado embutido em outro software.
- *Policy-violation* ou violação de privacidade, onde são classificados alertas que violam uma política de acesso, como um tráfego de um spyware por exemplo, com 22.718 alertas.
- *Misc-activity* ou atividades diversas, dentre elas tráfego de algum malware, com 24 alertas.
- *String-detected*, com 8 alertas, onde são categorizados os alertas de atividade *Worn (*verme*)*, como por exemplo em vírus anexados à e-mails.
- *Suspicious-filename-detected* ou nome de arquivo suspeito detectado, com 4 alertas, onde são identificados nomes comuns de arquivos nocivos. Nesta categoria foram encontrados 4 alertas.

Os dados apresentados foram obtidos via *query* SQL feito no banco de dados, conforme a Figura 42.

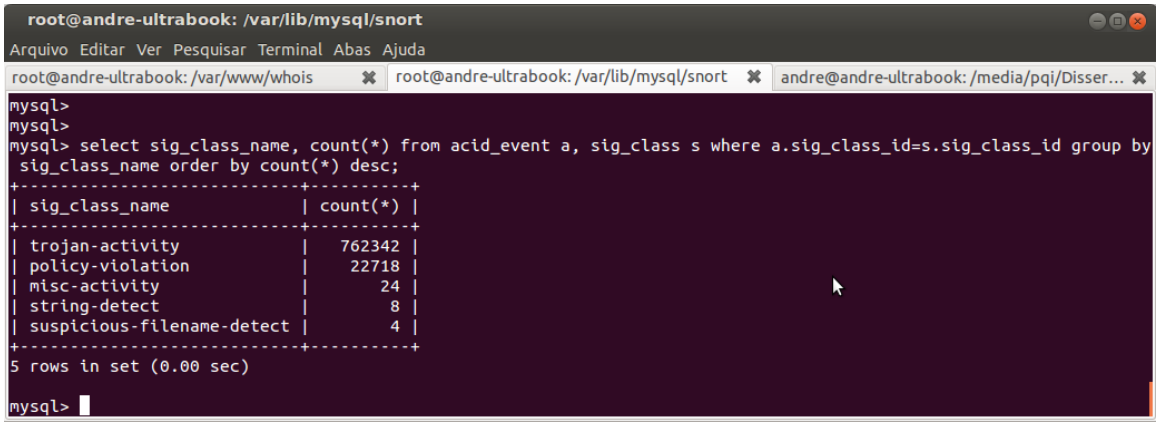

Figura 42: Categoria dos ataques internos.

Os dados sobre as assinaturas que estão englobadas nas 5 categorias acima foram obtidos via *query* SQL conforme a Figura 43.

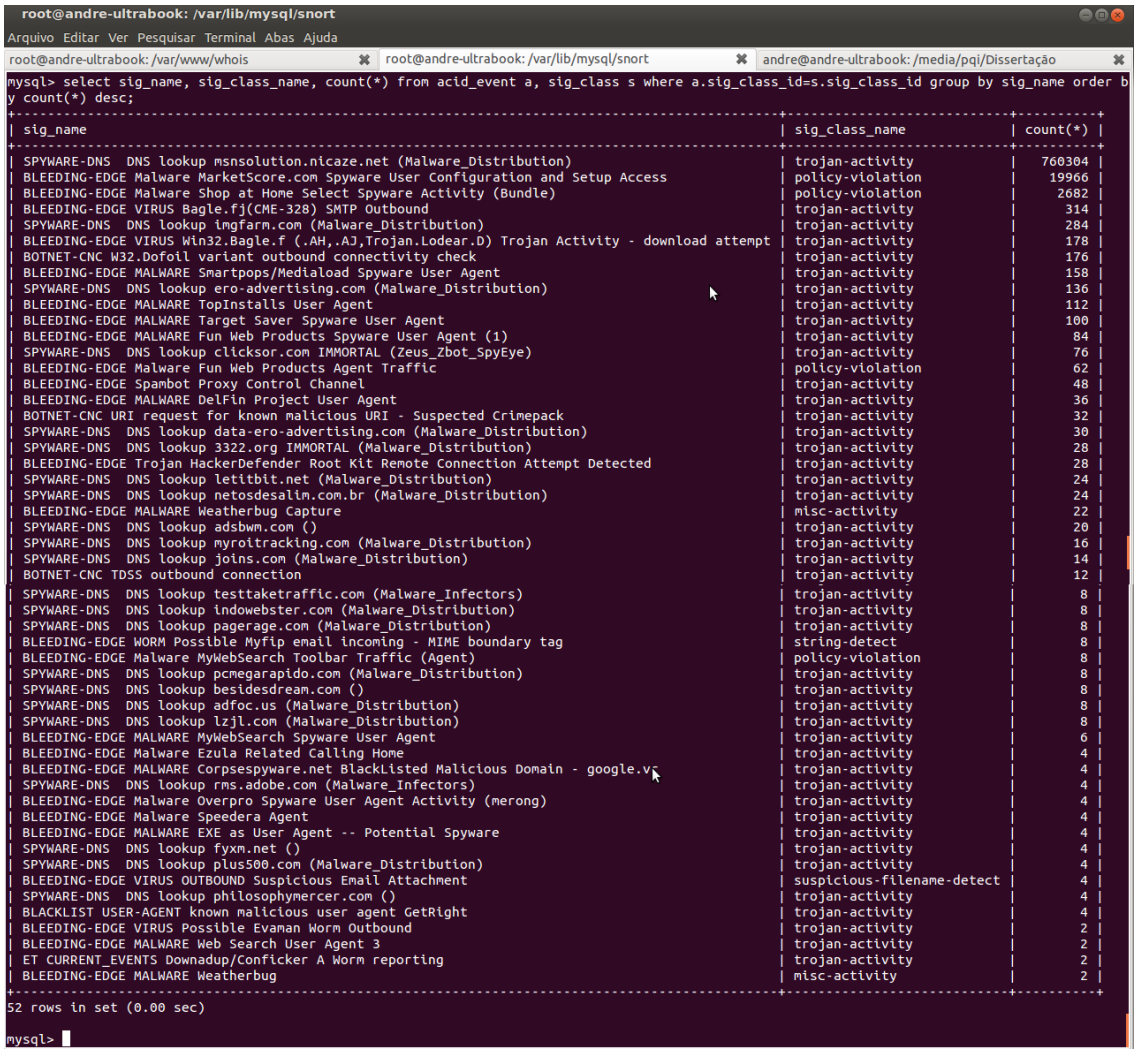

Figura 43: Assinaturas dos ataques externos.

Ao todo foram gerados 52 tipos de alertas diferentes, tendo como destaque as assinaturas "*SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_distribution)*" com 760.304 alertas, onde o host indicado na assinatura do alerta faz parte de uma rede de distribuição de *malwares*, mais especificamente *trojans*. *"BLEDDING-EDGE Malware MarketScore.com Spyware User Configuration and Setup Access*", onde identifica um tráfego gerado por um *spyware* com 19.966 alertas e "*BLEDDING-EDGE Malware Shop at Home Select Spyware Activity (Bundle)*", que também registra tráfego gerado por *spywares*.

Após a extração dos dados via *query* SQL, exportados como arquivo .csv e transformados em arquivo arff, os dados carregados no Weka, possibilitando analisar os alertas armazenados na etapa chamada pré-processamento. Os mesmos foram analisados de acordo com o horário do tráfego e relacionados com a categoria do ataque, onde é possível identificar que os alertas de tentativa de acessar *hosts* que distribuem *trojan* acontecem até as 14h e após esse horário os alertas que predominam são os da categoria de violação de política de acesso, conforme a Figura 44.

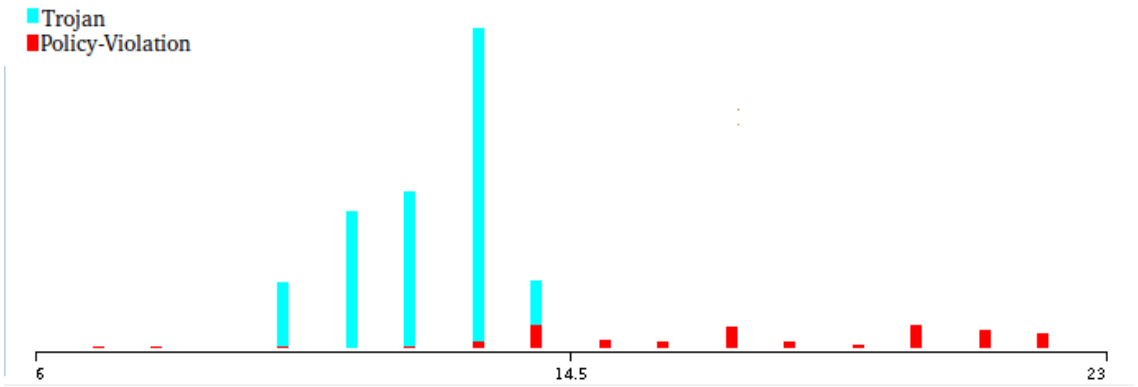

Figura 44: Assinaturas por hora – divididos por categoria do ataque.

Também foram analisados os alertas quanto ao protocolo da comunicação. Apenas dois protocolos geraram alertas no tráfego interno. O protocolo UDP sob o código 17 foi o protocolo com a grande maioria dos alertas (100.000 alertas). Já o protocolo TCP, código 6 acumulou 19.026 alertas, conforme a Figura 45:

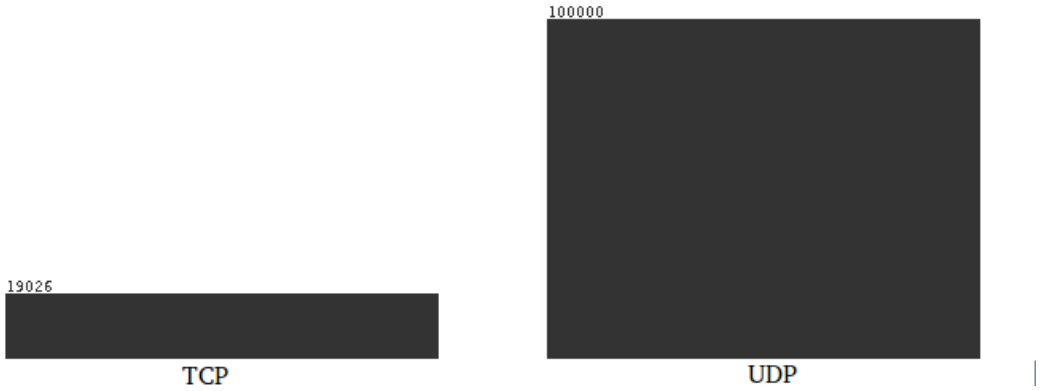

Figura 45: Assinaturas por protocolo.

Analisando a porta de destino dos alertas, é possível identificar que foram atacadas principalmente: a porta 53 UDP com 772.313 alertas, a porta 80 TCP com 27.386 alertas, a porta 137 UDP com 6.286 alertas e a porta 25 TCP com 328 alertas, conforme apresentado na Figura 46. Observando as portas, é possível identificar que os serviços HTTP (Web), DNS, SMTP (E-mail) estão novamente entre os mais atacados.

