Revista da Graduação

Vol. 6	No. 1	2013	14

Seção: Faculdade de Informática

Título: HAMDROID: FERRAMENTA MOBILE PARA CONTROLE DE INVASÕES

Autor: BRUNO PILZ E CLAUTER GATTI

Este trabalho está publicado na Revista da Graduação.
ISSN 1983-1374
http:// revistaseletronicas.pucrs.br/ojs/index.php/graduacao/article/view/13759

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA

BRUNO PILZ
CLAUTER GATTI

HAMDROID

Ferramenta Mobile para Controle de Invasões

Porto Alegre
2012

BRUNO PILZ
CLAUTER GATTI

HAMDROID

Ferramenta Mobile para Controle de Invasões

Trabalho de conclusão de curso de graduação apresentado à Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Ricardo M. Czekster

Porto Alegre
2012

DEDICATÓRIA

Dedico este trabalho a todas as pessoas que de alguma forma contribuíram para a realização do mesmo, direta ou indiretamente.

Bruno Pilz

Dedico este trabalho a minha esposa, família e amigos, pelo esforço, dedicação e compreensão em todos os momentos desta etapa, tornando possível a realização do mesmo.

Clauter Gatti

AGRADECIMENTOS

Agradeço a minha família, em especial meus pais por terem me proporcionado uma base sólida para a vida e terem me dado a oportunidade de concluir mais esta etapa. Não menos importante, agradeço a todos os meus amigos, pelos momentos de compreensão pela minha ausência, apoio quando necessitei e descontração para que eu pudesse vencer em mais esta etapa da minha vida. Finalmente, obrigado aos orientadores, Leticia Lopes Leite (durante o TCC I) e Ricardo Melo Czekster, que foram fundamentais para o sucesso deste trabalho.

Bruno Pilz

Agradeço a todos aqueles que me apoiaram de alguma forma na realização deste trabalho. Agradeço em especial a minha esposa e companheira Katiandry Rossini Gatti, que esteve sempre me incentivando, com muita compreensão e atenção. Aos amigos e colegas que mesmo sem saber, colaboraram para a melhoria deste. E aos orientadores, Leticia Lopes Leite (durante o TCC I) e Ricardo Melo Czekster, que foram fundamentais para um desenvolvimento de sucesso deste trabalho.

Clauter Gatti

EPÍGRAFE

Quem quer que ache que seu
problema pode ser resolvido usando
criptografia, não entende seu problema e
nem entende de criptografia.

Roger Needham

RESUMO

A segurança torna-se extremamente importante nos dias de hoje à medida que o número de ataques e intrusões a sistemas e redes de computadores aumenta de maneira significativa. Assim, para garantir o aumento da segurança, surgem novas técnicas e ferramentas. Sistema de Detecção de Intrusão (IDS) é um dos elementos essenciais à infraestrutura de segurança, que objetiva identificar tentativas de intrusão de modo que uma resposta apropriada possa ser invocada.

A utilização de dispositivos móveis como *Smartphones* e *Personal Digital Assistants* (PDAs) para acessar dados em qualquer lugar e a qualquer momento vem se tornando cada vez mais comum. Embora tais dispositivos apresentem recursos computacionais limitados, eles encurtam o espaço de tempo para tomada de decisões e são de grande valia para monitoramento e controle de disponibilidade de uma rede de computadores.

Neste trabalho procurou-se demonstrar os principais aspectos relativos à segurança de redes de computadores, os benefícios e limitações da mobilidade e por fim, na proposição de uma ferramenta para dispositivos móveis de controle de invasão e tomada de decisão.

Palavras-chave: Segurança de redes, IDS, Mobilidade e Ferramenta de Controle Remota.

ABSTRACT

Nowadays, security becomes extremely important as attacks and intrusions to systems and computers networks increase significantly. Different measures are used to guarantee security as well as new techniques and tools. Intrusion detection systems (IDS) are one of the elements used against intrusion. It presents an infrastructure whose objective is to identify attempts of intrusion suggesting appropriate measure to be invoked.

The use of mobile devices such as Smartphones and Personal Digital Assistants (PDAs) to access data anywhere and anytime is becoming more common. Such devices have limited computational resources, however, they reduce response time for taking decisions and they are valuable for both monitoring and management of network availability remotely.

In this work we demonstrate the most important aspects of computer network security, the benefits and limitations of mobility, and finally, we propose the project of a tool for mobile devices to enhance decision making and add more control of invasions.

Key-words: Network Security, IDS, Mobility, and Tool for Remote Control.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente, há uma grande necessidade por segurança em sistemas de computadores corporativos. Já é fato que corporações pelo mundo todo têm utilizado a Internet como principal meio de comunicação (CONORICH, 2004). De instituições financeiras a hospitais, muitas corporações se tornaram dependentes desse meio de comunicação, e ainda há casos extremos, como a Amazon e a Google, onde a Internet atua como a única plataforma para os seus negócios. Dessa maneira, esses sistemas, que são tão vitais para o nosso meio de vida e de trabalho, tornaram-se alvos constante de ataques por *hackers* (CONORICH, 2004). O número de ataques a redes de computadores está aumentando, e eles tem se tornado gradativamente mais sofisticados (RAGSDALE, 2000). Segundo o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERTBR), o número de incidentes de segurança reportados no Brasil foi quase 400.000 no ano de 2011.

A segurança dos dados tem impactado os negócios de empresas nos mais diversos ramos de atuação (RAGSDALE, 2000). Devido a este fato, a procura por ferramentas voltadas para a área de segurança tem sido grande. Um bom gerenciamento de segurança é necessário para todos os tipos de sistemas de uma empresa (BAZARA, 2010).

Um conjunto de tentativas que comprometem a segurança de um computador ou de um recurso da rede de computadores é conhecido como invasão (LUNDIN, E; JONSSON, 1999). Sistemas que detectam ataques, tanto externos como internos, em redes computacionais, e que através de medidas se encarregam de eliminar os ataques são conhecidos como *Intrusion Detection Systems* (IDS) (BAZARA, 2010).

Para uma pequena empresa, a necessidade de segurança não é diferente do que foi descrito anteriormente. Uma deficiência dos trabalhos recentes na área de IDS é o foco em ambientes de larga escala, não contemplando administradores de pequenos ambientes de um ou dois servidores.

O fator tempo é um elemento importante, pois quanto mais rápido for detectado um problema, maiores são as chances de minimizar o impacto deste problema na qualidade do serviço da rede. É importante manter um bom grau de resiliência, e para isso a mobilidade auxilia no gerenciamento de redes à distância.

Voltado exatamente para o público de administradores de pequenos ambientes, este trabalho visou desenvolver uma ferramenta de monitoração de ataques a um servidor. Com o apoio de um IDS, a ferramenta permite ao administrador visualizar os alertas de possíveis ataques através de um dispositivo móvel, possibilitando a tomada de decisão à distância.

Mas antes de discutir a proposição da ferramenta, este trabalho abordará conceitos de redes de computadores, requisitos de segurança da rede, suas principais ameaças e problemas, os tipos de ataques que ela pode sofrer e alguns mecanismos de prevenção de ataques. Além de conceitos de rede, este trabalho também abordará os benefícios e as limitações das tecnologias móveis. Ambos os conceitos serão abordados para facilitar o leitor a compreender os motivos que levaram a escolha deste trabalho e também para conceituar tecnologias e ferramentas de apoio utilizadas ao longo do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo, procurou-se descrever os vários aspectos dos temas Segurança de Redes de Computadores e Mobilidade.

Foram citadas definições retiradas de RFCs[1] (*Request for Comments)*. As RFCs são documentos produzidos por professores e pesquisadores nas diversas áreas de tecnologia, que descrevem todos os detalhes das implementações dessas tecnologias, tanto de redes quanto de linguagens, protocolos de comunicações, padrões de funcionamento de hardware e outros.

2.1 REDE DE COMPUTADORES

Tanto no ambiente explícito da informática quanto fora dele, todos nós estamos em contato com algum tipo de rede em maior ou menor grau. As redes de computadores surgiram da necessidade da troca de informações, onde é possível ter acesso a um dado que está fisicamente localizado longe de você (TORRES, 2001).

A RFC 2828 define rede de computadores como uma coleção de *hosts* interligados para troca de dados. *Host* é definido pela mesma RFC 2828 como sendo um computador ligado a uma rede de comunicação que possa usar os serviços providos pela rede para trocar dados com outros sistemas interligados. Não importa o tamanho ou tipo, a definição de rede de computadores é utilizada tanto para a Internet quanto para um simples computador pessoal interligado remotamente como um terminal de outro computador.

As redes de computadores ainda têm a vantagem de permitir o compartilhamento de periféricos, que podem significar uma redução nos custos de equipamentos. A Figura 1 representa um exemplo de compartilhamento de impressora (periférico) que está sendo usado por vários computadores.

[1] Disponíveis em http://www.ietf.org.

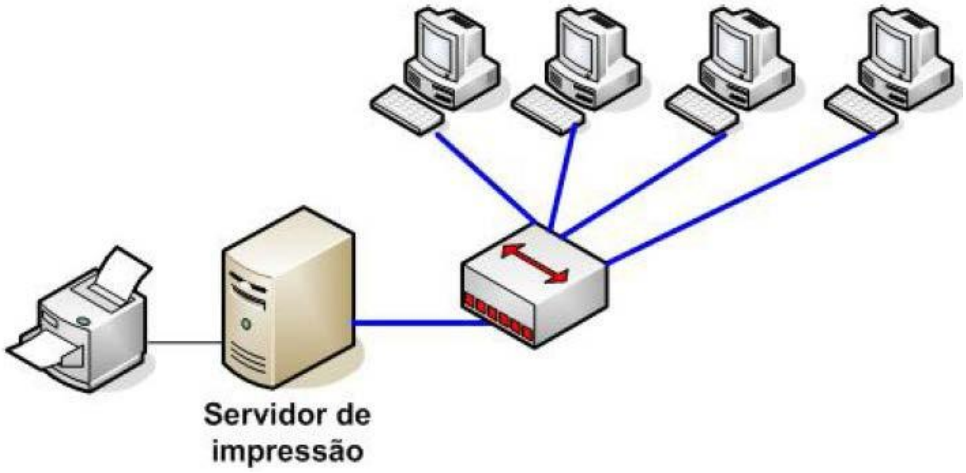


Figura 1: Exemplo de compartilhamento de periféricos.

Segundo Tanenbaum (1994), o termo “rede de computadores” pode ser especificado como um conjunto de computadores autônomos, interconectados, sendo capazes de trocar informações. Essas interconexões podem ser feitas através de fios de cobre, lasers, micro-ondas, satélites de comunicação e também por fibras óticas.

2.2 SEGURANÇA DE REDES DE COMPUTADORES

Não é fácil assimilar que, neste momento, milhares de pessoas estão interconectadas através de uma rede de computadores espalhada pelo mundo e trocando muitas informações, algumas de grande importância, em tempo real.

A popularização deste meio de comunicações revelou "[...] verdadeiras portas de entrada e saída para o mundo - os sistemas passaram a ficar vulneráveis aos ataques externos” (SOUSA e PUTTINI, 2006, p.1). E independente da solução utilizada “[...] nenhuma propriedade física é absolutamente segura contra o crime, nenhuma rede é completamente segura.” (COMER, 2006, p.359).

Mantido pelo Comitê Gestor da Internet no Brasil, O CERTBR é responsável por tratar incidentes de segurança em computadores que envolvam redes conectadas à Internet brasileira. Seu relatório anual de 2011 indicou um aumento significativo no número de incidentes de segurança reportados: 3.017 incidentes foram reportados em 1999 e quase 400.000 no ano de 2011 (Figura 2). Esses

ataques não estão se tornando apenas mais numerosos, mas também mais sofisticados.

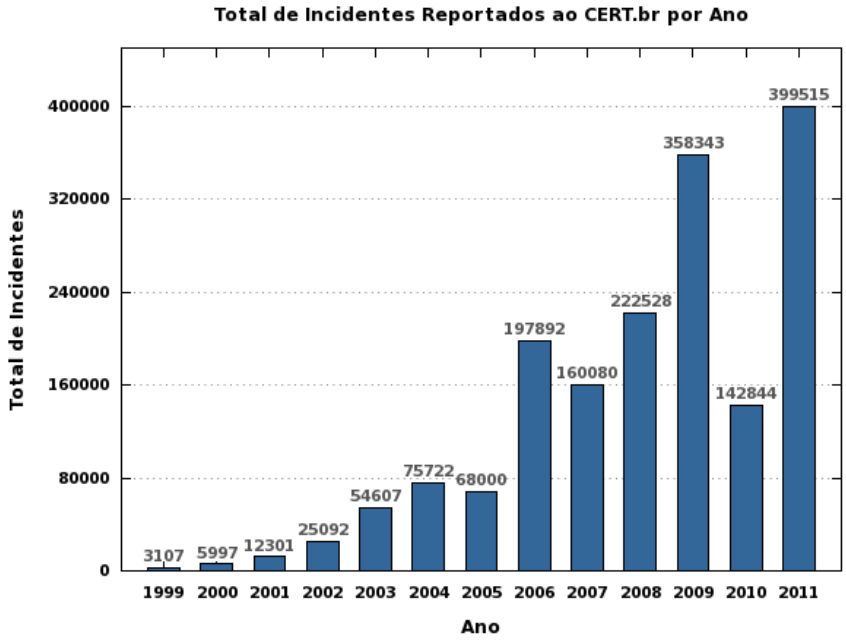


Figura 2: Incidentes Reportados ao CERT.br

Fonte: (CERTBR, 2012)

Para manter uma rede segura é necessário ter consciência da sua exposição, conhecer seus pontos fortes e suas vulnerabilidades. Invasões ou tentativas de ataques são feitas por pessoas capacitadas que possuem um bom conhecimento de programação, arquitetura de redes e muitas vezes, ferramentas avançadas. A partir desse cenário, os atuais bons administradores de rede devem conhecer muito bem as técnicas utilizadas por esses criminosos. Para Schetina e Carlson (2002, p.11), “[...] entender a visão mais ampla da segurança é a chave para a transição de um bom técnico para um verdadeiro profissional de segurança”.

Na próxima seção abordaremos as ameaças e problemas de segurança de redes de computadores.

2.3 AMEAÇAS E PROBLEMAS DE SEGURANÇA DE REDES DE COMPUTADORES

Nesta seção, descrevem-se algumas das ameaças à segurança das redes de computadores e alguns dos agentes que estão ligados a este assunto.

2.3.1 Hackers

O termo hacker é definido pela RFC 2828, como sendo alguma pessoa com interesse e conhecimento em tecnologia, que não tira benefício para si mesmo com eventuais descobertas de falhas de segurança. É comum ver este termo ser usado erroneamente como alguém que efetue crimes cibernéticos. O termo *cracker* ou *intruder*, esse sim, é definido pela mesma RFC 2828 como sendo alguém que tenta quebrar a segurança ou ganhar acesso a sistemas de outras pessoas sem ser convidado.

Com o passar dos anos e somado à evolução da tecnologia, percebe-se uma grande mudança no perfil do *cracker*. No passado, em sua maioria eram especialistas em informática, com alto grau de conhecimento em sistemas operacionais, redes e tecnologia em geral. Atualmente, devido a vários fatores como a grande facilidade de troca de informações pela Internet, o enorme aumento de *hosts* na Internet sem o devido preparo quanto à segurança, o surgimento das ferramentas automatizadas para *hacking* entre outros, percebe-se uma grande diminuição do conhecimento técnico necessário para realizar um ataque ou uma invasão (PELISSARI, 2002). A Figura 3 mostra com mais clareza as tendências atuais do perfil dos *crackers*:

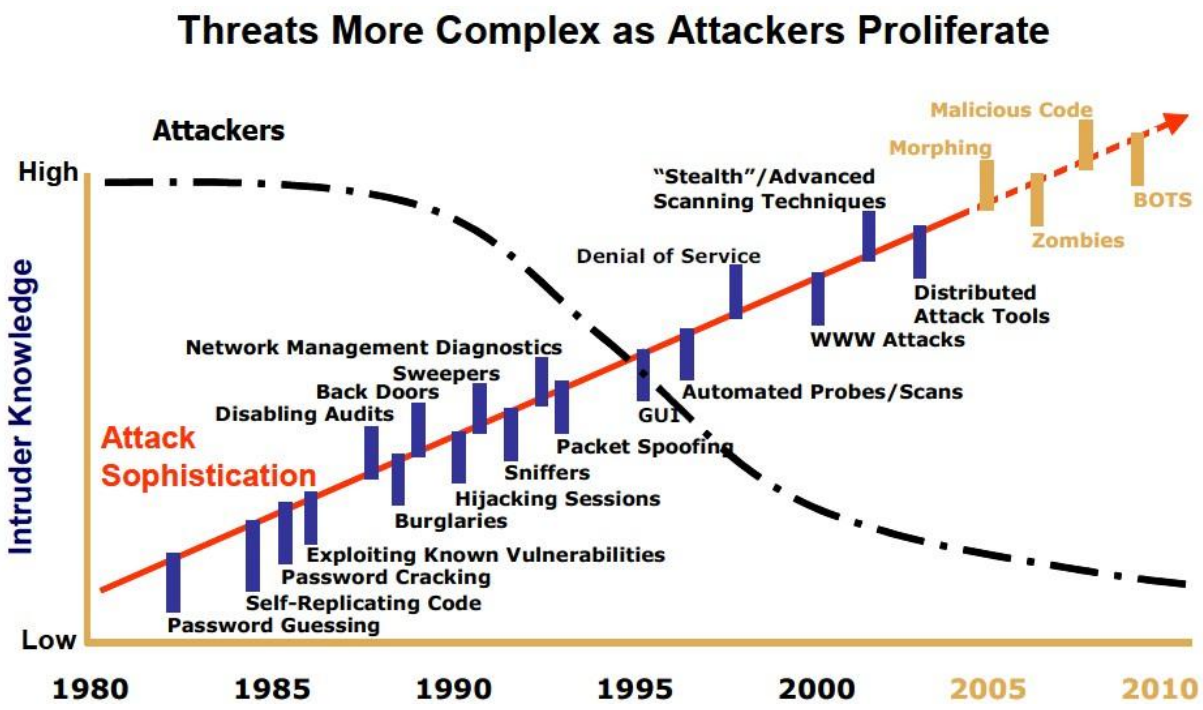


Figura 3: Conhecimento do Intruso x Sofisticação do ataque.
Fonte: Hahn, Guillen e Anderson (2005, p. 4)

Os fatos que levam o *cracker* a começar essa conduta são os mais variados. Pelissari (2002, p. 22) cita:

- **Lazer**: conforme entrevista do *cracker Master System*, 16 anos, afirma à revista Veja: "Gosto de invadir porque não tem nada para fazer em casa", Zakabi (2001).

- **Manter a supremacia do grupo**: Percebe-se uma competição entre os grupos de *crackers*, onde o número de sites invadidos é exibido como um troféu conseguido. Como exemplo, cita-se o grupo *Silver Lords*, que já estaria desfeito, mas alguns *crackers* brasileiros ainda atuam em seu nome para manter a supremacia do grupo, Zakabi (2001). Algo muito parecido com os pichadores de muros, onde os grupos sempre competem para pichar o local de acesso mais difícil ou o maior número de lugares.

- **Roubo de informações**: pode-se citar o exemplo do *cracker Phrozen_Byte*, "Raramente entro por diversão, só a trabalho". Na

entrevista a revista Veja, o *cracker* conta um trabalho a ele encomendado, onde invadiu o site da HP *Store* roubando os cadastros de mais de 1800 clientes, Segundo o *cracker*, "o trabalho foi encomendado por um concorrente", Zakabi (2001).

- **Protesto**: contra aumentos de impostos, contra o governo, contra a supremacia de algum país, por alguma ideologia e outros. Como exemplo cita-se a invasão do grupo *cracker Crime Boys* ao site do Supremo Tribunal Federal, "protestando contra o apagão". Zakabi (2001).

- **Desafio**: como exemplo, pode-se citar a campanha lançada pelo CEO da Oracle, Larry Ellison, no lançamento do Oracle 9i, onde declarou "Os hackers não conseguirão violar nosso site. É assim que nós nos diferenciamos dos fabricantes de games". Após essa declaração, viu o volume de ataques diários a seu site saltar de 3.000 para 30.000 por dia. (INFO EXAME, 2002).

2.3.2 Ataques e Incidentes a Redes de Computadores

Segundo Sofron e Tutanescu (2003, p. 266): "a singular tentativa de obter acesso não autorizado a um computador ou em uma rede de computadores é chamado ataque". Outro autor, Anderson (2008, p. 367) define um ataque à rede como “qualquer método, processo ou meios utilizados maliciosamente para a tentativa de comprometer a segurança da rede”.

Sofron e Tutanescu (2003, p. 266) definem incidente como:

> “O incidente é constituído por um conjunto de ataques, que são caracterizados pela existência de agressores e agredidos locais, e que utilizam de técnicas específicas e métodos de ataques sincronizados”.

Um ataque ou incidente de segurança à rede de computadores é definido como uma ameaça, intrusão, negação de serviço, ou ataque sobre uma infraestrutura de rede que irá analisar a sua rede e obter informações para desequilibrar ou corrompe-la. Em muitos casos, o intruso poderá não só estar

interessado em explorar aplicações, mas também de tentar obter acesso não autorizado aos dispositivos de rede. Dispositivos de rede sem ou com fraco controle são a principal fonte de vazamento de informações nas organizações. Na maior parte das organizações, cada mensagem de correio eletrônico, cada requisição de página web, cada *login* de usuário, ou troca de arquivo são manipulados por um dispositivo de rede. Há também serviços telefônicos e mensagens de voz que são manipuladas por dispositivos de rede. Se o intruso for capaz de apropriar de um único dispositivo da rede, ele poderá comprometer a rede toda. Ataques à rede acontecem em todas as categorias de *software* e tipos de plataforma.

Existem duas principais categorias de ataques: ataques passivos (interceptação de dados) e ataques ativos (interrupção do fluxo de dados, modificação e envio de dados “infectados”) (SOFRON E TUTANESCU, 2003). Veja os exemplos na Figura 4:

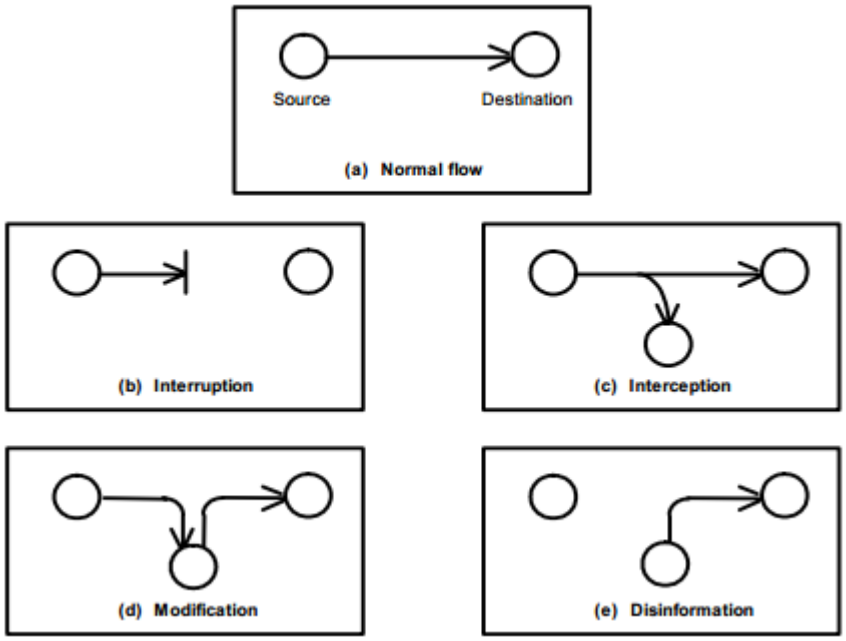


Figura 4: Fluxo normal (a), ataques passivos (c) e ativos (b, d, e).

Fonte: Sofron e Tutanescu (2003, p. 266).

Ataques passivos são caracterizados por:

- Eles violam a regras de confidencialidade.
    - Eles não causam danos (não excluem ou modificam dados).

- Transmissões de dados são interceptadas em redes sem fio, de rádio e outros.

Os ataques ativos são mais perigosos, porque eles modificam o *status* dos dados, computadores ou sistemas de comunicação (SOFRON E TUTANESCU, 2003). Os principais tipos de ataques ativos são:

- **Interceptação**: usam respostas de uma mensagem ou parte de uma mensagem para produzir um acesso não autorizado.

- **Modificação**: representa um ataque que modifica (através de inserção e/ou exclusão de caracteres) uma parte ou todo o dado transmitido.

- **Desinformação**: representa o tipo de ataque onde um usuário não autorizado finge ser um usuário autorizado. Por exemplo, um usuário que tenta substituir outro com a intenção de roubar um dado secreto.

2.3.3 Tipos de Ataques a Redes de Computadores

Anderson (2008, p. 367) diz que os tipos mais comuns de ataques estão relacionados com as vulnerabilidades mais comuns existentes:

> "Muitos dos ataques atuais envolvem combinações de vulnerabilidades. Exemplos de vulnerabilidade incluem *stack overflow* (onde você passar um parâmetro excessivamente longo para um programa que por descuido executa parte dele) e adivinhação de senhas, ambos estes tipos são usados nos tipos de ataque de Internet *worm"*.

O autor Anderson (2008, p. 368) ainda complementa seu raciocínio com o seguinte exemplo:

> "Uma estratégia comum é obter uma conta em qualquer máquina em uma rede visada, depois instalar um *password sniffer* para obter uma conta na máquina de destino, e então usar um *stack overflow para* modificar a senha do usuário *root"*.

Outro autor, Khayam (2010, p. 6) discute que os principais tipos de ataques estão evoluindo e mudando à medida que a velocidade da Internet e o poder de

processamento têm aumentado: “O aumento das conexões de Internet e de poder de processamento tem mudado os tipos de ataque, tornando-os mais eficazes, automatizados e difundidos”.

Como os fabricantes de *software* estão sempre trabalhando para liberar *patches* de correções de vulnerabilidades, e a velocidade da Internet e o poder de processamento têm aumentado relativamente rápidos, os tipos de ataques mais utilizados estão mudando constantemente.

Nessa seção teremos uma junção dos tipos mais comuns e mais usados de ataques listados pelos seguintes autores e fontes: Anderson (2008, p. 367), Khayam (2010, p. 6), Liska (2002, p. 210), Larson e Cockroft (2003) e OWASP (2010).

2.3.3.1 ***Sniffing* e *Port Scanning***

O primeiro passo em qualquer ataque bem sucedido é o *Sniffing*, utilizado para ver qual tipo de tráfego que está sendo transmitido em uma rede e também para acessar outros tipos de dados como senhas, números de cartões de crédito, e assim por diante. *Sniffing* é o termo geralmente utilizado para monitorização do tráfego de dentro de uma rede, já o *Port Scanning* é utilizado para descobrir informações sobre uma rede remota (LISKA 2002).

Ambos o *Sniffing* e o *Port Scanning* têm o mesmo objetivo de encontrar vulnerabilidades no sistema, contudo têm diferentes abordagens. *Sniffing* é usado em um ataque por alguém que já esteja dentro da rede e que quer coletar mais informações sobre a mesma (Figura 5). *Port Scanning* é usado por alguém que está interessado em encontrar vulnerabilidades em um sistema que ele não conhece.

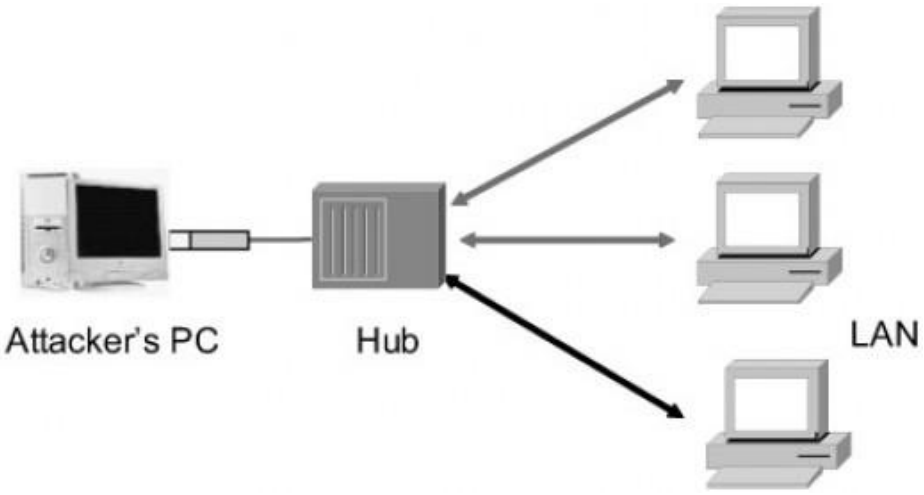


Figura 5: Exemplo de *Sniffing*.
Fonte: Liska (2002, p. 211).

Há várias ferramentas disponíveis para *Sniffing*, dentre delas está o *Snort*[2], que foi desenhada para ajudar a descobrir problemas de vulnerabilidade, mas também é usada para monitorar o tráfico da rede.

Se o pacote não for criptografado será possível ler as informações contidas nele. Ainda é possível monitorar as sequências de pacotes para ter acesso a toda a informação de uma transação.

Já para o *Scanning*, a ferramenta mais comum usada é a *nmap*[3]. Ela permite os seus usuários a entrarem com uma faixa de IP (*Internet Protocol*), escolher o tipo de *scan* desejado, e permite que o programa rode em segundo plano. Quando a execução termina, a ferramenta apresenta um relatório (Figura 6), mostrando as portas que responderam em cada dispositivo da rede.

[2] Disponível em http://www.snort.org.

[3] Disponível em http://www.insecure.org/nmap.

```
[root@test root]# nmap -sT www.datacenterwire.com
Starting nmap V. 2.99RC2 ( www.insecure.org/nmap/ )
Interesting ports on (66.150.201.102):
(The 1589 ports scanned but not shown below are in state: closed)
Port State Service
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
53/tcp open domain
80/tcp open http
81/tcp open hosts2-ns
110/tcp open pop-3
443/tcp open https
587/tcp open submission
3306/tcp open mysql
5432/tcp open postgres
10000/tcp open snet-sensor-mgmt
Nmap run completed -- 1 IP address (1 host up) scanned in 48 seconds
```

Figura 6: Exemplo de relatório do *nmap*.

Fonte: Liska (2002, p. 212).

Uma porta pode ter dois estados: fechada e aberta (*listening*/escutando), porém o *nmap* utiliza uma abordagem própria para identificar os estados das conexões. O *nmap* pode identificar as conexões como *open* (aberta), *closed* (fechada), *filtered* (filtrada), *unfiltered* (sem filtro), *open*|*filtered* (aberta|filtrada). O *nmap* classifica cada porta com esses status da seguinte maneira:

- ***Open* (aberta ou *listening*)**: quando realmente existe uma aplicação escutando na porta.

- ***Filtered* (filtrada)**: quando o *nmap* não consegue determinar o estado de conexão, pois algum *firewall* ou filtro está implementado no meio do caminho (roteador ou *firewall*) ou há algum problema na rede.

- ***Closed* (fechada)**: quando uma porta está fechada o *nmap* recebe e resposta um pacote do tipo *restart* (RST) que nenhuma aplicação possui uma porta aberta nesta porta.

- ***Unfiltered* (sem filtro)**: quando feito com um scan com o tipo de pacote *acknowledgement* (ACK). Neste caso o *nmap* não sabe dizer se a porta está aberta ou fechada, isso ocorre, pois a resposta recebida é um pacote do tipo RST, resposta quando uma porta está fechada e

quando uma porta recebe o pacote do tipo ACK, porém não há uma pré-conexão, logo não há como saber o estado da porta.

- ***Open|filtered*** **(aberta|filtrada)**: quando o *nmap* não consegue identificar se a porta está aberta ou filtrada, pois não há resposta. Neste cenário pode haver algum problema na rede, filtro de pacote, ou alguma regra de *firewall* não permitindo a resposta.

- ***Closed|filtered*** **(fechada|filtrada)**: quando o *nmap* não consegue identificar se a porta está fechada ou filtrada pois não há resposta. Este estado só ocorre com o *Idle Scan*.

Tendo acesso à lista de portas e nomes de servidores que estão abertas, o próximo passo é tentar explorar as fraquezas das diferentes possíveis configurações do servidor. Essa tarefa envolve o conhecimento das fraquezas de diferentes tipos de servidores (LISKA 2002).

2.3.3.2 IP *Spoofing*

Larson e Cockroft (2003) definem “Um ataque de *IP Spoofing* é um *hacker*, tanto interno ou externo da rede, que finge ser um computador confiável por estar usando o endereço IP de um computador da rede”. Neste tipo de ataque o hacker ou usa um endereço de IP dentro da faixa de endereços internos confiáveis da rede, ou um endereço externo confiável com acesso a alguns recursos específicos da rede. Segundo Liska (2002, p. 212), IP *Spoofing* é uma ferramenta comum usada como parte de outros tipos de ataques, como por exemplo, em algumas derivações do DoS (*Denial of Service*), onde ele esconde a identidade do *hacker*.

Ataques DoS baseados em IP *Spoofing* são relativamente simples. Um *hacker* envia um pacote a um *host* alvo com um endereço de IP forjado (SYN) – normalmente é um endereço de IP dentro da faixa especificada pela RFC 1918. O *host* alvo envia um pacote do tipo ACK e espera pela resposta. A resposta nunca é enviada, e a consulta não respondida fica mantida no buffer do dispositivo alvo. Isso

pode levar a um *buffer overflow* e o dispositivo de rede poderá vir a ficar instável ou cair (LISKA 2002).

