UNIVERSIDADE FEDERAL DA BAHIA

INSTITUTO DE MATEMÁTICA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Italo Valcy da Silva Brito

TRAIRA: uma ferramenta para o Tratamento de Incidentes de Rede Automatizado

Salvador

2010

Italo Valcy da Silva Brito

TRAIRA: uma ferramenta para o Tratamento de Incidentes de Rede Automatizado

Monografia apresentada ao curso de graduação em Ciência da Computação, Departamento de Ciência da Computação, Instituto de Matemática, Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profº . Luciano Porto Barreto
Co-orientador: Jerônimo Aguiar Bezerra

Salvador
2010

RESUMO

O crescimento atual da Internet tem alavancado o número de incidentes de segurança da informação. Segundo dados do CERT.Bahia, de Janeiro à Outubro de 2010 já foram reportados mais de 1500 incidentes de segurança às instituições de ensino e pesquisa monitoradas na Bahia.

Devido aos prejuízos causados por tais incidentes e sua dificuldade de prevenção, é necessário estabelecer políticas e mecanismos eficientes de tratamento e resposta a incidentes de segurança. Entretanto, um dos entraves à correta identificação de equipamentos comprometidos ou participantes em um incidente de segurança consiste na ampla existência de redes que utilizam equipamentos para traduzir ou mapear os endereços dos *hosts* internos à rede via NAT ou DHCP. O emprego dessas técnicas oculta a identificação precisa dos *hosts* internos, o que dificulta o tratamento adequado de incidentes.

Este trabalho descreve o projeto, a implementação e avaliação da ferramenta TRAIRA, a qual automatiza o procedimento de detecção, identificação e isolamento da máquina geradora do incidente. O custo da implantação da solução proposta é baixo e facilita sobremaneira o trabalho da equipe de tratamento de incidentes de segurança.

A ferramenta desenvolvida foi avaliada experimentalmente e em um ambiente real, na Universidade Federal da Bahia (UFBA). Atualmente, a equipe de segurança da UFBA utiliza essa ferramenta para tratar e responder a todos os incidentes de segurança reportados pelo CERT.Bahia. O baixo custo de execução e implantação da ferramenta indica, assim, a viabilidade de sua aplicação prática em ambientes corporativos.

Palavras-chave: segurança da informação, tratamento de incidentes de segurança, resposta a incidentes, código malicioso, NAT.

ABSTRACT

The current growth of the Internet has increased the number of computer security incidents. CERT.Bahia reported, from January to Octuber 2010, over 1500 security incidents involving education and research institutions of Bahia.

Since security incidents are hard to prevent and lead to financial losses, it is necessary to establish effective policies and mechanisms to cope with such kind of incidents. However, one of the most challenging issues related to the proper identification of affected systems is the wide use of techniques to translate or map the host internal address through NAT or DHCP. Thus, such techniques make the security incident handling process difficult to be performed since they hide the real host identification.

This work describes the design, implementation and evaluation of a tool called TRAIRA, which is a software designed to automate the process of detection, identification and containment of computers responsable for security incidents. The deployment requirements of TRAIRA are simple. In addition, TRAIRA helps the computer security incident response teams on their work.

TRAIRA was verified using experiments and a case study with the Federal University of Bahia (UFBA). At UFBA, with the use of TRAIRA it was possible to handle and respond to all security incidents reported by CERT.Bahia. The low costs deployment and utilization of TRAIRA indicate the viability of its use in real environments.

Keywords: information security, handling of security incidents, incident response, malicious software, NAT.

DEDICATÓRIA

Cerca de doze anos atrás (as lembranças daquele momento não são mais tão claras), maravilhado com um ato de pura nobreza de meus pais, fiquei a me perguntar como poderia retribuí-los. Daquele momento em diante, tenho me esforçado ao máximo para dar-lhes orgulho de seu filho e acredito que esse trabalho representa bem esse esforço.

Gostaria de dedicar esse trabalho à meus pais, pessoas honestas, trabalhadoras, de bom coração e pós-doutores na arte de educar. Eles sempre estiveram comigo durante minha caminhada, aconselhando a estudar nos momentos de desleixo e a relaxar nos momentos de estudo excessivo, afinal “a mente também precisa descansar”.