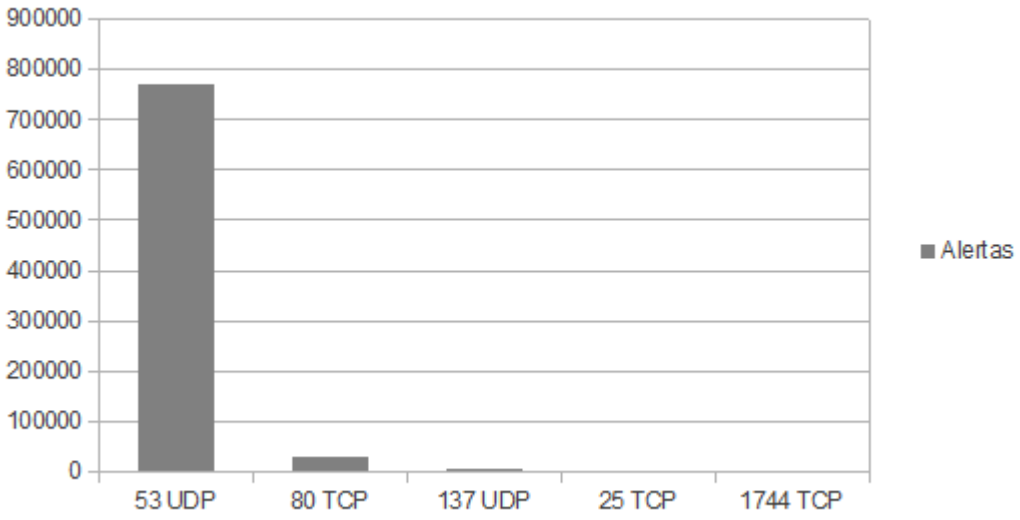

Figura 46: Alertas por porta.

A partir de uma lista de 16 atributos, incluídos na etapa de identificação dos atributos foram classificados 13 como essenciais e 3 atributos como irrelevantes, assim como nos dados externos. Os atributos excluídos foram ano, mês e minuto.

Na etapa de mineração de dados propriamente dita, foram selecionados novamente os classificadores *Apriori*, um classificador que gera regras de associação, *DecisionTable* e ZeroR que regras de classificação e o J48, baseado em árvore de decisão. Os resultados obtidos são mostrados respectivamente a seguir:

As regras formuladas pelo algoritmo apriori foram expressas no seguinte formato se → então, e na Tabela 4, são apresentadas as regras geradas, que ao todo foram 10. Todas as regras tiveram confiança igual a 1, ou seja, 100%.

Tabela 4: Resultado ao algoritmo apriori.

Se	Então
protocol=UDP	sig_class_name=trojan-activity
sig_class_name=trojan-activity	protocol=UDP
sig_name=SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_Distribution)	sig_class_name=trojan-activity
sig_name=SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_Distribution)	protocol=UDP
sig_name=SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_Distribution) e protocol=UDP	sig_class_name=trojan-activity
sig_name=SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_Distribution) e sig_class_name=trojan-activity	protocol=UDP
sig_name=SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_Distribution)	sig_class_name=trojan-activity e protocol=UDP
owner_dst=iff.edu.br	country_dst=BR
protocol=UDP e country_dst=BR	sig_class_name=trojan-activity
sig_class_name=trojan-activity e country_dst=BR	protocol=UDP

Ao se observar as regras, é possível relacionar que se o protocolo é o UDP, então a categoria dos alertas é o trojan-activity. Bem como, se a assinatura do alerta é a consulta dns a uma rede de distribuição de malware, a "DNS lookup msnsolution.nicaze.net", então a categoria é trojan-activity e o protocolo é o UDP. Se o domínio de origem é o "iff.edu.br" o país de destino é o próprio Brasil. Se O protocolo em questão é o UDP e o país de destino é o Brasil, a categoria do ataque é o "trojan-activity". Assim como se a categoria do alerta é "trojan-activity e o país de destino é o Brasil, significa que o protocolo utilizado na comunicação foi o UDP.

Utilizando o atributo ip_src como valor de entrada para o classificador ZeroR, foi possível identificar que o *host* "10.12.9.135" precisa passar por uma analise

detalhada, pois foi identificado como o gerador de 22,19% dos alertas. Já o classificador DecisionTable, com 99,28% das instâncias corretamente classificadas, gerou a Tabela 5 com os Ips de origem com maior número de alertas.

Tabela 5: Resultado ao algoritmo DecisionTable – ip de origem.

	TP_Rate	FP_Rate	Precision	Recall	F-Measure	ROC_Area	Class
	0,419	0	0,633	0,419	0,504	0,85	200.143.198.66
	0,374	0	1	0,374	0,544	0,90	200.143.198.49
	1	0,002	0,987	1	0,993	1,00	200.143.198.34
	0,818	0	0,955	0,818	0,881	0,98	200.143.198.110
	1	0	1	1	1	1,00	200.143.198.46
	0,995	0,001	0,996	0,995	0,995	1,00	10.12.9.135
	0,994	0,001	0,984	0,994	0,989	1,00	10.12.8.4
	0,997	0,002	0,993	0,997	0,995	1,00	10.12.8.1
	0,988	0	0,997	0,988	0,992	1,00	10.12.8.5
Weighted_Avg	0,993	0,001	0,993	0,993	0,992	1,00	

Ao executar o algoritmo J48, para que fosse possível gerar uma árvore de decisão com os dados coletados dentro da rede do Instituto, foi gerada uma árvore de tamanho 12.208 e com 9035 folhas, apesar de a mesma classificar corretamente 97,06% dos dados. Devido a este fato, foi reexecutado o classificador, removendo os atributos ip_src e ip_dst (Ips de origem e destino) e porta de origem (layer4_sport). De posse dos resultados dessa nova execução do algoritmo J48 é possível visualizar mais claramente que alguns domínios, como o "r.cloudfront.net", "ftp-osl.osuosl.org" e "securestudies.com" estão envolvidos na grande maioria das comunicações consideradas como alerta de segurança pelo IDS. Ainda é possível identificar quais os países de destino envolvidos nas comunicações e as portas de destino, bem como no caso de alguns alertas como o da assinatura "BLEEDING-EDGE VIRUS OUTBOUND Suspicious Email Attachment", gerada por anexos de e-mail com conteúdo suspeito que dependendo do horário o domínio envolvido é diferente. A árvore gerada, que teve 99,61% das instâncias corretamente classificadas, está abaixo representada na apêndice F devido ao seu tamanho (62) e quantidade de folhas (51). Abaixo a Figura 47 representa parte da árvore em questão, destacando seus pontos mais relevantes:

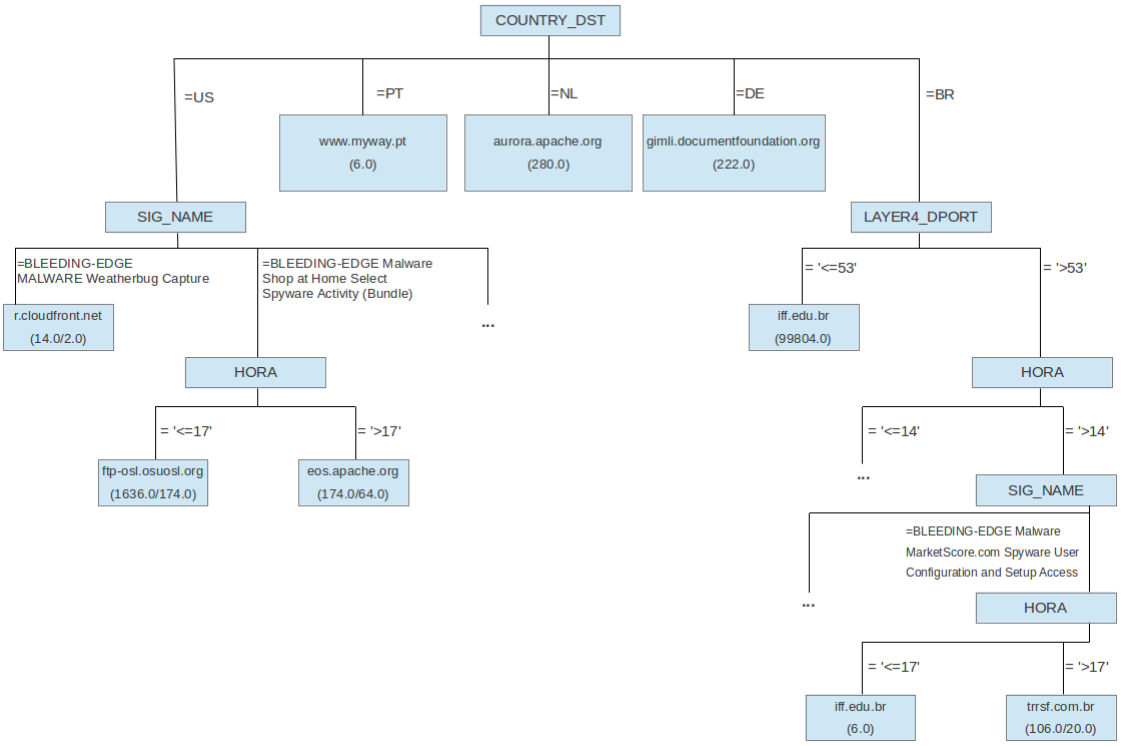

Figura 47: Árvore de Decisão – Dados internos.

Os resultados do tráfego interno indicam que quase a totalidade dos alertas é de máquinas infectadas, seja por trojan ou por spyware. Através dos resultados foi possível bloquear uma grande rede de distribuição de spyware e também visualizar que o tráfego gerado por trojans se concentra no período até as 14h, e após este horário apenas alertas gerados por spyware foram identificados. Outro resultado de grande valia foi gerado pelo algoritmo *ZeroR*, sendo possível identificar um host responsável por mais de 22% dos alertas.

5 CONCLUSÕES

A partir dos resultados pode-se concluir que a utilização das técnicas mineração de dados para a análise e extração de conhecimento a partir dos logs de segurança do tráfego de rede armazenados em banco de dados é de extrema valia para a segurança de rede, já sendo utilizado com sucesso no Instituto Federal Fluminense.

Já na etapa de pré-processamento, foi possível identificar com maior precisão vários dados como por exemplo: IPs, domínios, portas de origem e destino, horário dos ataques, protocolos e assinaturas com mais alertas de segurança, comparado com a interface web de gerência BASE do IDS utilizado SNORT. Esta interface devido ao grande volume de dados se tornou inutilizável, chegando a não exibir os dados em vários momentos.

Com os resultados dos classificadores na etapa de mineração de dados, pode-se relacionar alguns dados como, horário do alerta com país de origem e tipo do alerta. Esse tipo de relatório auxilia o administrador de rede a traçar o perfil das ameaças e proteger a rede bloqueando os tráfegos com tais características no *firewall* da instituição. Outro conhecimento extraído foi a relação dos tipos de alertas e os domínios envolvidos nas comunicações, o que após a analise do responsável da rede pode gerar regras de bloqueio para os tráfegos originados por tais domínios. Outro exemplo foi determinar os hosts com mais alertas, e em caso do host ser uma máquina interna do IFF, a mesma é encaminhada ao setor de manutenção para analise e correção de suas vulnerabilidades.

5.1 CONTRIBUIÇÕES

O trabalho desenvolvido auxiliou a análise do tráfego de rede do Instituto Federal Fluminense, extraindo conhecimento após a mineração dos *logs* do tráfego de rede do Instituto para que fosse possível que a equipe de T.I. do instituto bloqueie tráfegos indesejados quando necessário. Com isso toda a rede do instituto teve seu nível de proteção elevado contra ameaças e ataques, provendo uma melhoria no desempenho geral das comunicações.

A arquitetura proposta já foi integrada ao conjunto de ferramentas de monitoramento de rede do Instituto, sendo freqüentemente utilizada pela equipe de gerência de redes para acompanhar e filtrar tráfegos suspeitos e pelo setor de manutenção para monitorar possíveis máquinas infectadas na rede.

5.2 TRABALHOS FUTUROS

Este trabalho pode ser complementado com a criação de uma aplicação para automatizar todo o processo, desde a coleta dos dados, tratamento dos dados e mineração dos mesmos através da API de desenvolvimento do WEKA. Assim o profissional especialista teria uma plataforma com os conhecimentos já extraídos, apenas restando para o mesmo avaliar tais resultados e aplicar as novas regras de segurança.