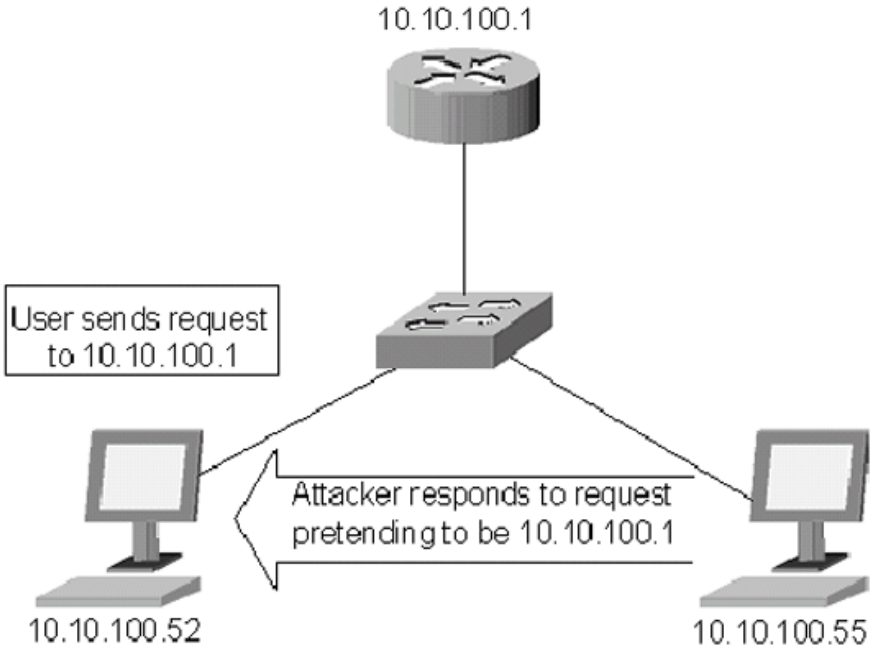


Figura 7: Exemplo de um ataque *Spoofing*.

Fonte: Liska (2002, p. 212).

A Figura 7 ilustra um ataque de *Spoofing*. O usuário envia um pedido à 10.10.100.1. O atacante finge ser 10.10.100.1 e envia uma resposta para esse usuário. Em seguida, o usuário encaminha todos os pacotes destinados ao 10.10.100.1 para o atacante.

Segundo Larson e Cockroft (2003) “os ataques de IP *Spoofing* podem ser reduzidos, mas não eliminados”. As medidas para deduzi-los são:

- **RFC 2827 *filtering***: basicamente como “melhores práticas” temos que nunca um IP privado, de uso específico ou seu próprio IP, deve ser aceito como tráfego *inbound* na *interface outside* de um roteador conectado a Internet.

- **RFC 1918 *filtering***: pacotes originários do mundo exterior com origem de redes privadas (endereços internos previstos na RFC 1918 e rede 127) devem ser bloqueados.

- **Non-IP *address authentication***: *IP Spoofing* é útil quando os dispositivos utilizar autenticação baseada em endereço IP. Se você utilizar outros métodos de autenticação, o IP *Spoofing* perde muito do seu valor.

2.3.3.3 ***Exploits Atacks***

Um *Exploit* permite que um intruso tire vantagem das falhas conhecidas dos sistemas operacionais ou aplicações para obter acesso a um servidor. O *Exploit* pode ser realizado de diversas formas, no entanto, é cada vez mais comum ele seja parte de uma aplicação pode ser facilmente apontado em qualquer servidor (LISKA, 2002).

Em Junho de 2002 uma falha séria de segurança foi encontrada no servidor web Apache, utilizado por mais de 18 milhões de sites ao redor do mundo. Este orifício de segurança pegou muitas pessoas desprevenidas, e dada a grande base instalada, o tempo para enviar um patch de segurança é considerável. Um bom atacante ciente disto irá verificar se o servidor está vulnerável.

2.3.3.4 **SQL *Injection***

É um ataque contra o banco de dados via web site. Nesse ataque, os *crackers* executam comandos não autorizados de SQL ao aproveitar sistemas inseguros que estão conectados na Internet. O SQL *Injection* é a primeira mais comum vulnerabilidade em aplicações web, de acordo com relatório da OWASP (2010).

Um cracker injeta a *query* SQL na aplicação usando um navegador web comum. O objetivo é injetar linguagem SQL maliciosa dentro do que a aplicação usa para fazer uma consulta no banco de dados. Tudo o que o criminoso precisa é de um navegador web, conhecimento em queries SQL e um trabalho criativo de adivinhação para saber nomes de tabelas e de campos. O *cracker* pode ler dados sensíveis do banco de dados e modificá-los. Além disso, ele pode realizar operações como o fechamento do Banco de Dados (BD) e potencialmente enviar comandos diretamente para o sistema operacional. (COMPUTERWORLD, 2012).

Neste tipo de ataque, dados importantes podem ser modificados ou enviados para outros computadores longe da empresa. Os *crackers* também podem usar as SQL *Injection* para conectar nos sistemas corporativos ou públicos como se fosse um usuário autorizado, driblando a necessidade de uma senha (COMPUTERWORLD, 2012).

O OWASP (2010) ilustra um ataque de SQL *Injection* com o seguinte exemplo: Em uma aplicação que usa dados não confiáveis na construção da seguinte chamada de SQL vulnerável:

```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") +"'";
```

O atacante modifica o parâmetro 'id' em seu browser e envia: *' or '1'='1*. Esta mudança significa que a consulta ira retornar todos os registros de contas do banco de dados, ao invés de retornar somente a do cliente em questão:

```
http://example.com/app/accountView?id=' or '1'='1
```

No pior caso, o atacante pode usar de uma vulnerabilidade para invocar *stored procedures* do banco de dados, permitindo assim o completo domínio do banco de dados do servidor.

É interessante evitar mensagens de erros detalhadas que podem dar pistas para os criminosos (COMPUTERWORLD, 2012). Use bibliotecas disponíveis nas diversas linguagens de programação que fazem o *parse* dos dados, de acordo com o tipo e tamanho esperado.

2.3.3.5 ***Cross-Site Scripting (XSS)***

O XSS, ou *Cross-Site Scripting* é a segunda mais comum vulnerabilidade em aplicações web, de acordo com relatório da OWASP (2010). O mesmo órgão define esse tipo de ataque como:

> “XSS é a mais prevalente falha em aplicações web. As falhas de XSS ocorrem quando uma aplicação inclui dados fornecidos dos usuários da página no navegador e envia-os sem validar adequadamente o conteúdo, os deixando serem desviados”.

A detecção da maior parte das falhas XSS é bastante fácil via teste ou análise de código.

O OWASP (2010) ilustra um ataque XSS com o seguinte exemplo: uma aplicação usa dados não confiáveis na construção da seguinte parte de um HTML, sem validação:

```
(String) page += "input name='creditcard' type='TEXT`
value='" + request.getParameter("CC") + "'";
```

O atacante modifica o parâmetro ‘CC’ em seu próprio *browser* para:

```
')(script)document.location=
'http://www.attacker.com/cgi-
bin/cookie.cgi?foo='+document.cookie(/script)'.
```

Esta modificação faz com que o ID da sessão da vítima seja enviado para o site do atacante, permitindo que ele rapte a sessão corrente do usuário.

Vulnerabilidades XSS tem sido encontradas em todos os tipos de site, até mesmo no Fbi.gov, Yahoo.com, ebay.com e muitos outros sites. Muitos administradores de sites falham em não dar a devida atenção a esse tipo de ataque, talvez o motivo seja que eles ainda conhecem os ataques XSS ou porque não o veem como um perigo em potencial (SEGURANÇA DIGITAL, 2012).

2.3.3.6 ***Denial of Service* (DoS) e *Distributed Denial of Service* (DDoS)**

Ataques *Denial of Service* (DoS) e *Distributed Denial of Service* (DDoS) visam indisponibilizar serviços oferecidos por servidores de redes, como Web, correio eletrônico ou servidor de nomes (DNS). Podem indisponibilizar, retirar o servidor de operação, ou deixar lento a tal ponto que o usuário abandone o serviço por causa da demora no tempo de resposta (CIL, 2006).

Segundo OWSAP (2008, p. 1): “DoS é focado em fazer um recurso se tornar indisponível (websites, aplicações, servidores) para o efeito que este foi concebido.”

Um ataque *Distributed Denial of Service* (DDoS), é um ataque em larga escala, utilizando dezenas, centenas ou milhares de sistemas ao mesmo tempo no ataque de um ou mais alvos selecionados (CIL, 2006).

Em um típico ataque DDoS, o atacante (*Attacker*) monta um exército que é composto por mestres (*Masters*) e escravos zumbis (*Slaves*). Os exércitos de ambas as categorias ficam em máquinas infectadas por códigos maliciosos. O atacante coordena os seus mestres zumbis e eles, por sua vez, coordenam os seus escravos zumbis. Mais especificamente, o atacante envia um comando de ataque ao mestre zumbi e este ativa todos os processos das suas máquinas escravas, que estão em hibernação, esperando pelo comando adequado para acordar e começa a atacar. Em seguida, os mestre zumbis, através desses processos, enviam comandos de ataque aos escravos zumbis, ordenando-lhes para montar um ataque DDoS contra a vítima. Dessa forma, os agentes das máquinas escravas zumbis começam a enviar um grande volume de pacotes para a vítima, encharcando seu sistema com carga inútil e desgastante aos seus recursos. A Figura 8 mostra esse tipo de ataque DDoS.

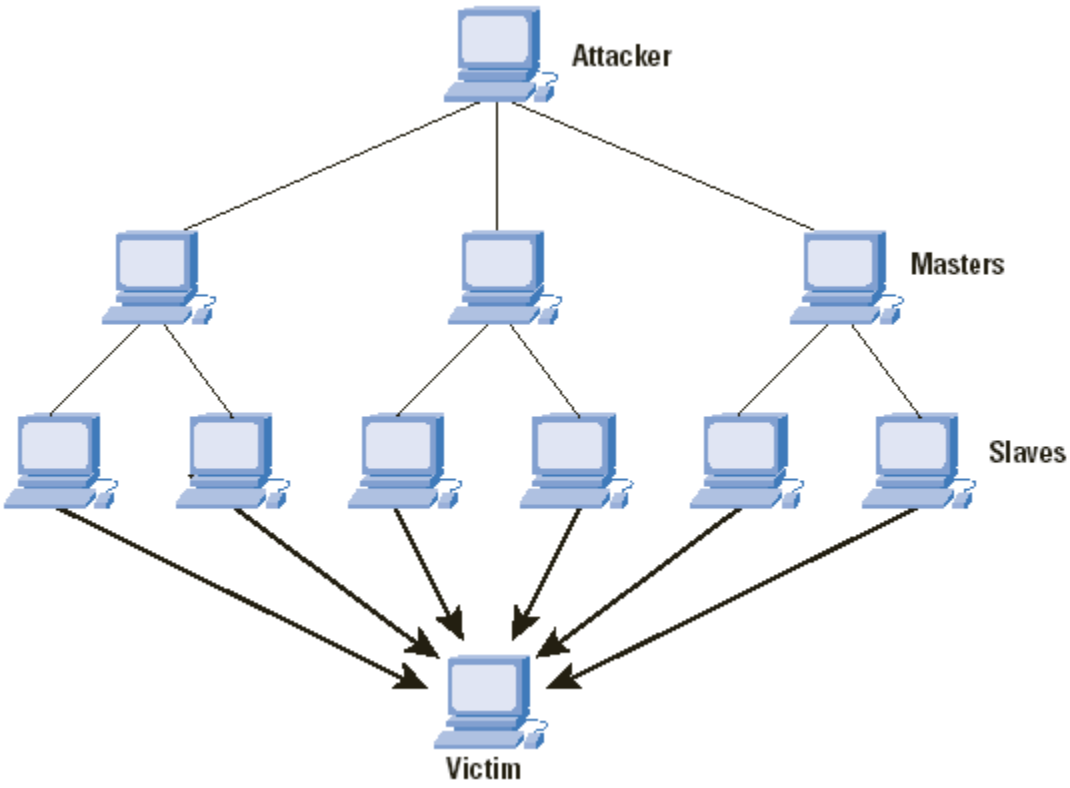


Figura 8: Ataque DDoS.

Fonte: THE INTERNET PROTOCOL JOURNAL (2012)

Existem ferramentas que automatizam os ataques, tais como TFN (*Tribe Flood Network*), Trinoo, stacheldraht e o TFN2000 (TFN2K).

Um ataque desse tipo é difícil de identificar, ainda mais se for um ataque distribuído vindo de diferentes pontos. É impossível impedir todos os ataque DoS, mas existem algumas precauções que os administradores de rede podem tomar para reduzir a incidência desses ataques. Por exemplo, desativar a resposta ICMP evitando um ataque de *Sniffing* ou, configurar o roteador para filtrar números IP externos iguais a números IP internos (ou vice-versa), evitando assim um ataque TFN (CIL, 2006).

2.3.3.7 ***Hijacking* ou *Man In The Middle (MITM)***

Também conhecido como ataque *Man In The Middle (MITM)*, o sequestro de sessões (*Hijacking)* engana um servidor ou cliente fazendo-os aceitar o *host upstream* como o *host* legítimo. Khayam (2010, p. 8) define o MITM como:

> *"Man-in-the-middle* (MITM) é um ataque que acontece quando um atacante é capaz de ler, inserir e modificar mensagens entre duas partes, sem que qualquer das partes saiba que o link entre eles tenha sido comprometido".

O MITM é um tipo de ataque que intercepta uma comunicação entre dois sistemas. Por exemplo, em uma transação HTTP (*Hypertext Transfer Protocol*) o alvo é a conexão TCP entre cliente e servidor. Por meio de diferentes técnicas, o intruso divide a conexão TCP original em duas novas ligações, uma entre o cliente e o agressor e o outro entre o atacante e o servidor, conforme indicado na Figura 9. Uma vez que a conexão TCP é interceptada, o atacante atua como um *proxy*, sendo capaz de ler, inserir e modificar os dados da comunicação capturada (OWASP, 2009).

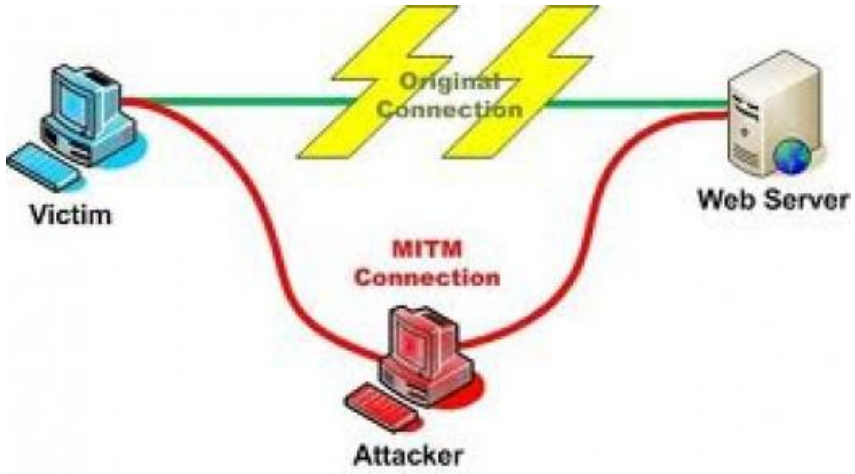


Figura 9: Ataque *Man In The Middle* (MITM)

Fonte: OWASP (2009).

O tipo de ataque MITM é muito eficaz devido à natureza do protocolo HTTP que tem sua transferência de dados baseado em ASCII (*American Standard Code for Information Interchange*). Desta forma, é possível visualizar e interver a transferência de dados do protocolo HTTP.

O ataque MITM também poderia ser feito através de uma ligação HTTPS (*Secure Hypertext Transfer Protocol*), utilizando a mesma técnica, a única diferença consiste na criação de duas sessões SSL independentes (uma sobre cada ligação TCP) (OWASP, 2009).

MITM não é apenas uma técnica de ataque, mas também é comum utilizar durante alguma etapa de desenvolvimento de um aplicativo web e para avaliações de vulnerabilidade.

Alguns exemplos de ferramentas que realizam este tipo de ataque são:

- PacketCreator[4]
- Ettercap[5]
- Dsniff[6]

[4] Disponível em http://www.colasoft.com/packet_builder.

[5] Disponível em http://ettercap.sourceforge.net.

[6] Disponível em http://monkey.org/~dugsong/dsniff.

2.3.3.8 Vírus de Computador

Vírus são códigos escritos com a intenção de se autoduplicar, tentando se alastrar de computador para computador se incorporando a um programa hospedeiro podendo danificar hardware, software ou arquivos (CIL, 2006). Segundo Sofron e Tutanescu (2003, p. 267): "vírus são pequenos programas inseridos em arquivos, que se autoduplicam".

O computador é infectado através de uma porta de entrada, que pode ser um site na Internet, um programa de correio eletrônico, um CD-ROM, USB *flash driver*, disquete, uma máquina infectada ligada à rede local, e outros (CIL, 2006).

Existem muitos tipos de vírus. Vírus que infectam somente alguns tipos de arquivos, que comprometem o desempenho do computador, que destroem o HD (Hard Disk), que impedem o acesso a Internet, que apagam arquivos, que se enviam através da lista de endereços de *e-mails* para infectar outras máquinas ou que vem em forma de um programa ou arquivo para solucionar algum tipo de problema do computador (CIL, 2006).

2.3.3.9 *Worm de Computador*

Assim como um vírus, um *Worm* cria cópias de si mesmo de um computador para outro automaticamente. Inicialmente ele controla recursos do computador que permitem o envio de arquivos ou informações. Depois que o *worm* contamina o sistema ele se desloca sozinho. O grande perigo dos *worms* é a sua capacidade de se replicar em grande volume. Por exemplo, um *worm* pode enviar cópias de si mesmo a todas as pessoas da agenda de endereços, e os computadores dessas pessoas passam a fazer o mesmo, causando um efeito dominó de alto tráfego de rede. Esse efeito pode prejudicar o funcionamento de uma rede tornando-a lenta. Quando novos *worms* são lançados, eles se alastram muito rapidamente, obstruem redes e fazem com que os usuários tenham que esperar um tempo maior para abrir páginas na Internet (CIL, 2006).

Worms podem consumir memória, o que pode levar a degradação de *performance* do computador infectado. Como os *worms* não precisam de um

programa ou arquivo "hospedeiro", eles também podem se infiltrar e permitir que outras pessoas controlem o computador infectado remotamente (CIL, 2006).

2.3.3.9 ***Trojan Horses***

Semelhante ao mitológico cavalo de Tróia, onde aparentemente os troianos estavam recebendo um presente, mas que na verdade este escondia soldados gregos em seu interior com o objetivo de tomar a cidade de Tróia, os *Trojan Horses*, da atualidade são programas de computador que parecem ser úteis, mas na verdade comprometem a sua segurança e causam muitos danos (CIL, 2006).

OWSAP (2008, p. 1) define o *Trojan Horses* como:

> Um Trojan Horse é um programa que utiliza código malicioso disfarçado como uma aplicação fidedigna. O código malicioso pode ser injetado em aplicações benigno, links no corpo de e-mails, ou às vezes escondido no JavaScript de páginas da Internet.

O *Trojan Horse* é instalado por seu usuário que busca alguma funcionalidade enganosamente oferecida por ele, contudo na verdade o seu principal objetivo é introduzir códigos maliciosos. Eles alastram quando as pessoas são seduzidas a abrir o programa por pensar que vem de uma fonte legítima. Eles não infectam outros arquivos. Se for encontrado na máquina um arquivo infectado provavelmente não é uma infecção, mas o próprio *Trojan*.

Os *Trojan Horses* também podem ser incluídos em software que você baixa gratuitamente. O OWSAP (2008, p. 1) aponta os seguintes tipos de Trojan como sendo os principais:

- ***Remote Access Trojan* (RAT)**: Provê total controle ao atacante à máquina infectada. Geralmente é mascarado como sendo utilitário.

- ***Data Sending Trojan***: Utilizado para capturar dados importantes como senhas, números de cartão de crédito e mensagens instantâneas. Uma vez capturado, os dados são enviados para o atacante.

- ***Destructive Trojan***: Destrói dados armazenados no computador da vítima.

- ***Proxy Trojan***: Usa o computador da vítima como um *proxy*, provendo ao atacante a oportunidade de executar atos ilícitos a partir do computador infectado, como fraudes bancárias.

- **FTP *Trojan***: Habilita a porta 21 da máquina da vítima, permitindo acesso via FTP.

- ***Security software disabler Trojan***: Desabilita os softwares de segurança da máquina da vítima, como firewall e antivírus.

- ***Denial-of-Service attack Trojan***: Permite ao atacante a oportunidade de realizar ataques de *Denial-of-Service* a partir da máquina da vítima.

2.4 IDS – INTRUSION DETECTION SYSTEM

Após termos abordados as principais ameaças e problemas de segurança de redes, vamos nesta seção abordar um recurso de detecção de ataques a uma rede de computadores.

A RFC 2828 define o termo detecção de intrusos como sendo um serviço que monitora e analisa eventos de uma rede com o propósito de encontrar e providenciar alertas em tempo real a acessos aos recursos da rede de maneira não autorizada. Ou seja, pode ser definido como um programa ou sistema, que está constantemente, em segundo plano, de maneira imperceptível para o usuário comum, monitorando o tráfego de uma rede de computadores a procura de indícios de invasões. Se acha-las, aciona as rotinas pré-definidas pela empresa a fim de inibir tal acesso.

Cruz (2001) complementa, definindo que uma intrusão pode ser “considerada qualquer conjunto de ações que tentem comprometer a integridade, confidencialidade ou disponibilidade dos dados e/ou sistema”. Ele também divide as intrusões em duas classes principais:

- I**ntrusões devido a mau uso do sistema**: são os ataques realizados a pontos fracos (conhecidos) do sistema. Podem ser descobertas através de comparações com padrões já estabelecidos;

- I**ntrusões devido à mudança de padrão**: são detectadas observando variações de uso em relação ao padrão normal de uso do sistema. Os padrões podem ser de utilização de CPU, número de conexões por minuto, número de processos por usuário, número de conexões, volume de dados trafegando no segmento de rede, entre outros. Uma variação significativa nesses padrões pode indicar uma invasão.

O mesmo autor define algumas características desejáveis nos IDS:

- Funcionar continuamente sem interação humana e ser segura o suficiente para permitir a sua operação em segundo plano;
- Ser tolerante a falhas, ou seja, a sua base de conhecimento não deve ser perdida em caso de falha do sistema operacional;
- Monitorar a si próprio evitando qualquer mudança na sua base;
- Causar o mínimo de impacto no funcionamento do sistema;
- Detectar mudanças no funcionamento padrão;
- Ser difícil de ser enganado;
- Detectar o menor número possível de Falsos Positivos, classificando uma ação legal como uma possível intrusão;
- Não permitir Falso Negativo (ocorre quando uma intrusão real acontece, mas o sistema a classifica como legítima);
- Não permitir a Subversão (ocorre quanto o intruso modifica a operação da ferramenta de IDS para forçar a ocorrência de falso negativo).

Allen, Christie e McHugh (2000) descrevem na Figura 10 o esquema de funcionamento de um IDS:

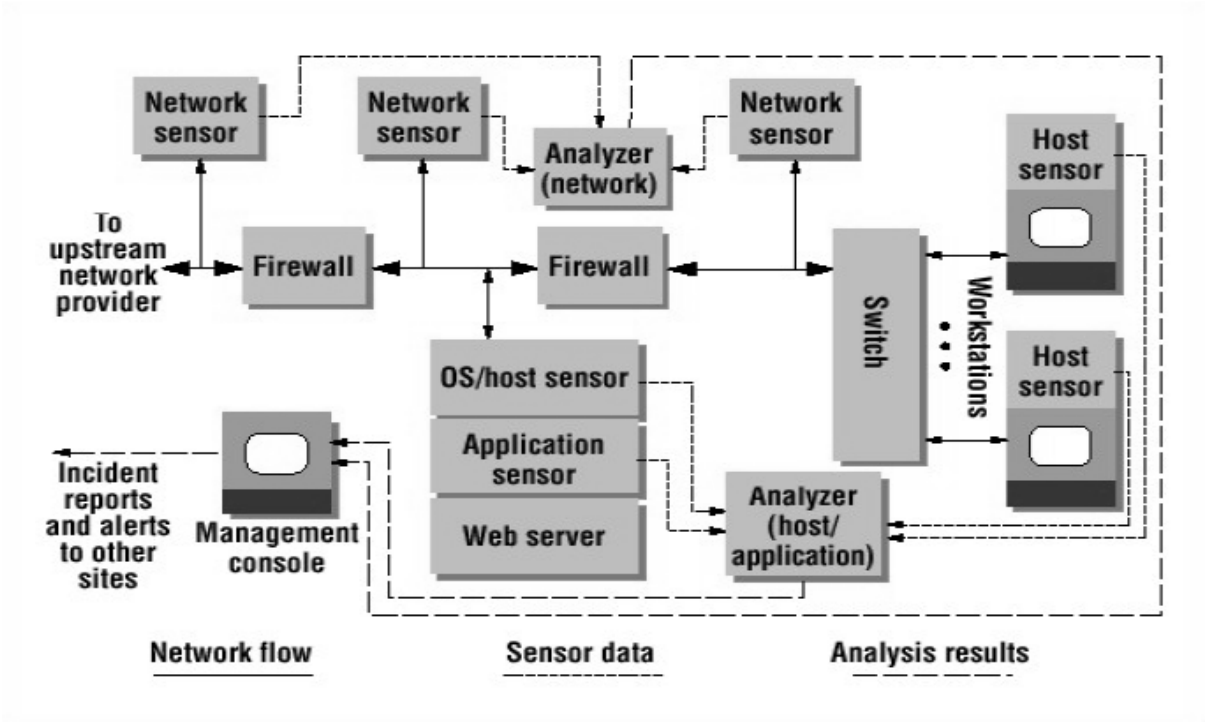


Figura 10: Esquema de um IDS.

Fonte: Allen, Christie e McHugh (2000).

Os mesmos autores descrevem que os IDS podem ser usados para vários propósitos:

- Gerar alarmes ao administrador de rede ou do sistema, em tempo real, sobre uma possível invasão;

- Disparar automaticamente mecanismos de segurança contra essa qualquer eventual suspeita. Exemplos:
  - Sistema poderia ser programado para “matar” o processo suspeito automaticamente;
  - Reconfigurar o *firewall* para impedir a tentativa de ataque;
  - Disparar um mecanismo de contra ataque ao suposto invasor;
  - Colher informações de intrusos para sua captura;
  - Diagnosticar e corrigir eventuais falhas de segurança.

Existem duas áreas a serem inspecionadas pelo IDS chamadas de *Host Based* (Baseada no hóspede) e *Network Based* (Baseada na rede).

O *Host Based* coleta dados locais encontrados em nível de sistema operacional para avaliar possíveis ataques de origem interna. Por sua vez, os dados encontrados possuem uma alta qualidade nas informações providas pela fonte, porém é ineficaz quando se trata de intrusões distribuídas. Neste caso, cada máquina possui vestígios de intrusão que somente serão flagradas se agrupadas e combinadas as informações.

O *Network Based* foca-se no tráfego da rede monitorando cada pacote recebido e enviado. Ao contrário do *Host Based,* tem maior facilidade de correlacionar as informações obtidas de cada máquina com base nas suas atividades na rede. Esta aplicação de segurança tem problemas de escalabilidade quando há uma carga muito elevada para se processar e quando são utilizados dados criptografados.

Alguns requisitos são importantes para o funcionamento do IDS no sistema, tais como: monitoramento e registro de intrusões de forma contínua, baixa taxa de alarmes falsos, adaptabilidade à topologia da rede e alterações de configuração, e escalabilidade para lidar com cargas computacionais elevadas. Além destes requisitos, desempenho também é importante, uma vez que a detecção de intrusão deverá ser informada imediatamente para evitar danos graves.

2.4.1 DS *Snort*

Snort é uma ferramenta de código aberto desenvolvida com intuito de manter a segurança em redes computacionais. Ele implementa as aplicações de segurança IDS e IPS e é uma das ferramentas de código aberto mais importantes da história, segundo a revista InfoWorld (2009, p. 25).

O *Snort* tem capacidade para analisar tráfego em tempo real, registros de pacote e correspondência entre conteúdos. Executa análise de protocolo, busca/associa padrões de conteúdo e pode ser usado para detectar uma variedade de ataques, tais como buffer overflows, *stealth port scans*, ataques CGI, SMB *probes*, OS *fingerprinting*, entre outras. Esta ferramenta é suportada em arquiteturas RISC e CISC e em plataformas das mais diversas, como várias distribuições Linux (Red Hat, Debian, Slackware, Mandrake entre outros), OpenBSD, FreeBSD, NetBSD, Solaris, SunOS, HP-UX, AIX, IRIX, Tru64 e MacOS X. Ele é de fácil

instalação e utilização, e conta com alguns fóruns de discussões disponíveis na Internet.

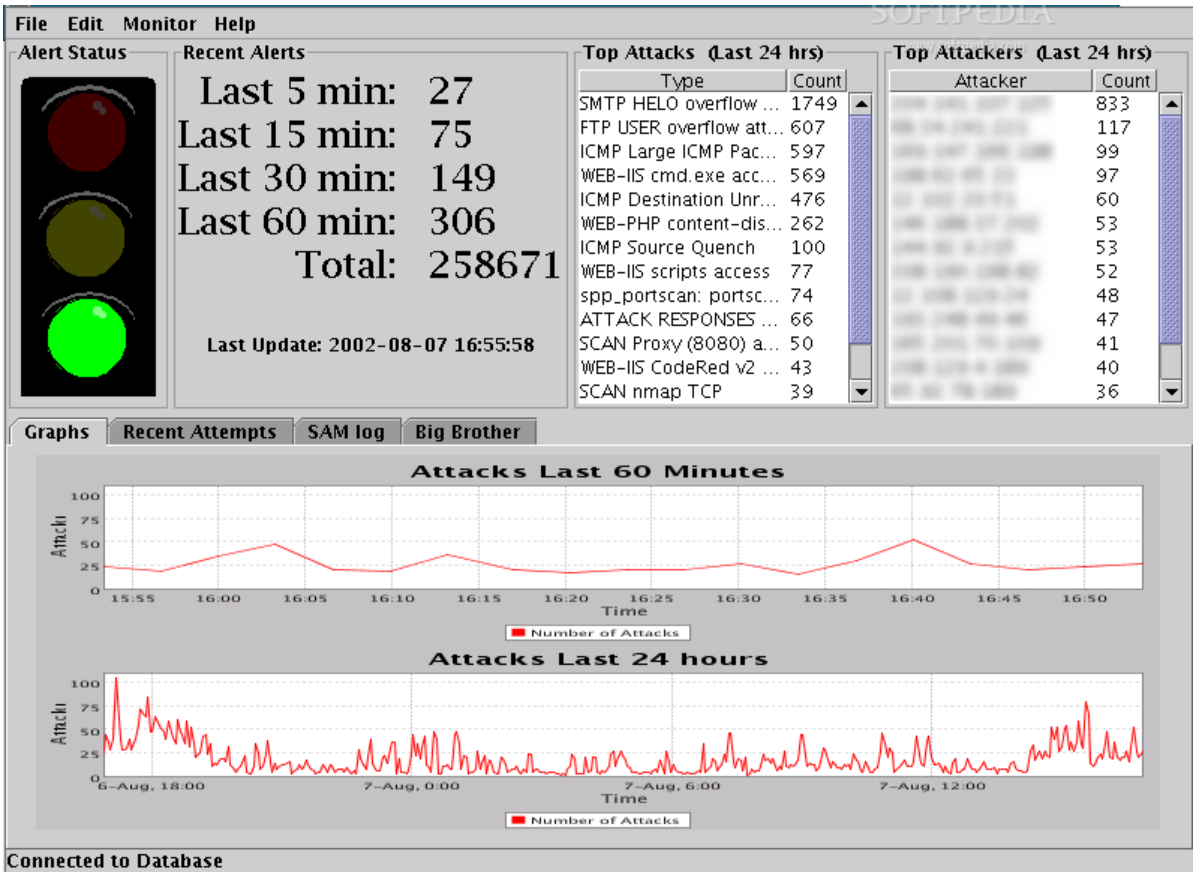


Figura 11: Interface de monitoramento de alertas do *Snort*

Fonte: http://linux.softpedia.com

O *Snort* também possui uma interface (Figura 11) que exibe os alertas de ataques e possui um semáforo do status atual da segurança da rede ou servidor monitorado.

2.4.2 utros IDS

Uma pesquisa foi realizada com intuito de estudar sobre as diversas ferramentas de segurança de código aberto que poderiam fazer parte do escopo deste projeto. OSSEC[7] é ferramenta semelhante ao *Snort*, mas implementa somente

[7] Disponível em http://communities.alienvault.com/download-ossim

o IDS e abrange a área *Host Based*. É caracterizada por analisar logs, verificar integridade, monitorar registros do Windows, detectar *rootkits* (softwares maliciosos de difícil percepção) e alertar sobre possíveis invasões, e opera em múltiplas plataformas (PANDA, 2009).

OSSIM[8] é outra ferramenta de código aberto com objetivo de fornecer um conjunto de ferramentas que permitem a um administrador uma visão geral de todos os aspectos relacionados à segurança do sistema. Trabalha como um IDPS e incorpora algumas funcionalidades das ferramentas já citadas como *Snort*, OSSEC e entre outras.

2.5 MOBILIDADE

Após termos abordados requisitos de segurança de redes de computadores, vamos agora abordar alguns conceitos de mobilidade que foram usados para fundamentar a criação da ferramenta proposta neste trabalho. O foco será abordar os benefícios, e também limitações, que os dispositivos móveis oferecem, e então relacionar de como podemos tirar beneficio da mobilidade para combater eventuais ataques as redes de computadores.

Segundo O Jornal de Hoje (2012), o uso da mobilidade implica:

> "Romper as barreiras físicas das empresas para os seus gestores e permitir que eles administrem à distância. Esse é o maior trunfo dos softwares de gestão em plataformas mobile (smartphones, tablets e celulares), ferramentas que dão agilidade e diminuem tempo e custos com a tomada de decisões estratégicas para as firmas".

A tecnologia móvel permitir ao indivíduo comunicar-se a qualquer momento e em qualquer lugar. As tecnologias móveis atingem a humanidade como um todo e, em especial, os usuários corporativos que, cada vez mais, precisam ter acesso, em tempo real, às aplicações corporativas que permitam tomadas de decisão imediata (PROMON, 2005).