AGRADECIMENTOS

Gostaria de agradecer a todos que, direta ou indiretamente, me ajudaram nesse projeto. Em particular, agradecer à Deus e à meus pais, Valdir e Ana, fundamentais nessa conquista; meus irmãos e demais parentes; meus amigos de todos os tempos; pessoal do Graco (Gestores da Rede Acadêmica de Computação – DCC/UFBA), DAComp (Diretório Acadêmico de Computação – UFBA), PoP-BA/RNP (Ponto de Presença da RNP na Bahia), CPD/UFBA (Centro de Processamento de Dados da UFBA), PSL-BA (Projeto Software Livre Bahia), professores e funcionários do DCC/IM/UFBA, e colegas das disciplinas.

Este trabalho é apenas a ponta de um *iceberg*, fruto de um projeto muito maior iniciado em 2006 com a inacreditável aprovação no vestibular da UFBA. Ao longo desses quatro anos e meio de vivência intensiva da UFBA, conheci pessoas incríveis, pessoas fundamentais na construção de quem sou hoje. Preferi não citar nomes aqui, evitando magoar os possíveis esquecidos, mas agradeço a todos que me ajudaram, uns mais que outros, a conquistar o que conquistei, desde as primeiras oportunidades, conselhos nos momentos de indecisão, críticas, apoio moral e técnico, compreensão etc.

Não pude, no entanto, deixar de relacionar as pessoas abaixo, pois tiveram contribuição especial na realização desse TCC: Jerônimo Bezerra - *PoP-BA/RNP*; Luciano Porto - *Orientador*; Daniel Batista - *USP*; equipe do PoP-BA/RNP, CPD/UFBA e CERT.Bahia; membros do Graco.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ARP	Address Resolution Protocol,	p. 38
CAIS	Centro de Atendimento a Incidentes de Segurança,	p. 18
CERT.Bahia	Grupo de Resposta a Incidentes de Segurança da Bahia/Brasil,	p. 19
CERT.br	Grupo de Resposta a Incidentes de Segurança no Brasil,	p. 18
CGI.br	Comitê Gestor da Internet no Brasil,	p. 18
CSA	Cisco Security Agent,	p. 26
CSIRT	Computer Security Incident Response Teams,	p. 17
DHCP	Dynamic Host Configuration Protocol,	p. 23
HIPS	Host Intrusion Prevention System,	p. 26
IDS	Intrusion Detection System,	p. 26
IETF	Internet Engineering Task Force,	p. 22
L2M	Layer 2 Manager,	p. 38
MAC	Media Access Control,	p. 24
NAPT	Network Address Port Translation,	p. 23
NAT	Network Address Translation,	p. 11
OTRS	Open Source Ticket Request System,	p. 27
PoP-BA/RNP	Ponto de Presença da RNP na Bahia,	p. 19
RNP	Rede Nacional de Ensino e Pesquisa,	p. 18
RT	Request Tracker,	p. 27
RTIR	Request Tracker for Incident Response,	p. 27
UFBA	Universidade Federal da Bahia,	p. 19

SUMÁRIO

1 INTRODUÇÃO

O crescimento atual da Internet tem influenciado diretamente no aumento do número de incidentes de segurança. Falhas relacionadas à segurança tornam-se não somente mais numerosas e diversas, como também mais prejudiciais à saúde financeira e desenvolvimento geral das instituições. Atividades preventivas baseadas, por exemplo, nos resultados da análise de risco das atividades de TI de uma instituição, contribuem na diminuição do número de incidentes, porém nem todos os incidentes podem ser prevenidos (SCARFONE; GRANCE; MASONE, 2008). Dessa forma, a capacidade de tratar e responder aos incidentes reportados à uma instituição é importante para rapidamente detectar o incidente, minimizar as perdas e destruições, mitigar as falhas que foram exploradas e restaurar os serviços computacionais afetados.

Não obstante, a identificação e tratamento de um incidente de segurança não é uma tarefa simples. Ao contrário, exige que todo um processo seja executado para detectar a máquina que gerou (ou sofreu) o incidente, verificar os impactos causados na rede, aplicar o plano de recuperação para o desastre, documentar as ações realizadas e rever as políticas de segurança com o propósito de diminuir a chance de repetição do incidente (fazendo o "levantamento das lições aprendidas"). Esse processo de tratamento de incidentes deve estar alinhado com a política de segurança e sua complexidade depende das características da rede da instituição. No caso das instituições de ensino e pesquisa, por exemplo, cada unidade possui demandas específicas, o que torna o ambiente bastante heterogêneo e dificulta a resposta aos incidentes.