Devido a quantidade de alertas que após análise que são considerados falsos positivos, o que é uma característica do softwares de IDS, não é recomendável que a própria plataforma automática aplique as regras, o que pode bloquear tráfegos legítimos.

REFERÊNCIAS BIBLIOGRÁFICAS

ABBOTT PA, LEE SM. **Data mining and knowledge discovery**. In: Saba VK, Mccormick KA. Essentials of nursing informatics. 4th ed. New York: McGraw-Hill Medical Pub. Division, 2006

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). **NBR ISO/IEC 17799: tecnologia da informação: técnicas de segurança - código de prática para a gestão da segurança da informação**. Rio de Janeiro, 2005. 120 p. Disponível em: <http://portal.cjf.jus.br/sigjus/arquivos-diversos/NBR-ISO-IEC-17799-2005.PDF/at_ download/file>. Acesso em: 07/03/2013.

ALIAS-I. **LingPipe**. Disponível em: <http://alias-i.com/lingpipe/>. Acessado em Maio de 2012.

ARAUJO JÚNIOR RH, TARAPANOFF K. Precisão no processo de busca e recuperação da informação: uso da mineração de textos**. Ciência da Informação,** Brasília, 35(3), 236-247, 2006.

BEZERRA, Romildo Martins. **Interconexão de redes no CEFET/BA**. Disponível em: <http://www.ifba.edu.br/professores/romildo/downloads/ifba/ interconexao.pdf> Acesso em 20/07/2012

CAMILO, Cássio Oliveira; SILVA, João Carlos da. **Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas - Relatório Técnico**. Instituto de Informática, Universidade Federal de Goiás. Agosto, 2009.

CARDOSO ONP; MACHADO RTM. Gestão do conhecimento usando data mining: estudo de caso na Universidade Federal de Lavras**. Revista de Administração Pública**, Lavras, SP, 42(3), 495-528, 2008.

CARNE, Bryan. **A professional's guide to data communication in a TCP/IP world**. London: Artech House Inc, 2004.

CASTELLI, Matthew J. **LAN Switching first-step**. Indianápolis: Cisco Press, 2004.

Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasl (CERT.BR). **Incidentes reportados ao CERT.br em 2012**. Disponível em: <http://www.cert.br/stats/incidentes/>. Acesso em 19/07/2012.

COIT, C.J.; STANIFORD, S.; MCALERNEY, J., Towards faster string matching for intrusion detection or exceeding the speed of Snort. **Proceedings of DARPA Information**, 2001.

CAMPO GRANDE (Município). **Diferenças entre Hub, Switch e Roteador**. Disponível em <http://www.campograndern.com.br/?tag=hub>. Acesso em: 20 de Novembro de 2012.

COMUTADORES. **Configuração e Administração de Switches - Switches 3Com 4500 – Configurando o Espelhamento de Porta (Port Mirroring)**. Disponível em: <http://www.comutadores.com.br/switches-3com-4500-configurando-o-espelhamento-de-porta-port-mirroring/> Acesso em: 20 de Novembro de 2012.

CORREIA, L. J. **Mineração de dados em arquivos de log gerados por servidores de páginas web.** 107p. Monografia (Graduação em Ciência da Computação) – Universidade Regional de Blumenal, Blumenal, SP, 2004.

COSER, Ezequial**. Automatização do processo de contenção de ameaças baseada em ferramenta de ids/ips - sistema de detecção e prevenção de intrusão.** 140p. Monografia (Graduação em Engenharia de Controle e Automação), 2011. Centro Universitário Univates, Lajeado, RS, 2011.

COSTA, LDA F. **Bioinformatics: perspectives for the future.** Genet Mol Res. 2004;3(4):564-74. Ribeirao Preto, SP, 2004

DAMATTO, Felipe César, RALL, Ricardo**. Estudo dos possíveis motivos do aumento de incidentes de malwares nas empresas**. Tékhne ε Lógos, Botucatu, SP, v.2, n,2, fev. 2011.

DARTMOUTH. **MDR**. Disponível em: <http://www.multifactordimensional ityreduction.org/>, acessado em Maio de 2012.

DONAHUE, Gary A. **Redes Robustas**. Rio de Janeiro: O`Reilly, 2008.

DOSTÁLEK, Libor; KABELOVÁ, Alena. **Understanding TCP/IP: a clear and comprehensive guide to TCP/IP protocols**. Birmingham: Packt Publishing, 2006.

DUHAM, M. H. **DATA MINING – Introductory and Advanced Topics**. New Jersey: Pearson Education: 2003. 314p.

FAYYAD UM, SHAPIRO GP, SMYTH P, UTHURUSAMY R. **Advances in knowledge discovery and data mining**. Menlo Park, Calif.: AAAI Press: MIT Press; 1996. 611p.

FAYYAD UM, SHAPIRO GP, SMYTH P, UTHURUSAMY R.The KDD process for extracting useful knowledge from volumes data. **Journal of the ACM**, 39(11):27-34, November 1996.

FRANK, E. **Aprendizado de Máquina utilizando WEKA**. New Zealand: University of Waikato, 2005.

FRANK, E. WITTENM I. H**. Data Mining – Pratical Machine Learning Tools and Techiniques.** 2 ed. San Francisco: Morgan Kaufmann Publishers; 2005

FUENTES, L.F. Malware, una amenaza de Internet. **Revista Digital Universitária**, México D. F., v. 9, n. 4, abril, 2012. Disponível em: <http://www.revista.unam.mx /vol.9/ num4/art22/int22.htm>. Acesso em: 13, jan. 2012.

GOLDSCHMIDT R; PASSOS E. **Data mining: um guia prático, conceitos, técnicas, ferramentas, orientações e aplicações**. São Paulo: Elsevier; 2005.

HALL, Eric. **Internet core protocols: The Definitive Guide**. Sebastopol: O'Reilly, 2000.

HELD, Gilbert. **Ethernet networks: Design, Implementation, Operation, Management.** 4. ed. Chichester: John Wiley & Sons, 2003.

HUNT, Craig**. TCP/IP network administration**. 3 ed. Sebastopol: O'Reilly, 2002.

International Business Machines (IBM). **Intelligent Miner**. Disponível em: <http://www-01.ibm.com/software/data/iminer/>, acessado em Maio de 2012.

JONES PBC. The commercialization of bioinformatics. **Electron J Biotechnol**. Valparaíso, 3(2): 33-4, 2000.

JULISCH, K., Mining alarm clusters to improve handling efficiency, **Proceedings of 17th Annual Computer Security Application Conference (ACSAC'01),** New Orleans, LA, 2001, p. 12.

LONG, J., SCHWARTZ, D.; STOECKLIN, S. Distinguishing False from True Alerts in Snort by Data Mining Patterns of Alerts**. Proceedings of 2006 SPIE Defense and Security Symposium**, pp. 62410B-1--62410B-10

LOPES, Isabel Maria. **Adopção de Políticas de Segurança de Sistemas de Informação na Administração Pública Local em Portugal**. 437p. Tese (Mestrado em Engenharia e Gestão de Sistemas de Informação), 2012. Universidade do Minho, Portugal, 2012.

MALUCELLI, A. et al. Classificação de microáreas de risco com uso de mineração de dados. **Revista de Saúde Pública**. Disponível em: <http://www.scielo.br/scielo.php ?pid=S0034-89102010000200009&script=sci_abstract &tlng=pt> vol.44, n.2, pp. 292-300, 2010.

MELLO, Thiago. **Nariz - um sistema de correlacionamento distribuído de alertas.** Dissertação, 2004. Universidade Federal do Paraná, Paraná, 2004.

MANSON, M. **Estudio sobre vírus informáticos**, 1999. Disponível em: <http://www.monografias.com/trabajos/estudiovirus/estudiovirus.shtml?monosearch. Acesso em: 12 set. 2009.

MARTINS S. N., CUNHA A. A., GOMES G. R. R. Aplicação de Técnicas de Mineração de Dados na Previsão de Evasão Escolar em Instituição Pública. ***Anais*** 2011, SIMPEP, Bauru-SP.

MATOS G, CHALMETA R, COLTELL O. Metodología para la extracción del conocimiento empresarial a partir de los datos**. Informatíon Tecnológica**. La Serena, 17(2): 81-8, 2006.

MCQUERRY, Steve. **Interconnecting cisco network devices, Part 1 (ICND1)**. 2 ed, Indianápolis: Cisco Press, 2008

MEIRA CAA, RODRIGUES LHA, MORAES SA. Análise da epidemia da ferrugem do cafeeiro com árvore de decisão. **Trop Plant Pathol**. Brasília, 2008;33(2):114-24.

MORAIS J. A. S. **Descoberta de conhecimento em logs de tentativas de intrusão: Um Estudo de Caso em Instituições de Ensino Superior**. 130p. Dissertação (Mestrado em Engenharia Informática), 2010. Instituto Superior de Engenharia do Porto. Porto, PT, 2010.

OLIVEIRA, Andreliza Mila Rosa De; NOGUEIRA, Rosane Corsini Silva; LEMES, Edinei Gonçalves. Segurança da informação nas empresas: enfocando a engenharia social. **Anais do Seminário de Produção Acadêmica da Anhanguera**, Anhanguera, SP, 2011.

ONLINE HELP. **Online Help** Disponível em: <http://onlinehelp.avs4you.com/AVS-Firewall/ Introduction /ComputerNetwork.aspx>. Acessado em 01/08/2012.

ORACLE**.** **Oracle**. Disponivel em: <http://www.oracle.com/technology /products/bi/odm/ index.html>, acessado em Maio de 2012.

PASCOAL, Vítor Rogério Gomes Pardal De Oliveira**. Definição de mecanismos para comunicação entre processadores e FPGAs**. 107p. Dissertação (Mestrado em Engenharia Electrónica e Telecomunicações), 2008. Universidade de Aveiro, Pará, 2008.

PEREIRA BB. Estatística em psiquiatria**. Revista Brasileira de Psiquiatria**. São Paulo, 23(3): 168-70, 2001.

PEREIRA GC, COUTINHO R, EBECKEN NFF. **Data mining for environmental analysis and diagnostic: a case study of upwelling ecosystem of Arraial do Cabo**. Braz J Oceanogr. São Paulo, 2008;56(1):1-12.

PIMIENTO. **Pimiento**. Disponível em: <http://erabaki.ehu.es/jjga/pimiento/>, acessado em Maio de 2012.

QUONIAM, L; TARAPANOFF, K; ARAÚJO JÚNIOR, RH; ALVARES, L. Inteligência obtida pela aplicação de data mining em base de teses francesas sobre o Brasil**. Ciência da Informação**, Brasília, 30(2): 20-8, 2001.

RIST, L.. **Glastopf project. The Honeynet Project (2009).** Diponivel em <http://www.honeynet.org/project> Acessado em 15/07/2012

SAS. **Enterprise Miner Suite**. Disponível em: <http://www.sas.com/technologies/ analytics/ datamining/miner/index.html>, acessado em Maio de 2012.

SAS. **SAS Text Miner**. Disponível em: <http://www.sas.com/technologies/analytics/ datamining/textminer/index.html>, acessado em Maio de 2012.

SCARPEL RA, MILIONI AZ. Otimização na formação de agrupamentos em problemas de composição de especialistas. **Pesquisa Operacional**. Rio de Janeiro, 27(1): 85-104, 2007.

SOARES, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sérgio. **Redes de Computadores: das LANs, MANs e WANs as Redes ATM**. 6 ed. Rio de Janeiro: Campus, 1995

SPORTACK, MARK A. **TCP/IP first-step**. Indianápolis: Cisco Press, 2004

STEINER, MTA; SOMA, NY; SHIMIZU, T; NIEVOLA, JC; STEINER NETO, PJ. Abordagem de um problema médico por meio do processo de KDD com ênfase à análise exploratória dos dados**. Gest Prod.** São Carlos, 2006;13(2):325-37.