Ainda segundo o artigo da Promon (2005):

> "Dada a evolução das aplicações em direção às interações em tempo real, o conceito de "estação de trabalho" ou mesmo de "desktop" já

[8] Disponível em http://www.ossec.net/main/downloads

> não é suficiente: uma empresa, em tempo real, requer acesso imediato à suas aplicações corporativas, possibilitando a tomada de decisões a qualquer hora e em qualquer lugar, com máxima flexibilidade".

Com a evolução da tecnologia aplicada aos dispositivos móveis (celulares, *tablets*, PDAs e outros) tornou-se cada vez mais usual a utilização destes dispositivos, antes considerados apenas para comunicação simples por voz. Muito mais do que agendas eletrônicas, os dispositivos móveis passaram a ser computadores que podem facilmente ser levados a qualquer lugar, e passaram a ser criados para atender profissionais e pessoas em movimento que necessitam de rapidez e facilidade no acesso a informações corporativas e pessoais. A Figura 12 ilustra alguns tipos de dispositivos móveis.

Figura 12: Dispositivos móveis.

Outro fator que tornou possível a adoção de dispositivos móveis como ferramenta para automatização de processos foi o desenvolvimento de linguagens de programação para as plataformas utilizadas por estes dispositivos (O'GRADY et al., 2005).

Além da portabilidade destes equipamentos (em geral podem ser manuseados com uso de apenas uma das mãos), um atrativo para a decisão de se

optar por seu uso está no custo dos aparelhos, que tem diminuído constantemente, e também pela capacidade de processamento e armazenagem de dados que tem aumentado (ANDERSON et al., 2004). Além disso, as grandes inovações trazidas pela tecnologia *Wireless* fizeram com que a indústria desse setor tenha tido um crescimento explosivo nos últimos anos, permitindo com que as pessoas comuniquem-se de forma barata e fácil sem ficarem presas aos seus telefones e computadores de mesa. Mas sempre existirá uma diferença entre os computadores do tipo desktop e notebooks (O'GRADY et al., 2005).

Dessa forma, adiante abordaremos as capacidades e limitações dos dispositivos móveis e aprofundaremos o estudo das vantagens do uso destes dispositivos na tomada de decisão.

2.5.1 Capacidade e Limitações de Dispositivos Móveis

Os dispositivos de computação móveis têm diferentes capacidades e limitações dependendo do tipo de dispositivo: computadores portáteis (*notebooks*), *tablet* PCs, PDAs e celulares. Nas próximas seções serão apresentadas avaliações das capacidades e limitações oferecidas por estes dispositivos. Os *tablet* PCs são dispositivos capazes de serem utilizados com teclado e mouse. No caso dos celulares, serão consideradas versões mais recentes destes dispositivos (*smartphones*), que incluem sistemas operacionais, como o Windows Mobile, iOS ou Android.

2.5.1.1 Entrada de Dados

Uma possível exigência de uma aplicação móvel é a necessidade de entrada de grandes volumes de dados. Os PDAs e celulares utilizam de um pequeno teclado ou da tela sensível ao toque como entrada de dados, o que torna esta atividade lenta, contudo é essa forma é útil para apoiar anotações curtas.

Conforme matéria publicada no site Ergoweb (2012), algumas pesquisas sugerem que a maioria dos celulares e PDAs disponíveis no mercado têm teclas de tamanho adequado ao dedo de uma criança de 5 anos de idade. E com a miniaturização cada vez maior destes dispositivos, outros mecanismos de entrada

de dados precisam ser fornecidos aos usuários. A Figura 13 apresenta os principais tipos de teclados dos celulares e PDAs.

Figura 13: Principais tipos de teclados dos celulares e PDAs.

Por outro lado, computadores portáteis e *tablet* PCs são dispositivos mais apropriados para apoiar processos de entrada intensiva de dados através de seus teclados. O processo de entrada de outros tipos de dados, tais como imagem, vídeo ou áudio, é operacionalmente semelhante para qualquer tipo de dispositivo de computação móvel.

Michael Dertouzos (2001) argumentou que a interação via voz deveria ser a principal abordagem na comunicação entre pessoas e máquinas, enquanto a visão seria mais adequada para a percepção de informações pelo homem. Mas isto só será possível quando as tecnologias para reconhecimento da fala estiverem maduras o suficiente.

Enfim, a capacidade e limitação de entrada de dados de cada dispositivo estão ligadas as características e capacidades de captar e armazenar os dados (SAKER e WELLS, 2003).

2.5.1.2 Tamanho da Tela

Os requisitos de tamanho de tela estão relacionados com o montante de informação que os usuários precisam ter acesso para apoiar suas atividades.

Segundo Chitarro (2006, p. 1): “As pessoas estão habituadas a confiar nas visualizações a compreender melhor os problemas que têm de resolver e tomar decisões melhores em menos tempo”. Aplicações com grande representação visual requerem grandes telas como as dos computadores portáteis. Embora dispositivos portáteis tenham sido criticados em algumas literaturas pelas suas pequenas telas (GUERRERO et al., 2004; KORTUEM et al., 2001), recentes técnicas de visualização melhoraram as capacidades destes dispositivos para exibir gráficos/informações detalhadas. Abaixo vamos ver exemplos de utilização de uma dessas técnicas apresentado por Chitarro (2006).

Imaginemos uma necessidade comum de usuários de dispositivos móveis: a visualização de uma base de dados geo-referenciados por pontos de interesse (POIs). Um típico mapeamento é baseado na associação de um ícone específico para cada categoria de POI e em seguida, coloca se o ícone apropriado para cada POI em um mapa de tal forma que as relações espaciais (posição e distância) entre POIs são visualmente mantidas. Um exemplo é ilustrado na Figura 14a (neste caso, hotéis e restaurantes são consideradas categorias).

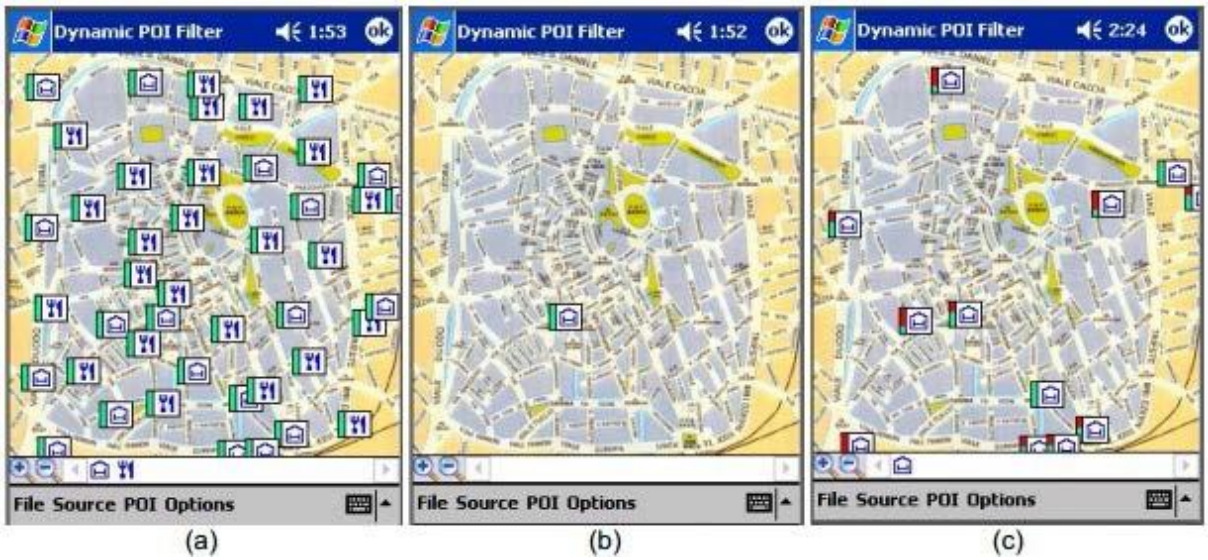


Figura 14: Visualização de POIs em um mapa a partir de uma aplicação de um dispositivo móvel.

Fonte: (CHITARRO, 2006).

Ao tocar no ícone escolhido, o usuário pode acessar o registo completo descrito pelo POI (por exemplo, preço, telefone, número de estrelas e outros). Embora a visualização seja clara e intuitiva, é extremamente improvável que um

usuário num contexto móvel irá sistematicamente tocar em cada ícone e ler o registo correspondente para chegar a uma decisão sobre qual hotel e restaurante a escolher. Este é um problema de seleção: a visualização não deveria simplesmente chamar todos os POIs personalizados como se fossem igualmente relevantes para o usuário. Uma típica forma de abordar esse problema é através da seleção de algoritmos que consomem apenas os POIs que satisfazem um conjunto de condicionalismos fornecidos pelo usuário. A limitação deste método é um ponto chave, pois se as exigências são demasiado soltas, depararemos novamente na situação da Figura 14a, enquanto se formos demasiadamente rigorosos na seleção, chegaremos a um mapa vazio, similar ao da Figura 14b, limitando assim a flexibilidade (por exemplo, o usuário poderá não gostar de ficar na rua, mas não tem ideia sobre quantas e quais restrições devem informar para encontrar uma melhor solução).

Através de um uma refinação no visual e uma melhor exploração da interatividade pode-se ajudar o usuário na consulta à base de dados. Por exemplo, a Figura 14c mostra a seguinte solução empregada: um mapeamento visual em cada ícone com uma indicação de quanto um POI satisfaz as limitações aplicadas pelo usuário, através do colorido de uma barra vertical no ícone. Se um POI satisfaz todas as restrições, a barra fica inteiramente verde; se algumas restrições não estiverem atendidas, a barra é dividida proporcional ao número de critérios atendidos (área verde) e não atendidos (zona vermelha). O usuário pode interativamente jogar com as restrições, visualmente percebendo o resultado das suas consultas em tempo real, observando alterações nas áreas de cor. Tocando em um POI, obtém-se um relatório sobre quais critérios específicos são satisfeitos ou não.

Outra técnica para a navegação em informações maiores que a capacidade do tamanho da tela de um dispositivo é trabalhar com efeitos de *scroll*, *zoom in* e *zoom out*.

Com a implementação de técnicas de navegação, como as vistas nesta seção, podemos contornar em alguns casos a limitação do tamanho das telas dos dispositivos móveis.

2.5.1.3 Privacidade

Dispositivos de computação móveis geralmente possuem pequenas telas, e assim, proporcionam melhor proteção à privacidade do que computadores portáteis e *tablet* PCs, já que facilitam esconder dados apresentados na tela de outras pessoas em espaços públicos. Além disso, a distância física entre o usuário e o dispositivo móvel durante a interação é mais curta do que a distância entre um usuário e o seu computador portátil ou *tablet* PC. Outra consideração na privacidade em computação móvel é a visibilidade dos usuários e o tráfico dos dados em redes públicas (KORTUEM et al., 2002). A garantia da segurança da informação é uma questão crítica em alguns casos, como por exemplo, em transações bancários e comércio eletrônico.

2.5.1.3 Capacidade de Armazenamento e de Memória

Restrições no desenvolvimento de software têm sido relatadas em algumas literaturas devido à capacidade de armazenamento e memória, especialmente relacionados com dispositivos portáteis (KORTUEM et al., 2002). No entanto, este tipo de dispositivos móveis tem constantemente melhorado a sua capacidade de armazenagem e de memória. As últimas versões destes dispositivos permitem as aplicações móveis gerir e armazenar tipos de dados complexos, como informações multimídia. Se a largura de banda da rede é estável e ampla, então a capacidade de armazenamento e memória destes dispositivos torna-se ainda menos importante, porque os dispositivos podem fazer *buffering*.

2.5.1.4 Poder de Processamento

Já há alguns anos, os computadores de mesa transacionaram de CPUs (*Central Processing Unit*) de arquitetura de um único núcleo para uma arquitetura de núcleos duplos, quádruplos e sêxtuplos. Contudo os consumidores só passaram a enxergar os reais benefícios de CPUs de múltiplos núcleos há poucos anos. Isto acontece porque o ecossistema de software necessário para alavancar o poder dessa nova arquitetura não estava imediatamente disponível para computadores de

mesa. Somente aplicações que foram desenvolvidas para utilizar vários núcleos utilizavam-se do benefício dessa nova arquitetura, e este tipo de aplicação só se tornaram disponíveis em quantidade há poucos anos após a introdução dos CPUs de múltiplos núcleos. Hoje os computadores de mesa estão vendo os inúmeros benefícios dessa arquitetura.

Para os dispositivos móveis, a transição um único núcleo para uma arquitetura de múltiplos núcleos tem sido muito mais rápida (NVIDIA, 2012). O ecossistema de software móvel tem evoluído a partir do trabalho já feito no espaço dos computadores de mesa e como resultado tem surgido aplicações para dispositivos portáteis que efetivamente suportam e se beneficiam CPUs de múltiplos núcleos. Por exemplo, o sistema operacional para dispositivos móveis Android, por ser baseado na plataforma do sistema operacional de computadores de mesa Linux, herdou todo o suporte nativo de multitarefa e multi-*threading*. As recentes versões do Android 3.0/3.1/3.2 têm adicionado várias funcionalidades para beneficiar-se do uso dos múltiplos núcleos.

B*rowsers* dos dispositivos móveis como o Firefox e o Webkit são baseados em suas versões para os computadores de mesa, e, portanto, nativamente incluem o suporte a multi-*threading*. O novo *browser* incluído no Android 3.0 além de suportar multi-*threading* também suporta múltiplas abas. A Figura 15 mostra um exemplo da utilização da CPU de um dispositivo móvel equipado com um CPU de múltiplos núcleos, o *Quad Core*. Estes *browsers* são capazes de utilizar desta melhoria na capacidade de processamento dos dispositivos móveis oferecida pelos CPUs de múltiplos núcleos e proporcionam melhor experiência de navegação na *Web*.

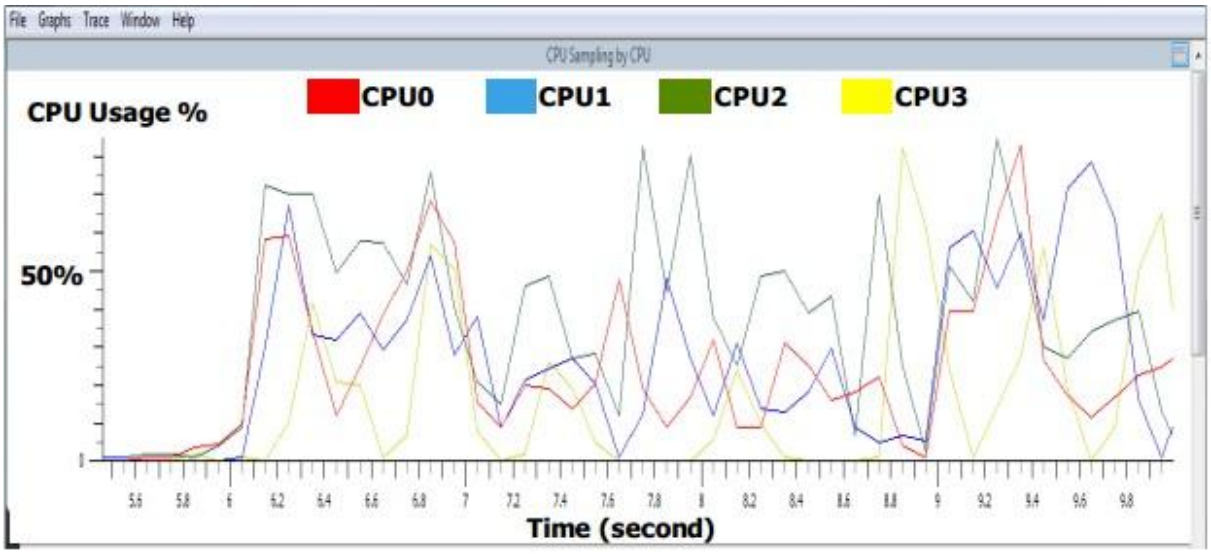


Figura 15: Utilização de CPU de um *browser* em um dispositivo móvel de CPU *Quad Core*.

Fonte: (NVIDIA, 2012).

Devido ao rápido crescimento dos jogos para dispositivos móveis, desenvolvedores estão portando populares jogos de computadores de mesa e *console games* para o ambiente móvel. Estes jogos foram desenvolvidos originalmente para plataformas computadores de mesa de múltiplos núcleos e, por isso irá alavancar nativamente utilizar-se dos CPUs de múltiplos núcleos dos dispositivos móveis em entregar benefícios imediatos para os jogadores móveis. CPUs *Quad core* proporcionam significativo poder de processamento para os programadores de jogos e permitem aos programadores incluírem efeitos de física avançada, inteligência artificial, detecção de colisão/esquiva, texturização, rede virtual de melhor jogabilidade e muito mais.

Utilizando um dispositivo móvel com um CPU *Quad core*, foi simulada a execução do jogo Glowball utilizando dois e quatro núcleos. O resultado foi comparado na Figura 16, onde o lado esquerdo é o jogo utilizando apenas dois núcleos, e o lado direito utilizando quatro núcleos.

Figura 16: Demonstração de jogo em um dispositivo móvel.

Fonte: (NVIDIA, 2012).

A utilização de quatro núcleos possibilitou a inserção de efeitos de realismo na imagem do jogo. *Quad core* CPUs permitem aos dispositivos móveis um melhor desempenho para as aplicações e para os desenvolvedores de jogos.

Enfim, a evolução do poder de processamento dos dispositivos móveis tem equiparado estes dispositivos aos tradicionais computadores de mesa.

2.5.1.5 Capacidade de Comunicação

Aplicativos para dispositivos móveis exigem comunicação síncrona ou assíncrona dependendo do tipo de atividade a ser suportada. Se uma comunicação assíncrona é necessária, a computação móvel pode prestar esse apoio baseado na mínima disponibilidade de rede. Por outro lado, se uma comunicação síncrona é necessária, um serviço de comunicação permanente e estável deve ser fornecido independentemente do ambiente em que o usuário se encontra (SARKER e WELLS, 2003). Segundo Fiorese (2005):

> "Com a evolução dos protocolos de comunicação sem fio e das redes de comunicação de celulares, se torna possível a transferência de

> dados entre dispositivos, notadamente um celular ou PDA e um computador, através das redes sem fio (wireless)".

Através do protocolo GPRS, que fornece acesso à Internet a celulares, é possível a transferência de dados de celulares e PDAs de qualquer lugar que se tenha cobertura da rede (Figura 17), permitindo que aplicações funcionando nesses dispositivos se comuniquem em tempo real com bancos de dados e outros sistemas através da Internet.

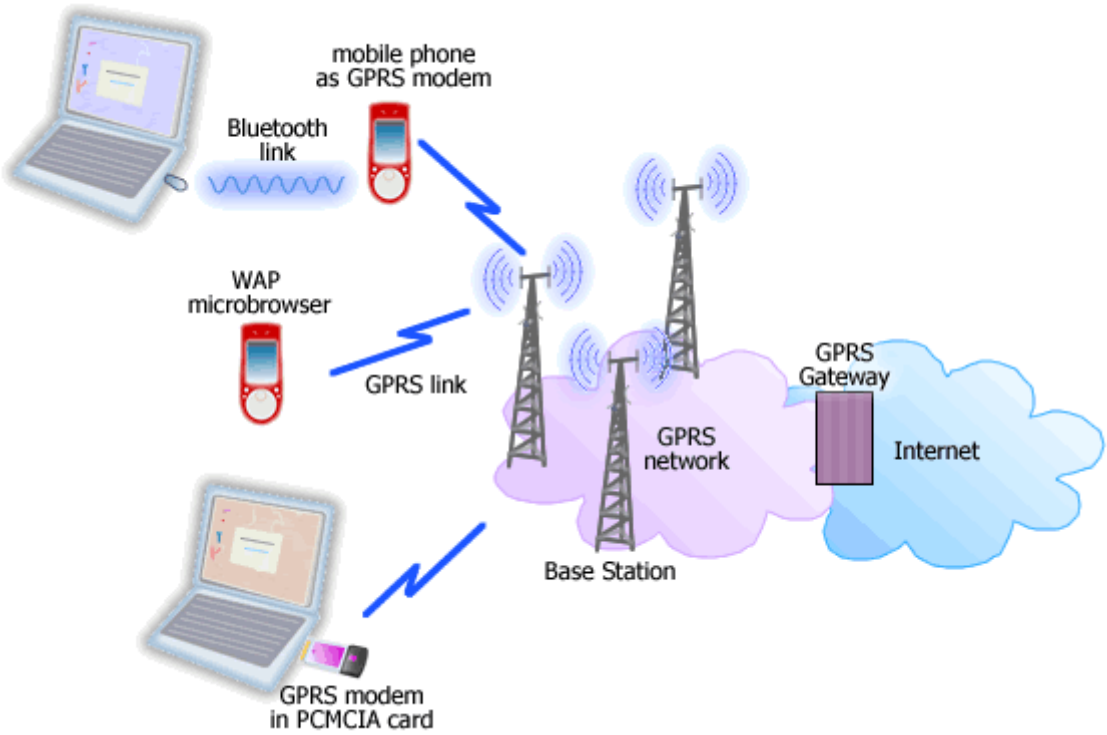


Figura 17: Rede GPRS.

Fonte: conniq.com.

Além do GPRS existem outros protocolos de comunicação capazes de facilitar a transmissão de dados entre dispositivos móveis e desktops, tais como as tecnologias Bluetooth, infravermelho e a 3G, a mais recente de todas. Qualquer uma pode ser utilizada desde que os equipamentos tenham suporte aos mesmos.

Celulares são normalmente a melhor opção para comunicação síncrona desde seu grande alcance de cobertura e boa estabilidade do sinal (MALLADI e AGRAWAL, 2002). No entanto, estas redes têm uma largura de banda limitada. Outra opção é fornecer comunicação síncrona para aplicações móveis utilizando uma infraestrutura de comunicação Wi-Fi (KOURTUEM et al. 2001). Embora a

largura de banda é melhor que as redes celulares, a estabilidade do sinal Wi-Fi depende do ambiente físico onde é implantado (ALDUNATE et al. 2005). Além disso, este tipo de redes possui uma cobertura limitada gama (MALLADI e AGRAWAL, 2002).

2.5.1.6 Capacidade de Utilização

A capacidade de utilização de um dispositivo móvel é limitado pela autonomia de sua bateria. Muitos pesquisadores têm identificado este problema como crítico para suportar a colaboração móvel (KORTUEM et al. 2001; GUERRETO et al. 2004). No entanto, técnicas de desenvolvimento de aplicações onde se otimiza o uso do processador, consumo de banda da conexão e até as cores da interface da aplicação, fornecem uma maneira de otimizar o uso de alimentação resultando em maior durabilidade da bateria. Por outro lado, é sempre possível levar baterias extras quando PDAs, computadores portáteis ou *Tablet* PCs são utilizados.

O consumo de energia de um dispositivo móvel tende a ser menor que o de um computador de mesa, já que os primeiros normalmente são alimentados por baterias recarregáveis.

A capacidade de utilização não é tão crítica no caso dos celulares porque estes dispositivos são capazes de trabalhar por muitas horas sem ser recarregada (HAKKILA e MANTYJARVI, 2005).

2.5.2 Benefícios Adicionados Pelos Dispositivos Móveis

No mundo de hoje, o uso de dispositivos móveis enriquece os serviços eletrônicos e multiplica as possibilidades de apoio à decisão. Os seus usuários podem tomar decisões em tempo real baseadas em dados atuais acessados via dispositivos móveis. As transações de negócios tais como lojas *on-line*, bancos, e outros, podem ser concluídas em um ambiente seguro de um dispositivo móvel. Os viajantes podem otimizar as suas viagens e intercalar com alguns dias de descanso em lugares turísticos através de recomendações que estão disponíveis no seu dispositivo móvel (CARLSSON et al., 2006 e NIELSEN, 2004). Um negociante de ações pode controlar sua carteira de investimento em um celular que fornece alertas

sobre o comportamento de suas ações (KARGUPTA et al., 2002), vejamos exemplo da Figura 18:

Figura 18: Exemplo de aplicativo de acompanhamento de bolsa de valores.

Fonte: uol.com.br.

A tecnologia móvel está cada vez mais evoluindo em conjugação com a capacidade das redes sem fios. Os avanços nas comunicações sem fios e da tecnologia móveis trouxeram à existência novos termos como *mobile era*, *mobile commerce*, e *mobile workers* (MENNECKE e STRADER, 2002). Seu sucesso e importância tem realçado a necessidade de crescimento também na área de apoio à tomada de decisão. Ela criou oportunidades para seus usuários de reunir informações precisas e em tempo real para a tomada de decisões imediatas. Além da disponibilidade de informações atualizadas ou em tempo real, o potencial de tais ferramentas em poupar tempo e aumentar a produtividade tem atraído bastante novos usuários (CARLSSON et al., 2005).

Enquanto sistemas de apoio à decisão têm sido tipicamente associados com computadores de mesa (*desktops*), o desenvolvimento de novas tecnologias móveis e compactas proporcionam novas oportunidades (ARONSON et al., 2005). Ambientes de trabalho que são móveis por natureza e que não podiam se beneficiar de sistemas de apoio à tomada de decisão, por até então só estarem disponíveis para computadores de mesa, poderão agora ser utilizados através do uso das tecnologias móveis. Essa tecnologia também pode ser adaptada aos atuais locais de trabalho para resolver limitações dos sistemas atuais (SHARAF e CHRYSANTHIS,

2002). Acessar informações em tempo real é essencial para uma boa tomada de decisão, a utilidade de dispositivos móveis para apoio à decisão, em nossa opinião, é incalculável.

Na atual sociedade consumista, "somos bombardeados com publicidade que nos mostra que podemos ter tudo instantaneamente" (TARICA, 2001). Comerciantes estão atualmente focados no *mobile commerce* e tem direcionado sua publicidade para os dispositivos móveis, onde os consumidores recebem publicidade via SMS (serviço de mensagens curtas) e MMS (serviço de mensagens multimídia) em seu telefone móvel, adaptadas às suas preferências pessoais (PANIS et al. 2002). Instituições financeiras oferecem serviços *on-line* que podem ser acessados pelo usuário via *web* através de seu computador pessoal de mesa (*Desktop* PC), celular e até televisor inteligente. Para os consumidores, estas transações *on-line* incluem consulta de saldo de conta, transferências de fundos, pagamentos eletrônicos e outros. Para as empresas, serviços bancários *on-line* incluem a monitorização dos saldos de seu portfólio de investimento, pagamento de salários e outros. “Pagamentos eletrônicos com cartões de crédito também são muito frequentes e fazem parte da vida quotidiana dos consumidores” Hartman e Bretzke (1999). Com a introdução desses novos serviços para dispositivos móveis, o consumidor passa a ter que tomar decisões difíceis ao responder a todo esse novo marketing direcionado, obrigando-os a tentar gerir de forma eficiente o seu orçamento com o risco real de obter elevados níveis de dívidas que muitas vezes são difíceis de controlar.

Como ponto negativo, a mobilidade introduz alguns pontos de incerteza em um ambiente para a tomada de decisão. Em primeiro lugar, a informação detida em um dispositivo móvel é suscetível de estar incompleta ou desatualizada e pode não prover dados confiáveis ao seu usuário em situações críticas tais como gerenciamento de sistemas de saúde, defesa nacional, ou previsões meteorológicas. Por esse motivo, algumas literaturas expõem a necessidade de oferecer ao tomador de decisão, um indicar do quanto os dados ou informações oferecidos pelo sistema de apoio à decisão (DSS - *Decision Support System*) são confiáveis, indicador este que essas literaturas chamam de qualidade dos dados (QoD - *Quality of Data*) (SAN PEDRO et al., 2003 e BURSTEIN et al., 2004). O QoD pode ser calculado através de diferentes modelos.

Um modelo QoD visa ajudar os utilizadores móveis na seleção da melhor opção, tendo em conta a confiabilidade dos dados que tal opção foi baseada. A Figura 19 representa um modelo de QoD criado por Burstein et al. (2004) para decisão móvel, e que utiliza de atributos para a qualificação.

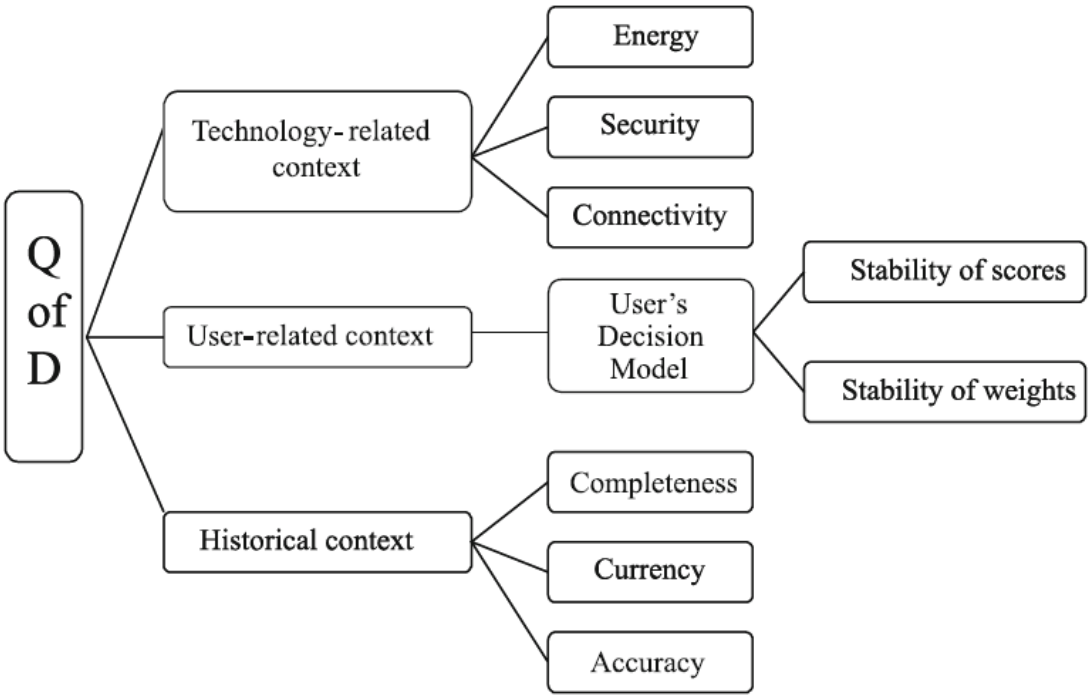


Figura 19: Modelo de representação de QoD.

Fonte: (BURSTEIN et al., 2004).

Estes atributos estão relacionados à tecnologia, usuário e históricos. Esses atributos são ainda subdivididos em algumas métricas relativas à energia, segurança, tecnologia e conectividade. Alguns destes indicadores podem ser calculados comparando dados atuais com dados padrão.

Com relação à implementação de sistemas de decisão móveis, eles podem ser implementado de várias formas, dependendo dos requisitos do usuário, disponibilidade de recursos tecnológicos, frequência de acesso aos dados, urgência de recuperação de dados e outros. Como essa tecnologia é relativamente nova, ainda é questionável a arquitetura ideal para cada contexto. Para este trabalho consideramos como componentes fundamentais do DSSs o bando de dados (DB), interface do usuário (UI) e o modelo analítico (AM) (ARONSON et al., 2005). A Figura 20 ilustra cinco tipos possíveis de arquiteturas de DSS para dispositivos móveis.

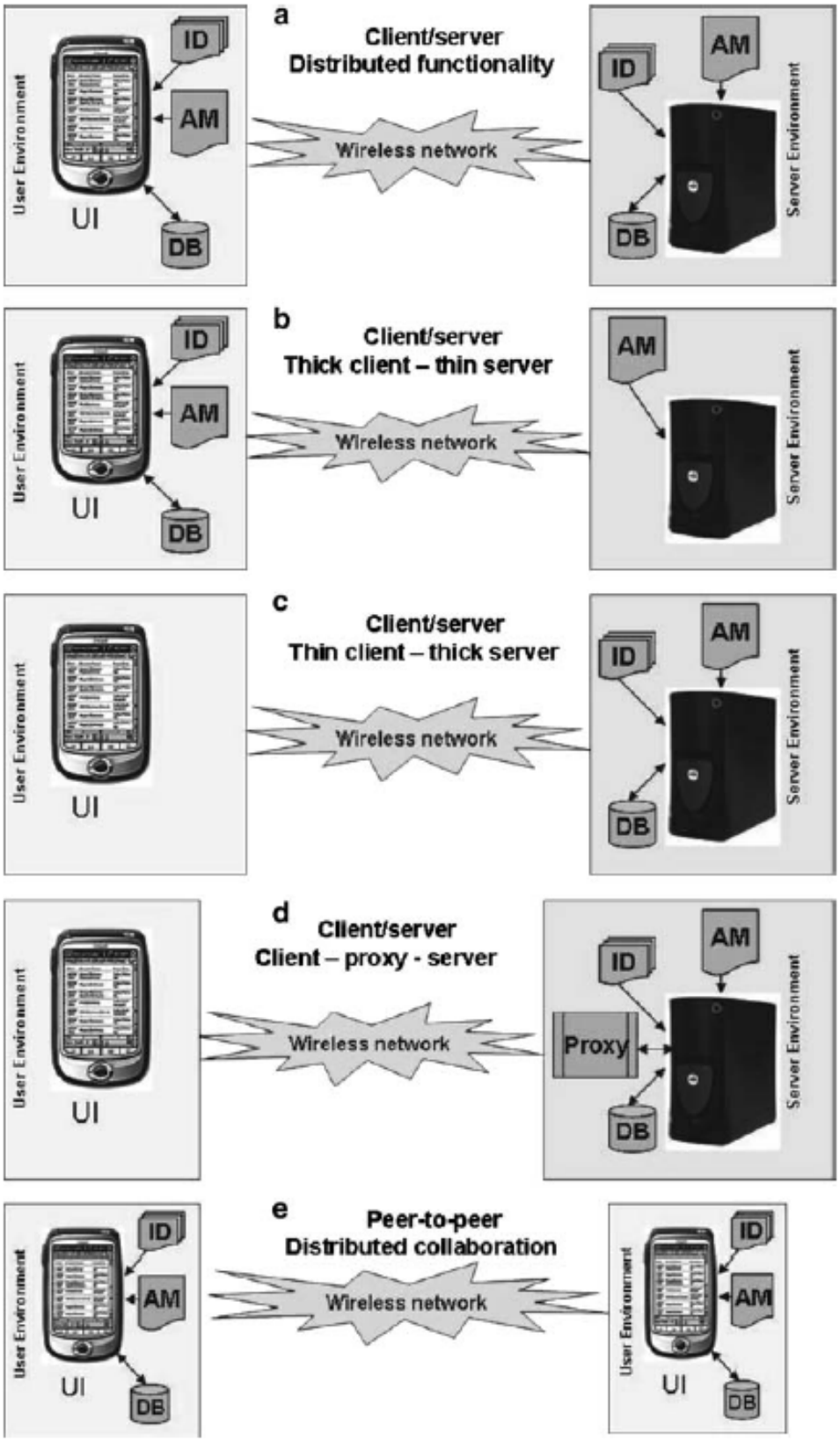


Figura 20: Possíveis arquiteturas de um sistema de decisão móvel (UI *user interface*, ID *input data*, AM *analytical model*, DB *data base*).