Em particular, este trabalho tem um foco especial nas instituições de ensino e pesquisa da Bahia que se conectam à Internet através da *Rede Nacional de Ensino e Pesquisa* (RNP) e como estas instituições têm participado do cenário de (in)segurança da Internet.

1.1 MOTIVAÇÃO

A *Rede Nacional de Ensino e Pesquisa* (RNP) opera, desde 1991, a rede acadêmica nacional (também conhecida, a partir de 2005, como *Rede Ipê*). Financiada pelos Ministérios da

Ciência e Tecnologia e da Educação, a RNP foi concebida para atender à comunidade de ensino e pesquisa brasileira, interconectando instituições e redes regionais em nível nacional e oferecendo enlaces próprios para a rede de produção (*Internet commodity*) (RNP, 2007a). Por exemplo, algumas instituições clientes da RNP na Bahia são: UFBA, UFRB, IFBA, UNEB, UEFS, UESC, UESB, IFBaiano, Fiocruz, Embrapa, UCSAL, UNIFACS.

A política de uso da RNP (RNP, 2007b) especifica que as instituições podem usar a *Rede Ipê* para promover atividades de ensino e pesquisa, exceto para:

- produção ou transmissão de dados ou materiais considerados ilegais;
- transmissão de mensagens ou material de propaganda não solicitados pelo destinatário;
- atividades que contribuam para ineficiência ou esgotamento dos recursos na rede;
- atividades que promovam a corrupção ou destruição de dados de usuários.

Por outro lado, observando-se particularmente os clientes da RNP na Bahia, percebe-se que a rede acadêmica também tem contribuído no aumento dos incidentes de segurança na Internet, conforme pode ser visto no gráfico da Figura 1.1, gerado pelo *Grupo de Resposta a Incidentes de Segurança da Bahia/Brasil* (CERT.Bahia)(CERT.BAHIA, 2010b). Esse gráfico mostra a quantidade de tentativas de ataques ao sensor de *honeypot* do CERT.Bahia, com a linha superior (linha verde) representando o total de acessos e a linha inferior (linha azul) aqueles cujo endereço de origem pertence a um dos clientes da RNP na Bahia. Tais acessos indicam, com alta probabilidade, um possível comprometimento da máquina de origem com *vírus/worms*[1], ou até mesmo que a máquina pode estar sendo usada como *bot*[2]. Ainda, segundo dados do CERT.Bahia sobre as notificações de incidentes de segurança recebidas em 2010 (dados coletados até Outubro), foram notificados 1431 incidentes do tipo *host possivelmente infectado com vírus/worm* (93.8%), 45 incidentes de *envio de spam* (3%), 23 incidentes relacionados à *violação de copyright* (1.5%), dentre outros. Em outras palavras, a partir desses dados é possível ter uma noção do impacto da comunidade baiana conectada à RNP na geração de incidentes de

[1] Segundo (CERT.BR, 2006b), *worm* é um programa malicioso capaz de se propagar automaticamente através de redes, enviando cópias de si mesmo de computador para computador. Diferente do vírus, o worm não inclui cópias de si mesmo em outros programas ou arquivos e não necessita ser explicitamente executado para se propagar. Sua propagação se dá através da exploração de vulnerabilidades existentes ou falhas na configuração de softwares instalados em computadores.

[2] Segundo (CERT.BR, 2006b), *bot* é um programa malicioso capaz de se propagar automaticamente (similar ao *worm*), explorando vulnerabilidades existentes ou falhas na configuração de softwares instalados em um computador. Adicionalmente ao *worm*, possui mecanismos para se comunicar com o invasor e ser controlado remotamente (o invasor, ao se comunicar com o *bot*, pode orientá-lo a realizar ataques contra outros computadores, furtar dados, enviar spam, etc.)