TANENBAUM, Andrew S. **Redes de Computadores**. 4 ed. Rio de Janeiro: Campus, 1997.

TANENBAUM, Andrew S. **Computer networks**. 4 ed. New Jersey: Prentice Hall, 2003.

TARAPANOFF K; ARAÚJO JÚNIOR RH; CORMIER, PMJ. Sociedade da informação e inteligência em unidades de informação. **Ciência da Informação**. Brasília, 29(3): 91-100, 2000.

TORRES, Gabriel**. Redes de computadores**. Rio de Janeiro: Novaterra, 2009.

ULBRICH, Henrique Cesar; VALLE, James Della. **Universidade H4ck3r**. 3 ed. São Paulo: Digerati books, 2003

VASCONCELOS, L. M. R; CARVALHO, C. L. **Aplicação de regras de associação para mineração de dados na web:** Technical report. Cidade de Goiás: Universidade Federal de Goiás, 2004.

KNIME.COM. **KNIME**. Disponível em: <http://www.knime.org/>, acessado em Maio de 2012.

KXEN. **KXEN**. Disponível em: <http://www.aexis.com/KXEN-Analytic-Framework.htm>, acessado em Maio de 2012.

WICKERT E; MARCONDES J; LEMOS MV; LEMOS EGM. Nitrogen assimilation in Citrus based on CitEST data mining**. Genet Mol Biol.** Ribeirão Preto, 2007;30(3 Suppl):810-8.

WIMAX-INDUSTRY, **Wimax-Industry.** Disponível em <http://www.wimax-industry.com/sp/gbm/ gbmprod3.htm > Acessado em 01/08/2012

WUU, L., CHEN, S. Building intrusion pattern miner for Snort network intrusion detection system**, Proceedings of IEEE 37th Annual 2003 International Carnahan Conference on Security Technology (ICCST).** Taipei, TW, 2003, p. 477–484.

ZHU D, PORTER A, CUNNINGHAM S, CARLISIE J, NAYAK A. A process for mining science & technology documents databases, illustrated for the case of "knowledge discovery and data mining". **Ciência da Informação**. Brasília, 28(1): 7-14, 1999.

APÊNDICE A: SISTEMA DE CONSULTA WHOIS

Arquivo whois.php

```
<?php
require "conn.php";
require "functions_whois.php";

//rodar quantas vezes?
$linhas = 10;

for($i=0; $i<$linhas; $i++){

        $linha = $i+1;
        $SQL = "select timestamp, sig_name, INET_NTOA(ip_src) as ip_origem,
INET_NTOA(ip_dst) as ip_destino, ip_proto, layer4_sport, layer4_dport from
acid_event e, sig_class sc where e.sig_class_id=sc.sig_class_id and country_src is null
limit ".$linha.",1;";


        $result = @mysql_query($SQL) or
trigger_error(mysql_error(),E_USER_ERROR);
        $total = @mysql_num_rows($result);

        // Caso já exista linhas sem pais de origem o número de linhas será 1..
        if($total == 1)
        {
                $dados = @mysql_fetch_array($result);
                $codigo = $dados["timestamp"];
                echo $ip_src = $dados["ip_origem"];
                echo "<br>";
                echo $ip_dst = $dados["ip_destino"];
                echo "<br>";
                echo "<br>";


                $whois_src = get_whois($ip_src);
                //echo "<pre>$whois_src</pre>";
                $owner = strpos($whois_src,'OrgName:');
```

```
                    $owner2 = strpos($whois_src,'OrgId:');
                    $casas = $owner2 - $owner;
                    echo $country_src = substr($whois_src,$owner+8,$casas-8);
                    $country = strpos($whois_src,'Country:');
                    echo $owner_src = substr($whois_src,$country+8,10);

                    $whois_dst = get_whois($ip_dst);
                    echo "<pre>$whois_dst</pre>";
                    $owner = strpos($whois_dst,'OrgName:');
                    $owner2 = strpos($whois_dst,'OrgId:');
                    $casas = $owner2 - $owner;
                    echo $country_dst = substr($whois_dst,$owner+8,$casas-8);
                    $country = strpos($whois_dst,'Country:');
                    echo $owner_dst = substr($whois_dst,$country+8,10);

                    echo "<br>";

                    $sql_update = "UPDATE acid_event set country_src = '" .$country_src. "', owner_src = '" .$owner_src. "', country_dst = '" .$country_dst. "', owner_dst = '" .$owner_dst. "' where timestamp = '" .$codigo. "' ";
                    $result_update = @mysql_query($sql_update) or trigger_error(mysql_error(),E_USER_ERROR);

                }

        }

        ?>
```

Arquivo conn.php

```
        <?php

        //$hostname_conn = "localhost:3306:/var/lib/mysql/mysql.sock";
        $hostname_conn = "localhost:3306";
        $database_conn = "snort";
        $username_conn = "snort";
        $password_conn = "shadow";
        $conn = mysql_pconnect($hostname_conn, $username_conn, $password_conn) or trigger_error(mysql_error(),E_USER_ERROR);

        mysql_select_db($database_conn);

        ?>
```

Arquivo function_whois.php

```
<?php

//whois.iana.org
//whois.lacnic.net
//whois.arin.net


/**
	Program to perform ip whois
	Silver Moon
	m00n.silv3r@gmail.com
*/

//$ip = "74.65.112.23";

//$whois = get_whois($ip);

//echo "<pre>$whois</pre>";

/**
	Get the whois content of an ip by selecting the correct server
*/
function get_whois($ip)
{
	$w = get_whois_from_server('whois.iana.org' , $ip);

	preg_match('@whois\.[\w\.]*@si' , $w , $data);

	$whois_server = $data[0];

	//echo $whois_server;

	//now get actual whois data
	$whois_data = get_whois_from_server($whois_server , $ip);

	return $whois_data;
}

/**
	Get the whois result from a whois server
	return text
*/
function get_whois_from_server($server , $ip)
{
	$data = '';

	#Before connecting lets check whether server alive or not
```

```
		#Create the socket and connect
		$f = fsockopen($server, 43, $errno, $errstr, 3);	//Open a new connection
		if(!$f)
		{
			return '';
		}

		#Set the timeout limit for read
		if(!stream_set_timeout($f , 3))
		{
			die('Unable to set set_timeout');	#Did this solve the problem ?
		}

		#Send the IP to the whois server
		if($f)
		{
			fputs($f, "$ip\r\n");
		}

		/*
			Theory : stream_set_timeout must be set after a write and before a read for it to take effect on the read operation
			If it is set before the write then it will have no effect : http://in.php.net/stream_set_timeout
		*/

		//Set the timeout limit for read
		if(!stream_set_timeout($f , 3))
		{
			die('Unable to stream_set_timeout'); #Did this solve the problem ?
		}
		//Set socket in non-blocking mode
		stream_set_blocking ($f, 0 );
		//If connection still valid
		if($f)
		{
			while (!feof($f))
			{
				$data .= fread($f , 128);
			}
		}

		//Now return the data
		return $data;
	}?>
```