Fonte: (ARONSON et al., 2005).

Dispositivos portáteis podem agir como plataformas computacionais para aplicações de tarefas específicas, coletando, armazenamento, processamento e fornecendo dados. Navarro et al. (2006) dizem que: “O uso de recursos de um dispositivo ou de um servidor criam uma distinção entre os possíveis tipos de DSSs móveis que podem ser fornecidos”. Uma arquitetura de um sistema de decisão móveis pode ser baseada no cliente, no servidor, em um *proxy o*u distribuídas em uma rede *ad hoc* com vários pares de dispositivos semelhantes. O tipo de arquitetura depende de onde a informação é armazenada e onde os cálculos são realizados. Estas diversas implementações têm as suas vantagens e desvantagens. A utilidade de cada tipo depende significativamente dos requisitos de aplicação e de infraestrutura. Por exemplo, sistemas centrados no acesso a grades volumes de dados são mais propensos a utilizar uma arquitetura baseada no servidor, isto devido a limitações de armazenamento e processamento dos dispositivos móveis. Atualmente, a mais popular arquitetura é a ilustrada na Figura 20a, onde a funcionalidade é distribuída em todo o ambiente cliente-servidor com uma interface de usuário (UI) localizada no dispositivo portátil, usando dados distribuídos por ambos cliente e servidor, com dados sensíveis que resida no dispositivo do utilizador, enquanto grandes quantidades de dados, incluindo histórico, estão localizados em um servidor. Nesta configuração, o modelo analítico (AM) também é distribuído entre o cliente e o servidor, onde o dispositivo móvel efetuar cálculos elementares e delega os mais complexos, que exigem muitos recursos computacionais, para o servidor.

Arquiteturas que concentram o processamento em um camada cliente servidor representam casos mais casos extremos e portanto mais configurações mais raras (ver Figura 20b). Dada a alta probabilidade de desligamentos em ambientes *wireless*, alguns sistemas podem usar o conceito de uma arquitetura *proxy*, onde um processo *proxy* que fica localizado do lado do servidor representa um cliente e, se a conectividade for boa, dados e cálculos são simplesmente canalizado do servidor para o cliente (ver Figura 20d). No entanto, se um cliente for desligado (por exemplo, condução através de um túnel), então o *proxy* assume plena funcionalidade do cliente e mantem os dados e os resultados em *cache* até que o cliente é reativado. Com a proliferação de computação *peer-to-peer*, tornou-se possível formar redes *ad hoc* de dispositivos semelhantes, tornando possível aplicar

o AM em um ambiente de computação distribuída, tornando-o um recurso mais eficaz (ver Figura 20e). Essas diferentes possibilidades de implementações de arquiteturas permitem uma suficiente flexibilidade e escalabilidade para os DSS.

Concluímos que apesar de algumas limitações, a computação móvel em dispositivos de mão, como PDAs e celulares, gera grandes benefícios, principalmente em aplicações de tomada de decisão, onde o tempo e a mobilidade são cruciais. Sendo assim, a mobilidade pode ser usada como uma valiosa funcionalidade em prol da segurança de redes, e foi isto que fundamentou a construção de uma ferramenta que usa do poder da mobilidade para a monitoração de ataques contra redes de computadores e de também rápida tomada de ação para evitar a penetração da rede.

Nas próximas seções deste trabalho, iniciaremos a abordagem da ferramenta em si.

2.6 DISCUSSÃO DA FERRAMENTA PROPOSTA

A preocupação com a segurança de redes não é uma particularidade de pequenas, médias e grandes empresas ou órgãos públicos, mas também de micro entidades, como pequenos administradores de redes. O fato de um administrador ter apenas um ou dois servidores expostos a vulnerabilidades não diminui o risco de prejuízos. Uma microempresa, por exemplo, que tem todo o seu faturamento vinculado à disponibilidade de uma página na Internet, também tem a total preocupação em monitor se seu servidor não está sendo alvo deu um ataque. Embora que, para microempresas similares ao exemplo anterior, o monitoramento do servidor seja vital, é comum não ter nenhum tipo de monitoramento devido à falta de recursos financeiros para aplicar em alguma solução que cubra o monitoramento em tempo integral.

Utilizando-se dos benefícios que a mobilidade oferece, este trabalho tem o objetivo de cobrir as necessidades das microempresas em ter uma ferramenta de baixo custo, e que também proporcione o monitoramento de uma rede de computadores, a qualquer hora e em qualquer lugar, permitindo a tomada de decisão remota em um eventual ataque. Para isso, foi desenvolvida a ferramenta *Hamdroid*.

Este projeto ao final irá mostrar as vantagens da utilização da sua ferramenta.

3 ***HAMDROID***

Neste capítulo serão apresentadas as especificações de software do *Hamdroid* bem como suas características e arquitetura.

3.1 INTRODUÇÃO

O *Hamdroid* tem como objetivo ser uma ferramenta de controle de segurança a uma rede de computadores remota, permitindo ao seu usuário visualizar alertas de segurança produzidos por um IDS e tomar decisões em um eventual caso de ataque.

A ferramenta ainda visa atender ao nicho de mercado de pequenas empresas ou pessoas que possuem um ambiente simples de um a dois servidores conectados à Internet e que precisam de uma solução IDS eficaz e de baixo custo e que permita a monitoração remotamente. Para isso, este trabalho desenvolveu um protótipo cliente-servidor, que utilizará a *Application Programming Interface* (API) do *Snort*, possibilitando o monitoramento de um servidor através de alertas gerados pelo IDS em um dispositivo móvel. Além disso, algumas funcionalidades foram implementadas para que o usuário consiga tomar alguma decisão rápida mediante os alertas, desde o desligamento do servidor até o simples bloqueio de um IP. O usuário final deverá ter conhecimentos em redes de computadores para o bom uso de todas as funcionalidades presentes no protótipo.

3.2 TRABALHOS RELACIONADOS

Nesta seção serão apresentados trabalhos relacionados com este trabalho e um posterior comparativo entre os mesmos.

3.2.1 ***Swinedroid***

Swinedroid é um aplicativo disponível no mercado que mais se aproxima com as funcionalidades da ferramenta proposta neste trabalho. Disponível exclusivamente para clientes Android e servidores Linux, este é um aplicativo de

monitoração remota através de dispositivos móveis que trabalha em conjunto com o *Snort*. Desenvolvida informalmente pelos programadores William Budington, a ferramenta é oferecida gratuitamente em seu site (BUDINGTON, 2012). Em sua versão atual, o *Swinedroid* permite visualizar as estatísticas de ataques de um servidor (Figura 21), mostra os alertas mais recentes gerados pelo *Snort*, permite consultas aos alertas (por severidade, nome e linha de tempo), e mostra os detalhes referentes a um alerta (Figura 22).

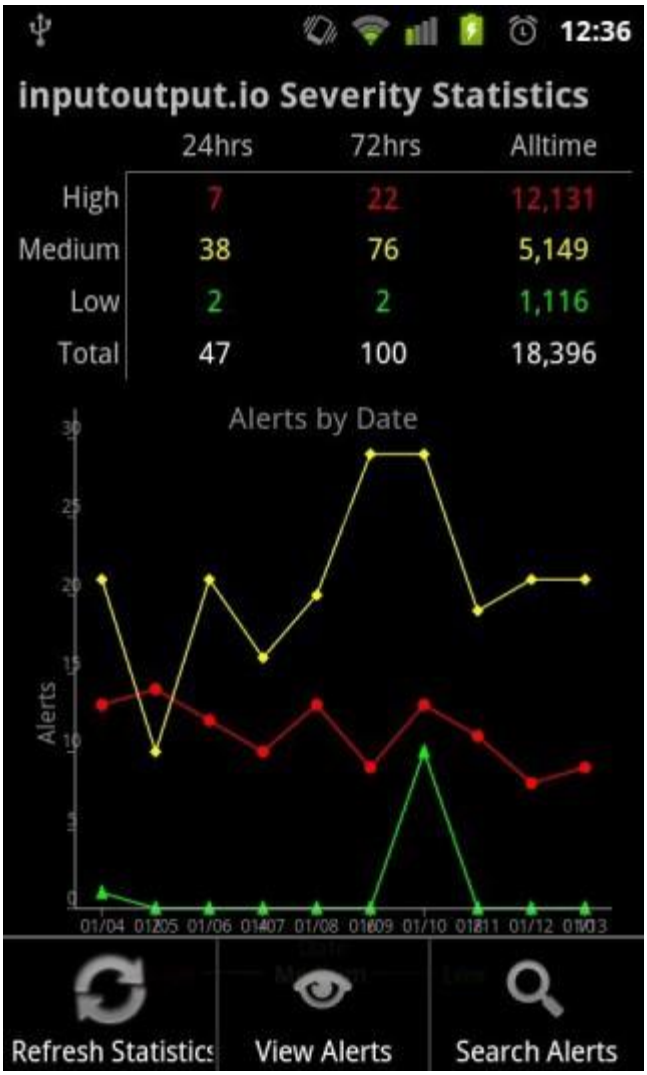


Figura 21: *Swinedroid* – Estatísticas do servidor.

Fonte: inputoutput.io

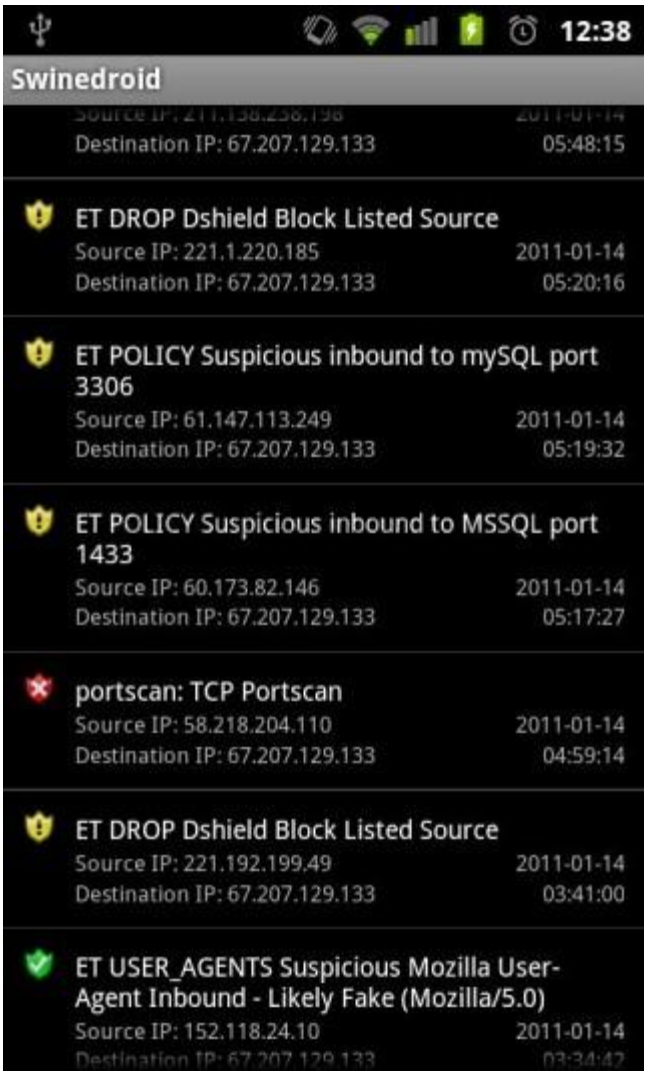


Figura 22: *Swinedroid* – Alertas do servidor.

Fonte: inputoutput.io

O *Swinedroid* consiste em duas camadas: a camada cliente – que é compatível apenas com Android, e a camada servidor – que é compatível apenas com distribuições Linux, e que interage com os logs gerados pelo *Snort* e provê os dados para a camada cliente. Toda a comunicação cliente e servidor é feita através de uma conexão *Secure Socket Layer* (SSL).

3.2.2 Comparativo

O *Hamdroid*, ferramenta desenvolvida nesse trabalho, é similar ao *Swinedroid*, pois ele tem as mesmas funcionalidades vistas anteriormente no *Swinedroid*, porém ele soma as seguintes funcionalidades: suporte a múltipla

plataformas na camada do servidor – distribuições Linux e Windows Server, disponibilidade de tomada de ações no servidor em caso de um eventual ataque e um *Dashboard* (painel de controle) de fácil visualização do status do servidor – verde para normal, branco para ausência de comunicação com a rede de computadores monitorada e vermelho para ataque ou invasão.

Diferente também do *Swinedroid*, o *Hamdroid* tem em seu componente na camada do servidor um algoritmo que mais do que somente ler os logs gerados pelo *Snort*, ela se propõe a interpretar os logs e gerar informações significativas para o usuário.

3.3 ARQUITETURA E PRÉ-REQUISITOS

Nesta seção iremos apresentar a arquitetura e as ferramentas de apoio utilizadas para a criação de todo o ambiente e infraestrutura da ferramenta *Hamdroid*.

3.3.1 Arquitetura, Instalação e Inicialização do *Snort*

De acordo com Caswell et al. (2003), a arquitetura do *Snort* é composta de 4 componentes básicos: o farejador, o pré-processador, o mecanismo de detecção e os *plug-ins* de saída. Ele é projetado para pegar pacotes e processá-los através do pré-processador e depois verificar esses pacotes com relação a uma série de regras. A Figura 23 oferece uma visão de alto nível da arquitetura do *Snort*.

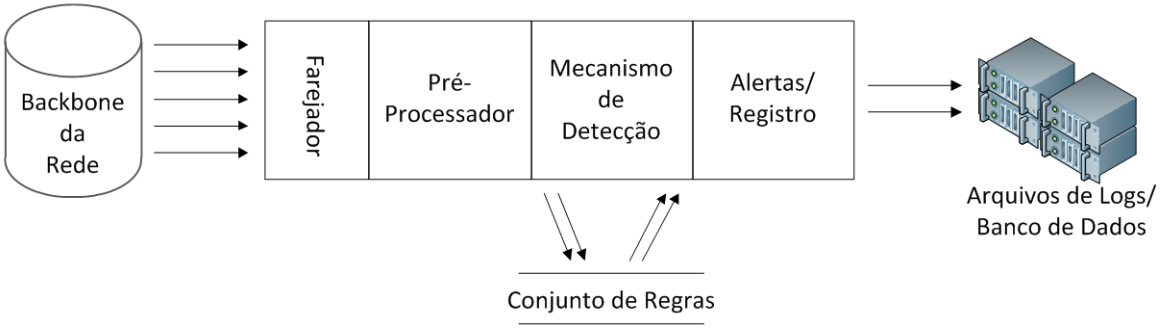


Figura 23: Arquitetura do *Snort*.
Fonte: (CASWELL et al., 2003).

O *Snort* em uma visão mais simples pode ser descrito como:

1. **Mecanismo de captura/decodificador de pacotes** – O tráfego é obtido do link de rede, através de um farejador.

2. ***Plug-ins* de pré-processadores** – Os pacotes passam então por um conjunto de pré-processadores. Eles são examinados e manipulados antes de serem enviados ao mecanismo de detecção. Cada pré-processador verifica se esse pacote é algo que ele deve examinar, alertar a respeito ou modificar.

3. **Mecanismo de detecção** – Em seguida, os pacotes são enviados para o mecanismo de detecção. O mecanismo de detecção verifica cada pacote em relação a várias opções listadas nos arquivos de regras do *Snort*. Cada uma das opções de palavra-chave da regra é vinculada a um *plugin* de detecção que pode realizar testes adicionais.

4. ***Plug-ins* de saída** – O *Snort* produz a saída dos alertas do mecanismo de detecção, dos pré-processadores ou do mecanismo de decodificação.

Para esse projeto foi utilizada a versão 2.9.2.3 do *Snort* para *Windows*, que foi obtida de seu site oficial[9].

Toda a configuração do Snort está em um arquivo chamado “snort.conf”, presente na instalação padrão em: “c:\snort\etc\”. O arquivo deste projeto pode ser encontrado no Anexo B desta monografia. No “snort.conf” estão todos os caminhos que o Snort utilizará para carregar as regras, a definição dos endereços IP da rede, dos servidores DNS, HTTP, bem como as portas que serão farejadas, entre outras opções de configuração.

Algumas configurações feitas no arquivo de configuração “snort.conf” devem ser destacadas, pois afetam diretamente o projeto. São elas:

[9] Disponível em http://www.snort.org.

- **IPs dos hosts monitorados**: Na variável HOME_NET foi configurada com os endereços IPs dos hosts que serão monitorados:

  ```
  var HOME_NET 192.168.0.11/1
  ```

  Os endereços devem ser representados da seguinte forma: IP/CIDR (*Classless InterDomain Routing*). O operador de negação "!" pode ser usado em conjunto com um endereço IP, para negar o valor. Por exemplo !192.168.200.0/24. Pode-se também especificar intervalos de endereço em um formato de lista, usando colchetes para englobar intervalos de IP's, com cada elemento separado por vírgula, como observado a seguir: [192.168.100.0/24, 192.168.200.0/24, 192,168,20.5]. O coringa "any" pode ser utilizado para definir qualquer endereço.

- **Caminho das Regras**: As variáveis RULE_PATH, SO_RULE_PATH e PREPROC_RULE_PATH foram configuradas para apontar os diretórios que armazenam as regras do *Snort* que foram utilizadas:

  ```
  var RULE_PATH c:\snort\rules
  var SO_RULE_PATH c:\snort\so_rules
  var PREPROC_RULE_PATH c:\snort\preproc_rules
  ```

- **Caminho dos pré-processadores**:

  ```
  dynamicpreprocessor directory
  C:\Snort\lib\snort_dynamicpreprocessor
  ```

- **Caminho dos núcleo do *Snort***:

  ```
  dynamicengine
  C:\Snort\lib\snort_dynamicengine\sf_engine.dll
  ```

Para inicializar o *Snort* deve se acessar o *prompt* de comandos do Windows, acessar a pasta onde ele foi instalado em: "c:\snort\bin\" e executar o *Snort* digitando o comando "snort" conforme a Figura 24 abaixo.

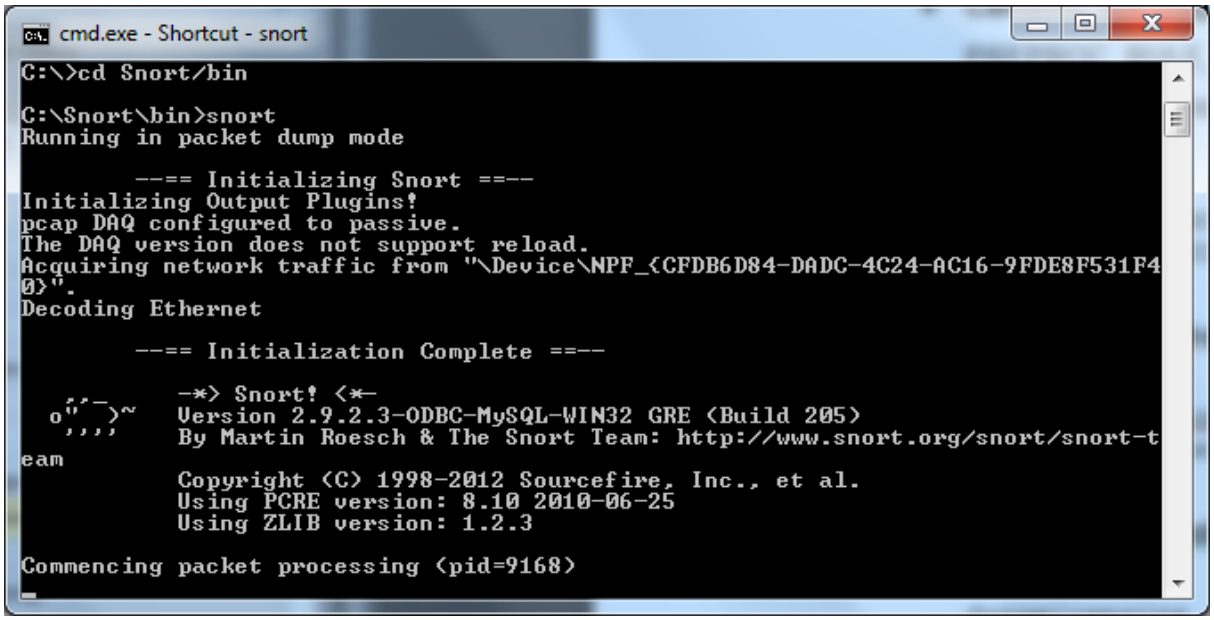


Figura 24: Inicializando o *Snort*.

O *Snort* aceita vários parâmetros de configurações, a fim de simplificar as configurações iniciais, registramos todas as configurações necessárias para o funcionamento do *Snort* no arquivo “snort.conf”, presente na instalação padrão em: “c:\snort\etc\”. Estas configurações estão presentes no Anexo B desta monografia.

Para importar as configurações do arquivo “snort.conf” digitamos no *Snort* o comando “snort – c c:\snort\etc\snort.conf -i x”. Onde “x” é o número do dispositivo de rede que o *Snort* usará em sua análise, que pode ser encontrado pelo comando “snort – W”.

3.3.2 ***Winpcap***

Conforme visto na seção anterior, o *Snort* utiliza de dispositivos farejadores para prover os dados para o pré-processadores. Os farejadores de tráfego de redes são dispositivos de hardware ou software que tem o objetivo de capturar os pacotes que trafegam na rede de computadores (CASWELL et al., 2003).

Alguns farejadores são capazes de analisar diferentes protocolos para transformar os dados de maneira legível. A Figura 25 mostra o funcionamento do farejador.

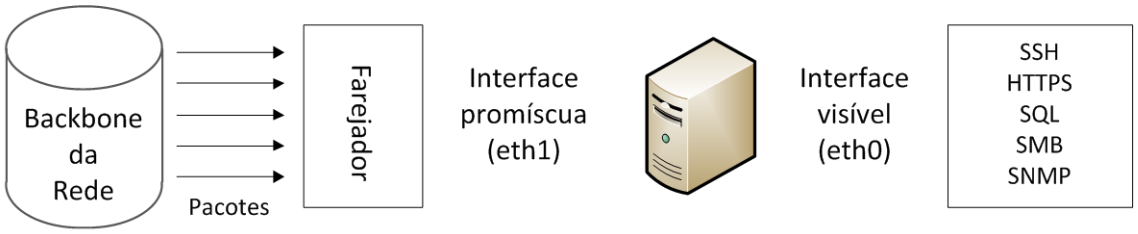


Figura 25: Esquema de funcionamento de um farejador de pacotes.
Fonte: (CASWELL et al., 2003).

Para que seja possível capturar tráfego em uma rede, o *Snort* utiliza no Windows a biblioteca *Winpcap*, permitindo capturar os dados que chegam no *host* e os que são destinados a outras máquinas também. O *Winpcap* ainda adiciona a capacidade de filtrar e guardar em um *buffer* os pacotes capturados.

Neste trabalho foi instalada a versão 4.1.2 do *Winpcap* que foi obtida de seu site oficial[10].

3.3.3 *MySQL* Hospedado na Internet

O *Snort* possui diversos módulos de saída dos alertas gerados, dentre deles está módulo de saída para banco de dados. Esse módulo permite ao *Snort* trabalhar com uma variedade de servidores de bancos de dados. É preciso definir detalhes da conexão como o endereço do servidor, nome da base de dados, usuário e senha com as devidas autorizações para manipular os registros. Todas essas configurações são realizadas no arquivo de configuração "snort.conf", que fica no subdiretório "/etc" do diretório raiz do *Snort*.

Para a arquitetura desse trabalho foi utilizado o *plug-in Snort-MySQL*, que permite gerar os alertas diretamente em um servidor *MySQL*.

Como há uma considerável oferta de servidores de *MySQL* grátis na Internet, este projeto optou por usar um desses serviços na nuvem oferecido pelo provedor Xeround[11], pois além de diminuir o esforço de instalação e configuração de um servidor na infraestrutura local do projeto, também ganha-se em segurança, pois ao

[10] Disponível em http://www.winpcap.org.

[11] Serviço oferecido em http://www.xeround.com.

não ter uma porta aberta para o banco de dados, o fato do atacante terá mais dificuldades em descobrir que existe uma comunicação entre o *host* alvo e uma ferramenta de monitoração de ataques.

Sendo assim, a seguinte configuração foi feita no arquivo “snort.conf”:

```
output database: alert, mysql, user=clautergatti
password=oracle1234 dbname=snortdb
host=instance18577.db.xeround.com port=
output database: log, mysql, user=clautergatti
password=oracle1234 dbname=snortdb
host=instance18577.db.xeround.com port=12335
```

A próxima seção irá inicialmente conceituar o que é arquitetura de software e então descrever a arquitetura implantada para a ferramenta *Hamdroid*.

3.3.4 Arquitetura do *Hamdroid*

Diversas definições são dadas para arquitetura de software na literatura. Dentre elas, destacam-se:

“Descrição dos elementos a partir dos quais os sistemas são construídos (componentes), interações entre estes elementos (conectores), padrões que guiam sua composição, e restrições sobre estes padrões” (SHAW e GARLAN, 1996).

“A arquitetura de um sistema consiste da(s) estrutura(s) de suas partes (incluindo as partes de software e hardware envolvidas no tempo de execução, projeto e teste), da natureza e das propriedades relevantes que são externamente visíveis destas partes (módulos com interfaces, unidades de hardware, objetos), e dos relacionamentos e restrições entre elas” (D’SOUZA e WILLS, 1999).

A arquitetura desempenha um papel importante no desenvolvimento de softwares, pois é através dela que se compreendem questões estruturais importantes, como a divisão dos sistemas em componentes ou subsistemas, interconexões entre os componentes, seleção de alternativas de projeto e atribuição de funcionalidades aos componentes de projeto e outros.

Os aspectos estruturais não são os únicos a serem considerados nas definições de arquitetura para se obter uma completa e compreensível descrição arquitetural de um software. É necessário também considerar aspectos como

distribuição física, processo de comunicação e sincronização, entre outros. Cada um destes aspectos deve ser tratado em uma diferente visão arquitetural.

Voltando o foco para a ferramenta *Hamdroid*, esta foi concebida para trabalhar divida em três camadas físicas: uma rede local com o(s) servidor(s) monitorado(s), um dispositivo móvel e uma terceira camada com um servidor de aplicação e um banco de dados.

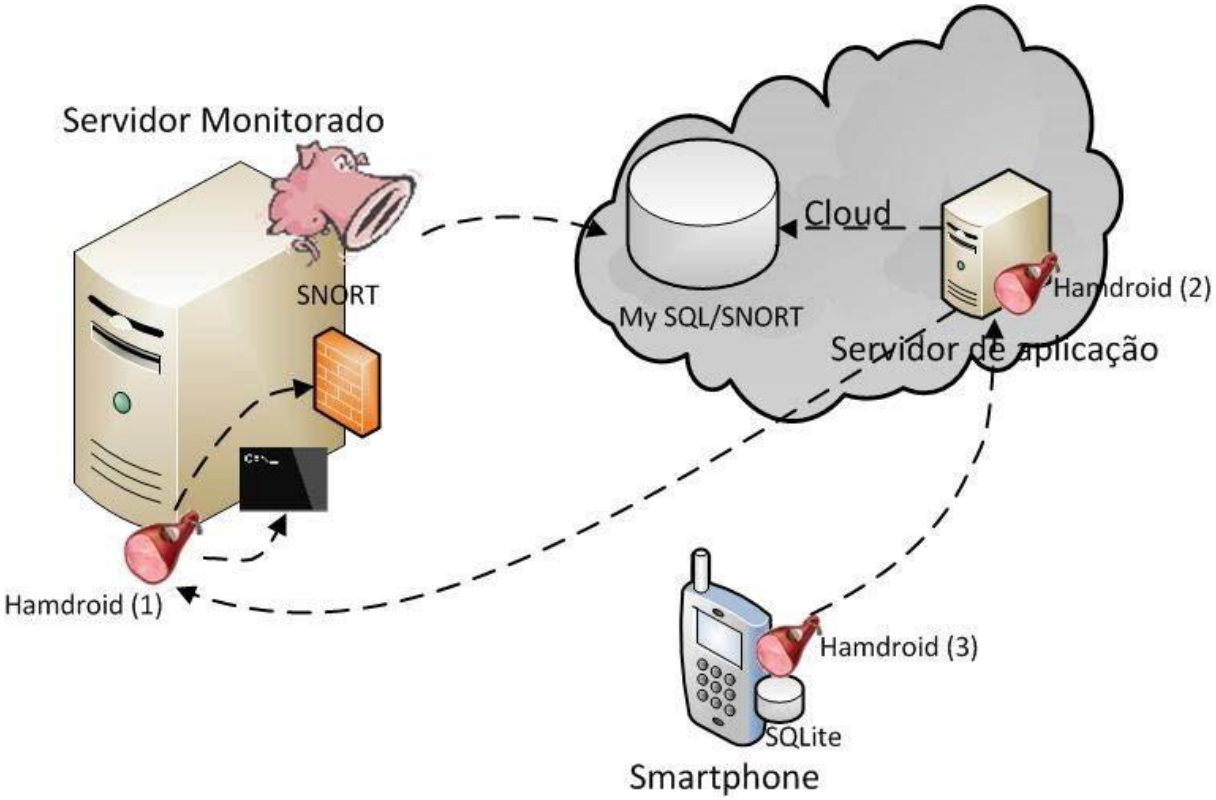


Figura 26: Arquitetura do *Hamdroid*.

Entre seus componentes (conforme representado na Figura 26), destacamos os seguintes:

- **Servidor Monitorado**: componente exposto na Internet sujeito as vulnerabilidades já abordadas por este trabalho.

- ***Snort:*** IDS instalado junto ao servidor monitorado, com suas devidas regras de monitoramento e com acesso a Internet.

- **MySQL**: banco de dados hospedado na Internet e que fica acessível ao IDS, para que o mesmo possa persistir os seus alertas.

- **Servidor de Aplicação**: componente que contém a camada “servidor” do *Hamdroid.* Este é quem faz a ponte entre o banco de dados, servidor monitorado e o componente “cliente” do *Hamdroid*.

- ***Hamdroid* Servidor**: camada responsável para ler os alertas persistidos no banco de dados e disponibilizá-los em formato de *Web Service* para a camada “cliente”. Esta ainda disponibiliza um segundo *Web Service* para o recebimento da tomada de ação a ser executada no servidor monitorado.

- ***Hamdroid* Cliente**: camada que fica no dispositivo móvel e que faz a interface com o usuário.

- **SQLite**: banco de dados local para armazenar as configurações de conexão, alertas baixados da camada “servidor” e as tomadas de ação personalizadas.

- **Smartphone**: dispositivo móvel com sistema operacional *Android* e com acesso a Internet.

O servidor de aplicação veio a ser adicionado à arquitetura devido a dificuldades encontradas para fazer um componente DAO (*Data Access Object*) na camada “cliente”.

3.4 CASOS DE USO DA FERRAMENTA

Nesta seção estão dispostos os casos de uso que foram levados em consideração para a ferramenta proposta para esse trabalho.

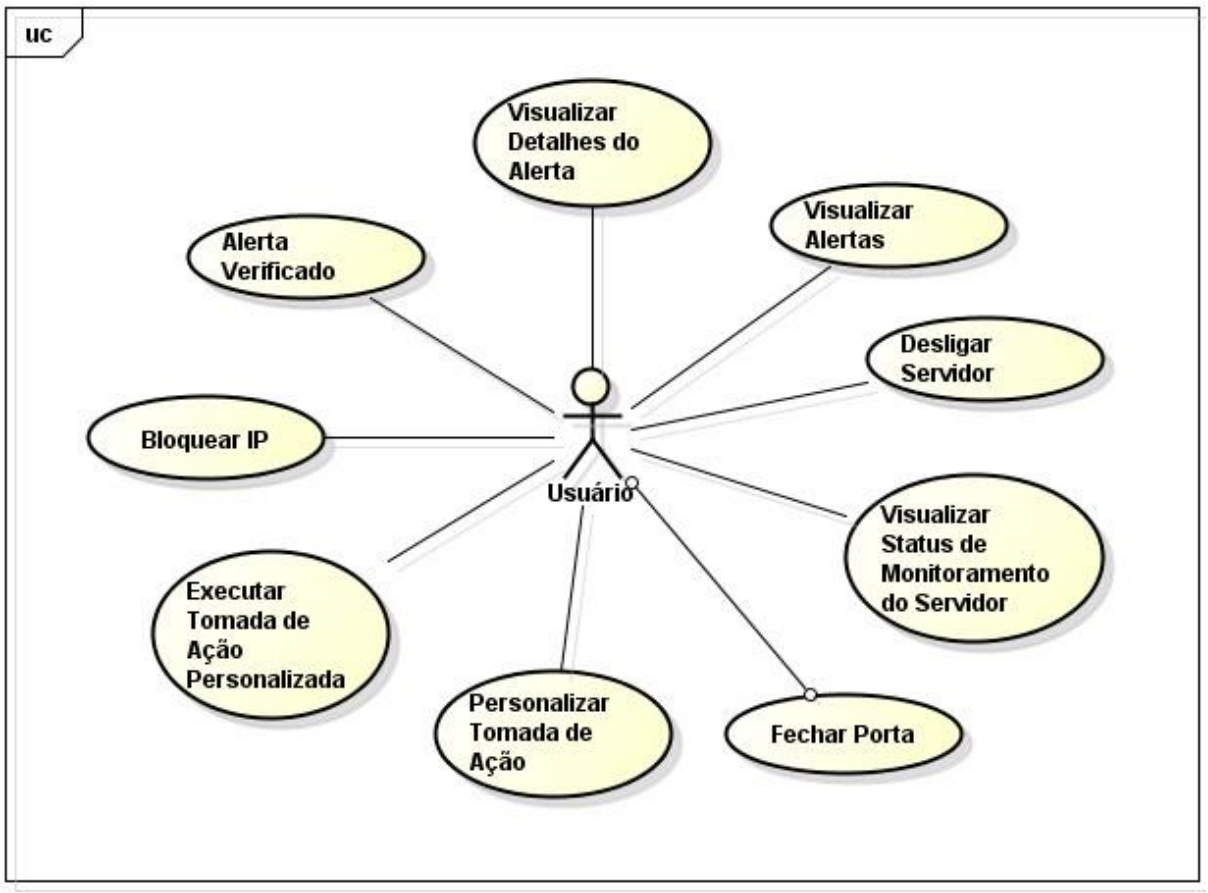


Figura 27: Diagrama de casos de uso.