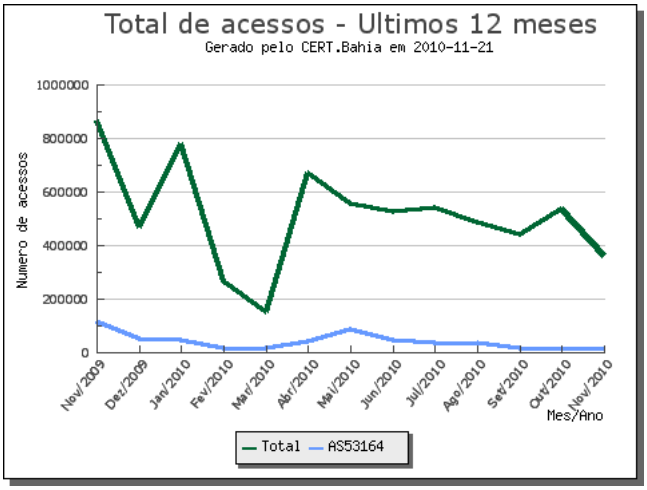

Figura 1.1: Número total de acessos ao *honeypot* versus acessos originados nos clientes da RNP na Bahia (CERT.BAHIA, 2010b)

segurança na Internet, principalmente no que concerne a incidentes do tipo *host possivelmente infectado com vírus/worm*.

O tratamento dessa gama de incidentes de segurança em instituições de ensino e pesquisa constitui um grande desafio para as equipes de segurança, principalmente levando-se em consideração o volume de notificações recebidas e a heterogeneidade da rede. Mesmo as grandes instituições, geralmente, não possuem uma equipe e ferramentas de segurança necessárias para tratar todos os incidentes. Além disso, a maior parte dos incidentes, aqueles relacionados às máquinas infectadas com *vírus/worm*, são causados por ferramentas que comprometem a máquina de forma automatizada e, no geral, são simples de resolver, apesar de trabalhosos (KAISER et al., 2006). Obviamente, a infecção automatizada de máquinas com software malicioso não pode ser tratada de forma manual. Uma alternativa é tornar o processo de tratamento desses incidentes o mais automatizado possível. Por exemplo, usando ferramentas para detectar e isolar as máquinas comprometidas de forma automatizada e contando com o apoio de uma equipe de apoio (*helpdesk*) para trabalhar na desinfecção das máquinas. Dessa maneira, reserva-se os analistas de segurança (cujo custo de contratação é comumente alto) para o tratamento de incidentes mais importantes ou complexos, e para as outras atividades de segurança da instituição.

Uma ferramenta de tratamento automatizado de incidentes deve lidar com uma dificuldade frequentemente reportada pelas instituições de ensino e pesquisa da Bahia (quiçá de grande parte das instituições): a correspondência entre o endereço IP contido na notificação recebida e aquele da máquina da rede interna[3] que, de fato, gerou o incidente. Essa dificuldade é oca-

[3]Neste trabalho, considera-se que a instituição possui uma rede interna, na qual os *hosts* são endereçados utilizando-se endereços IP privados (definidos na RFC 1918 (REKHTER et al., 1996)), e, para acessar a Internet, tais endereços são traduzidos em endereços IP roteáveis na Internet através de um dispositivo de NAT.

sionada, principalmente, pelo uso da técnica de *tradução de endereços de rede* (NAT, do inglês *Network Address Translation*), que esconde os endereços IP da rede interna da instituição (REKHTER et al., 1996), em conjunto com os mecanismos de configuração dinâmica de endereços IP (DHCP), que possibilitam que uma máquina possa ser endereçada por dois ou mais diferentes endereços IP ao longo do dia.

1.2 PROPOSTA

Os desafios supracitados motivaram o desenvolvimento de uma ferramenta e a confecção de documentos de boas práticas que, aplicados em conjunto, permitem um tratamento dos incidentes de segurança altamente automatizado. A esse conjunto de programas, interfaces e documentos de boas práticas que compõem a solução de tratamento de notificações de incidentes, deu-se o nome TRAIRA (*Tratamento de Incidentes de Rede Automatizado*). Em sua primeira versão, o TRAIRA atua nas duas primeiras fases do tratamento de incidentes (SCARFONE; GRANCE; MASONE, 2008) (*preparação* e *detecção e análise* - descrito na Seção 2.3) de forma que a detecção da máquina interna que gerou o incidente, etapa trabalhosa do processo, é totalmente automatizada. A arquitetura da ferramenta prevê ainda uma atuação na etapa de *isolamento* das máquinas comprometidas.