APÊNDICE B: ARQUIVO DE CONFIGURAÇÃO DO SNORT

Arquivo Snort.conf

```
#--------------------------------------------------
#   http://www.snort.org     Snort 2.8.5.2 Ruleset
#     Contact: snort-sigs@lists.sourceforge.net
#--------------------------------------------------
# $Id$
#
######################################################
# This file contains a sample snort configuration.
# You can take the following steps to create your own custom configuration:
#
#  1) Set the variables for your network
#  2) Configure dynamic loaded libraries
#  3) Configure preprocessors
#  4) Configure output plugins
#  5) Add any runtime config directives
#  6) Customize your rule set
#
######################################################
# Step #1: Set the network variables:
#
# You must change the following variables to reflect your local network. The
# variable is currently setup for an RFC 1918 address space.
#
# You can specify it explicitly as:
#
# var HOME_NET 10.1.1.0/24
#
# if Snort is built with IPv6 support enabled (--enable-ipv6), use:
#
# ipvar HOME_NET 10.1.1.0/24
#
# or use global variable $<interfacename>_ADDRESS which will be always
# initialized to IP address and netmask of the network interface which you run
# snort at.  Under Windows, this must be specified as
# $(<interfacename>_ADDRESS), such as:
```

```
# $(\Device\Packet_{12345678-90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP addresses for HOME_NET
# by separating the IPs with commas like this:
#
# var HOME_NET [10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY SPACES IN YOUR LIST!
#
# or you can specify the variable to be any IP address
# like this:

#var HOME_NET any
var HOME_NET [10.0.0.0/8,200.143.198.0/24]

# Set up the external network addresses as well.  A good start may be "any"
#var EXTERNAL_NET any
var EXTERNAL_NET !$HOME_NET

# Configure your server lists.  This allows snort to only look for attacks to
# systems that have a service up.  Why look for HTTP attacks if you are not
# running a web server?  This allows quick filtering based on IP addresses
# These configurations MUST follow the same configuration scheme as defined
# above for $HOME_NET.

# List of DNS servers on your network
var DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
var SMTP_SERVERS $HOME_NET

# List of web servers on your network
var HTTP_SERVERS $HOME_NET

# List of sql servers on your network
var SQL_SERVERS $HOME_NET

# List of telnet servers on your network
var TELNET_SERVERS $HOME_NET

# List of telnet servers on your network
var FTP_SERVERS $HOME_NET

# List of snmp servers on your network
var SNMP_SERVERS $HOME_NET

# Configure your service ports.  This allows snort to look for attacks destined
# to a specific application only on the ports that application runs on.  For
```

```
# example, if you run a web server on port 8180, set your HTTP_PORTS variable
# like this:
#
# portvar HTTP_PORTS 8180
#
# Ports you run web servers on
portvar HTTP_PORTS 80

# NOTE:  If you wish to define multiple HTTP ports, use the portvar
# syntax to represent lists of ports and port ranges.  Examples:
## portvar HTTP_PORTS [80,8080]
## portvar HTTP_PORTS [80,8000:8080]
# And only include the rule that uses $HTTP_PORTS once.
#
# The pre-2.8.0 approach of redefining the variable to a different port and
# including the rules file twice is obsolete.  See README.variables for more
# details.

# Ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# Ports you might see oracle attacks on
portvar ORACLE_PORTS 1521

# Ports for FTP servers
portvar FTP_PORTS 21

# other variables
#
# AIM servers.  AOL has a habit of adding new AIM servers, so instead of
# modifying the signatures when they do, we add them to this list of servers.
var AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# Configure the snort decoder
# ============================
#
# Snort's decoder will alert on lots of things such as header
# truncation or options of unusual length or infrequently used tcp options
#
#
```

```
# Stop generic decode events:
#
# config disable_decode_alerts
#
# Stop Alerts on experimental TCP options
#
# config disable_tcpopt_experimental_alerts
#
# Stop Alerts on obsolete TCP options
#
# config disable_tcpopt_obsolete_alerts
#
# Stop Alerts on T/TCP alerts
#
# In snort 2.0.1 and above, this only alerts when a TCP option is detected
# that shows T/TCP being actively used on the network.  If this is normal
# behavior for your network, disable the next option.
#
# config disable_tcpopt_ttcp_alerts
#
# Stop Alerts on all other TCPOption type events:
#
# config disable_tcpopt_alerts
#
# Stop Alerts on invalid ip options
#
# config disable_ipopt_alerts
#
# Alert if value in length field (IP, TCP, UDP) is greater than the
# actual length of the captured portion of the packet that the length
# is supposed to represent:
#
# config enable_decode_oversized_alerts
#
# Same as above, but drop packet if in Inline mode -
# enable_decode_oversized_alerts must be enabled for this to work:
#
# config enable_decode_oversized_drops
#

# Configure the detection engine
# ==============================
#
# Use a different pattern matcher in case you have a machine with very limited
# resources:
#
# config detection: search-method lowmem

# Configure Inline Resets
# =======================
```

```
#
# If running an iptables firewall with snort in InlineMode() we can now
# perform resets via a physical device. We grab the indev from iptables
# and use this for the interface on which to send resets. This config
# option takes an argument for the src mac address you want to use in the
# reset packet.  This way the bridge can remain stealthy. If the src mac
# option is not set we use the mac address of the indev device. If we
# don't set this option we will default to sending resets via raw socket,
# which needs an ipaddress to be assigned to the int.
#
# config layer2resets: 00:06:76:DD:5F:E3

###################################################
# Step #2: Configure dynamic loaded libraries
#
# If snort was configured to use dynamically loaded libraries,
# those libraries can be loaded here.
#
# Each of the following configuration options can be done via
# the command line as well.
#
# Load all dynamic preprocessors from the install path
# (same as command line option --dynamic-preprocessor-lib-dir)
#
dynamicpreprocessor directory /usr/lib/snort_dynamicpreprocessor/
#
# Load a specific dynamic preprocessor library from the install path
# (same as command line option --dynamic-preprocessor-lib)
#
#                                   dynamicpreprocessor                                   file
/usr/lib/snort_dynamicpreprocessor/libdynamicexample.so
#
# Load a dynamic engine from the install path
# (same as command line option --dynamic-engine-lib)
#
dynamicengine /usr/lib/snort_dynamicengine/libsf_engine.so
#
# Load all dynamic rules libraries from the install path
# (same as command line option --dynamic-detection-lib-dir)
#
# dynamicdetection directory /usr/lib/snort_dynamicrule/
#
# Load a specific dynamic rule library from the install path
# (same as command line option --dynamic-detection-lib)
#
# dynamicdetection file /usr/lib/snort_dynamicrule/libdynamicexamplerule.so
#

###################################################
# Step #3: Configure preprocessors
```

```
#
# General configuration for preprocessors is of
# the form
# preprocessor <name_of_processor>: <configuration_options>

# frag3: Target-based IP defragmentation
# -------------------------------------
#
# Frag3 is a brand new IP defragmentation preprocessor that is capable of
# performing "target-based" processing of IP fragments.  Check out the
# README.frag3 file in the doc directory for more background and configuration
# information.
#
# Frag3 configuration is a two step process, a global initialization phase
# followed by the definition of a set of defragmentation engines.
#
# Global configuration defines the number of fragmented packets that Snort can
# track at the same time and gives you options regarding the memory cap for the
# subsystem or, optionally, allows you to preallocate all the memory for the
# entire frag3 system.
#
# frag3_global options:
#   max_frags: Maximum number of frag trackers that may be active at once.
#              Default value is 8192.
#    memcap: Maximum amount of memory that frag3 may access at any given time.
#            Default value is 4MB.
#     prealloc_frags: Maximum number of individual fragments that may be processed
#                     at once.  This is instead of the memcap system, uses static
#                     allocation to increase performance.  No default value.  Each
#                     preallocated fragment typically eats ~1550 bytes.  However,
#                     the exact amount is determined by the snaplen, and this can
#                     go as high as 64K so beware!
#
# Target-based behavior is attached to an engine as a "policy" for handling
# overlaps and retransmissions as enumerated in the Paxson paper.  There are
# currently five policy types available: "BSD", "BSD-right", "First", "Linux"
# and "Last".  Engines can be bound to standard Snort CIDR blocks or
# IP lists.
#
# frag3_engine options:
#   timeout: Amount of time a fragmented packet may be active before expiring.
#            Default value is 60 seconds.
#   ttl_limit: Limit of delta allowable for TTLs of packets in the fragments.
#              Based on the initial received fragment TTL.
#    min_ttl: Minimum acceptable TTL for a fragment, frags with TTLs below this
#            value will be discarded.  Default value is 0.
```

```
#   detect_anomalies: Activates frag3's anomaly detection mechanisms.
#   policy: Target-based policy to assign to this engine.  Default is BSD.
#   bind_to: IP address set to bind this engine to.  Default is all hosts.
#
# Frag3 configuration example:
#preprocessor frag3_global: max_frags 65536, prealloc_frags 65536
#preprocessor frag3_engine: policy linux \
#                   bind_to [10.1.1.12/32,10.1.1.13/32] \
#                   detect_anomalies
#preprocessor frag3_engine: policy first \
#                   bind_to 10.2.1.0/24 \
#                   detect_anomalies
#preprocessor frag3_engine: policy last \
#                   bind_to 10.3.1.0/24
#preprocessor frag3_engine: policy bsd

preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies overlap_limit 10

# stream5: Target Based stateful inspection/stream reassembly for Snort
# ---------------------------------------------------------------------
# Stream5 is a target-based stream engine for Snort.  It handles both
# TCP and UDP connection tracking as well as TCP reassembly.
#
# See README.stream5 for details on the configuration options.
#
# Example config
preprocessor stream5_global: max_tcp 8192, track_tcp yes, \
                    track_udp no
preprocessor stream5_tcp: policy first
# Not recommended in production systems
# preprocessor stream5_tcp: policy first, use_static_footprint_sizes
# preprocessor stream5_udp: ignore_any_rules


# Performance Statistics
# ----------------------
# Documentation for this is provided in the Snort Manual.  You should read it.
# It is included in the release distribution as doc/snort_manual.pdf
#
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000

# http_inspect: normalize and detect HTTP traffic and protocol anomalies
#
# lots of options available here. See doc/README.http_inspect.
# unicode.map should be wherever your snort.conf lives, or given
# a full path to where snort can find it.
preprocessor http_inspect: global \
   iis_unicode_map unicode.map 1252
```

```
preprocessor http_inspect_server: server default \
   profile all ports { 80 8080 8180 } oversize_dir_length 500

#
#  Example unique server configuration
#
#preprocessor http_inspect_server: server 1.1.1.1 \
#    ports { 80 3128 8080 } \
#    server_flow_depth 0 \
#    ascii no \
#    double_decode yes \
#    non_rfc_char { 0x00 } \
#    chunk_length 500000 \
#    non_strict \
#    oversize_dir_length 300 \
#    no_alerts


# rpc_decode: normalize RPC traffic
# --------------------------------
# RPC may be sent in alternate encodings besides the usual 4-byte encoding
# that is used by default. This plugin takes the port numbers that RPC
# services are running on as arguments - it is assumed that the given ports
# are actually running this type of service. If not, change the ports or turn
# it off.
# The RPC decode preprocessor uses generator ID 106
#
# arguments: space separated list
# alert_fragments - alert on any rpc fragmented TCP data
# no_alert_multiple_requests - don't alert when >1 rpc query is in a packet
# no_alert_large_fragments - don't alert when the fragmented
#                            sizes exceed the current packet size
# no_alert_incomplete - don't alert when a single segment
#                       exceeds the current packet size

preprocessor rpc_decode: 111 32771

# bo: Back Orifice detector
# -------------------------
# Detects Back Orifice traffic on the network.
#
# arguments:
#   syntax:
#     preprocessor bo: noalert { client | server | general | snort_attack } \
#                      drop    { client | server | general | snort_attack }
#   example:
#     preprocessor bo: noalert { general server } drop { snort_attack }
#
#
# The Back Orifice detector uses Generator ID 105 and uses the
```

```
# following SIDS for that GID:
#  SID    Event description
# -----   -------------------
#   1      Back Orifice traffic detected
#   2      Back Orifice Client Traffic Detected
#   3      Back Orifice Server Traffic Detected
#   4      Back Orifice Snort Buffer Attack

preprocessor bo

# ftp_telnet: FTP & Telnet normalizer, protocol enforcement and buff overflow
# ----------------------------------------------------------------------------
# This preprocessor normalizes telnet negotiation strings from telnet and
# ftp traffic.  It looks for traffic that breaks the normal data stream
# of the protocol, replacing it with a normalized representation of that
# traffic so that the "content" pattern matching keyword can work without
# requiring modifications.
#
# It also performs protocol correctness checks for the FTP command channel,
# and identifies open FTP data transfers.
#
# FTPTelnet has numerous options available, please read
# README.ftptelnet for help configuring the options for the global
# telnet, ftp server, and ftp client sections for the protocol.

#####
# Per Step #2, set the following to load the ftptelnet preprocessor
# dynamicpreprocessor file <full path to libsf_ftptelnet_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_ftptelnet_preproc.so>

preprocessor ftp_telnet: global \
  encrypted_traffic yes \
  inspection_type stateful

preprocessor ftp_telnet_protocol: telnet \
  normalize \
  ayt_attack_thresh 200

# This is consistent with the FTP rules as of 18 Sept 2004.
# CWD can have param length of 200
# MODE has an additional mode of Z (compressed)
# Check for string formats in USER & PASS commands
# Check nDTM commands that set modification time on the file.
preprocessor ftp_telnet_protocol: ftp server default \
  def_max_param_len 100 \
  alt_max_param_len 200 { CWD } \
  cmd_validity MODE < char ASBCZ > \
  cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
  chk_str_fmt { USER PASS RNFR RNTO SITE MKD } \
```

```
    telnet_cmds yes \
    data_chan

  preprocessor ftp_telnet_protocol: ftp client default \
    max_resp_len 256 \
    bounce yes \
    telnet_cmds yes

  # smtp: SMTP normalizer, protocol enforcement and buffer overflow
  # ------------------------------------------------------------------------
  # This preprocessor normalizes SMTP commands by removing extraneous spaces.
  # It looks for overly long command lines, response lines, and data header lines.
  # It can alert on invalid commands, or specific valid commands.  It can optionally
  # ignore mail data, and can ignore TLS encrypted data.
  #
  # SMTP has numerous options available, please read README.SMTP for help
  # configuring options.

  #####
  # Per Step #2, set the following to load the smtp preprocessor
  # dynamicpreprocessor file <full path to libsf_smtp_preproc.so>
  # or use commandline option
  # --dynamic-preprocessor-lib <full path to libsf_smtp_preproc.so>

  preprocessor smtp: \
    ports { 25 587 691 } \
    inspection_type stateful \
    normalize cmds \
    normalize_cmds { EXPN VRFY RCPT } \
    alt_max_command_line_len 260 { MAIL } \
    alt_max_command_line_len 300 { RCPT } \
    alt_max_command_line_len 500 { HELP HELO ETRN } \
    alt_max_command_line_len 255 { EXPN VRFY }

  # sfPortscan
  # ----------
  # Portscan detection module.  Detects various types of portscans and
  # portsweeps.  For more information on detection philosophy, alert types,
  # and detailed portscan information, please refer to the README.sfportscan.
  #
  # -configuration options-
  #    proto { tcp udp icmp ip all }
  #      The arguments to the proto option are the types of protocol scans that
  #      the user wants to detect.  Arguments should be separated by spaces and
  #      not commas.
  #    scan_type { portscan portsweep decoy_portscan distributed_portscan all }
  #      The arguments to the scan_type option are the scan types that the
  #      user wants to detect.  Arguments should be separated by spaces and not
```

```