A seguir será descrito todos os casos de uso ilustrados na Figura 27.

3.5 DESCRIÇÃO RESUMIDA DOS CASOS DE USO

Os casos de uso oferecem uma alternativa para que o ator do sistema tenha maior controle sobre seu servidor Linux ou Windows em termos de segurança. O *Hamdroid* provê uma gama de funcionalidades com acesso imediato aos servidores que estão sobre seu monitoramento. Foram selecionados nove casos de uso que apresentaram maior relevância para o controle de servidores através de dispositivos móveis. Levou-se em consideração a distância em que o usuário se encontra do acesso ao servidor, a simplicidade e o tratamento das informações que serão disponibilizados num *smartphone*.

As operações do *Hamdroid* resumem-se em tomar medidas rápidas para se defender de ataques em tempo real. Para isso, estão presentes operações como a visualização de status do servidor (sobre ataque, sem ataques, sem conexão), o

bloqueio de IP's de invasores para evitar ataques futuros destes endereços, ações personalizadas e dinâmicas criadas pelo usuário para serem executadas a qualquer momento, e o recebimento de alertas que permitem ao usuário tomar uma decisão rápida mediante o tipo do ataque apresentado.

3.6 DETALHAMENTO DOS CASOS DE USO

Nesta seção serão apresentados os detalhamentos de cada caso de uso, cuja finalidade é descrever com detalhes o fluxo de eventos do caso de uso e o fluxo de eventos do caso de uso para que o cliente e os usuários possam compreendê-lo.

3.6.1 Caso de Uso – Bloquear IP

A Tabela 1 abaixo detalha o caso de uso de bloqueio de um endereço IP.

Tabela 1: Detalhamento do caso de uso Bloquear IP.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	1. Bloquear IP
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Usuário solicita serviço via dispositivo móvel.
Pré-Condições	- Servidor deve estar operacional e conectado a Internet. - Dispositivo móvel deverá estar conectado a Internet.
Pós-condições	IP bloqueado pelo servidor.
Regras de Negócios	- O usuário só pode informar IP's válidos - Esta é uma operação reativa para o usuário bloquear IP's que que foram considerados perigosos de acordo com alerta *Snort* vindo do servidor.
Descrição: O usuário recebe em seu dispositivo móvel um alerta *Snort* contendo informações de IP's que estão invadindo o servidor. Para resolver este incidente, o usuário pode, através do dispositivo móvel, bloquear o acesso do IP invasor no servidor.	
DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	O usuário entra na opção de bloquear IP.
2	O usuário informa o IP e clica em bloquear.
3	O Sistema informa que o IP foi bloqueado.
Fluxo alternativo	
A 1	2.1 IP inválido.
A 2	3.1 Falha de comunicação com o servidor.

3.6.2 Caso de Uso – Visualizar Alertas

A Tabela 2 abaixo detalha o caso de uso de visualização dos alertas gerados pelo Snort:

Tabela 2: Detalhamento do caso de uso Visualizar Alertas.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	2. Visualizar Alertas
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Dispositivo móvel faz uma requisição de alertas no servidor.
Pré-Condições	- Servidor deve estar operacional e conectado a Internet. - Dispositivo móvel deverá estar conectado a Internet.
Pós-condições	Usuário é informado dos alertas gerados pelo servidor
Descrição: O usuário é notificado da existência de alertas ainda não verificados.	
DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	O usuário abre a aplicação *Hamdroid* e na tela inicial é apresentada uma lista contendo todos os alertas ainda não verificados.

3.6.3 Caso de Uso – Visualizar *Status* de Monitoramento do Servidor

A Tabela 3 abaixo detalha o caso de uso de monitoramento do *Status* do servidor:

Tabela 3: Detalhamento do caso de uso Visualizar *Status* de Monitoramento do Servidor.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	3. Visualizar Status de Monitoramento do Servidor
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Dispositivo móvel verifica status do servidor.
Pré-Condições	- Dispositivo móvel deverá estar conectado a Internet.
Pós-condições	Usuário informado do status do servidor
Regras de Negócios	Existem 3 status possíveis para o servidor: • Status Verde: Nenhum alerta foi encontrado no servidor; • Status Vermelho: Existem alertas que precisam ser verificados; • Status Branco: Sem conexão com servidor; A verificação de status no servidor será realizada constantemente pelo dispositivo móvel em um curto espaço de tempo.
Descrição: Quando há mudança de status, o dispositivo móvel sinaliza o usuário de tal evento informando a cor correspondente ao atual status do servidor.	

DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	Sistema informa a mudança de status ou o próprio usuário verifica em que status se encontra o servidor.

3.6.4 Caso de Uso – Fechar Porta

A Tabela 4 abaixo detalha o caso de uso de fechamento de porta no servidor monitorado:

Tabela 4: Detalhamento do caso de uso Fechar Porta.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	4. Fechar Porta
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Usuário solicita serviço via dispositivo móvel.
Pré-Condições	- Servidor deve estar operacional e conectado a Internet. - Dispositivo móvel deverá estar conectado a Internet.
Pós-condições	Porta fechada do servidor.
Regras de Negócios	- O usuário só pode informar portas válidas - Esta é uma operação reativa para o usuário fechar portas que podem estar sendo invadidas.
Descrição: O usuário recebe uma notificação de alerta no sistema contendo informações de portas que estão sobre ataque no servidor. Para resolver este incidente, o usuário vai até a opção fechar porta e digita o número da porta que deseja que o servidor feche.	
DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	O usuário entra na opção de fechar porta.
2	O usuário informa porta e clica em fechar.
3	O Sistema informa que o porta foi fechada.
Fluxo alternativo	
A 1	3.1 Porta inválida.
A 2	3.2 Erro ao tentar fechar a porta.

3.6.5 Caso de Uso – Desligar Servidor

A Tabela 5 abaixo detalha o caso de uso de desligamento do servidor monitorado:

Tabela 5: Detalhamento do caso de uso Desligar Servidor.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	5. Desligar Servidor
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Usuário solicita serviço via dispositivo móvel.
Pré-Condições	- Servidor deve estar operacional e conectado a Internet. - Dispositivo móvel deverá estar conectado a Internet. - Status do servidor verde ou vermelho.
Pós-condições	Servidor desligado e status alterado para branco.
Regras de Negócios	- Esta é uma operação reativa para o usuário evitar que a invasão resulte em dados comprometidos no servidor, como roubo, alteração exposição e outros.
Descrição: O usuário recebe em seu dispositivo móvel um alerta *Snort* contendo informações de IP's de invasores ou portas sendo invadidas no servidor. Para remediar o problema, o usuário pode desligar o servidor monitorado através do dispositivo móvel.	
DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	O usuário entra na opção de desligar servidor
2	O usuário clica em desligar.
3	Status alterado para branco (sem conexões com servidor).
Fluxo alternativo	
A 1	3.1 Não foi possível desligar o servidor, status inalterado.
A 2	3.2 Falha de comunicação com o servidor.

3.6.6 Caso de Uso – Personalizar Tomada de Ação

A Tabela 6 abaixo detalha o caso de uso de personalização de uma tomada de ação:

Tabela 6: Detalhamento do caso de uso Personalizar Tomada de Ação.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	6. Personalizar Tomada de Ação
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Usuário solicita serviço via dispositivo móvel.
Pré-Condições	- Servidor deve estar operacional e conectado a Internet. - Dispositivo móvel deverá estar conectado a Internet.
Pós-condições	Tomada de ação adicionada no sistema.
Regras de Negócios	- Esta é uma operação de criação de novas ações, permitindo maior flexibilidade e controle do administrador com seu servidor. Depois de criada a nova ação, o usuário estará apto a executá-la a qualquer momento.
Descrição: O usuário envia uma nova ação contendo o título e o comando contendo parâmetros para serem executados no servidor.	

DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	O usuário entra na opção de criar nova ação.
2	O sistema informa os campos de título, texto de comando e campo de adição de parâmetros.
3	Usuário preenche título, o texto de comando, adiciona zero ou mais parâmetro informado o nome de cada parâmetro e clica em salvar.
4	O sistema informa que o comando foi inserido com sucesso.
Fluxo alternativo	
A 1	4.1 Título ou comando inválido

3.6.7 Caso de Uso – Executar Tomada de Ação Personalizada

A Tabela 7 abaixo detalha o caso de uso de execução de uma tomada de ação personalizada junto ao servidor monitorado:

Tabela 7: Detalhamento do caso de uso Executar Tomada de Ação Personalizada.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	7. Executar Tomada de Ação Personalizada
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Usuário solicita serviço via dispositivo móvel.
Pré-Condições	- Servidor deve estar operacional e conectado a Internet. - Dispositivo móvel deverá estar conectado a Internet.
Pós-condições	Ação executada com sucesso.
Regras de Negócios	- Esta é uma operação para tomar uma ação personalizada do administrador.
Descrição: Esta é uma operação em que o usuário escolhe uma de suas ações personalizadas para ser por linha de comando no servidor.	
DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	O usuário entra na opção lista de operações personalizadas.
2	O sistema informa a lista.
3	Usuário clica na ação desejada
4	O sistema informa o título e os parâmetros de entrada para executar tal ação.
5	O usuário clica em executar.
6	O sistema informa o resultado da execução
Fluxo alternativo	
A 1	2.1 Não há ações personalizadas.
A 2	2.2 Falha de comunicação com o servidor.

3.6.8 Caso de Uso – Visualizar Detalhes do Alerta

A Tabela 8 abaixo detalha o caso de uso de visualização dos detalhes de um alerta gerado pelo Snort:

Tabela 8: Detalhamento do caso de uso Visualizar Detalhes do Alerta.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	8. Visualizar Detalhes do Alerta
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Usuário solicita serviço via dispositivo móvel.
Pré-Condições	- Servidor deve estar operacional e conectado a Internet. - Dispositivo móvel deverá estar conectado a Internet. - Existe pelo menos um alerta gerado pelo sistema.
Pós-condições	Usuário é informado dos detalhes do alerta selecionado.
Regras de Negócios	O usuário deverá estar na tela que possui a lista de alertas para ver visualizar os detalhes de cada alerta.
Descrição: O usuário seleciona um alerta da lista e em seguida o sistema apresenta uma lista contendo os detalhes da origem deste alerta.	
DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	O usuário seleciona o alerta que deseja verificar os detalhes.
2	O sistema informa ao usuário os detalhes deste alerta.
Fluxo alternativo	
A 1	2.1 Erro de conexão.

3.6.9 Caso de Uso – Alerta Verificado

A Tabela 9 abaixo detalha o caso de uso marcar um alerta como já verificado:

Tabela 9: Detalhamento do caso de uso Alerta Verificado.

DESCRIÇÃO DO CASO DE USO	
CASO DE USO	9. Alerta Verificado
Ator Iniciante	Usuário administrador do servidor.
Evento Iniciante	Usuário solicita serviço via dispositivo móvel.
Pré-Condições	- Servidor deve estar operacional e conectado a Internet. - Dispositivo móvel deverá estar conectado a Internet. - Existe pelo menos um alerta gerado pelo sistema. - Usuário deverá estar na tela de detalhes do alerta.
Pós-condições	Alerta é marcado como verificado e em seguida é removido do dispositivo móvel.

Descrição: O usuário marca como verificado os alertas que receberam algum tipo de tratamento ou que não apresentam impacto na segurança.	
DESCRIÇÃO DOS FLUXOS	
Fluxo básico	
1	O usuário seleciona o alerta que deseja verificar os detalhes.
2	O sistema informa ao usuário os detalhes deste alerta.
3	O usuário marca o alerto como verificado.
4	O alerta é removido do smartphone.
Fluxo alternativo	
A 1	2.1 Erro de conexão.

3.7 DIAGRAMAS UML

Nesta seção estão dispostos alguns diagramas UML da ferramenta *Hamdroid*.

Seguindo a estrutura cliente/servidor da ferramenta, os dois primeiros diagramas de classe são referentes à camada “cliente” e o terceiro à camada “servidor”.

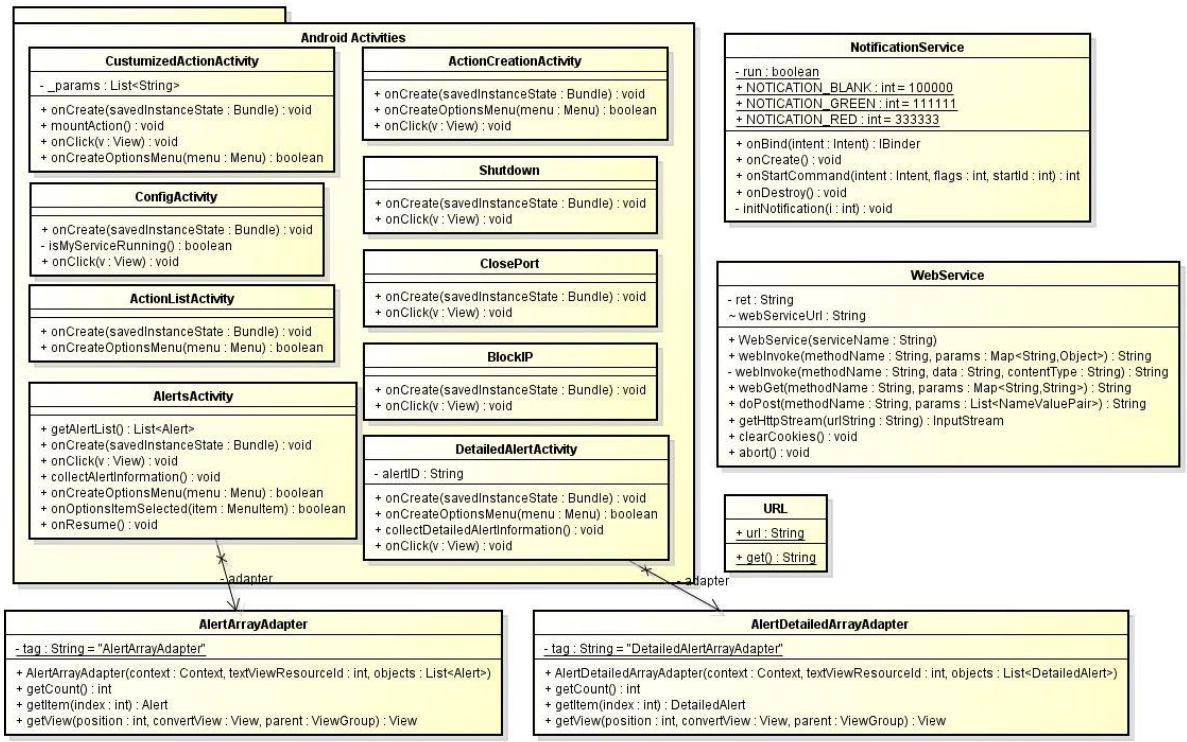


Figura 28: Diagrama de classes da camada cliente (alertas e ações).

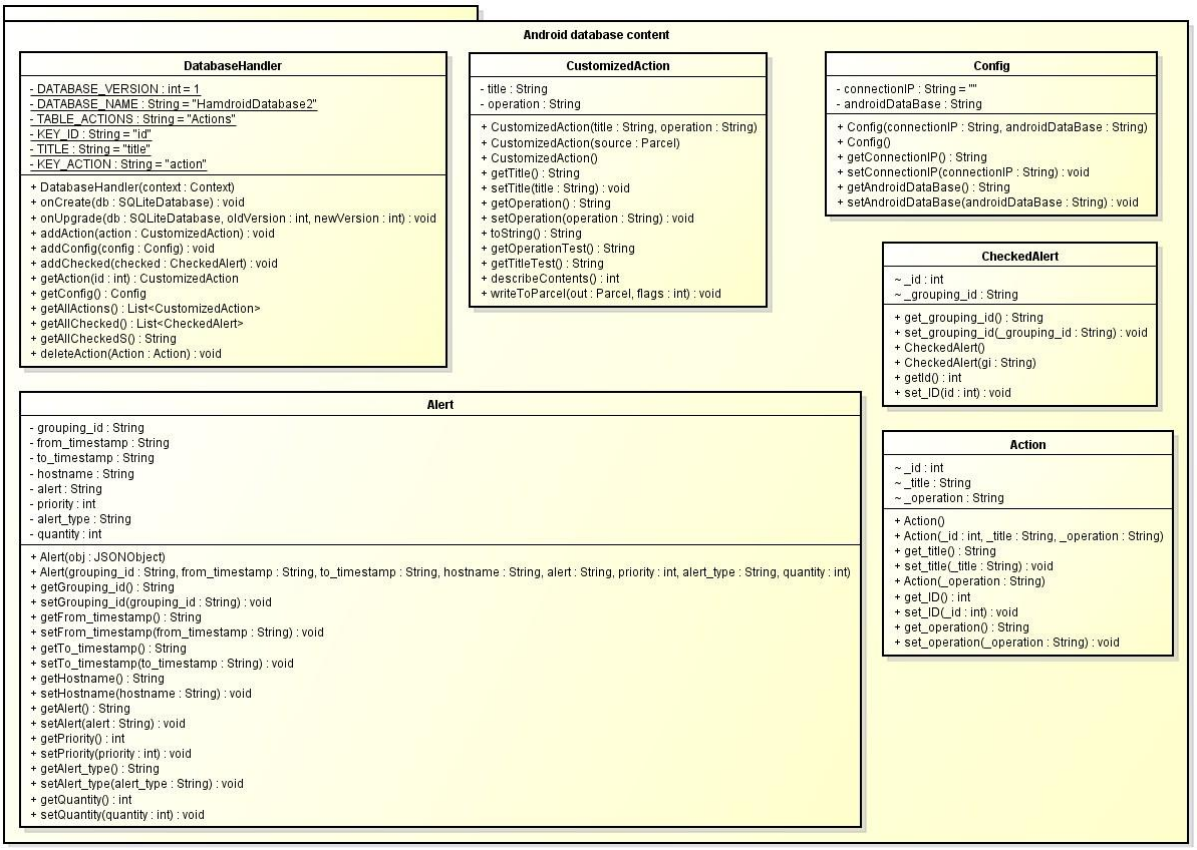


Figura 29: Diagrama de classes da camada cliente (conexão ao banco de dados).

Dentre os diagramas da camada cliente, a Figura 28 representa as classes referentes aos alertas e as tomadas de ações, enquanto a Figura 29 representa as classes referentes à conexão com o banco de dados.

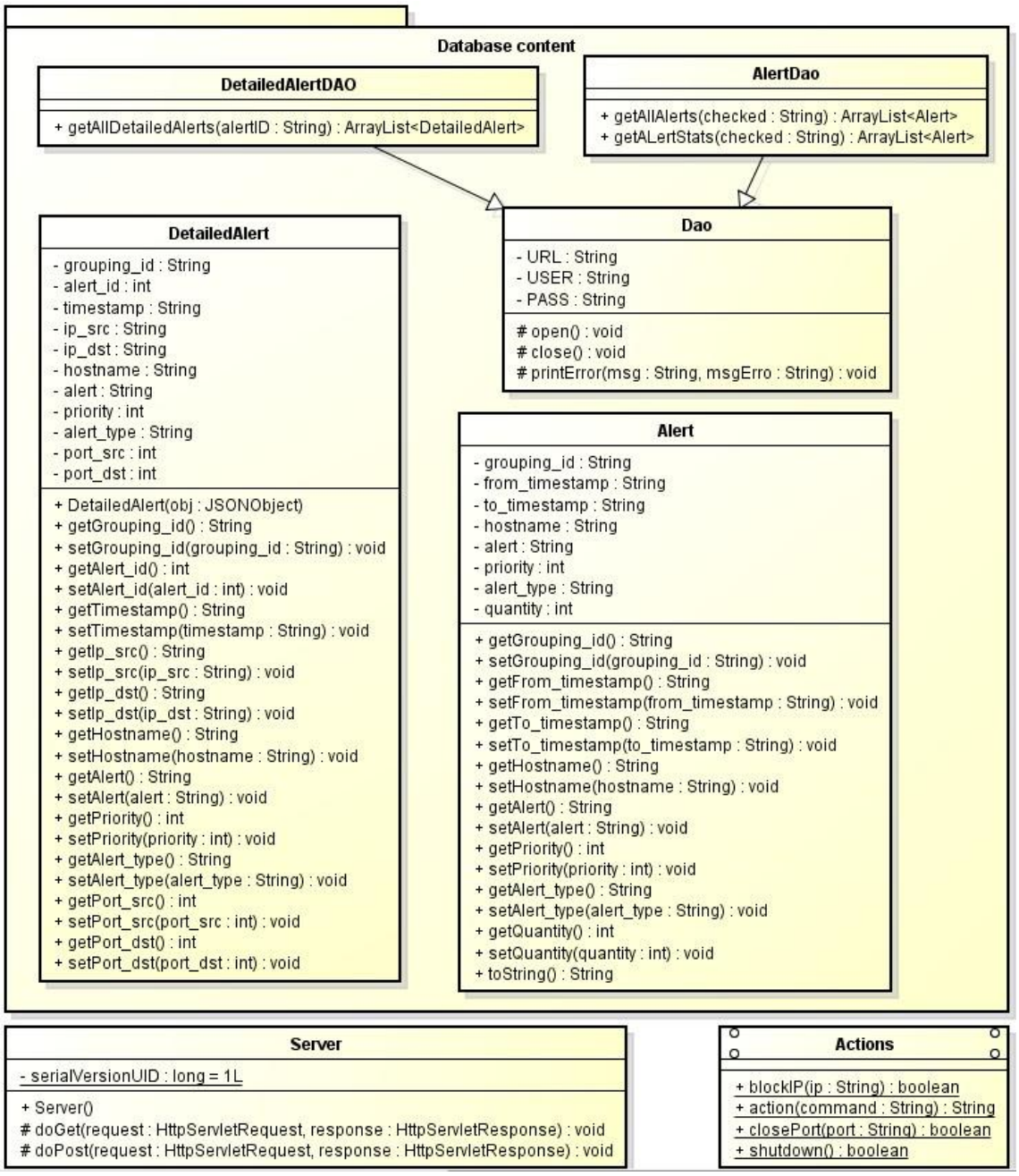


Figura 30: Diagrama de classes da camada servidor.

Já a Figura 30 representa as classes da camada “servidor” da ferramenta.

Para ilustrar como o *Hamdroid* se porta na execução de uma tomada de ação, abaixo (Figura 31), em forma de diagrama UML, está à ilustração da sequência de execução da ação de fechar porta no servidor monitorado.

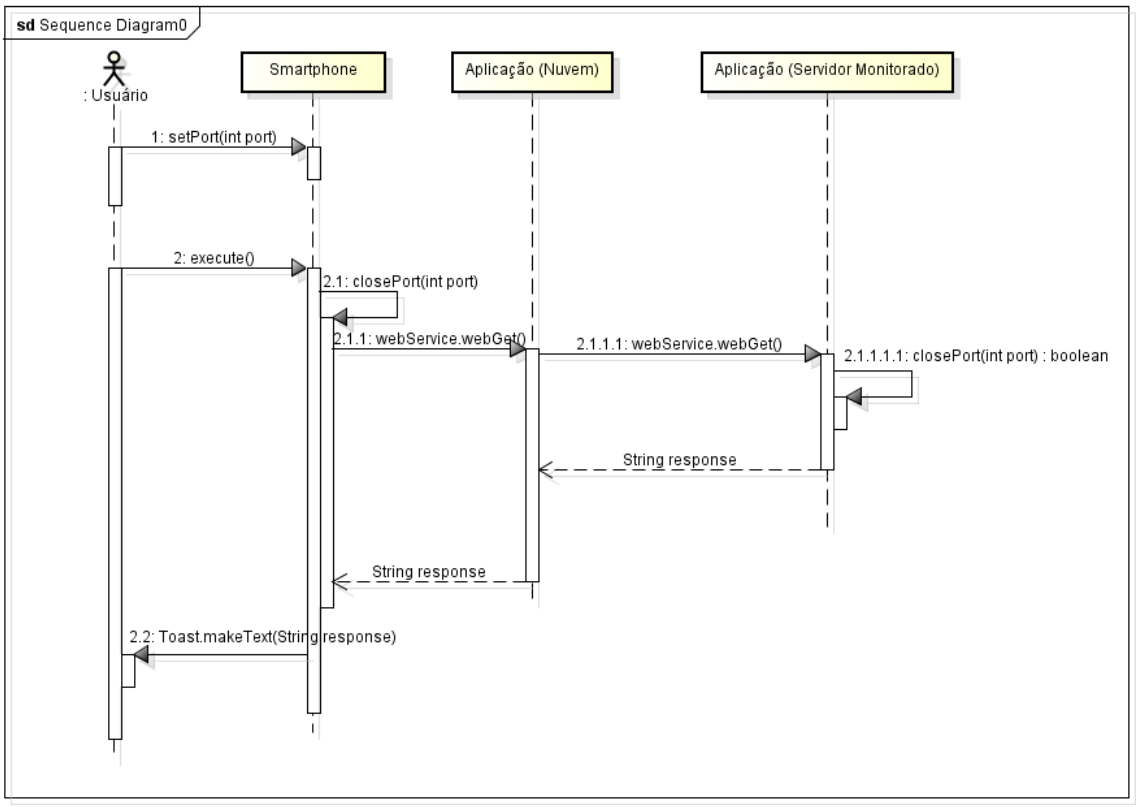


Figura 31: Diagrama de sequência da ação de fechar porta.

O usuário informa o valor da porta e clica no botão executar. Em seguida, o smartphone envia uma mensagem via *Web Service* para o servidor de aplicação na nuvem, que repassa essa informação para o servidor monitorado. Por fim, é executado o bloqueio da porta no próprio servidor e uma mensagem contendo o resultado deste processo é retornado para o usuário.

3.8 DIAGRAMAS ER

A versão do *Snort* utilizada por este trabalho acompanha scripts para criação do banco de dados, um para cada tipo de banco de dados. Conforme abordado anteriormente, foi escolhido o banco de dados *MySQL*, e portanto o script utilizado foi o “create_mysql.sql”, este disponível no subdiretório “/schemas” do diretório raiz do *Snort*.

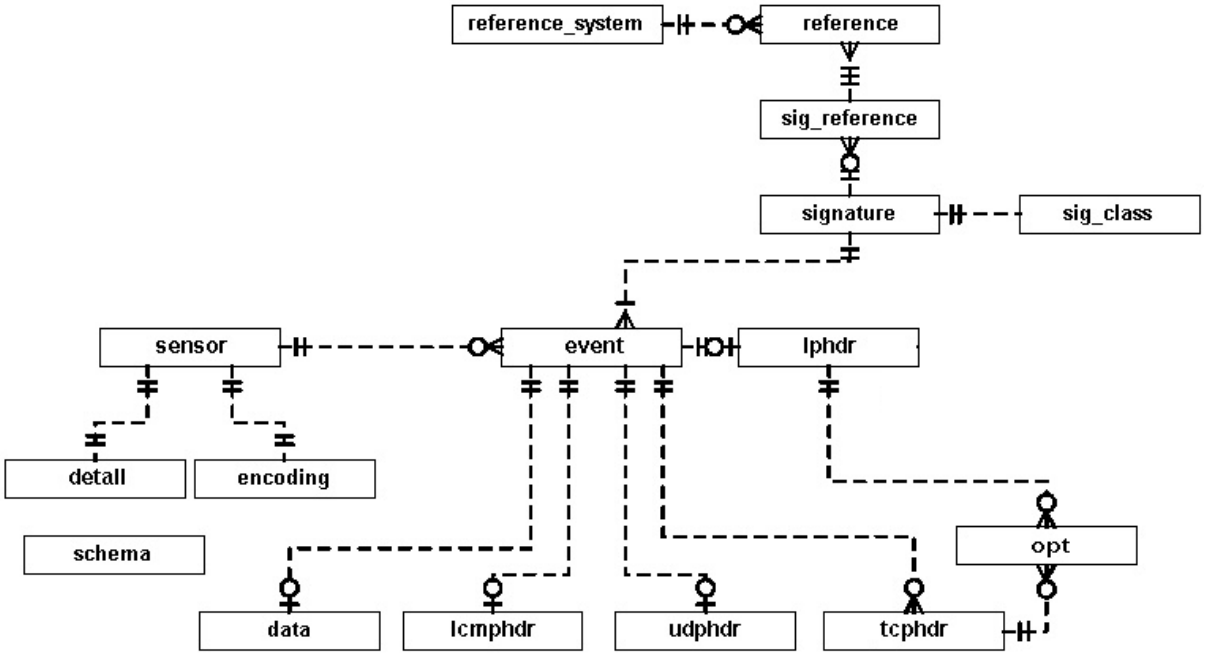


Figura 32: Diagrama ER do banco de dados do *Snort*.

Fonte: (ACIDLAB, 2012)

A Figura 32 mostra o diagrama de entidade relacionamento do banco de dados do *Snort*, que destaca-se o fato de ter uma tabela para cada tipo de alerta gerado pelos protocolos de redes ICMP, UDP e TCP.

Além das tabelas exigidas pelo *Snort*, ainda foi criada uma tabela adicional para persistir as preferencias de configuração da ferramenta *Hamdroid*. A tabela adicional foi nomeada como *Preferences*, e funciona como uma tabela *Hash* para armazenar a preferencia e o valor.

Os atributos da tabela *Preferences* são representados a seguir na Tabela 10:

Tabela 10: Atributos da tabela *Preferences*.

Coluna	**Tipo de Dados**	**Chave Primária**
Preference	varchar(100)	Sim
value_int	int(11)	
value_varchar	varchar(300)	
value_timestamp	Timestamp	

3.9 VISÕES DE BANCO DE DADOS

Os registros de alertas gerados pelo *Snort* permitem ao administrador da rede verificar as tentativas de intrusão e no máximo, tomar conhecimento que elas estão acontecendo. Entretanto, analisar um conjunto de registros distribuídos entre tabelas de um banco de dados não é uma tarefa fácil para um ser humano.

O *Snort* na sua instalação padrão não possui nenhuma ferramenta que agrupe, resuma e ordene os alertas por prioridade, o que facilitaria o trabalho do administrador da rede, podendo assim concentrar seus esforços nas vulnerabilidades do sistema e não no garimpo de informações de *log*.

Pensando nisso, este projeto criou visões de banco de dados com a finalidade de agrupar, resumir e ordenar os alertas por prioridade.

Colares (2007, p. 63) define visão de banco de dados como:

> "Uma visão de banco de dados ou *view* é uma tabela virtual, uma tabela resultante de uma consulta efetuada sobre uma ou mais tabelas. As *views* suprem varias funções. Podem, por exemplo: substituir consultas longas e complexas por algo mais fácil de ser entendido e manipulado; funcionar como elemento de segurança, a partir do momento em que conseguem limitar o acesso dos usuários a determinados grupos de informações armazenadas em um banco de dados".

Foram criadas duas visões, uma com a finalidade de prover os dados resumidos para a tela principal do *Hamdroid*, ponderando as limitações de tamanho de tela dos dispositivos móveis, e uma segunda visão com maiores detalhes dos alertas. A Tabela 11 mostra detalhes das duas visões:

Tabela 11: Visões de banco de dados do *Hamdroid*.

View	**Descrição**	**Colunas**	**Clausula *Where***
Alerts	Consolida e resume os alertas	*grouping_id, from_timestamp, to_timestamp, hostname, alert, priority, alert_type, quantity*	Excluem alertas mais antigos que o limite definido na tabela de preferencias do usuário.
Alert_Details	Detalha os alertas	*alert_id, grouping_id, timestamp, hostname, alert, priority, alert_type,*	

		ip_src, port_src, ip_dst, port_dst	

A visão Alerts ainda possui uma cláusula *where* que exclui os alertas mais antigos que um limite definido pela tabela de preferências do usuário. Essa funcionalidade tem como objetivo fazer com que a sincronização entre o dispositivo móvel e o banco de dados do *Snort* seja rápida, não perdendo tempo crucial em um eventual caso de ataque a rede de computadores.

O script em SQL que gera as visões de banco de dados no *MySQL* está apresentado no Anexo A.

3.10 PLANO DE TESTES

A ideia inicial era elaborar casos de testes que cobrissem desde a simulação de um ataque a uma rede de computadores até a usabilidade e desempenho da ferramenta *Hamdroid*, contudo ao longo do desenvolvimento deste projeto foi constatado que esta abordagem seria inviável. O racional para esta constatação foi que a simulação de ataque a uma rede colocaria a mesma em risco, pois a simulação envolve fazer o ataque em si usando vírus e trojans e também entendemos que o objetivo do trabalho é a ferramenta de monitoramento e não testar os IDS. Esta última constatação foi o que levou a focar os testes especificamente na ferramenta, e a decisão foi em simular os alertas gerados pelo IDS e não propriamente os ataques para que então o IDS gerasse os alertas.