#       commas.
#     sense_level { low|medium|high }
#       There is only one argument to this option and it is the level of
#       sensitivity in which to detect portscans.  The 'low' sensitivity
#       detects scans by the common method of looking for response errors, such
#       as TCP RSTs or ICMP unreachables.  This level requires the least
#       tuning.  The 'medium' sensitivity level detects portscans and
#       filtered portscans (portscans that receive no response).  This
#       sensitivity level usually requires tuning out scan events from NATed
#       IPs, DNS cache servers, etc.  The 'high' sensitivity level has
#       lower thresholds for portscan detection and a longer time window than
#       the 'medium' sensitivity level.  Requires more tuning and may be noisy
#       on very active networks.  However, this sensitivity levels catches the
#       most scans.
#     memcap { positive integer }
#       The maximum number of bytes to allocate for portscan detection.  The
#       higher this number the more nodes that can be tracked.
#     logfile { filename }
#       This option specifies the file to log portscan and detailed portscan
#       values to.  If there is not a leading /, then snort logs to the
#       configured log directory.  Refer to README.sfportscan for details on
#       the logged values in the logfile.
#     watch_ip { Snort IP List }
#     ignore_scanners { Snort IP List }
#     ignore_scanned { Snort IP List }
#       These options take a snort IP list as the argument.  The 'watch_ip'
#       option specifies the IP(s) to watch for portscan.  The
#       'ignore_scanners' option specifies the IP(s) to ignore as scanners.
#       Note that these hosts are still watched as scanned hosts.  The
#       'ignore_scanners' option is used to tune alerts from very active
#       hosts such as NAT, nessus hosts, etc.  The 'ignore_scanned' option
#       specifies the IP(s) to ignore as scanned hosts.  Note that these hosts
#       are still watched as scanner hosts.  The 'ignore_scanned' option is
#       used to tune alerts from very active hosts such as syslog servers, etc.
#     detect_ack_scans
#       This option will include sessions picked up in midstream by the stream
#        module, which is necessary to detect ACK scans.  However, this can lead
to
#         false alerts, especially under heavy load with dropped packets; which is
why
#       the option is off by default.
#
preprocessor sfportscan: proto  { all } \
                  memcap { 10000000 } \
                  sense_level { low }

# arpspoof
#--------------------------------------
# Experimental ARP detection code from Jeff Nathan, detects ARP attacks,
# unicast ARP requests, and specific ARP mapping monitoring.  To make use of
```

```
        # this preprocessor you must specify the IP and hardware address of hosts on
        # the same layer 2 segment as you.  Specify one host IP MAC combo per line.
        # Also takes a "-unicast" option to turn on unicast ARP request detection.
        # Arpspoof uses Generator ID 112 and uses the following SIDS for that GID:

        #  SID    Event description
        # -----   -------------------
        #   1      Unicast ARP request
        #   2      Etherframe ARP mismatch (src)
        #   3      Etherframe ARP mismatch (dst)
        #   4      ARP cache overwrite attack

        #preprocessor arpspoof
        #preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00

        # ssh
        # ----------------------------
        # The SSH preprocessor detects the following exploits: Challenge-Response
        # Authentication overflow, CRC 32 overflow, Secure CRT version string
overflow,
        # and protocol version mismatches.
        #
        # Both Challenge-Response Auth and CRC 32 attacks occur after the key
exchange,
        # and are therefore encrypted.  Both attacks involve sending a large payload
        # (20kb+) to the server immediately after the authentication challenge.
        # To detect the attacks, the SSH preprocessor counts the number of bytes
        # transmitted to the server.  If those bytes exceed a pre-defined limit,
        # set by the option "max_client_bytes", an alert is generated. Since
        # the Challenge-Response Auth overflow only affects SSHv2, while CRC 32
only
        # affects SSHv1, the SSH version string exchange is used to distinguish
        # the attacks.
        #
        # The Secure CRT and protocol mismatch exploits are observable before
        # the key exchange.
        #
        # SSH has numerous options available, please read README.ssh for help
        # configuring options.

        #####
        # Per Step #2, set the following to load the ssh preprocessor
        # dynamicpreprocessor file <full path to libsf_ssh_preproc.so>
        # or use commandline option
        # --dynamic-preprocessor-lib <full path to libsf_ssh_preproc.so>
        #
        preprocessor ssh: server_ports { 22 } \
                        max_client_bytes 19600 \
                        max_encrypted_packets 20 \
                        enable_respoverflow enable_ssh1crc32 \
```

```
                enable_srvoverflow enable_protomismatch

        # DCE/RPC
        #--------------------------------------
        #
        # The dcerpc preprocessor detects and decodes SMB and DCE/RPC traffic.
        # It is primarily interested in DCE/RPC data, and only decodes SMB
        # to get at the DCE/RPC data carried by the SMB layer.
        #
        # Currently, the preprocessor only handles reassembly of fragmentation
        # at both the SMB and DCE/RPC layer.  Snort rules can be evaded by
        # using both types of fragmentation; with the preprocessor enabled
        # the rules are given a buffer with a reassembled SMB or DCE/RPC
        # packet to examine.
        #
        # At the SMB layer, only fragmentation using WriteAndX is currently
        # reassembled.  Other methods will be handled in future versions of
        # the preprocessor.
        #
        # Autodetection of SMB is done by looking for "\xFFSMB" at the start of
        # the SMB data, as well as checking the NetBIOS header (which is always
        # present for SMB) for the type "SMB Session".
        #
        # Autodetection of DCE/RPC is not as reliable.  Currently, two bytes are
        # checked in the packet.  Assuming that the data is a DCE/RPC header,
        # one byte is checked for DCE/RPC version (5) and another for the type
        # "DCE/RPC Request".  If both match, the preprocessor proceeds with that
        # assumption that it is looking at DCE/RPC data.  If subsequent checks
        # are nonsensical, it ends processing.
        #
        # DCERPC has numerous options available, please read README.dcerpc for
help
        # configuring options.

        #####
        # Per Step #2, set the following to load the dcerpc preprocessor
        # dynamicpreprocessor file <full path to libsf_dcerpc_preproc.so>
        # or use commandline option
        # --dynamic-preprocessor-lib <full path to libsf_dcerpc_preproc.so>
        #
        #preprocessor dcerpc: \
        #    autodetect \
        #    max_frag_size 3000 \
        #    memcap 100000


        # DCE/RPC 2
        #--------------------------------------
        # See doc/README.dcerpc2 for explanations of what the
        # preprocessor does and how to configure it.
```

```
#
preprocessor dcerpc2
preprocessor dcerpc2_server: default


# DNS
#--------------------------------------
# The dns preprocessor (currently) decodes DNS Response traffic
# and detects a few vulnerabilities.
#
# DNS has a few options available, please read README.dns for
# help configuring options.

#####
# Per Step #2, set the following to load the dns preprocessor
# dynamicpreprocessor file <full path to libsf_dns_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_dns_preproc.so>

preprocessor dns: \
    ports { 53 } \
    enable_rdata_overflow

# SSL
#--------------------------------------
# Encrypted traffic should be ignored by Snort for both performance reasons
# and to reduce false positives.  The SSL Dynamic Preprocessor (SSLPP)
# inspects SSL traffic and optionally determines if and when to stop
# inspection of it.
#
# Typically, SSL is used over port 443 as HTTPS.  By enabling the SSLPP to
# inspect port 443, only the SSL handshake of each connection will be
# inspected.  Once the traffic is determined to be encrypted, no further
# inspection of the data on the connection is made.
#
# If you don't necessarily trust all of the SSL capable servers on your
# network, you should remove the "trustservers" option from the configuration.
#
#   Important note: Stream5 should be explicitly told to reassemble
#                   traffic on the ports that you intend to inspect SSL
#                   encrypted traffic on.
#
#   To add reassembly on port 443 to Stream5, use 'port both 443' in the
#   Stream5 configuration.

preprocessor ssl: noinspect_encrypted, trustservers


##################################################################
####
```

```
# Step #4: Configure output plugins
#
# Uncomment and configure the output plugins you decide to use.  General
# configuration for output plugins is of the form:
#
# output <name_of_plugin>: <configuration_options>
#
# alert_syslog: log alerts to syslog
# ----------------------------------
# Use one or more syslog facilities as arguments.  Win32 can also optionally
# specify a particular hostname/port.  Under Win32, the default hostname is
# '127.0.0.1', and the default port is 514.
#
# [Unix flavours should use this format...]
# output alert_syslog: LOG_AUTH LOG_ALERT
#
# [Win32 can use any of these formats...]
# output alert_syslog: LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname, LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname:port, LOG_AUTH LOG_ALERT

# log_tcpdump: log packets in binary tcpdump format
# -------------------------------------------------
# The only argument is the output file name.
#
output log_tcpdump: tcpdump.log

# database: log to a variety of databases
# ---------------------------------------
# See the README.database file for more information about configuring
# and using this plugin.
#
# output database: log, mysql, user=root password=test dbname=db
host=localhost
# output database: alert, postgresql, user=snort dbname=snort
# output database: log, odbc, user=snort dbname=snort
# output database: log, mssql, dbname=snort user=snort password=test
# output database: log, oracle, dbname=snort user=snort password=test
#
# On Debian Systems, the database configuration is kept in a separate file:
# /etc/snort/database.conf.
# This file can be empty, if you are not using any database information
# If you are using databases, please edit that file instead of this one, to
# ensure smoother upgrades to future versions of this package.
include database.conf


# unified: Snort unified binary format alerting and logging
# -------------------------------------------------------------
```

```
        # The unified output plugin provides two new formats for logging and
generating
        # alerts from Snort, the "unified" format.  The unified format is a straight
        # binary format for logging data out of Snort that is designed to be fast and
        # efficient.  Used with barnyard (the new alert/log processor), most of the
        # overhead for logging and alerting to various slow storage mechanisms such as
        # databases or the network can now be avoided.
        #
        # Check out the spo_unified.h file for the data formats.
        #
        # Two arguments are supported.
        #    filename - base filename to write to (current time_t is appended)
        #    limit    - maximum size of spool file in MB (default: 128)
        #
        # output alert_unified: filename snort.alert, limit 128
        # output log_unified: filename snort.log, limit 128


        # prelude: log to the Prelude Hybrid IDS system
        # ------------------------------------------
        #
        # profile = Name of the Prelude profile to use (default is snort).
        #
        # Snort priority to IDMEF severity mappings:
        # high < medium < low < info
        #
        # These are the default mapped from classification.config:
        # info   = 4
        # low    = 3
        # medium = 2
        # high   = anything below medium
        #
        # output alert_prelude
        # output alert_prelude: profile=snort-profile-name


        # You can optionally define new rule types and associate one or more output
        # plugins specifically to that type.
        #
        # This example will create a type that will log to just tcpdump.
        # ruletype suspicious
        # {
        #   type log
        #   output log_tcpdump: suspicious.log
        # }
        #
        # EXAMPLE RULE FOR SUSPICIOUS RULETYPE:
        # suspicious tcp $HOME_NET any -> $HOME_NET 6667 (msg:"Internal IRC
Server";)
        #
```

```
# This example will create a rule type that will log to syslog and a mysql
# database:
# ruletype redalert
# {
#   type alert
#   output alert_syslog: LOG_AUTH LOG_ALERT
#   output database: log, mysql, user=snort dbname=snort host=localhost
# }
#
# EXAMPLE RULE FOR REDALERT RULETYPE:
# redalert tcp $HOME_NET any -> $EXTERNAL_NET 31337 \
#   (msg:"Someone is being LEET"; flags:A+;)

#
# Include classification & priority settings
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\classification.config
#

include classification.config

#
# Include reference systems
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\reference.config
#

include reference.config

##################################################################
####
# Step #5: Configure snort with config statements
#
# See the snort manual for a full set of configuration references
#
# config flowbits_size: 64
#
# New global ignore_ports config option from Andy Mullican
#
# config ignore_ports: <tcp|udp> <list of ports separated by whitespace>
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53


##################################################################
####
# Step #6: Customize your rule set
#
# Up to date snort rules are available at http://www.snort.org
#
```

```
# The snort web site has documentation about how to write your own custom snort
# rules.

#==================================================
# Include all relevant rulesets here
#
# The following rulesets are disabled by default:
#
#   web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
#   chat, multimedia, and p2p
#
# These rules are either site policy specific or require tuning in order to not
# generate false positive alerts in most enviornments.
#
# Please read the specific include file for more information and
# README.alert_order for how rule ordering affects how alerts are triggered.
#==================================================

include $RULE_PATH/local.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/community-exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/community-dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules

# Specific web server rules:
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules
include $RULE_PATH/community-sql-injection.rules
include $RULE_PATH/community-web-client.rules
include $RULE_PATH/community-web-dos.rules
include $RULE_PATH/community-web-iis.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-web-php.rules
```

```
# Rules for other services:
include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/community-oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/snmp.rules
include $RULE_PATH/community-ftp.rules
include $RULE_PATH/smtp.rules
include $RULE_PATH/community-smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/community-imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules

include $RULE_PATH/nntp.rules
include $RULE_PATH/community-nntp.rules
include $RULE_PATH/community-sip.rules
include $RULE_PATH/other-ids.rules

# Attack-in-progress rules:
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/community-bot.rules
include $RULE_PATH/community-virus.rules
# This ruleset is almost useless currently:
# include $RULE_PATH/virus.rules
# Note: this rule is extremely chatty, enable with care
# include $RULE_PATH/shellcode.rules

# Policy related rules:
# include $RULE_PATH/policy.rules
# include $RULE_PATH/community-policy.rules
# include $RULE_PATH/porn.rules
# include $RULE_PATH/community-inappropriate.rules
# include $RULE_PATH/chat.rules
# include $RULE_PATH/multimedia.rules
# include $RULE_PATH/p2p.rules
# include $RULE_PATH/community-game.rules
# include $RULE_PATH/community-misc.rules

# Extremely chatty rules:
# include $RULE_PATH/info.rules
# include $RULE_PATH/icmp-info.rules
# include $RULE_PATH/community-icmp.rules
```

```
# Experimental rules:
# NOTICE: this is currently empty
include $RULE_PATH/experimental.rules

# include $PREPROC_RULE_PATH/preprocessor.rules
# include $PREPROC_RULE_PATH/decoder.rules

# Include any thresholding or suppression commands. See threshold.conf in the
# <snort src>/etc directory for details. Commands don't necessarily need to be
# contained in this conf, but a separate conf makes it easier to maintain them.
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\threshold.conf
# Uncomment if needed.
# include threshold.conf
```

APÊNDICE C: SCRIPT DE INICIALIZAÇÃO

Arquivo /etc/init.d/snortbarn

```
#! /bin/sh
#
### BEGIN INIT INFO
# Provides: snortbarn
# Required-Start: $remote_fs $syslog mysql
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# X-Interactive: true
# Short-Description: Start Snort and Barnyard
### END INIT INFO
. /lib/init/vars.sh
. /lib/lsb/init-functions
mysqld_get_param() {
/usr/sbin/mysqld --print-defaults | tr " " "\n" | grep -- "--$1" | tail -n 1 | cut -d= -f2
}
do_start()
{
log_daemon_msg "Starting Snort and Barnyard" ""
# Make sure mysql has finished starting
ps_alive=0
while [ $ps_alive -lt 1 ];
do
pidfile=`mysqld_get_param pid-file`
if [ -f "$pidfile" ] && ps `cat $pidfile` >/dev/null 2>&1; then ps_alive=1; fi
sleep 1
done
/sbin/ifconfig eth1 up
/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth1 &
/usr/local/bin/barnyard2 -q -c /etc/snort/barnyard2.conf -d /var/log/snort -f
snort.log -w /etc/snort/bylog.waldo -G /etc/snort/gen-msg.map –S
/etc/snort/sid-msg.map -C /etc/snort/classification.config 2> /dev/null &
log_end_msg 0
return 0
}
do_stop()
{
```

```
log_daemon_msg "Stopping Snort and Barnyard" ""
kill $(pidof snort) 2> /dev/null
kill $(pidof barnyard2) 2> /dev/null
log_end_msg 0
return 0
}
case "$1" in
start)
do_start
;;
stop)
do_stop
;;
restart)
do_stop
do_start
;;
*)
echo "Usage: snort-barn {start|stop|restart}" >&2
exit 3
;;
esac
exit 0
```

APÊNDICE D: SCRIPT DE CRIAÇÃO DO BANCO MYSQL

```
# Copyright (C) 2000-2002 Carnegie Mellon University
#
# Maintainer: Roman Danyliw <rdd@cert.org>, <roman@danyliw.com>
#
# Original Author(s): Jed Pickel <jed@pickel.net>    (2000-2001)
#                     Roman Danyliw <rdd@cert.org>
#                     Todd Schrubb <tls@cert.org>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License Version 2 as
# published by the Free Software Foundation.  You may not use, modify or
# distribute this program under any other version of the GNU General
# Public License.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

CREATE TABLE `schema` ( vseq       INT      UNSIGNED NOT NULL,
                        ctime      DATETIME NOT NULL,
                        PRIMARY KEY (vseq));
INSERT INTO `schema`  (vseq, ctime) VALUES ('107', now());

CREATE TABLE event  ( sid          INT      UNSIGNED NOT NULL,
                        cid        INT      UNSIGNED NOT NULL,
                        signature  INT      UNSIGNED NOT NULL,
                        timestamp  DATETIME NOT NULL,
                        PRIMARY KEY (sid,cid),
                        INDEX      sig (signature),
                        INDEX      time (timestamp));
```

```
CREATE TABLE signature ( sig_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
                         sig_name     VARCHAR(255) NOT NULL,
                         sig_class_id INT UNSIGNED NOT NULL,
                         sig_priority INT UNSIGNED,
                         sig_rev      INT UNSIGNED,
                         sig_sid      INT UNSIGNED,
                         sig_gid      INT UNSIGNED,
                         PRIMARY KEY (sig_id),
                         INDEX   sign_idx (sig_name(20)),
                         INDEX   sig_class_id_idx (sig_class_id));

CREATE TABLE sig_reference (sig_id  INT    UNSIGNED NOT NULL,
                            ref_seq INT    UNSIGNED NOT NULL,
                            ref_id  INT    UNSIGNED NOT NULL,
                            PRIMARY KEY(sig_id, ref_seq));

CREATE TABLE reference (  ref_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
                          ref_system_id INT UNSIGNED NOT NULL,
                          ref_tag       TEXT NOT NULL,
                          PRIMARY KEY (ref_id));

CREATE TABLE reference_system ( ref_system_id    INT UNSIGNED NOT NULL AUTO_INCREMENT,
                                ref_system_name VARCHAR(20),
                                PRIMARY KEY (ref_system_id));

CREATE TABLE sig_class ( sig_class_id INT    UNSIGNED NOT NULL AUTO_INCREMENT,
                         sig_class_name      VARCHAR(60) NOT NULL,
                         PRIMARY KEY (sig_class_id),
                         INDEX       (sig_class_id),
                         INDEX       (sig_class_name));

# store info about the sensor supplying data
CREATE TABLE sensor ( sid INT UNSIGNED NOT NULL AUTO_INCREMENT,
                      hostname    TEXT,
                      interface   TEXT,
                      filter      TEXT,
                      detail      TINYINT,
                      encoding    TINYINT,
                      last_cid    INT     UNSIGNED NOT NULL,
                      PRIMARY KEY (sid));

# All of the fields of an ip header
CREATE TABLE iphdr  ( sid INT UNSIGNED NOT NULL,
                      cid       INT     UNSIGNED NOT NULL,
                      ip_src    INT     UNSIGNED NOT NULL,
```

```
                ip_dst      INT      UNSIGNED NOT NULL,
                ip_ver      TINYINT  UNSIGNED,
                ip_hlen     TINYINT  UNSIGNED,
                ip_tos      TINYINT  UNSIGNED,
                ip_len      SMALLINT UNSIGNED,
                ip_id       SMALLINT UNSIGNED,
                ip_flags    TINYINT  UNSIGNED,
                ip_off      SMALLINT UNSIGNED,
                ip_ttl      TINYINT  UNSIGNED,
                ip_proto            TINYINT  UNSIGNED NOT NULL,
                ip_csum             SMALLINT UNSIGNED,
                PRIMARY KEY (sid,cid),
                INDEX ip_src (ip_src),
                INDEX ip_dst (ip_dst));

# All of the fields of a tcp header
CREATE TABLE tcphdr(  sid           INT      UNSIGNED NOT NULL,
                cid         INT      UNSIGNED NOT NULL,
                tcp_sport   SMALLINT UNSIGNED NOT NULL,
                tcp_dport   SMALLINT UNSIGNED NOT NULL,
                tcp_seq     INT      UNSIGNED,
                tcp_ack     INT      UNSIGNED,
                tcp_off     TINYINT  UNSIGNED,
                tcp_res     TINYINT  UNSIGNED,
                tcp_flags   TINYINT  UNSIGNED NOT NULL,
                tcp_win     SMALLINT UNSIGNED,
                tcp_csum    SMALLINT UNSIGNED,
                tcp_urp     SMALLINT UNSIGNED,
                PRIMARY KEY (sid,cid),
                INDEX       tcp_sport (tcp_sport),
                INDEX       tcp_dport (tcp_dport),
                INDEX       tcp_flags (tcp_flags));

# All of the fields of a udp header
CREATE TABLE udphdr(  sid           INT      UNSIGNED NOT NULL,
                cid         INT      UNSIGNED NOT NULL,
                udp_sport   SMALLINT UNSIGNED NOT NULL,
                udp_dport   SMALLINT UNSIGNED NOT NULL,
                udp_len     SMALLINT UNSIGNED,
                udp_csum    SMALLINT UNSIGNED,
                PRIMARY KEY (sid,cid),
                INDEX       udp_sport (udp_sport),
                INDEX       udp_dport (udp_dport));

# All of the fields of an icmp header
CREATE TABLE icmphdr( sid           INT      UNSIGNED NOT NULL,
                cid         INT      UNSIGNED NOT NULL,
                icmp_type   TINYINT  UNSIGNED NOT NULL,
                icmp_code   TINYINT  UNSIGNED NOT NULL,
                icmp_csum   SMALLINT UNSIGNED,
```

```
                icmp_id     SMALLINT UNSIGNED,
                icmp_seq    SMALLINT UNSIGNED,
                PRIMARY KEY (sid,cid),
                INDEX       icmp_type (icmp_type));

        # Protocol options
        CREATE TABLE opt    ( sid        INT      UNSIGNED NOT NULL,
                cid         INT      UNSIGNED NOT NULL,
                optid       INT      UNSIGNED NOT NULL,
                opt_proto   TINYINT  UNSIGNED NOT NULL,
                opt_code    TINYINT  UNSIGNED NOT NULL,
                opt_len     SMALLINT,
                opt_data    TEXT,
                PRIMARY KEY (sid,cid,optid));

        # Packet payload
        CREATE TABLE data   ( sid          INT      UNSIGNED NOT NULL,
                cid          INT      UNSIGNED NOT NULL,
                data_payload  TEXT,
                PRIMARY KEY (sid,cid));

        # encoding is a lookup table for storing encoding types
        CREATE TABLE encoding(encoding_type TINYINT UNSIGNED NOT
NULL,
                encoding_text TEXT NOT NULL,
                PRIMARY KEY (encoding_type));
        INSERT INTO encoding (encoding_type, encoding_text) VALUES (0, 'hex');
        INSERT INTO encoding (encoding_type, encoding_text) VALUES (1,
'base64');
        INSERT INTO encoding (encoding_type, encoding_text) VALUES (2, 'ascii');

        # detail is a lookup table for storing different detail levels
        CREATE TABLE detail  (detail_type TINYINT UNSIGNED NOT NULL,
                detail_text TEXT NOT NULL,
                PRIMARY KEY (detail_type));
        INSERT INTO detail (detail_type, detail_text) VALUES (0, 'fast');
        INSERT INTO detail (detail_type, detail_text) VALUES (1, 'full');

        # be sure to also use the snortdb-extra tables if you want
        # mappings for tcp flags, protocols, and ports
        ------------------------------------------------------------------------------------
```