O estudo de como simular os alertas do *Snort* levou até ao estudo de como funcionam as regras de geração de alertas, pois uma vez entendido como elas funcionam, foi possível criar regras para gerar alertas falsos positivos e então simular os dados para os cenários pretendidos.

Nas próximas seções vamos abordar os tipos de testes cobertos por esse projeto, então falaremos um pouco de como as regras de alerta do *Snort* funcionam, quais foram os alertas modificados para gerar os falsos positivos e finalmente abordaremos os casos de testes.

3.10.1 Tipos de Testes

O que se espera de softwares é que eles atendam a certos atributos de qualidade e, para isso, é preciso identificar seus objetivos ou requisitos de qualidade, requisitos funcionais, de desempenho e de custos. Para entender melhor entender e identificar esses atributos de qualidade, vamos falar rapidamente sobre processos e tipos de testes, e então definir os tipos de testes que foram planejados para este projeto.

Mas antes de tudo é importante a identificação de alguns termos para uma melhor compreensão do assunto. A seguir, algumas definições, de acordo com Magela (2006):

- Erro: Resultado inconsistente na execução do software devido a uma ou mais falhas. Visão de verificação.
- Falha: Violação da especificação presente no código-fonte de um componente.
- Defeito: Conjunto de falhas que, quando provoca um comportamento inconsistente do software. Visão de validação.

Os tipos de teste de softwares não são aplicados ao esmo, mas sim dentro de um planejamento já no inicio do projeto do desenvolvimento do software, onde deve haver uma fase só de elaboração do planejamento do teste. Segundo Crespo (2004), essa fase de planejamento de teste compreende a definição dos seguintes itens: O nível de teste, isto é, a definição da fase do desenvolvimento do software em que o teste será aplicado; A técnica de teste a ser utilizada; O critério de teste a ser adotado; O tipo de teste a ser aplicado no software.

A Figura 36 ilustra graficamente a ligação existente entre os níveis de teste, as técnicas de teste, os tipos de teste e os critérios de teste que podem ser adotados ao se definir uma estratégia de teste (CRESPO 2004).

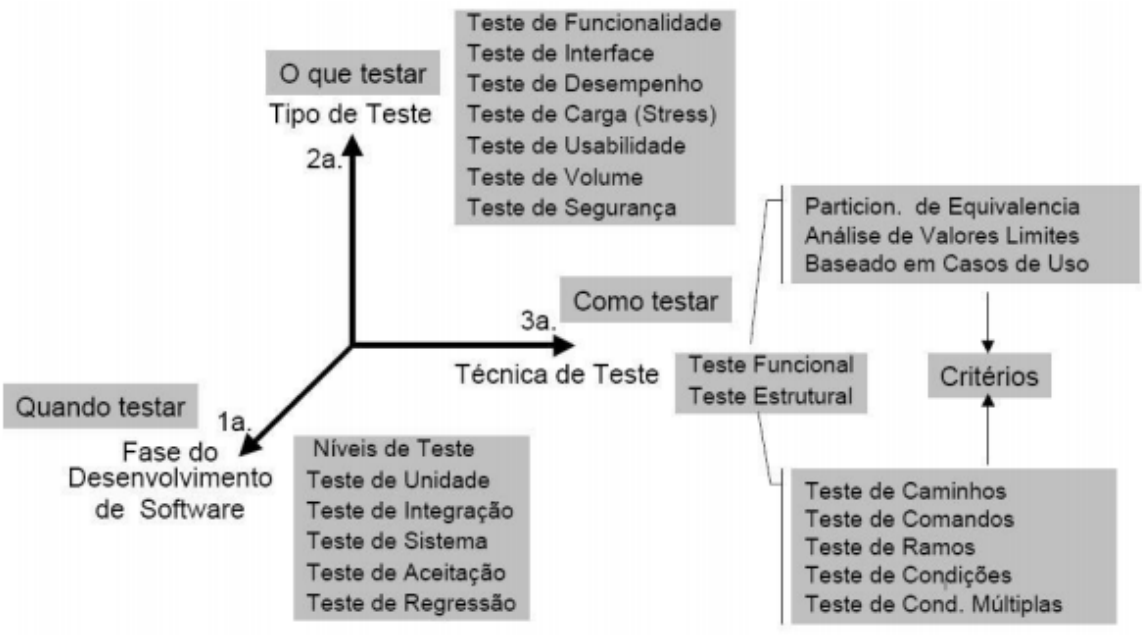


Figura 33: Relação entre níveis, técnicas e tipos de testes.

Fonte: (CRESPO, 2004)

Existem dois tipos de processo de testes que são facilmente confundidos: verificação e validação. Podemos diferencia-los como validação sendo a análise se o produto que está sendo construído é de fato o produto esperado, e verificação sendo a análise se o produto está sendo construído corretamente. Para esses dois tipos de processos são usados geralmente os termos testes funcionais e testes não funcionais.

Abaixo abrangeremos alguns destes tipos de teste:

- **Testes Funcionais**: De acordo com Magela (2006) os testes funcionais são um processo de tentativa para encontrar discrepâncias entre o programa e as especificações externas que são descrições precisas do funcionamento do programa do ponto de vista do usuário final. Os testes funcionais usados em pequenos programas são uma atividade de caixa preta.

- **Não Funcionais**: Segundo Magela (2006) os testes funcionais não fornecem a garantia necessária para um software entrar em produção e exemplifica da seguinte forma: embora todos os requisitos do usuário possam ser atendidos, se para cada tarefa o usuário esperar dez

minutos para sua execução, o software está com problema de desempenho e sem dúvida, esse problema inviabiliza a liberação do software, por isso os testes não-funcionais são necessários. São apresentados a seguir alguns exemplos de testes não-funcionais: usabilidade, volume, ambiente, segurança, desempenho, instalação e recuperação.

- **Usabilidade**: está relacionada à eficácia e eficiência da interface diante do usuário e pela reação do usuário diante da interface. É um teste que foca em verificar se, por exemplo, as saídas do software são de fácil entendimento e do usuário, ou se o software é de fácil operação pelo usuário.

- **Volume**: Segundo Magela (2006), este tipo testa a capacidade do sistema de suportar um grande volume de dados sem preocupação nenhuma com o tempo, que é o teste de carga.

- **Ambiente**: Os testes de ambiente suportam os requisitos de ambiente. Limitações como temperatura local, umidade e interferência dentre outros pontos devem ser verificadas e validadas (MAGELA, 2006).

- **Desempenho (*performance*)**: tem como finalidade assegurar a qualidade do sistema tornado sua execução rápida, confiável e segura. Seu objetivo principal é encontrar erros, mas este tipo de teste também serve para se obter medidas como confiabilidade ou desempenho, através da realização de testes de desempenho podem–se obter medidas para verificar como é o comportamento do sistema em alguns aspectos como, por exemplo, se um número muito grande se usuários estiverem utilizando o sistema simultaneamente ou quando houver uma massa muito grande de dados para ser processada.

Após entender alguns dos tipos de testes, voltamos nosso foco para a ferramenta *Hamdroid*, pois deseja-se definir os tipos de testes que seriam mais

interessantes para o contexto deste trabalho. Visto que uma das questões abordadas por esse projeto era a limitação e benefícios da mobilidade, logo entendeu-se que um teste de usabilidade poderia avaliar se o objetivo perseguido de ter uma ferramenta fácil e ágil para alertar e tomar decisões foi alcançado ou não. Outro ponto crucial era de que não só bastaria ter uma ferramenta de boa usabilidade, mas também deveria ser rápida o bastante para lidar com ataques DDOS, cujo volume de alertas seria grande, e a ferramenta teria que lidar com esse volume sem impactar a agilidade em tomar uma decisão.

Concluiu se então que os tipos de testes cobertos por esse projeto seriam o de Teste de Usabilidade e o de Teste de Carga. No próximo capitulo vamos entender melhor estes dois tipos de testes.

3.10.1.2 Testes de Usabilidade

Segundo USDHHS (2012), U.S. *Department of Health & Human Services*, os principais fatores de medição de usabilidade em um teste são:

- **Efetividade**: Habilidade do usuário em conseguir encontrar informações e executar suas tarefas em um *software*.

- **Eficiência**: A capacidade de um usuário de executar suas tarefas rapidamente, com facilidade e sem frustração.

- **Satisfação**: Quanto um usuário gostou de usar um *software*.

- **Frequência de erros e severidade**: Com que frequência os usuário se deparam com erros durante a utilização do sistema, quanto graves são estes erros e como fazer os usuários contornarem estes erros.

- **Memorização**: Se o usuário tiver utilizado o sistema antes, o quanto ele consegue relembrar do que já aprendeu ou se ele tem que aprender tudo novamente.

A usabilidade é relacionada à eficácia e eficiência da interface diante do usuário e pela reação do usuário diante da interface (FERREIRA, 2002). O teste de usabilidade é um processo onde alguns usuários avaliam o grau o *software* se encontra em relação a critérios específicos de usabilidade.

Segundo Ferreira (2002, p. 14) define que em um teste de usabilidade, o processo se inicia com a escolha de um avaliador do teste. O avaliador é responsável por tudo que ocorre durante a sessão de teste. Sua função é interagir com o participante, coletar informações, compilar e comunicar o resultado dos testes para a equipe de desenvolvimento. Em seguida vem a definição do plano de teste, que consiste na base de todo o teste, especificando como, quando onde, quem, o porquê e o quê sobre o teste de usabilidade. Esse plano deve conter o propósito, a descrição do problema, objetivos, perfil dos usuários que irão executar os testes, a metodologia, lista de tarefas, ambiente de teste, papel do avaliador, medidas de avaliação e o relatório esperado com os resultados do teste. O mesmo autor ainda define que a seleção dos participantes é algo crucial para o sucesso efetivo do processo de teste, envolvendo a identificação e descrição de habilidades relevantes e o conhecimento do pessoal que irá usar o produto. A determinação do perfil e caracterização do usuário são determinados nos primeiros estágios do desenvolvimento do produto e servem como base para a seleção dos participantes. Por fim, ele define que o avaliador deve ter um roteiro que serve como guia para orientar o avaliador durante a sessão de teste. Bastante semelhante ao plano de teste, descreve o ambiente que será utilizado, as funções do avaliador, o perfil do participante, tarefas do sistema, procedimentos e uma lista dos formulários utilizados.

3.10.1.2 Testes de Desempenho

Os testes de desempenho precisam ser projetados para assegurar que o sistema possa processar sua carga pretendida, isso envolve planejar uma série de testes, onde a carga é constantemente aumentada até que o desempenho do sistema se torne inaceitável (SOMMERVILLE, 2003). Ainda segundo Sommerville (2003), os testes de desempenho podem ser divididos em basicamente três tipos:

- **Carga**: têm por finalidade testar o desempenho do sistema levando-se em conta uma carga de transações simultâneas, onde é possível observar alguns fatores como o tempo de resposta do sistema e ocorrência de erros.

- **Estresse**: visam avaliar o desempenho do sistema com um grande número de usuários, geralmente são inseridos usuários além da capacidade máxima que o sistema pode suportar. Os testes de estresse continuam além da carga máxima para a qual o sistema foi projetado, até que o sistema falhe.

- **Volume**: testa a quantidade de dados que o sistema pode gerenciar, o objetivo deste teste é determinar a capacidade do sistema em lidar com o volume de dados especificado nos seus requisitos. Em geral, este tipo de teste usa grandes quantidades de dados, o que permite determinar os limites em que o sistema falha. Além disso, costumam ser utilizados na identificação da carga máxima ou volume de dados que o sistema pode gerenciar em um dado período de tempo (PRESSMAN, 1995).

Voltando para a ferramenta *Hamdroid*, lembramos que ela foi desenvolvida para dispositivos móveis que possuem certa limitação de capacidade de processamento e de memória gerando a dúvida de até quantos alertas simultâneos a ferramenta conseguirá lidar. Para entendermos qual seria a capacidade da ferramenta em lidar com volumes grandes alertas então estabelecemos um teste de carga.

Na próxima seção será abrangido como funcionam os alertas do *Snort*, pois é necessário conhecer como se funcionam as regras de geração de alertas para poder alcançar os devidos cenários para a execução dos testes estabelecidos nesta seção.

3.10.2 Alertas do *Snort*

As regras de alerta do *Snort* são as peças vitais de seu funcionamento. Saber criá-las pode ser uma excelente arma para o administrador de redes e para este trabalho conhecer o funcionamento delas foi vital para poder simular alertas de ataques desejados para os testes da ferramenta *Hamdroid*.

O *Snort* possui uma linguagem fácil na construção de novas regras que podem ser codificadas em uma única linha. Como pode ser observado usa-se o termo regra e não assinatura. O termo assinatura refere-se a nada mais do que uma definição básica de ataque, parecido com uma pegada deixada na lama pelo sapato de alguém invadindo uma casa. Uma regra define a metodologia de ataque em termos de identificação do invasor, análogo à identificação do modo como o ladrão entrou na casa, na esperança de capturá-lo.

As regras do *Snort* são divididas em duas partes lógicas: cabeçalho e miolo. O cabeçalho contém a ação da regra, protocolo, endereço IP, máscara de rede e porta. O miolo contém mensagem de alerta e informações sobre que parte do pacote deverá ser examinada. As regras do *Snort* obedecem a seguinte sintaxe descrita na Figura 34:

```
<tipo_de_alerta> <protocolo> <rede_origem> <porta_origem>
     -> <rede_destino> <porta_destino>
     (Cabeçalho da Regra; <opções>; sid:X;...);
```

Figura 34: Sintaxe básica para definição de regras do *Snort*.

Um exemplo bem simples pode ser visto na Figura 35. Neste exemplo, o tipo de alerta é "*alert*", o protocolo é TCP, a porta de origem é qualquer, a rede de destino é 192.168.1.0/24 e a porta de destino é 111:

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86
a5|"; msg:"mountd access";sid:11)
```

Figura 35: Exemplo de definição de regra do *Snort*.

A partir desse modelo, é possível construir regras complexas e bem úteis para manusear o *Snort*. As possibilidades de manuseio dos campos das regras são:

Para o campo <tipo_de_alerta> temos:

- ***Activate***: É a mais poderosa do *Snort*. Ela gera um alerta e depois inicia a regra dinâmica especificada. Ela é muito útil para capturar ataques complexos.
- ***Dynamic***: Essa ação permanece inativa até que uma regra active a dispare;
- ***Alert***: Registra o pacote e depois alerta o usuário da forma especificada no arquivo de configuração;
- ***Pass***: Ignora esse pacote, continuando a análise nos pacotes subsequentes;
- ***Log***: Registra o pacote (Não Alerta).

Essas são as cinco ações pré-existentes no *Snort*, mas ainda pode-se se criar outros tipos de regras personalizadas.

Quanto ao miolo da regra, o formato é dividido em seções separadas por ponto e vírgula (;). Cada seção define uma opção, seguida do valor da opção desejada. O miolo da regra fica entre parênteses. Vamos ver algumas opções que foram usadas por este trabalho:

- ***nocase***: Não faz distinção entre maiúscula e minúscula;
- ***content***: procura por um molde no conteúdo do pacote. A busca pode ser realizada por conteúdo ASCII ou binários, podendo negar uma expressão com o ponto de exclamação (!). A expressão a ser localizada deve vir entre aspas dupla. A seguir segue um exemplo de regra, que gera um alerta sempre que uma conexão Telnet for realizada com a palavra root, não fazendo distinção entre maiúscula e minúscula. Exemplo:

```
alert tcp any any -> any 23 (content: "root"; nocase;);
```

- ***msg***: imprime uma mensagem em alerta e registra o pacote;
- ***sid***: é usada unicamente para identificar regras do *Snort*. Esta informação permite aos plug-ins que interagem com o *Snort* identificar

regras facilmente. Deve-se atentar a numeração de ID que precisa obedecer ao seguinte padrão: Menor que 100 – reservada para uso futuro; Entre 100 e 1000000 – regras incluídas com a distribuição do Snort; Maior que 1000000 – regras personalizadas localmente.

- ***classtype***: permite orientar a categoria de ataque que se está sendo detectado. Trabalha em cima de uma tabela com nome, descrição e prioridade. O usuário então pode especificar a prioridade que determinado tipo de ataque tem ao ser detectado. Formato:

```
classtype: <nome da classe>
```

As classificações de regras são definidas no arquivo *classification.config*. Este arquivo utiliza a seguinte sintaxe:

```
config classification: <nome da classe>, <descrição da classe>, <prioridade>
```

Um exemplo de regra contendo essa opção segue abaixo na Figura 36. Podemos então notar que a *classtype* descrita se chama "*attempted-recon*", que possui entrada em *classification.config*:

```
alert tcp any any -> any 25 (msg:"SMTP expn \
root"; flags:A+; content:"expn root";nocase; \
classtype:attempted-recon;) config classification: \
attempted-recon,Attempted Information Leak,2
```

Figura 36: Exemplo de regra com a opção *classtype*.

Uma vez entendido o funcionamento das regras de alertas, definimos criar três regras com prioridades diferentes e que pudessem ser facilmente disparadas pelo *Snort*. Estas regras foram definidas com o intuito de cobrir os casos de testes da ferramenta *Hamdroid* que serão abordados logo a seguir.

As regras definidas e suas características foram:

1. *Port Scan Detected for a Non-Critcal Port*: Regra para identificar um ataque de *Port Scan* vinda de qualquer *host* para faixa de portas de 61

a 1024 do servidor monitorado, cuja prioridade foi configurada como baixa (prioridade 3). Configuração da regra em *scan.rules*:

```
alert tcp any any -> $HOME_NET 61:1024 (msg:"Port Scan Detected for a Non-Critcal Port"; classtype:network-scan-low; sid:1000001;)
```

Configuração da classificação em *classifition.config*:

```
config classification: network-scan-low,Detection of a Network Scan,3
```

2. *Port Scan Detected for a Semi-Critical Port*: Regra para identificar um ataque de *Port Scan* vinda de qualquer *host* para a faixa de portas de 23 a 60 do servidor monitorado, cuja prioridade foi configurada como média (prioridade 2). Configuração da regra em *scan.rules*:

```
alert tcp any any -> $HOME_NET 23:60 (msg:"Port Scan Detected for a Semi-Critical Port"; classtype:network-scan-medium; sid:1000002;)
```

Configuração da classificação em *classifition.config*:

```
config classification: network-scan-medium,Detection of a Network Scan to a Critical Port,2
```

3. *Port Scan Detected for a Critical Port*: Regra para identificar um ataque de *Port Scan* vinda de qualquer *host* para a faixa de portas de 21 a 22 do servidor monitorado, cuja prioridade foi configurada como alta (prioridade 1). Configuração da regra em *scan.rules*:

```
alert tcp any any -> $HOME_NET 21:22 (msg:"Port Scan Detected for a Critical Port"; classtype:network-scan-critical; sid:1000003;)
```

Configuração da classificação em *classifition.config*:

```
config classification: network-scan-
critical,Detection of a Network Scan to FTP port,1
```

Com a definição dessas regras, um *Port Scan* no servidor monitorado irá gerar um conjunto de alertas bem completos, com a grande maioria de alertas sendo de baixa prioridade, alguns de média e apenas um crítico.

Na próxima seção iremos apresentar os casos de testes que se utilizaram das regras recém-descritas.

3.10.3 Casos de Teste

Conforme abordado em seções anteriores, este trabalho propôs dois tipos distintos de testes a ferramenta *Hamdroid*, o teste de usabilidade e o teste de carga. Abaixo iremos detalhar estes dois tipos de testes através de seus casos de teste.

3.10.3.1 Caso de Teste – Teste de Usabilidade

A Tabela 12 abaixo detalha o caso de teste de usabilidade:

Tabela 12: Caso de teste de usabilidade.

DESCRIÇÃO DO CASO DE TESTE	
CASO DE TESTE	Teste de Usabilidade
Propósito do Teste	O propósito deste teste é verificar a *performance* alcançada pelos participantes e o entendimento das funções do sistema, com a finalidade de responder apenas aos alertas de alta prioridade (prioridade 1) em meio a uma gama de alertas de *port scan*, tomando a ação de bloquear todas as portas listadas por esses alertas de alta prioridade. Será medido o tempo gasto para a realização das tarefas e serão identificados erros e dificuldades envolvendo a utilização do protótipo em tarefas rotineiras.
Declaração dos Problemas	- Responder rapidamente os alertas, com a finalidade de evitar qualquer dado à rede de computadores sob ataque.
Perfil do Usuário	- Usuário com mais de um ano de experiência em uso de dispositivos móveis com sistema operacional Android; - Superior completo em curso relacionado à informática; - Conhecimento básico em inglês; - Conhecimento básico em segurança de redes de computadores.
Quantidade de Execuções	3

Metodologia	- Avaliador irá treinar o usuário nas funcionalidades do *Hamdroid*; - Avaliador passará a lista de tarefas ao usuário; - Depois de passadas as orientações, será permitido que o participante utilize a ferramenta livremente por dois minutos. - Logo depois, será requisitado ao participante retornar à tela inicial do Android; - Avaliador irá gerar o *port scan* para que o *Hamdroid* capture os alertas de diferentes prioridades; - Usuário irá executar as tarefas; - Depois de completadas todas as tarefas, o participante preencherá um questionário de avaliação do sistema pelo participante cuja finalidade é coletar informações preferenciais do participante.
Ambiente de Teste / Equipamento	Dispositivo móvel *Samsung Galaxy* Y DUO, sistema operacional Android 2.3, tela de 3.14 polegadas, peso 98 gramas com processador de 832 Mhz.
Papel do Avaliador	O avaliador se sentará ao lado do participante durante a realização do teste e registrará o tempo gasto nas tarefas, erros e observações. O avaliador não poderá ajudar o participante na realização das tarefas. Ele somente poderá orientar se surgir uma questão acerca do procedimento de teste.
Medidas de Avaliação	As seguintes medidas de avaliação serão coletadas e calculadas: 1. Tempo gasto para completar cada tarefa por participante; 2. Número de erros cometidos na realização de cada tarefa por participante.

FORMULÀRIOS

Lista de Tarefas

T 1	Ao receber os alertas, acessar a ferramenta *Hamdroid*.
T 2	Acessar somente o alerta de prioridade alta (prioridade 1).
T 3	Acessar os detalhes do alerta acima.
T 4	Tomar a decisão de fechar todas as portas.
T 5	Marcar a tarefa como *checked.*
T 6	Retornar a tela de alertas do *Hamdroid*.

Questionário de Avaliação do *Hamdroid*

Q 1	Facilidade de utilização	Difícil					Fácil
		0	1	2	3	4	5
Q 2	Organização das informações	Ruim					Boa
		0	1	2	3	4	5
Q 3	*Layout* das telas	Confuso					Claro
		0	1	2	3	4	5
Q 4	Nomenclatura utilizada nas telas (nome de comandos, títulos, campos, etc.)	Confuso					Claro
		0	1	2	3	4	5
Q 5	Mensagens do sistema	Confusas					Claras
		0	1	2	3	4	5

3.10.3.2 Caso de Teste – Teste de Carga

A Tabela 13 abaixo detalha o caso de teste de usabilidade:

Tabela 13: Caso de teste de carga.

DESCRIÇÃO DO CASO DE TESTE	
CASO DE TESTE	Teste de Carga
Propósito do Teste	O propósito deste teste é verificar o desempenho do sistema levando-se em conta uma carga de transações simultâneas, observando o fator tempo de resposta do sistema e ocorrência de erros.
Declaração dos Problemas	- Responder rapidamente os alertas, com a finalidade de evitar qualquer dado à rede de computadores sob ataque.
Metodologia	- Avaliador irá executar o *port scan* initerruptamente para que o IDS gere os alertas para a ferramenta *Hamdroid*. - Cada execução do *port scan* irá gerar 122 alertas, sendo 2 alertas de prioridade alta (prioridade 1), 38 alertas de prioridade média (prioridade 2) e 82 alertas de prioridade baixa (prioridade 3). - A ferramenta *Hamdroid* estará ligada e conectada ao banco de dados *MySQL* disponível na Internet, e através de uma mensagem de log será computada o tempo decorrido para o *Hamdroid* sincronizar as mensagens e alertar a existência de uma mensagem crítica (prioridade 1).
Ambiente de Teste / Equipamento	Dispositivo móvel *Samsung Galaxy* Y DUO, sistema operacional Android 2.3, tela de 3.14 polegadas, peso 98 gramas com processador de 832 Mhz.
Quantidade de Ciclos de Teste	5
Papel do Avaliador	O avaliador ira executar o port scan bem como capturar o log dos tempos de sincronização do *Hamdroid* e de seu alerta pela existência de uma mensagem crítica (prioridade 1).
Medidas de Avaliação	As seguintes medidas de avaliação serão coletadas e calculadas: 1. Tempo gasto para o *Hamdroid* sincronizar as mensagens e alertar a existência de uma mensagem crítica (prioridade 1).; 2. Capacidade máxima do *Hamdroid* para lidar com os alertas.

4 RESULTADOS

Este capítulo apresentará inicialmente as telas e funcionalidades resultantes do desenvolvimento da ferramenta *Hamdroid*, depois apresentará os resultados da aplicação dos casos de testes acima explorados e por fim irá discutir os resultados obtidos.

4.1 APRESENTAÇÃO DE TELAS E FUNCIONALIDADES DO *HAMDROID*

A seguir encontram-se algumas telas utilizadas na ferramenta *Hamdroid* e suas respectivas funcionalidades.

4.1.1 ***Dashboard***

Similar a um painel de controles de um automóvel, um *Dashboard* visa prover indicadores de *performance* ou *status* para uma gerência com um único olhar (DEBUSK et al., 2003).

Dentre as funcionalidades oferecias pelo *Hamdroid,* destaca-se o *Dashboard* de status de segurança da rede de computadores monitora. Este *Dashboard* utiliza de diferentes cores para associar o status corrente da rede monitorada, onde a cor verde (Figura 37a) define que a rede se encontra estável sem nenhum alerta, branca (Figura 37c) é dada para ausência de comunicação com a rede monitorada e vermelha (Figura 37b) define que existe um ou mais alertas de um possível ataque ou invasão.

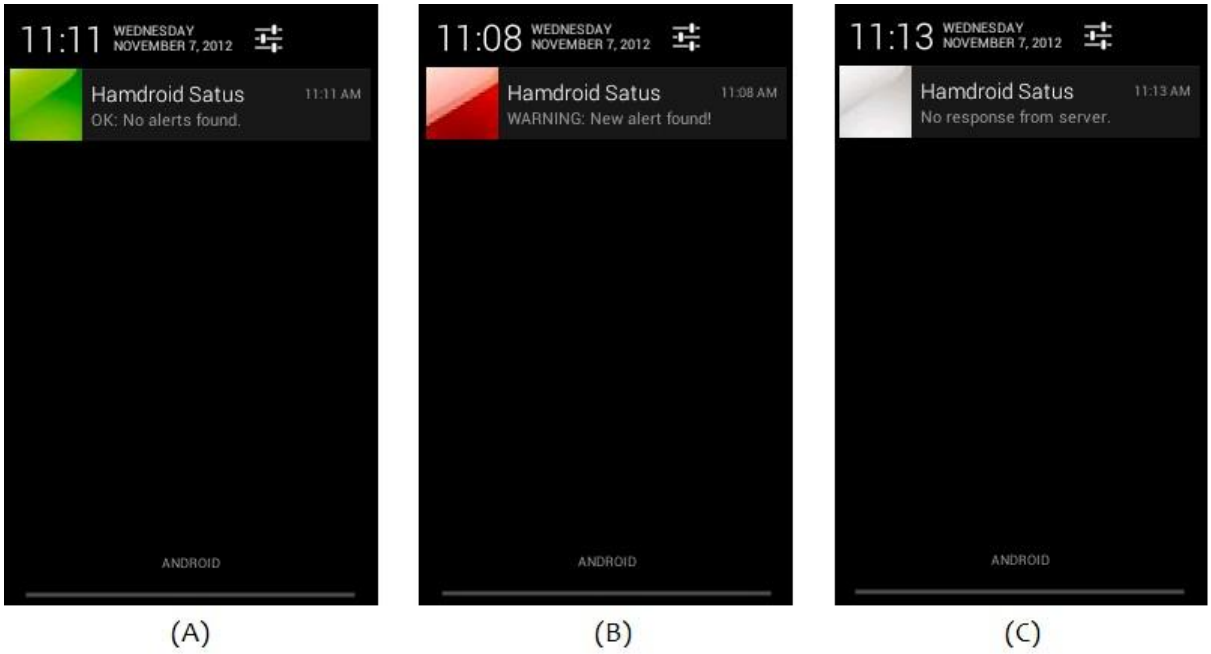


Figura 37: *Dashboard* de status de segurança da rede monitorada.

Sempre que há alguma alternância entre os status do *Dashboard*, o *Hamdroid* exibe, em forma de um ícone grande, a cor e a descrição da cor associada ao novo status. Após o usuário tomar ciência do novo status, o *Hamdroid* minimiza o *Dashboard* para um pequeno ícone no canto superior direito da tela do dispositivo móvel. Veja o exemplo da Figura 38 abaixo:

Figura 38: *Dashboard* minimizado na tela do dispositivo móvel.

Dessa forma, o usuário sempre terá visível um controle geral do status corrente de sua rede.

4.1.2 Telas de Alertas e de Tomada de Ação

A principal funcionalidade do *Hamdroid* é a visualização dos alertas gerados pelo IDS *Snort*. Ao abrir a ferramenta, esta automaticamente faz a atualização com o banco de dados (Figura 39a). A ferramenta divide a visualização dos alertas em duas telas, a primeira com focada em apresentar os aletas de forma sumária e ordenado pela devida criticidade e a segunda com todos os detalhes dos alertas. Na primeira tela (Figura 39b), a de informações sumárias, a ferramenta agrupa os alertas por criticidade, nome, tipo e janela de tempo em que os alertas aconteceram. Ao selecionar o grupo de mensagem na primeira tela, através de um toque na tela, a ferramenta direciona o usuário para uma segunda tela (Figura 39c) com todos os detalhes do grupo de mensagens selecionado, como os endereços de IPs e as portas envolvidas no ataque.

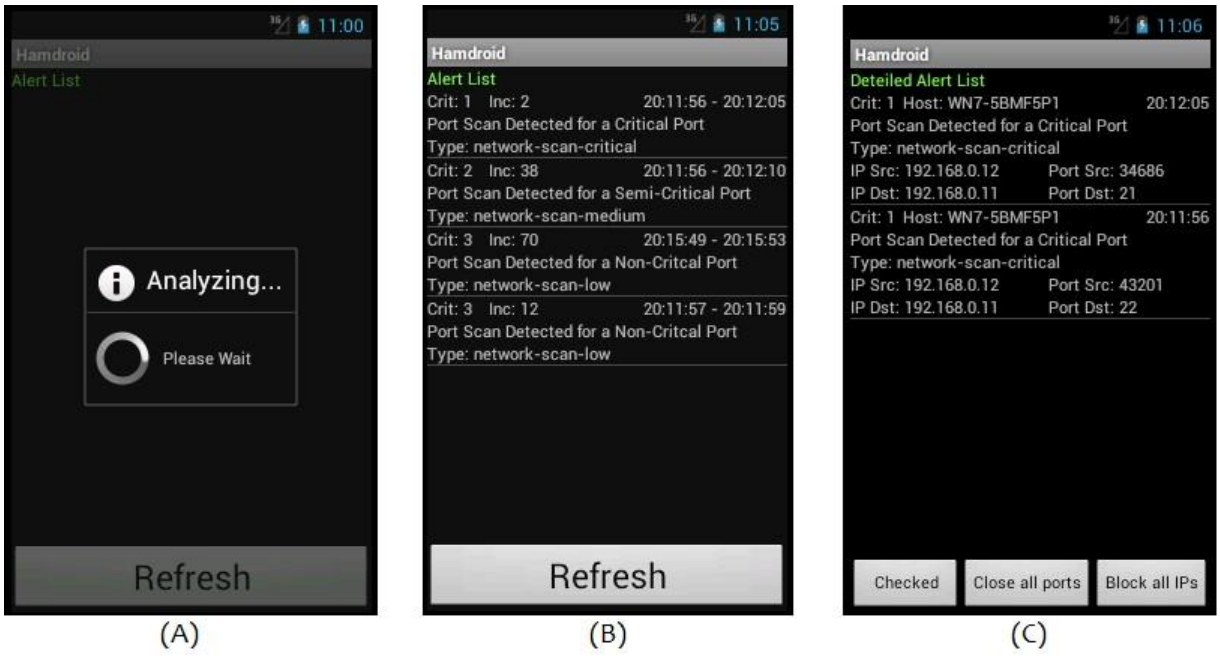


Figura 39: Visualização dos alertas na ferramenta *Hamdroid*.

Ainda na tela de detalhes dos alertas, além do botão *Checked* para marcar como lido ao alerta, a ferramenta disponibiliza os botões para a tomada de decisão

mediante ao alerta de ataque. São eles: o botão *Close all ports*, para fechar todas as portas envolvidas no grupo de alertas selecionado, e o botão *Block all IPs*, para bloquear todos os endereços de IPs envolvidos no grupo de alertas selecionado.

Voltando para primeira tela, a de alertas sumarizados, o *Hamdroid* ainda oferece um conjunto de toma de ações adicional. Eles são acessados através da tecla de *menu* do Android. Vejamos as opções oferecidas pela ferramenta na Figura 40:

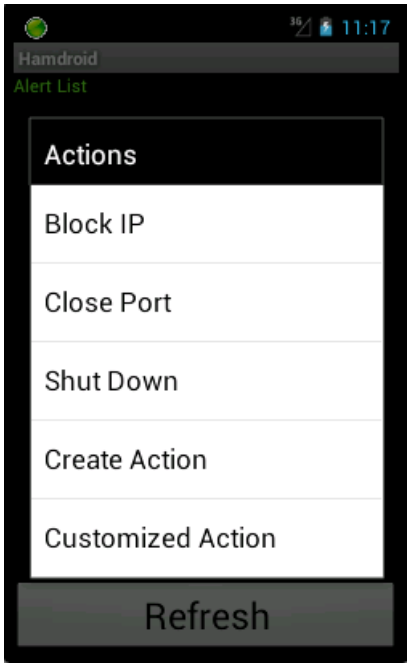


Figura 40: Opções adicionais de tomada de ações.

A opção *Block IP* irá abrir uma nova tela para que o usuário possa manualmente entrar com o endereço de IP que ele deseja bloquear no servidor monitorado (Figura 41a). A opção *Close Port* é semelhante a anterior e também irá solicitar ao usuário o número de porta a ser bloqueada (Figura 41b). A opção *Shut Down* é para a ação drástica de desligar o servidor monitorado (Figura 41c).

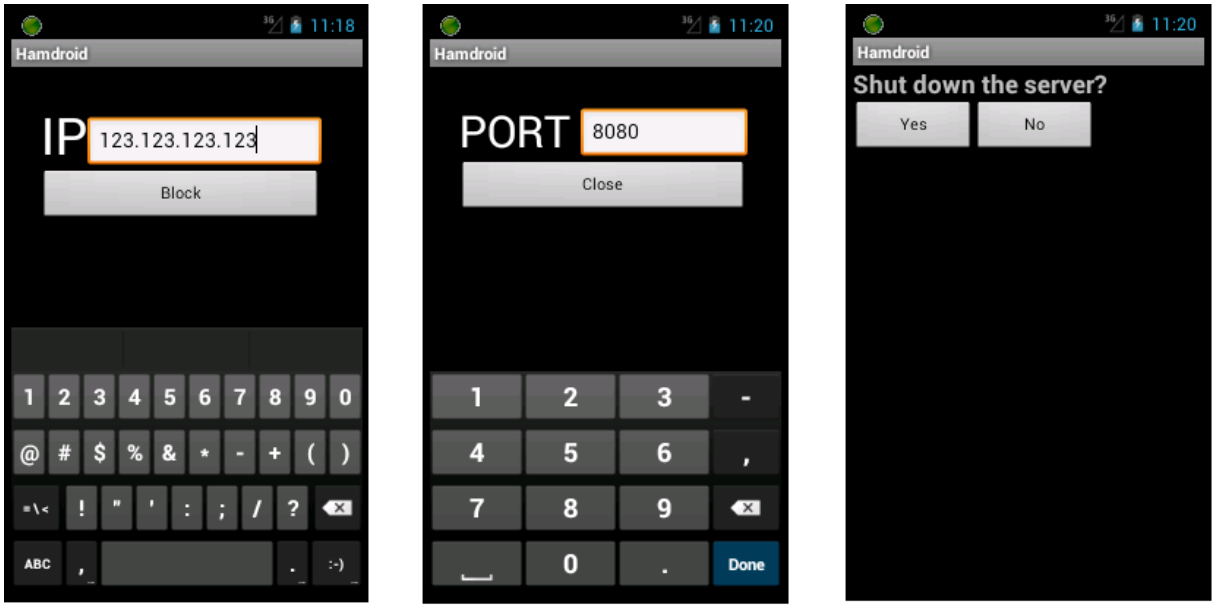


Figura 41: Ações Adicionais.

A ferramenta ainda oferece as opções de criação de uma tomada de ação personalizada e a de criação de uma nova ação personalizada. Estas duas funcionalidades serão abordados na próxima seção.

4.1.3 Telas de Customização de Tomada de Ações

Ao selecionar a opção de customizar uma tomada de ação, a ferramenta direciona o usuário para a tela de configuração dessa nova ação (Figura 42a). O usuário deverá digitar o comando que ele deseja executar no sistema operacional do servidor monitorado, os parâmetros dinâmicos que ele irá informar no momento que ele querer executar a ação e então dar um nome para essa ação.

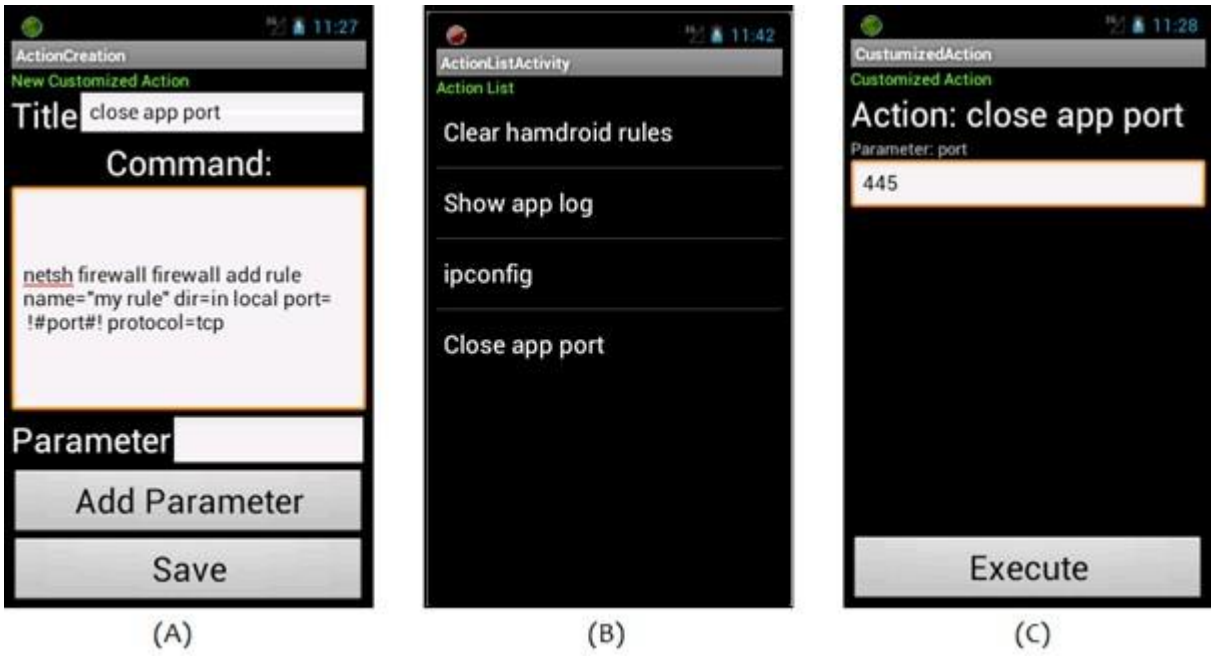


Figura 42: Telas de Tomada de Ações Personalizadas.

Todas as ações personalizadas pelo usuário estarão disponíveis em forma de lista (Figura 42b). Ao selecionar a ação personalizada, a ferramenta abre uma nova tela com todos os parâmetros dinâmicos definidos para a determinada ação personalizada. A Figura 42c mostra um exemplo para a ação personalizada *close app port*.

4.1.4 Tela de Configuração Inicial

O *Hamdroid* tem uma tela (Figura 43) de configuração dos parâmetros de conexão com o bando de dados MySQL, banco este que, na arquitetura proposta por este trabalho, irá persistir os alertas gerados pelo *Snort*.

Figura 43: Tela de configuração inicial do *Hamdroid*.

Nesta tela, o usuário deverá informar no campo *Connection String* o endereço HTTP de conexão com o servidor. Ainda nesta tela, o usuário terá o indicador *Notification Service* que informa se a conexão com o banco está ativa ou não.

4.2 RESULTADO DOS TESTES

Esta seção tem por objetivo apresentar os resultados dos testes de usabilidade e de carga da ferramenta *Hamdroid*. Primeiramente são apresentadas as medidas coletadas durante o teste de usabilidade e as respostas aos questionários propostos aos usuários. Em seguida, são expostos os resultados dos testes de carga.

4.2.1 Resultados dos Testes de Usabilidade

Os testes de usabilidade foram realizados entre os dias 7 a 9/11/2012. Foram utilizados 3 participantes com mais de um ano de experiência em uso de dispositivos móveis com sistema operacional Android, com superior completo na área de informática (Análise de Sistemas, Ciência da Computação e Sistemas de Informação), com conhecimentos avançados de Inglês e informática.

Os participantes preencheram o questionário sobre satisfação do produto. O avaliador capturou o tempo decorrido para a execução de cada tarefa em cada sessão de teste.

O resultado apurado nas sessões de teste é apresentado a seguir na Tabela 14:

Tabela 14: Resultado dos testes de usabilidade.

Tarefa	Tempo de Execução da Tarefa em Segundos				
	Usuário 1	Usuário 2	Usuário 3	Média	Desvio
T 1	4	5	5	4,7	0,6
T 2	3	4	3	3,3	0,6
T 3	9	3	4	5,3	3,2
T 4	5	5	6	5,3	0,6
T 5	3	4	4	3,7	0,6
T 6	1	2	2	1,7	0,6
Total ->	**25**	**23**	**24**	**24,0**	**1,0**

O único problema encontrado durante as sessões e anotado pelo avaliador foi pelo na sessão de teste do Usuário 1 durante a execução da Tarefa 3, que é acessar os detalhes de um alerta. O usuário teve alguns minutos de dúvida em como detalhar o alerta, mas logo, sem a ajuda do avaliador, se recordou em como fazer a tarefa.

Ao final não houve grandes discrepâncias na média de execução das tarefas, e a média total ficou em 24 segundos.

O resultado do questionário aplicado após a execução dos testes é apresentado abaixo na Tabela 15:

Tabela 15: Resultados do questionário aplicado durante os testes de usabilidade.

Questão	Grau de Satisfação				
	Usuário 1	Usuário 2	Usuário 3	Média	Desvio
Q 1	4	3	4	3,7	0,6
Q 2	4	4	5	4,3	0,6
Q 3	3	2	3	2,7	0,6
Q 4	5	4	5	4,7	0,6
Q 5	5	5	5	5,0	0,0
Total ->	**4,2**	**3,6**	**4,4**	**4,1**	**0,5**

Os usuários 1 e 3 adicionaram um comentário quanto a satisfação com o *Dashboard* da ferramenta. A média total de satisfação ficou em 4,1.

4.2.2 Resultados dos Testes de Carga

Os testes de carga foram realizados no dia 4/11/2012, onde contou com um dispositivo móvel *Samsung Galaxy* Y DUO, com sistema operacional Android 2.3 de processador de 832 Mhz, O teste foi feito usando uma Internet a cabo de 20 Mb, visto que o banco de dados MySQL usado foi hospedado em um servidor gratuito na Internet.

Foram executados 5 ciclos de testes, onde foi simulado um ataque de *port scan* initerruptamente para que o IDS com suas mensagens de alerta personalizadas gerassem conjuntos de 122 alertas em cada *port scan*. Em todos os ciclos de teste, o *Hamdroid* não suportou e gerou erro de insuficiência de espaço quando ele atingiu o número de 3.294 alertas. O resultado está apresentado abaixo na Tabela 16:

Tabela 16: Resultados dos testes de carga.

Quantidade de Alertas	Tempo de Resposta da Ferramenta em Segundos						
	Ciclo de Teste 1	Ciclo de Teste 2	Ciclo de Teste 3	Ciclo de Teste 4	Ciclo de Teste 5	Média	Desvio
122	2	8	3	3	5	4,2	2,4
244	5	2	3	2	4	3,2	1,3
366	5	3	2	2	4	3,2	1,3
488	6	7	4	3	4	4,8	1,6
610	7	5	5	5	6	5,6	0,9
732	6	6	5	5	7	5,8	0,8
854	7	7	6	8	6	6,8	0,8
976	8	8	5	6	8	7,0	1,4
1098	7	6	7	7	8	7,0	0,7
1220	8	7	9	8	8	8,0	0,7
1342	8	7	7	7	10	7,8	1,3
1464	9	9	7	8	9	8,4	0,9
1586	8	9	8	10	9	8,8	0,8
1708	11	12	10	9	10	10,4	1,1
1830	9	10	9	9	9	9,2	0,4
1952	11	12	12	10	10	11,0	1,0
2074	11	13	11	12	10	11,4	1,1
2196	10	13	11	12	11	11,4	1,1
2318	12	9	12	11	10	10,8	1,3
2440	12	10	10	13	10	11,0	1,4
2562	9	10	9	10	11	9,8	0,8
2684	9	11	10	11	11	10,4	0,9

2806	11	10	10	11	11	10,6	0,5
2928	13	9	12	13	13	12,0	1,7
3050	10	8	10	12	12	10,4	1,7
3172	12	9	11	12	11	11,0	1,2
Média ->	**8,7**	**8,5**	**8,0**	**8,4**	**8,7**	**8,5**	**1,1**

O resultado foi uma média de 8,5 segundos (com desvio de 1,1, ou seja, os tempos variaram de por volta de 7,5 a 9,5 segundos) para o tempo de resposta da ferramenta *Hamdroid* para sincronizar as mensagens e alertar a existência de uma mensagem crítica (prioridade 1).

Para detalhar o erro de falta de memória gerado pela ferramenta ao atingir o número de 3.294 alertas, acompanhamos a utilização de memória do dispositivo móvel durante a realização dos ciclos de teste 4 e 5. Vejamos o resultado na Tabela 17 abaixo:

Tabela 17: Utilização de Memória.

Quantidade de Alertas	**Utilização de Memória em Mb**	
	Ciclo de Teste 4	**Ciclo de Teste 5**
122	158	158
244	158	158
366	159	159
488	159	159
610	160	160
732	160	160
854	161	161
976	161	161
1098	162	162
1220	162	162
1342	163	163
1464	163	163
1586	163	163
1708	164	164
1830	164	164
1952	165	165
2074	165	165
2196	166	166
2318	166	166
2440	167	167
2562	167	167
2684	167	167

2806	168	168
2928	168	168
3050	169	169
3172	169	169
3294	170	170

Coma a capacidade máxima de memória do dispositivo usado nos testes era de 170 Mb, podemos comprovar que o erro foi dado devido a limitação do hardware, e não da ferramenta.

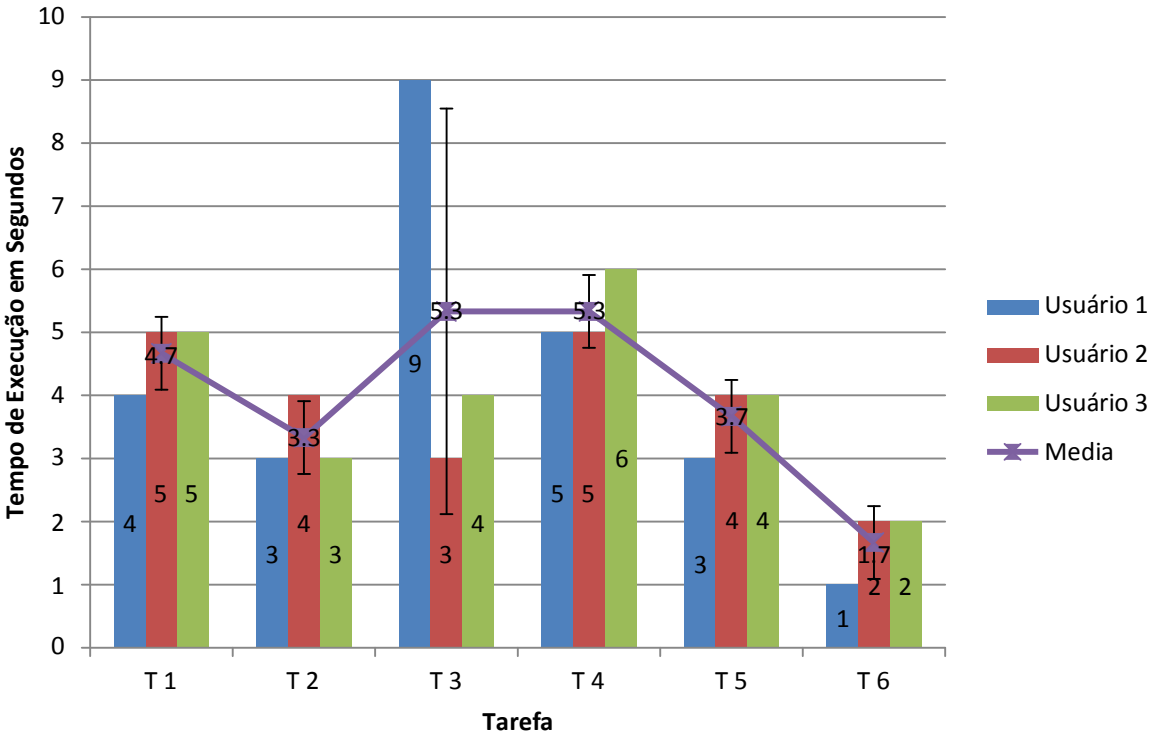
Na próxima seção iremos discutir os resultados de ambos os testes de usabilidade e de carga.

4.3 DISCUSSÃO DOS RESULTADOS

Nesta seção iremos discutir os resultados obtidos nos testes de usabilidade e de carga.

Começando pelo teste de usabilidade, o resultado mostrou que as principais tarefas puderam ser executadas rapidamente. Vejamos o plotagem dos resultados no Gráfico 1 a seguir:

Gráfico 1: Tempo de Execução das Tarefas do Teste de Usabilidade.

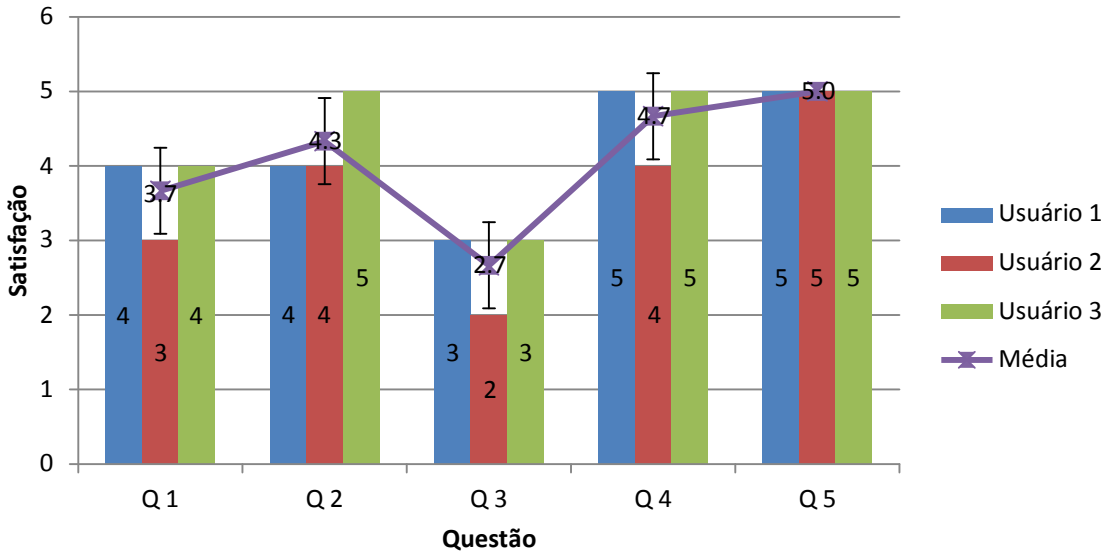


Com exceção de um incidente anotado pelo avaliador durante a execução da Tarefa 3 pelo Usuário 1, as outras tarefas não passaram dos 6 segundos. A média de execução de todas das tarefas não ultrapassou os 5,3 segundos. Considerando que nesses tempos há também o tempo de resposta da ferramenta para acessar o banco de dados, estes resultados se mostram aceitáveis.

Observamos ainda, que o fato de todos os usuários terem familiaridade com a navegação padrão do sistema operacional Android, ajudou positivamente nos resultados obtidos. Baseado nessa observação, este trabalho recomenda que o usuário final da ferramenta *Hamdroid* tenha o perfil de ser experiente com o sistema operacional Android.

Quanto à satisfação dos usuários em relação à experiência de uso da ferramenta, vejamos os resultados plotados no Gráfico 2 a seguir:

Gráfico 2: Satisfação dos Usuários do *Hamdroid*.

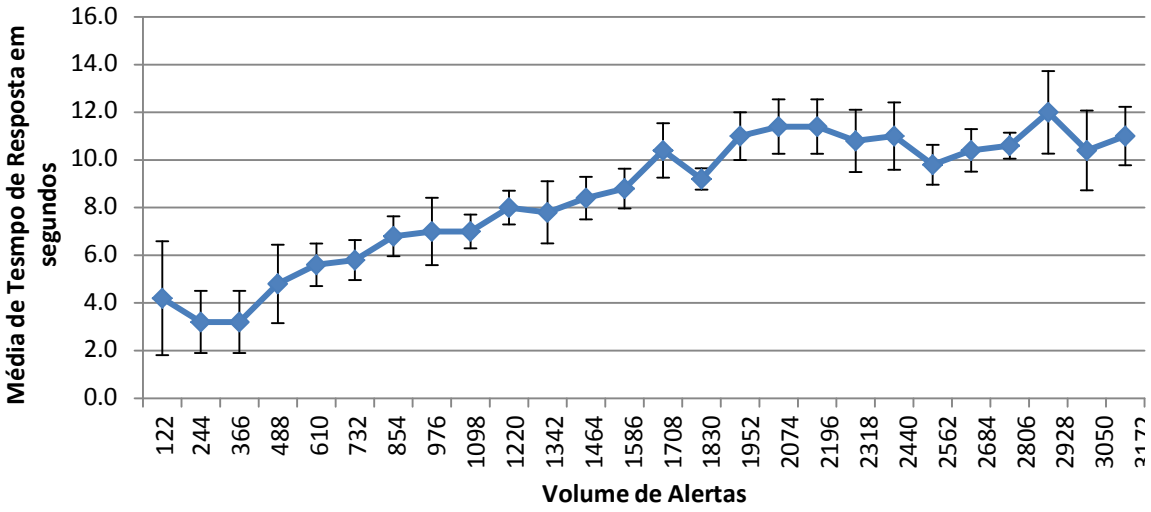


A pior nota atribuída ao *Hamdroid* foi 2 para o requisito de satisfação quanto ao *Layout* das telas. O Usuário 2 achou um pouco confuso todas as mensagens e seus atributos serem da mesma cor, pois ele esperava que as mensagens de maior prioridade se destacasse perante as outras de menos prioridade. Embora notado essa insatisfação, a média final dada por esse usuário foi 2,7. No total, a média de satisfação de todos os usuários foi 4,1, o que faz concluir que a ferramenta foi bem aceita entre seus usuários.

Outro destaque, segundo os usuários, foi o *Dashboard*. Dois dos usuários ressaltaram a satisfação com esta funcionalidade.

Os testes de carga também se mostraram satisfatórios. Vejamos os resultados plotados no Gráfico 3 abaixo:

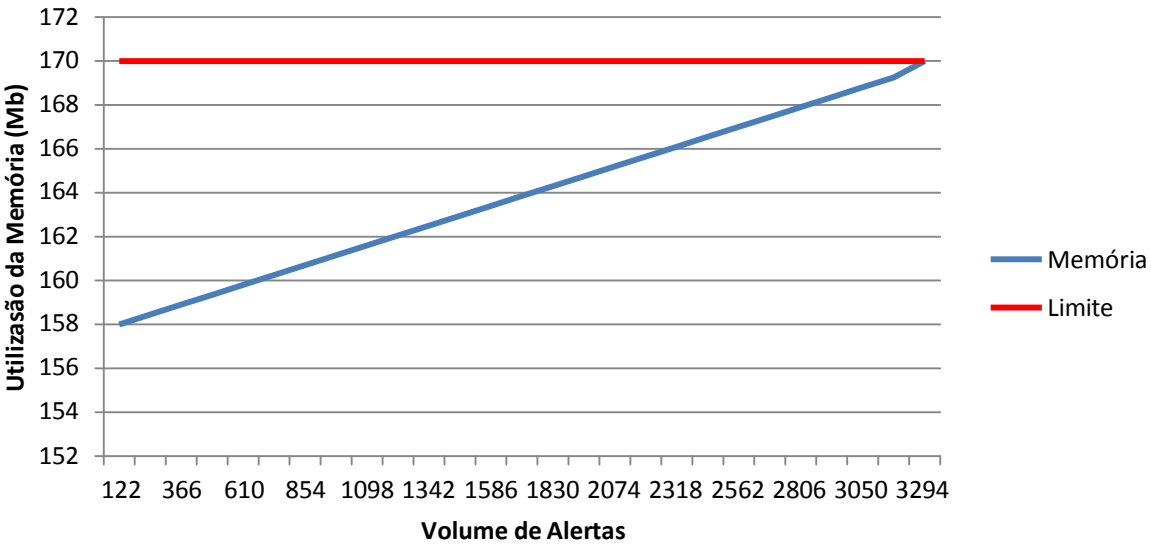
Gráfico 3: Tempo de Resposta do *Hamdroid*.



O tempo de resposta demostrou estável após o volume de 1708, e se sustentou abaixo dos 12 segundos até o volume de 3.172 alertas.

Embora a ferramenta não suportou o número de 3.294 alertas, o acompanhamento da utilização da memória do dispositivo móvel mostrou que o limite se deu devido a limitação do hardware e não da ferramenta. No Gráfico 4 está plotado as resultados do monitoramento da memória do dispositivo com capacidade de 170 Mb:

Gráfico 4: Utilização de Memória do Dispositivo Móvel.



Podemos concluir que a ferramenta *Hamdroid* teve um bom desempenho em seu tempo de resposta, mesmo trabalhando com números elevados de alertas. Além disso, a ferramenta foi bem aceita pelos usuários que experimentaram o seu uso.

Na próxima seção vamos discutir a principais dificuldades encontradas ao longo do desenvolvimento deste trabalho.

4.4 DIFICULDADES ENCONTRADAS

A principal dificuldade encontrada durante o desenvolvimento deste projeto foi a escassez de materiais relacionados ao uso do *Snort* em sistemas operacionais *Windows*. Pelo fato dos integrantes deste projeto não terem familiaridade com ambientes *Linux*, o projeto optou por focar todo o desenvolvimento da arquitetura da ferramenta *Hamdroid* em *Windows*, contudo na fase de instalação do *Snort* houveram grandes dificuldades em encontrar documentação que explicasse passo a passo as configurações necessárias para que o IDS funcionasse corretamente neste tipo de sistema operacional.

Além disso, houve dificuldade em definir a arquitetura da ferramenta. Por se tratar de um projeto voltado para segurança de redes de computadores, houve uma preocupação em não definir uma arquitetura que adicionasse pontos de invasão à rede monitorada.

Por fim ainda citamos o *ramp up* inicial de desenvolvimento para plataforma Android, pois nenhum dos integrantes deste trabalho tinha experiência ou conhecimento anterior nesta tecnologia.

5 CONSIDERAÇÕES FINAIS

Este trabalho mostrou a importância da segurança de redes de computadores, visto que além do número de incidentes reportados terem aumentado significantemente e constantemente, também vimos que os danos causados por um ataque podem ser irreversíveis, podendo levar até a falência da empresa.

Também foi abordada a importância da mobilidade para o trabalhador nômade, onde com o uso de dispositivos móveis, o tempo de resposta a um evento é reduzido.

Considerado a importância da segurança de redes e o benefício agregado pela mobilidade, compartilhamos a visão de que a criação de uma ferramenta mobile para controle de uma pequena rede de computadores é de grande valia, especialmente para administradores de pequenas redes de computadores que não contam com uma grande infraestrutura de monitoramento. Dessa forma, esse público alvo pode se beneficiar da ferramenta proposta por esse trabalho e minimizar o risco de prejuízo com ataques a sua rede de computadores.

A ferramenta *Hamdroid* ainda acrescenta a possibilidade de respostas a um alerta de ataque, oferecendo algumas tomadas de decisão, permitindo ao seu usuário ter autonomia de resposta a um ataque mesmo distante fisicamente de sua rede de computadores.

Por fim, os resultados dos testes de usabilidade e de carga aplicados à ferramenta *Hamdroid*, mostraram que ela é de fácil uso, suporta números elevados de alertas e permite uma ação rápida em resposta a um ataque à rede de computadores monitorada.

5.1 TRABALHOS FUTUROS

Para projetos futuros, pode ser realizado a pesquisa e uma possível implementação de uma ação automática em resposta a um alerta de ataque, ponderando o *trade off* do quanto de fato é positivo ao administrador de rede perder a autonomia de uma resposta racional a um ataque. A sugestão é implementar as regras de ações automáticas na camada “servidor” do Hamdroid, pois assim,

mitigaríamos o risco de contar com um dispositivo móvel baseado em uma bateria, para monitorar uma rede a distância.

Outra sugestão de trabalho futuro é estudar o impacto da arquitetura desta ferramenta à rede de computadores monitorada. A proposta seria verificar se a implantação dessa ferramenta não adicionou nenhuma potencial vulnerabilidade. Ainda como sugestão, pode-se propor melhor a arquitetura atual do *Hamdroid*, incluindo criptografia e chaves de sessão.

REFERÊNCIAS

ACIDLAB, Analysis Console for Intrusion Databases. **Snort DB ER**, Disponível em:<http://acidlab.sourceforge.net/acid_db_er_v102.html>. Acesso em: 06 out. 2012.

ALDUNATE, R., S; OCHOA, F; NUSSBAUM, M. **Robust Mobile Ad-hoc Space for Collaboration to Support Disaster Relief Efforts Involving Critical Physical Infrastructure**. ASCE J. of Computing in Civil Engineering, American Society of Civil Engineers, in press. 2005.

ALLEN, Julia; CHRISTIE, Alan; McHOUGH, John. **The IDS life cycle.** Infosec Outlook, 2000. Disponível em: <http://www.cert.org>. 26 maio 2012.

ANDERSON, Ross. **Security Engineering**. 2. Ed. Inglaterra: Wiley, 2008. 562 p.

ANDERSON, P; BACKWOOD, A. **Mobile and PDA Technologies and their future use in education**. JISC Technology and Standards Watch. 2004.

ARONSON, J; LIANG, T; TURBAN, E. **Decision support systems and intelligent systems**. Pearson, Upper Saddle River, 2005.

BAZARA, I. A.; STAVROULAKIS, P.; STAMP, M. **Intrusion Detection Systems**. Handbook of Information and Communication Security. Springer Berlin Heidelberg, (2010).

BUDINGTON, William. Disponível em: <http://www.inputoutput.io>. Acesso em 2 abr. de 2012.

BURSTEIN, F; SAN PEDRO, J. C; ZASLAVSKY, A; HODGKIN, J. **Pay by cash, credit or EFTPOS?**. Proceedings of the 3th international conference on mobile business, m[business 2004, Institute of Technology and Enterprise, Polytechnic University, New York, 2004.

CARLSSON, C; CARLSSON, J; DENK, M; WALDEN, P. **Mobile commerce: insights from expert surveys in Austria and Finland**. Proceedings of the 18th bled eCommerce conference, 2005.

CARLSSON, C; CARLSSON, J; WALDEN, P. **Mobile travel and tourism services on the finnish market**. Proceedings of the 24th Euro CHRIE Congress, 2006.

CARLSSON, C; HYVONEN, K; REPO, P, WALDEN, P. **Adoption of mobile services across different technologies**. Proceedings of the 18th bled eCommerce conference, Bled, 2005.

CASWELL, Brian et al. **Snort 2: Sistema de detecção de intrusão**. Rio de Janeiro: Alta Books, 2003.

CERTBR, Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. **Estatísticas dos Incidentes Reportados ao CERT.br**, Disponível em:<http://www.cert.br/stats/incidentes>. Acesso em: 27 mar. 2012.

CIL, Renato. **Segurança de Redes de Computadores e Comunicação de Dados**. Londrina: Universidade Estadual de Londrina, 2006. Disponível em: <http://www.uel.br>. Acesso em: 24 maio 2012.

CHITTARO, Luca. **Visualizing Information on Mobile Devices**. Udine: Universidade de Udine, Itália, 2006. Disponível em: <http://hcilab.uniud.it/publications/2006-03/VisualizingInformationMobile_IEEECOMPUTER.pdf>. Acesso em: 31 maio 2012.

COLARES, Flavio M. **Análise Comparativa de Banco de Dados Gratuitos**. Faculdade Lourenço Filho, Fortaleza, 2007. Disponível em: <http://www.flf.edu.br/revista-flf/monografias-computacao/monografia_flaviocolares.pdf>. Acesso em: 10 oct. 2012.

COMER, Douglas E. **Interligação em redes com TCP/IP, vol. 1 princípios, protocolos e arquitetura**. Rio de Janeiro: Elsevier, 2006 – 2ª reimpressão. il.

COMPUTERWORLD. São Paulo: UOL. Diário. Disponível em <http://computerworld.uol.com.br/slide-shows/como-funcionam-os-ataques-de-sql-injection/paginador/pagina_1>. Acesso em: 7 maio 2012.

CONORICH, Douglas G. **Monitoring Intrusion Detection Systems**: From Data to Knowledge. Information security journal 13.2 (2004):19-19.

CRESPO, A. N. et. Al. **Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo**, Simpósio Brasileiro de Qualidade de Software, Maio de 2004, p.271-285.

CROASMUN, J. **Are Ergonomists Really Consulted in Mobile Phone Design?** Disponível em: <http://www.ergoweb.com/news/detail.cfm?id=961>. Acessado em: 7 de jun. de 2012.

CRUZ, Frank. **Entendendo um sistema de detecção de invasões**, 2001. Disponível em: <http://www.timaster.com.br>. Acesso em: 25 maio 2012.

DEBUSK, G. K.; BROWN, R. M.; KILLOUGH, L. N. **Components and relative weights in utilization of dashboard systems like the Balanced Scorecard**. The British Accounting Review, v. 35, p. 215-231, 2003.

DERTOUZOS, M. **The Unfinished Revolution**: Human-centeredComputers and What They Can Do for us. Nova Iorque, 2001.

D'SOUZA, D; WILLS, A. **Objects, Components and Frameworks with UML.** Addison-Wesley, 1999.

FERREIRA, Katia G. **Teste de Usabilidade**. UFMG: Universidade Federal de Minas Gerais, Brasil, 2002. Disponível em: <http://conteudo.imasters.com.br/3206/usabilidade.pdf>. Acesso em: 4 out. 2012.

FIORESE, V. **Wireless**: Introdução às Redes de Telecomunicação Móveis Celulares. Editora Brasport: 2005.

GUERRERO, L.; PINO, J. C.; COLLAZOS, A.; OCHOA, S. **Mobile Support for Collaborative Work**. 2004, Lecture Notes in Computer Science 3198, 363–375.

HAHN, J.; GUILLEN D. P.; ANDERSON, T. **Process Control Systems in the Chemical Industry: Safety vs. Security**. In: Annual CCPS International Conference, 20., 2005, EUA. Disponível em <http://www.inl.gov>. Acesso em: 5 maio 2012.

HAN, Bing. **Analysis and Research of System Security Based on Android**. Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference on. (2012). :581.

HAKKILA, J; MANTYJARVI, J. **Collaboration in Context-Aware Mobile Phone Applications Proc. of HICSS**: 2005. IEEE Computer Society Press.

HARTMANN, J; BRETZKE, S. **Financial services for the future - mobile, flexible, and agent-based**, 1999. Disponível em <http://citeseer.ist.psu.edu/correct/287340>. Acesso em: 10 jun. 2012.

INFO EXAME. Revista, São Paulo. Edição 190, p. 44, Jan 2002.

INFOWOLRD. Revista, Estados Unidos da América. Edição 98, p. 25, 17 de agosto de 2009.

KARGUPTA, H; PARK, B; PITTIE, S; LIU, L; KUSHRAJ, D; SARKAR, K. **Mobimine**: monitoring the stock market from a PDA. SIGKDD Explor, 2002.

KHAYAM, Syed A. **Design and Development of an Open Source Enterprise Network Security Solution**. Islamabad: National University of Sciences & Technology, 2010. Disponível em: <http://wisnet.seecs.nust.edu.pk/projects/ENS/Deliverables/LiteratureReview/Literature_Survey.pdf>. Acesso em: 6 maio 2012.

KORTUEM, G. J.; SCHNEIDER, D.; PREUITT, T. G. C.; THOMPSON, S.; Segall. Z.. **When Peer-to-Peer Comes Face-to-Face**: Collaborative Peer-to-Peer Computing in Mobile Ad-hoc Networks. Procs. First International Conference on Peer-to-Peer Computing. 2001. 27–29, pp. 75–91.

LARSON, R. E.; COCKROFT, L. **CCSP - Cisco Certified Security Professional Certification All-in-One Exam Guide**. 1. ed. EUA: CISCO, 2003.

LISKA, Allan. **The Practice of Network Security:** Deployment Strategies for Production Environments. 1. Ed. EUA: Prentice Hall, 2002. 416 p.

LUNDIN, E; JONSSON, E. **Provacy vs intrusion detection analysis**. Proceedings of the 2nd Internation Workshop on the Recent Advances in Intrusion Detection – RAID'99, West Lafayette, IN, Setembro 7-9.

MAGELA, R. **Engenharia de software aplicada: fundamentos**. 1ª edição. Ed. Rio de Janeiro: Alta Books, 2006.

MALLADI, R; AGRAWAL, D. **Current and Future Applications of Mobile and Wireless Networks**. Communications of the ACM 45(10), 144–146. 2002.

MENNECKE, B. E; STRADER, T. J. **Mobile commerce**: technology, theory, and applications. Idea Group, Hershey: 2002.

NAVARRO, F; SCHULTER, A; KOCH, F; ASSUNCAO, M; WESTPHALL, C. **Grid middleware for mobile decision support systems, networking**. International conference on systems and international conference on mobile communications and learning technologies, 2006.

NIELSEN, L. B. **Post disney experience paradigm?**: Some implications for the development of content to mobile tourist services. Mariji J, Sol HG, Wagenaar RW (eds), ICEC'04, 6th international conference on electronic, 2004.

NVIDIA. **The Benefits of Quad Core CPUs in Mobile Devices**, 2012. Disponível em:<http://www.nvidia.com/content/PDF/tegra_white_papers/tegra-whitepaper-0911a.pdf>. Acesso em: 31 maio 2012.

O JORNAL DE HOJE. Natal. Diário. Disponível em: <http://jornaldehoje.com.br/programa-de-gestao-promete-reduzir-custos-administrativos-em-ate-40/>. Acesso em: 27 maio 2012.

O'GRADY, M; O'HARE, G. **Mobile Devices and intelligent agents**: towards a new generation of applications and services. 2005.

OWASP, Open Web Application Security Project. **Top 10 Application Security Risks - 2010**, Disponível em:<http://www.cert.br/stats/incidentes>. Acesso em: 06 maio 2012.

OWASP, Open Web Application Security Project. **Man-in-the-middle attack**, 2009. Disponível em:<https://www.owasp.org/index.php/Man-in-the-middle_attack>. Acesso em: 24 maio 2012.

OWASP, Open Web Application Security Project. **Trojan Horse**, 2008. Disponível em:<https://www.owasp.org/index.php/Trojan_Horse>. Acesso em: 24 maio 2012.

PANDA, Partha. **OSSEC PCI Solution**, 2009.Disponível em <http://www.ossec.net/ossec-docs/ossec-PCI-Solution.pdf >. Acesso em 8 abr. de 2012.

PANIS, S; MORPHIS, N; FELT, E; REUFENHEUSER, B; BOHM, A; NITZT, J; SAARLOT, P. **Service scenarios and business models for mobile commerce**, 2002. Disponível em <http://citeseer.nj.nec.com/panis02service.html>. Acesso em 10 jun. de 2012.

PELISSARI, Fernando A. B. **Segurança de Redes e Análise Sobre a Conscientização das Empresas da Cidade de Bauru (SP) Quanto ao Problema**. Bauru: Universidade Estadual de São Paulo, 2002. Disponível em: <http://www.enesp.br>. Acesso em: 5 maio 2012.

PRESSMAN, Roger S. **Engenharia de software**, São Paulo: Makron Books, 1995.

PROMON, Business & Technology Review. **Mobilidade**: A grande tendência do futuro, 2005.Disponível em <http://www.teleco.com.br/promon/pbtr/Mobilidade_4Web.pdf>. Acesso em 27 maio de 2012.

RAGSDALE, D J. **Adaptation techniques for intrusion detection and intrusion response systems**. Systems, Man, and Cybernetics, 2000 IEEE International Conference on. New York, NY: IEEE, (2000). :2344.

SAN PEDRO, J. C; BURSTEIN, F. V; ZASLAVSKY, A. B. **Support for real-time decision-making in mobile business applications**. In the 2nd International conference on mobile business: 2003, Vienna.

SARKER, S; WELLS, J. Understanding Mobile Handheld Device Use and Adoption. 2003: Communications of the ACM 46(12), 35–40.

SCHETINA, Erik; CARLSON, Jacob. **Sites seguros: aprenda a desenvolver e construir**. Rio de Janeiro: Campus, 2002.

SEGURANÇA DIGITAL. Revista, São Paulo. Diário. Disponível em <http://www.segurancadigital.info/dicas/49-seguranca-da-informacao/419-cross-site-scripting-xss>. Acesso em: 7 maio 2012.

SHARAF, M; CHRYSANTHIS, P. **Data and content**: facilitating mobile decision making. Proceedings of the 2nd ACM mobicom international workshop on mobile commerce, 2002.

SHAW, M; GARLAN, D. **Software Architecture**: perspectives on an emerging discipline, 1 ed, Nova Jersey, Prentice-Hall, 1996.

SILVA, Lino Sarlo. **Virtual Private Network – VPN**: Aprenda a Construir Redes Privadas em Plataformas Linux e Windows. São Paulo: Editora Novatec, 2003.

SOFRON, E.; TUTANESCU, I. **Anatomy and Types of Attacks against Computer Networks**. In: RoEduNet Conference, 2003, Iasi. Disponível em <http://conference.iasi.roedu.net/site/conference/papers/TUTANESCU_I-Anatomy_and_Types_of_Attacks_against_Computer_..pdf>. Acesso em: 5 maio 2012.

SOMMERVILLE, I. **Engenharia de software**. 6ª ed. São Paulo: Addison Wesley, 2003.

SOUSA, R. T; PUTTINI, R. S. **Firewall.** 2006. Disponível em: <http://www.redes.unb.br/security/*firewall*/*firewall*.html>. Acesso em: 5 maio 2012.

TANENBAUM, Andrew S. **Redes de Computadores**. Rio de Janeiro: Campus, 1994.

TARICA, E. **Real budgets for real people**. The Age, Australia. 2001.

THE INTERNET PROTOCOL JOURNAL. San Jose: CISCO. Mensal. Disponível em <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-4/ipj_7-4.pdf>. Acesso em: 25 maio 2012.

TORRES, Gabriel. **Redes de Computadores Curso Completo.** Rio de Janeiro: Axcel Books do Brasil Editora Ltda, 2001.

USDHHS, U.S Department of Health & Human Services. **Measuring Usability**. Disponível em:< http://www.usability.gov/basics/measured/index.html>. Acesso em: 4 out. 2012.

XIAOMING, Kou. **Intrusion detection model based on Android**. Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference on. (2011). :624.

ZAKABI , Rosana. Hackers brazucas detonam. **Revista da Web**, São Paulo. Edição 22, Jul. 2001.

_____. Hackers: os nossos são campeões. **Revista Veja**, São Paulo. Edição 1716, Set. 2001.

ANEXO A

Script em SQL que gera as visões de banco de dados no *MySQL*:

```
CREATE OR REPLACE VIEW alert_details AS
SELECT event.cid alert_id
     , TRUNCATE(TIMESTAMPDIFF(MINUTE, TIMESTAMP('2012-01-01'),
event.timestamp) / (SELECT preferences.value_int FROM
preferences WHERE preferences.preference = 'TIMEFRAME'), 0)
       +
TRUNCATE(CONV(SUBSTRING(CAST(SHA(CONCAT(SUBSTRING(sensor.hostn
ame, 1, INSTR(sensor.hostname, ':') - 1) , ',',
signature.sig_name, ',', signature.sig_priority, ',',
sig_class.sig_class_name)) AS CHAR), 1, 16), 16, 10) /
1000000000, 0)
       grouping_id
     , event.timestamp
     , SUBSTRING(sensor.hostname, 1,
INSTR(sensor.hostname, ':') - 1) hostname
     , signature.sig_name alert
     , signature.sig_priority priority
     , sig_class.sig_class_name alert_type
     , inet_ntoa(iphdr.ip_src) ip_src
     , ports.sport port_src
     , inet_ntoa(iphdr.ip_dst) ip_dst
     , ports.dport port_dst
  FROM event
     , sensor
     , signature
     , sig_class
     , iphdr
     , (SELECT sid
             , cid
             , udp_sport sport
             , udp_dport dport
         FROM udphdr
        UNION ALL
        SELECT sid
             , cid
             , tcp_sport sport
             , tcp_dport dport
         FROM tcphdr) ports
 WHERE TIMESTAMPDIFF(MINUTE, event.timestamp,
CURRENT_TIMESTAMP) <= (SELECT preferences.value_int FROM
preferences WHERE preferences.preference = 'LIMIT')
   AND sensor.sid = event.sid
   AND signature.sig_id = event.signature
   AND sig_class.sig_class_id = signature.sig_class_id
   AND iphdr.sid = event.sid
```

```
   AND iphdr.cid = event.cid
   AND ports.sid = event.sid
   AND ports.cid = event.cid;

CREATE OR REPLACE VIEW alerts AS
SELECT alert_details.grouping_id
     , MIN(alert_details.timestamp) from_timestamp
     , MAX(alert_details.timestamp) to_timestamp
     , alert_details.hostname
     , alert_details.alert
     , alert_details.priority
     , alert_details.alert_type
     , COUNT(*) quantity
  FROM alert_details
 GROUP BY alert_details.grouping_id
       , alert_details.hostname
       , alert_details.alert
       , alert_details.priority
       , alert_details.alert_type;
```

ANEXO B

Arquivo de configuração do *Snort*:

```
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
#     http://www.snort.org                    Snort Website
#     http://vrt-sourcefire.blogspot.com/    Sourcefire VRT
Blog
#
#     Mailing list Contact:      snort-
sigs@lists.sourceforge.net
#     False Positive reports:    fp@sourcefire.com
#     Snort bugs:                bugs@snort.org
#
#     Compatible with Snort Versions:
#     VERSIONS : 2.9.2.3
#
#     Snort build options:
#     OPTIONS : --enable-ipv6 --enable-gre --enable-mpls --
enable-targetbased --enable-decoder-preprocessor-rules --
enable-ppm --enable-perfprofiling --enable-zlib --enable-
active-response --enable-normalizer --enable-reload --enable-
react --enable-flexresp3
#
#     Additional information:
#     This configuration file enables active response, to run
snort in
#     test mode -T you are required to supply an interface -i
<interface>
#     or test mode will fail to fully validate the
configuration and
#     exit with a FATAL error
#--------------------------------------------------

######################################################
# This file contains a sample snort configuration.
# You should take the following steps to create your own
custom configuration:
#
#  1) Set the network variables.
#  2) Configure the decoder
#  3) Configure the base detection engine
#  4) Configure dynamic loaded libraries
#  5) Configure preprocessors
#  6) Configure output plugins
#  7) Customize your rule set
```

```
#  8) Customize preprocessor and decoder rule set
#  9) Customize shared object rule set
##################################################

##################################################
# Step #1: Set the network variables.  For more information,
see README.variables
##################################################

# Setup the network addresses you are protecting
var HOME_NET 192.168.0.11/1

# Set up the external network addresses. Leave as "any" in
most situations
var EXTERNAL_NET any

# 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 ssh servers on your network
var SSH_SERVERS $HOME_NET

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

# List of sip servers on your network
var SIP_SERVERS $HOME_NET

# List of ports you run web servers on
portvar HTTP_PORTS
[80,81,311,591,593,901,1220,1414,1830,2301,2381,2809,3128,3702
,4343,5250,7001,7145,7510,7777,7779,8000,8008,8014,8028,8080,8
088,8118,8123,8180,8181,8243,8280,8800,8888,8899,9080,9090,909
1,9443,9999,11371,55555]

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
```

```
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
var AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.
12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.1
88.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]

# 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 c:\snort\rules
var SO_RULE_PATH c:\snort\so_rules
var PREPROC_RULE_PATH c:\snort\preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are
relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars
work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH c:\snort\rules
var BLACK_LIST_PATH c:\snort\rules

##################################################
# Step #2: Configure the decoder.  For more information, see
README.decode
##################################################

# 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
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 th
elength of the packet
# config enable_decode_oversized_alerts

# Same as above, but drop packet if in Inline mode (requires
enable_decode_oversized_alerts)
# config enable_decode_oversized_drops

# Configure IP / TCP checksum mode
config checksum_mode: all

# Configure maximum number of flowbit references.  For more
information, see README.flowbits
# config flowbits_size: 64

# Configure ports to ignore
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53

# Configure active response for non inline operation. For more
information, see REAMDE.active
# config response: eth0 attempts 2

# Configure DAQ related options for inline operation. For more
information, see README.daq
#
# config daq: <type>
# config daq_dir: <dir>
# config daq_mode: <mode>
# config daq_var: <var>
#
# <type> ::= pcap | afpacket | dump | nfq | ipq | ipfw
# <mode> ::= read-file | passive | inline
# <var> ::= arbitrary <name>=<value passed to DAQ
# <dir> ::= path as to where to look for DAQ module so's
```

```
# Configure specific UID and GID to run snort as after
dropping privs. For more information see snort -h command
line options
#
# config set_gid:
# config set_uid:

# Configure default snaplen. Snort defaults to MTU of in use
interface. For more information see README
#
# config snaplen:
#

# Configure default bpf_file to use for filtering what traffic
reaches snort. For more information see snort -h command line
options (-F)
#
# config bpf_file:
#

# Configure default log directory for snort to log to.  For
more information see snort -h command line options (-l)
#
# config logdir:


##################################################
# Step #3: Configure the base detection engine.  For more
information, see  README.decode
##################################################

# Configure PCRE match limitations
config pcre_match_limit: 3500
config pcre_match_limit_recursion: 1500

# Configure the detection engine  See the Snort Manual,
Configuring Snort - Includes - Config
config detection: search-method ac-split search-optimize max-
pattern-len 20

# Configure the event queue.  For more information, see
README.event_queue
config event_queue: max_queue 8 log 3 order_events
content_length

##################################################
## Configure GTP if it is to be used.
## For more information, see README.GTP
##################################################

# config enable_gtp
```

```
##################################################
# Per packet and rule latency enforcement
# For more information see README.ppm
##################################################

# Per Packet latency configuration
#config ppm: max-pkt-time 250, \
#   fastpath-expensive-packets, \
#   pkt-log

# Per Rule latency configuration
#config ppm: max-rule-time 200, \
#   threshold 3, \
#   suspend-expensive-rules, \
#   suspend-timeout 20, \
#   rule-log alert

##################################################
# Configure Perf Profiling for debugging
# For more information see README.PerfProfiling
##################################################

#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks

##################################################
# Configure protocol aware flushing
# For more information see README.stream5
##################################################
config paf_max: 16000

##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort -
Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory
C:\Snort\lib\snort_dynamicpreprocessor

# path to base preprocessor engine
dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll

# path to dynamic rules libraries
# dynamicdetection directory /usr/local/lib/snort_dynamicrules

##################################################
# Step #5: Configure preprocessors
```

```
# For more information, see the Snort Manual, Configuring
Snort - Preprocessors
##################################################

# GTP Control Channle Preprocessor. For more information, see
README.GTP
# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see
README.normalize
# Does nothing in IDS mode
# preprocessor normalize_ip4
# preprocessor normalize_tcp: ips ecn stream
# preprocessor normalize_icmp4
# preprocessor normalize_ip6
# preprocessor normalize_icmp6

# Target-based IP defragmentation.  For more inforation, see
README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies
overlap_limit 10 min_fragment_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly.  For
more inforation, see README.stream5
preprocessor stream5_global: track_tcp yes, \
   track_udp yes, \
   track_icmp no, \
   max_tcp 262144, \
   max_udp 131072, \
   max_active_responses 2, \
   min_response_seconds 5
preprocessor stream5_tcp: policy windows, detect_anomalies,
require_3whs 180, \
   overlap_limit 10, small_segments 3 bytes 150, timeout 180,
\
    ports client 21 22 23 25 42 53 79 109 110 111 113 119 135
136 137 139 143 \
        161 445 513 514 587 593 691 1433 1521 2100 3306 6070
6665 6666 6667 6668 6669 \
        7000 8181 32770 32771 32772 32773 32774 32775 32776
32777 32778 32779, \
    ports both 80 81 311 443 465 563 591 593 636 901 989 992
993 994 995 1220 1414 1830 2301 2381 2809 3128 3702 4343 5250
7907 7001 7145 7510 7802 7777 7779 \
        7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910
7911 7912 7913 7914 7915 7916 \
        7917 7918 7919 7920 8000 8008 8014 8028 8080 8088 8118
8123 8180 8243 8280 8800 8888 8899 9080 9090 9091 9443 9999
11371 55555
preprocessor stream5_udp: timeout 180
```

```
# performance statistics.  For more information, see the Snort
Manual, Configuring Snort - Preprocessors - Performance
Monitor
# preprocessor perfmonitor: time 300 file
/var/snort/snort.stats pktcnt 10000

# HTTP normalization and anomaly detection.  For more
information, see README.http_inspect
preprocessor http_inspect: global iis_unicode_map unicode.map
1252 compress_depth 65535 decompress_depth 65535
preprocessor http_inspect_server: server default \
    http_methods { GET POST PUT SEARCH MKCOL COPY MOVE
    LOCK
UNLOCK NOTIFY POLL BCOPY BDELETE BMOVE LINK UNLINK OPTIONS
HEAD DELETE TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE
PROPFIND PROPPATCH BPROPFIND BPROPPATCH RPC_CONNECT
PROXY_SUCCESS BITS_POST CCM_POST SMS_POST RPC_IN_DATA
RPC_OUT_DATA RPC_ECHO_DATA } \
    chunk_length 500000 \
    server_flow_depth 0 \
    client_flow_depth 0 \
    post_depth 65495 \
    oversize_dir_length 500 \
    max_header_length 750 \
    max_headers 100 \
    max_spaces 0 \
    small_chunk_length { 10 5 } \
    ports { 80 81 311 591 593 901 1220 1414 1830 2301 2381
2809 3128 3702 4343 5250 7001 7145 7510 7777 7779 8000 8008
8014 8028 8080 8088 8118 8123 8180 8181 8243 8280 8800 8888
8899 9080 9090 9091 9443 9999 11371 55555 } \
    non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \
    enable_cookie \
    extended_response_inspection \
    inspect_gzip \
    normalize_utf \
    unlimited_decompress \
    normalize_javascript \
    apache_whitespace no \
    ascii no \
    bare_byte no \
    directory no \
    double_decode no \
    iis_backslash no \
    iis_delimiter no \
    iis_unicode no \
    multi_slash no \
    utf_8 no \
    u_encode yes \
    webroot no
```

```
# ONC-RPC normalization and anomaly detection.  For more
information, see the Snort Manual, Configuring Snort -
Preprocessors - RPC Decode
preprocessor rpc_decode: 111 32770 32771 32772 32773 32774
32775 32776 32777 32778 32779 no_alert_multiple_requests
no_alert_large_fragments no_alert_incomplete

# Back Orifice detection.
preprocessor bo

# FTP / Telnet normalization and anomaly detection.  For more
information, see README.ftptelnet
preprocessor ftp_telnet: global inspection_type stateful
encrypted_traffic no
preprocessor ftp_telnet_protocol: telnet \
    ayt_attack_thresh 20 \
    normalize ports { 23 } \
    detect_anomalies
preprocessor ftp_telnet_protocol: ftp server default \
    def_max_param_len 100 \
    ports { 21 2100 3535 } \
    telnet_cmds yes \
    ignore_telnet_erase_cmds yes \
    ftp_cmds { ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP } \
    ftp_cmds { CEL CLNT CMD CONF CWD DELE ENC EPRT } \
    ftp_cmds { EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \
    ftp_cmds { LPSV MACB MAIL MDTM MIC MKD MLSD MLST } \
    ftp_cmds { MODE NLST NOOP OPTS PASS PASV PBSZ PORT } \
    ftp_cmds { PROT PWD QUIT REIN REST RETR RMD RNFR } \
    ftp_cmds { RNTO SDUP SITE SIZE SMNT STAT STOR STOU } \
    ftp_cmds { STRU SYST TEST TYPE USER XCUP XCRC XCWD } \
    ftp_cmds { XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ XSEM } \
    ftp_cmds { XSEN XSHA1 XSHA256 } \
    alt_max_param_len 0 { ABOR CCC CDUP ESTA FEAT LPSV NOOP
PASV PWD QUIT REIN STOU SYST XCUP XPWD } \
    alt_max_param_len 200 { ALLO APPE CMD HELP NLST RETR RNFR
STOR STOU XMKD } \
    alt_max_param_len 256 { CWD RNTO } \
    alt_max_param_len 400 { PORT } \
    alt_max_param_len 512 { SIZE } \
    chk_str_fmt { ACCT ADAT ALLO APPE AUTH CEL CLNT CMD } \
    chk_str_fmt { CONF CWD DELE ENC EPRT EPSV ESTP HELP } \
    chk_str_fmt { LANG LIST LPRT MACB MAIL MDTM MIC MKD } \
    chk_str_fmt { MLSD MLST MODE NLST OPTS PASS PBSZ PORT } \
    chk_str_fmt { PROT REST RETR RMD RNFR RNTO SDUP SITE } \
    chk_str_fmt { SIZE SMNT STAT STOR STRU TEST TYPE USER } \
    chk_str_fmt { XCRC XCWD XMAS XMD5 XMKD XRCP XRMD XRSQ } \
    chk_str_fmt { XSEM XSEN XSHA1 XSHA256 } \
    cmd_validity ALLO < int [ char R int ] > \
    cmd_validity EPSV < [ { char 12 | char A char L char L } ]
> \
```

```
    cmd_validity MACB < string > \
    cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ]
string > \
    cmd_validity MODE < char ASBCZ > \
    cmd_validity PORT < host_port > \
    cmd_validity PROT < char CSEP > \
    cmd_validity STRU < char FRPO [ string ] > \
    cmd_validity TYPE < { char AE [ char NTC ] | char I |
char L [ number ] } >
preprocessor ftp_telnet_protocol: ftp client default \
    max_resp_len 256 \
    bounce yes \
    ignore_telnet_erase_cmds yes \
    telnet_cmds yes


# SMTP normalization and anomaly detection.  For more
information, see README.SMTP
preprocessor smtp: ports { 25 465 587 691 } \
    inspection_type stateful \
    b64_decode_depth 0 \
    qp_decode_depth 0 \
    bitenc_decode_depth 0 \
    uu_decode_depth 0 \
    log_mailfrom \
    log_rcptto \
    log_filename \
    log_email_hdrs \
    normalize_cmds \
    normalize_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO
EMAL ESAM ESND ESOM ETRN EVFY } \
    normalize_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU
QUIT RCPT RSET SAML SEND SOML } \
    normalize_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY
X-ADAT X-DRCP X-ERCP X-EXCH50 } \
    normalize_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR
XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \
    max_command_line_len 512 \
    max_header_line_len 1000 \
    max_response_line_len 512 \
    alt_max_command_line_len 260 { MAIL } \
    alt_max_command_line_len 300 { RCPT } \
    alt_max_command_line_len 500 { HELP HELO ETRN EHLO } \
    alt_max_command_line_len 255 { EXPN VRFY ATRN SIZE BDAT
DEBUG EMAL ESAM ESND ESOM EVFY IDENT NOOP RSET } \
    alt_max_command_line_len 246 { SEND SAML SOML AUTH TURN
ETRN DATA RSET QUIT ONEX QUEU STARTTLS TICK TIME TURNME VERB
X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE
XSTA XTRN XUSR } \
    valid_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL
ESAM ESND ESOM ETRN EVFY } \
```

```
    valid_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU
QUIT RCPT RSET SAML SEND SOML } \
    valid_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-
ADAT X-DRCP X-ERCP X-EXCH50 } \
    valid_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50
XGEN XLICENSE XQUE XSTA XTRN XUSR } \
    xlink2state { enabled }

# Portscan detection.  For more information, see
README.sfportscan
preprocessor sfportscan: proto  { all } memcap { 10000000 }
sense_level { low }

# ARP spoof detection.  For more information, see the Snort
Manual - Configuring Snort - Preprocessors - ARP Spoof
Preprocessor
# preprocessor arpspoof
# preprocessor arpspoof_detect_host: 192.168.40.1
f0:0f:00:f0:0f:00

# SSH anomaly detection.  For more information, see README.ssh
preprocessor ssh: server_ports { 22 } \
                  autodetect \
                  max_client_bytes 19600 \
                  max_encrypted_packets 20 \
                  max_server_version_len 100 \
                  enable_respoverflow enable_ssh1crc32 \
                  enable_srvoverflow enable_protomismatch

# SMB / DCE-RPC normalization and anomaly detection.  For more
information, see README.dcerpc2
preprocessor dcerpc2: memcap 102400, events [co ]
preprocessor dcerpc2_server: default, policy WinXP, \
    detect [smb [139,445], tcp 135, udp 135, rpc-over-
    http-
server 593], \
    autodetect [tcp 1025:, udp 1025:, rpc-over-http-server
1025:], \
    smb_max_chain 3, smb_invalid_shares ["C$", "D$", "ADMIN$"]

# DNS anomaly detection.  For more information, see README.dns
preprocessor dns: ports { 53 } enable_rdata_overflow

# SSL anomaly detection and traffic bypass.  For more
information, see README.ssl
preprocessor ssl: ports { 443 465 563 636 989 992 993 994 995
7801 7802 7900 7901 7902 7903 7904 7905 7906 7907 7908 7909
7910 7911 7912 7913 7914 7915 7916 7917 7918 7919 7920 },
trustservers, noinspect_encrypted

# SDF sensitive data preprocessor.  For more information see
README.sensitive_data
```

```
preprocessor sensitive_data: alert_threshold 25

# SIP Session Initiation Protocol preprocessor.  For more
information see README.sip
preprocessor sip: max_sessions 40000, \
   ports { 5060 5061 5600 }, \
   methods { invite \
             cancel \
             ack \
             bye \
             register \
             options \
             refer \
             subscribe \
             update \
             join \
             info \
             message \
             notify \
             benotify \
             do \
             qauth \
             sprack \
             publish \
             service \
             unsubscribe \
             prack }, \
   max_uri_len 512, \
   max_call_id_len 80, \
   max_requestName_len 20, \
   max_from_len 256, \
   max_to_len 256, \
   max_via_len 1024, \
   max_contact_len 512, \
   max_content_len 2048

# IMAP preprocessor.  For more information see README.imap
preprocessor imap: \
   ports { 143 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0

# POP preprocessor. For more information see README.pop
preprocessor pop: \
   ports { 110 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0
```

```
# Modbus preprocessor. For more information see README.modbus
preprocessor modbus: ports { 502 }

# DNP3 preprocessor. For more information see README.dnp3
preprocessor dnp3: ports { 20000 } \
   memcap 262144 \
   check_crc

# Reputation preprocessor. For more information see
README.reputation
preprocessor reputation: \
   memcap 500, \
   priority whitelist, \
   nested_ip inner, \
   whitelist $WHITE_LIST_PATH\white_list.rules, \
   blacklist $BLACK_LIST_PATH\black_list.rules

##################################################
# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring Snort -
Output Modules
##################################################

# unified2
# Recommended for most installs
# output unified2: filename merged.log, limit 128, nostamp,
mpls_event_types, vlan_event_types

# Additional configuration for specific types of installs
# output alert_unified2: filename snort.alert, limit 128,
nostamp
# output log_unified2: filename snort.log, limit 128, nostamp

# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT

# pcap
# output log_tcpdump: tcpdump.log
# output alert_fast: alerts.ids
# output alert_full: alerts.full

# database
output database: alert, mysql, user=clautergatti
password=oracle1234 dbname=snortdb
host=instance18577.db.xeround.com port=
output database: log, mysql, user=clautergatti
password=oracle1234 dbname=snortdb
host=instance18577.db.xeround.com port=12335

# prelude
```

```
# output alert_prelude

# metadata reference data.  do not modify these lines
include C:\Snort\etc\classification.config
include C:\Snort\etc\reference.config


##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
##################################################

# site specific rules
include $RULE_PATH\local.rules

include $RULE_PATH\attack-responses.rules
include $RULE_PATH\backdoor.rules
include $RULE_PATH\bad-traffic.rules
include $RULE_PATH\blacklist.rules
include $RULE_PATH\botnet-cnc.rules
include $RULE_PATH\chat.rules
include $RULE_PATH\content-replace.rules
include $RULE_PATH\ddos.rules
include $RULE_PATH\dns.rules
include $RULE_PATH\dos.rules
include $RULE_PATH\exploit.rules
include $RULE_PATH\file-identify.rules
include $RULE_PATH\finger.rules
include $RULE_PATH\ftp.rules
include $RULE_PATH\icmp.rules
include $RULE_PATH\icmp-info.rules
include $RULE_PATH\imap.rules
include $RULE_PATH\info.rules
include $RULE_PATH\misc.rules
include $RULE_PATH\multimedia.rules
include $RULE_PATH\mysql.rules
include $RULE_PATH\netbios.rules
include $RULE_PATH\nntp.rules
include $RULE_PATH\oracle.rules
include $RULE_PATH\other-ids.rules
include $RULE_PATH\p2p.rules
include $RULE_PATH\phishing-spam.rules
include $RULE_PATH\policy.rules
include $RULE_PATH\pop2.rules
include $RULE_PATH\pop3.rules
include $RULE_PATH\rpc.rules
include $RULE_PATH\rservices.rules
include $RULE_PATH\scada.rules
include $RULE_PATH\scan.rules
```

```
include $RULE_PATH\shellcode.rules
include $RULE_PATH\smtp.rules
include $RULE_PATH\snmp.rules
include $RULE_PATH\specific-threats.rules
include $RULE_PATH\spyware-put.rules
include $RULE_PATH\sql.rules
include $RULE_PATH\telnet.rules
include $RULE_PATH\tftp.rules
include $RULE_PATH\virus.rules
include $RULE_PATH\voip.rules
include $RULE_PATH\web-activex.rules
include $RULE_PATH\web-attacks.rules
include $RULE_PATH\web-cgi.rules
include $RULE_PATH\web-client.rules
include $RULE_PATH\web-coldfusion.rules
include $RULE_PATH\web-frontpage.rules
include $RULE_PATH\web-iis.rules
include $RULE_PATH\web-misc.rules
include $RULE_PATH\web-php.rules
include $RULE_PATH\x11.rules

#---------------------------------------------------
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc_rules
#---------------------------------------------------

# decoder and preprocessor event rules
# include $PREPROC_RULE_PATH/preprocessor.rules
# include $PREPROC_RULE_PATH/decoder.rules
# include $PREPROC_RULE_PATH/sensitive-data.rules

#---------------------------------------------------
# Step #9: Customize your Shared Object Snort Rules
# For more information, see http://vrt-
sourcefire.blogspot.com/2009/01/using-vrt-certified-shared-
object-rules.html
#---------------------------------------------------

# dynamic library rules
# include $SO_RULE_PATH/bad-traffic.rules
# include $SO_RULE_PATH/chat.rules
# include $SO_RULE_PATH/dos.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/icmp.rules
# include $SO_RULE_PATH/imap.rules
# include $SO_RULE_PATH/misc.rules
# include $SO_RULE_PATH/multimedia.rules
# include $SO_RULE_PATH/netbios.rules
# include $SO_RULE_PATH/nntp.rules
# include $SO_RULE_PATH/p2p.rules
# include $SO_RULE_PATH/smtp.rules
```

```
# include $SO_RULE_PATH/snmp.rules
# include $SO_RULE_PATH/specific-threats.rules
# include $SO_RULE_PATH/web-activex.rules
# include $SO_RULE_PATH/web-client.rules
# include $SO_RULE_PATH/web-iis.rules
# include $SO_RULE_PATH/web-misc.rules

# Event thresholding or suppression commands. See
threshold.conf
include C:\Snort\etc\threshold.conf
```