APÊNDICE E: ÁRVORE DE DECISÃO – DADOS EXTERNOS

J48 pruned tree
---

sig_class_name = attempted-dos: COMMUNITY SIP TCP/IP message flooding directed to SIP proxy (100000.0)
sig_class_name = misc-activity
| country_src = BR: ICMP Destination Unreachable Communication Administratively Prohibited (2361.41/171.0)
| country_src = US
| | hora <= 5: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (707.0/82.0)
| | hora > 5
| | | hora <= 15: ICMP Destination Unreachable Communication Administratively Prohibited (712.05/278.0)
| | | hora > 15
| | | | hora <= 16: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (20.0/4.0)
| | | | hora > 16
| | | | | hora <= 18: ICMP Destination Unreachable Communication Administratively Prohibited (35.0/15.0)
| | | | | hora > 18: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (538.0/125.0)
| country_src = JP
| | hora <= 11
| | | hora <= 10: ICMP Destination Unreachable Communication Administratively Prohibited (15.01/3.0)
| | | hora > 10: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (3.01/0.01)
| | hora > 11: ICMP Destination Unreachable Communication Administratively Prohibited (10.02)
| country_src = VG
| | hora <= 10: ICMP Destination Unreachable Communication Administratively Prohibited (2.0)
| | hora > 10: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (5.01/2.01)
| country_src = DE: ICMP Destination Unreachable Communication Administratively Prohibited (41.04/2.0)

| country_src = AL: ICMP Destination Unreachable Communication Administratively Prohibited (3.0/1.0)
| country_src = GB: ICMP Destination Unreachable Communication Administratively Prohibited (58.06/16.0)
| country_src = NL
| | hora <= 17
| | | hora <= 3: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (7.0/1.0)
| | | hora > 3: ICMP Destination Unreachable Communication Administratively Prohibited (30.05/11.0)
| | hora > 17: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (8.0/1.0)
| country_src = CA
| | hora <= 14
| | | hora <= 3: ICMP Destination Unreachable Communication Administratively Prohibited (2.0)
| | | hora > 3: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (9.01/2.01)
| | hora > 14: ICMP Destination Unreachable Communication Administratively Prohibited (6.0)
| country_src = IL: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (14.01/0.01)
| country_src = CZ: ICMP Destination Unreachable Communication Administratively Prohibited (2.0/1.0)
| country_src = CY: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = FR
| | hora <= 11: ICMP Destination Unreachable Communication Administratively Prohibited (5.01)
| | hora > 11: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (16.01/0.01)
| country_src = CN
| | hora <= 3: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (14.0/3.0)
| | hora > 3: ICMP Destination Unreachable Communication Administratively Prohibited (17.03/2.0)
| country_src = KR: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = PE: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = CH: ICMP Destination Unreachable Communication Administratively Prohibited (5.01/1.0)
| country_src = PT: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (23.02/6.02)
| country_src = AO: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (5.01/0.01)
| country_src = EU: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = TW: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (27.03/0.03)

| country_src = RU: ICMP Destination Unreachable Communication Administratively Prohibited (7.01/3.0)
| country_src = HK: ICMP Destination Unreachable Communication Administratively Prohibited (13.01/2.0)
| country_src = SG: ICMP Destination Unreachable Communication Administratively Prohibited (10.01/1.0)
| country_src = ES: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = RO: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = IN: ICMP Destination Unreachable Communication Administratively Prohibited (3.0)
| country_src = DK: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = HU: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = QA: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = A2: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = CL: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (12.01/0.01)
| country_src = MX: ICMP Destination Unreachable Communication Administratively Prohibited (3.0)
| country_src = PL: ICMP Destination Unreachable Communication Administratively Prohibited (4.0)
| country_src = SE: ICMP Destination Unreachable Communication Administratively Prohibited (3.0)
| country_src = IT: ICMP Destination Unreachable Communication Administratively Prohibited (5.01)
| country_src = AR: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = IE: ICMP Destination Unreachable Communication Administratively Prohibited (7.01/1.0)
| country_src = VA: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = MY: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = NA: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = BE: ICMP Destination Unreachable Communication Administratively Prohibited (6.01/2.0)
| country_src = ZA: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = SK: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (5.01/1.01)
| country_src = CO: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = TR: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)

| country_src = AU: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (36.04/5.04)
| country_src = MK: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (3.0/0.0)
| country_src = MD: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = MZ: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = UA: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = TN: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = PH: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (8.01/0.01)
| country_src = PA: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = CV: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = LU: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = AT: ICMP Destination Unreachable Communication Administratively Prohibited (3.0)
| country_src = SC: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = EC: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = NO: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = NG: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = VE: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = VN: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = CR: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = SI: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = FI: ICMP Destination Unreachable Communication Administratively Prohibited (0.0)
| country_src = LT: ICMP Destination Unreachable Communication Administratively Prohibited (25.03)
| country_src = LV: ICMP Destination Unreachable Communication Administratively Prohibited (17.02)
| country_src = IR: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (4.0/0.0)
| country_src = LK: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (1.0/0.0)
| country_src = AE: ICMP Destination Unreachable Communication Administratively Prohibited (3.0)

| country_src = GR: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (2.0/0.0)
| country_src = SA: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = TH: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (2.0/0.0)
| country_src = UY: ICMP Destination Unreachable Communication Administratively Prohibited (2.0)
| country_src = GH: ICMP Destination Unreachable Communication with Destination Host is Adm. Prohibited (5.01/0.01)
| country_src = AP: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = LA: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
| country_src = ID: ICMP Destination Unreachable Communication Administratively Prohibited (1.0)
sig_class_name = successful-admin: TFTP GET passwd (8.0)
sig_class_name = attempted-admin: TFTP PUT filename overflow attempt (2.0)
sig_class_name = bad-unknown: BAD-TRAFFIC same SRC/DST (24.0)
sig_class_name = port_scan: (portscan) Open Port (100000.0/25.0)

Number of Leaves : 97
Size of the tree : 112

APÊNDICE F: ÁRVORE DE DECISÃO – DADOS INTERNOS

J48 pruned tree

```
------------------
country_dst = US
|   sig_name = BLEEDING-EDGE MALWARE Weatherbug Capture: r.cloudfront.net (14.0/2.0)
|   sig_name = BLEEDING-EDGE MALWARE Weatherbug: amazonaws.com (2.0)
|   sig_name = BLEEDING-EDGE Malware Shop at Home Select Spyware Activity (Bundle)
|   |   hora <= 17: ftp-osl.osuosl.org (1636.0/174.0)
|   |   hora > 17: eos.apache.org (174.0/64.0)
|   sig_name = BLEEDING-EDGE Malware MarketScore.com Spyware User Conf. and Setup: securestudies.com (14700.0/134.0)
|   sig_name = BLEEDING-EDGE Malware Fun Web Products Agent Traffic: securestudies.com (0.0)
|   sig_name = BLEEDING-EDGE Malware MyWebSearch Toolbar Traffic (Agent): securestudies.com (0.0)
|   sig_name = SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_Distribution)
|   |   hora <= 11: ftp-osl.osuosl.org (80.0)
|   |   hora > 11
|   |   |   hora <= 12: eos.apache.org (12.0)
|   |   |   hora > 12: ftp-osl.osuosl.org (66.0)
|   sig_name = SPYWARE-DNS DNS lookup imgfarm.com (Malware_Distribution): securestudies.com (0.0)
|   sig_name = SPYWARE-DNS DNS lookup clicksor.com IMMORTAL (Zeus_Zbot_SpyEye): securestudies.com (0.0)
|   sig_name = SPYWARE-DNS DNS lookup myroitracking.com (Malware_Distribution): securestudies.com (0.0)
|   sig_name = SPYWARE-DNS DNS lookup adsbwm.com (): securestudies.com (0.0)
|   sig_name = SPYWARE-DNS DNS lookup testtaketraffic.com (Malware_Infectors): securestudies.com (0.0)
|   sig_name = BLEEDING-EDGE VIRUS OUTBOUND Suspicious Email Attachment
|   |   hora <= 17: hotmail.com (2.0)
|   |   hora > 17: ye-in-f.e.net (2.0)
```

| sig_name = BLEEDING-EDGE WORM Possible Myfip email incoming - MIME boundary tag: securestudies.com (0.0)
country_dst = NL: aurora-.apache.org (280.0)
country_dst = DE: gimli.documentfoundation.org (222.0)
country_dst = BR
| layer4_dport <= 53: iff.edu.br (99804.0)
| layer4_dport > 53
| | hora <= 14
| | | sig_name = BLEEDING-EDGE MALWARE Weatherbug Capture: anankecdn.net.br (0.0)
| | | sig_name = BLEEDING-EDGE MALWARE Weatherbug: anankecdn.net.br (0.0)
| | | sig_name = BLEEDING-EDGE Malware Shop at Home Select Spyware Activity (Bundle): iff.edu.br (16.0)
| | | sig_name = BLEEDING-EDGE Malware MarketScore.com Spyware User Conf. and Setup : anankecdn.net.br (132.0/62.0)
| | | sig_name = BLEEDING-EDGE Malware Fun Web Products Agent Traffic: anankecdn.net.br (0.0)
| | | sig_name = BLEEDING-EDGE Malware MyWebSearch Toolbar Traffic (Agent): anankecdn.net.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_Distribution): anankecdn.net.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup imgfarm.com (Malware_Distribution): anankecdn.net.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup clicksor.com IMMORTAL (Zeus_Zbot_SpyEye): anankecdn.net.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup myroitracking.com (Malware_Distribution): anankecdn.net.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup adsbwm.com (): anankecdn.net.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup testtaketraffic.com (Malware_Infectors): anankecdn.net.br (0.0)
| | | sig_name = BLEEDING-EDGE VIRUS OUTBOUND Suspicious Email Attachment: anankecdn.net.br (0.0)
| | | sig_name = BLEEDING-EDGE WORM Possible Myfip email incoming - MIME boundary tag: anankecdn.net.br (0.0)
| | hora > 14
| | | sig_name = BLEEDING-EDGE MALWARE Weatherbug Capture: trrsf.com.br (0.0)
| | | sig_name = BLEEDING-EDGE MALWARE Weatherbug: trrsf.com.br (0.0)
| | | sig_name = BLEEDING-EDGE Malware Shop at Home Select Spyware Activity (Bundle): iff.edu.br (22.0)
| | | sig_name = BLEEDING-EDGE Malware MarketScore.com Spyware User Configuration and Setup Access
| | | | hora <= 17: iff.edu.br (6.0)
| | | | hora > 17: trrsf.com.br (106.0/20.0)
| | | sig_name = BLEEDING-EDGE Malware Fun Web Products Agent Traffic: iff.edu.br (8.0)

| | | sig_name = BLEEDING-EDGE Malware MyWebSearch Toolbar Traffic (Agent): trrsf.com.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup msnsolution.nicaze.net (Malware_Distribution): trrsf.com.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup imgfarm.com (Malware_Distribution): trrsf.com.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup clicksor.com IMMORTAL (Zeus_Zbot_SpyEye): trrsf.com.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup myroitracking.com (Malware_Distribution): trrsf.com.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup adsbwm.com (): trrsf.com.br (0.0)
| | | sig_name = SPYWARE-DNS DNS lookup testtaketraffic.com (Malware_Infectors): trrsf.com.br (0.0)
| | | sig_name = BLEEDING-EDGE VIRUS OUTBOUND Suspicious Email Attachment: trrsf.com.br (0.0)
| | | sig_name = BLEEDING-EDGE WORM Possible Myfip email incoming - MIME boundary tag: trrsf.com.br (0.0)
country_dst = PT: www.myway.pt (6.0)