O TRAIRA, na verdade, originou-se a partir de um conjunto de *scripts* que eram utilizados na *Universidade Federal da Bahia* (UFBA) para busca nos *logs* dos equipamentos por evidências que indicassem a máquina interna que gerou o incidente (pois, com o uso de NAT, o endereço que é listado em uma notificação, na maioria das vezes, não é mapeado diretamente na máquina da rede interna que causou o incidente). Esses *scripts*, no entanto, eram específicos para o ambiente de rede da UFBA e de difícil extensão. Assim, o TRAIRA foi criado como uma refatoração dos *scripts* anteriores de forma a transformá-los em uma ferramenta modular e facilitar sua adaptação à outros cenários de rede, acrescentando algumas funcionalidades observadas pela equipe da UFBA. Dessa forma, o TRAIRA poderá ser usado em outras instituições e ajudar a diminuir as estatísticas apresentadas anteriormente.

Segue abaixo uma listagem não exaustiva das principais contribuições desse trabalho.

- Criação de uma ferramenta de tratamento automatizado de incidentes de segurança, o TRAIRA, a partir da refatoração do código-fonte de *scripts* utilizados na UFBA com propósito similar, que permita modularidade nas várias etapas que compõem o processo de tratamento de uma notificação e, consequentemente, utilização em diversos cenários de rede;

- Integrar a ferramenta criada, o TRAIRA, em um software que permita fazer a gerência completa do incidente de segurança. Por gerência do incidente de segurança entende-se o registro ou execução das ações relacionadas às etapas do tratamento de incidentes (SCARFONE; GRANCE; MASONE, 2008). Por exemplo, documentar as ações tomadas para um determinado incidente e indexá-las de forma a facilitar sua recuperação para consulta futura, extrair informações dos incidentes e organizá-las de forma a facilitar a geração de novas estatísticas, permitir diferentes níveis de acesso e atualização sobre uma notificação de incidente (importante quando equipes com privilégios diferentes trabalham em um incidente, por exemplo a equipe de *helpdesk* e os analistas de segurança, ou ainda o próprio usuário relacionado a um incidente), dentre outras;

- Aproveitar da arquitetura modular do TRAIRA e adicionar módulos para etapas do tratamento em que as ferramentas usadas nas instituições variam com maior probabilidade. Por exemplo, uma etapa que provavelmente difere de instituição para instituição é no mapeamento do endereço IP que aparece na notificação para a máquina da rede interna que gerou o incidente: caso utilize-se NAT na instituição, o endereço IP externo (roteável) primeiramente deverá ser mapeado para um endereço IP da rede privada e então deve-se mapear esse IP interno para a máquina que gerou a notificação (via endereço MAC); caso a instituição não utilize NAT, mas os endereços IP roteáveis sejam dinamicamente distribuídos nas máquinas internas, basta localizar a máquina da rede interna (via endereço MAC) que gerou a notificação; caso a instituição não utilize NAT e suas máquinas possuem endereço IP estático (cenário bastante incomum), esse mapeamento não é um problema. Ainda para os casos em que utiliza-se NAT, dispositivos de NAT distintos possuem formas diferentes de armazenar os *logs* das traduções;

- Implementar uma aplicação para fazer *logging* das traduções de NAT do *IPTables/Netfilter*, o *firewall* nativo do Linux. O *IPTables* não faz *logging* das traduções NAT incluindo os endereços traduzidos (IP traduzido e porta traduzida) e muito menos a duração que uma conexão de NAT levou (ou, pelo menos, não o faz de forma direta, simples e eficiente). Essas informações são essenciais no tratamento dos incidentes, pois sem elas não será garantido o correto tratamento de uma notificação (pode não ser possível detectar o IP da rede interna ou detectá-lo incorretamente, caso o IP/porta original seja diferente do IP/porta traduzida em uma conexão NAT);

- Trabalhar na etapa de tomada de ações para mitigar o problema reportado. Ao se conhecer a máquina da rede interna que gerou o incidente, é necessário aplicar uma série de medidas para resolver o problema e evitar que a máquina gere outra notificação. Exemplos

de ações são: corrigir uma configuração incorreta da máquina, executar uma varredura com anti-vírus, reinstalar o sistema comprometido, etc. Dependendo do número de notificações recebidas, pode ser inviável tratar cada máquina em tempo hábil de evitar que o incidente volte a ocorrer. Assim, uma abordagem alternativa é isolar a máquina comprometida da rede temporariamente até que a equipe de apoio execute as medidas de mitigação;

- Coletar estatísticas sobre os incidentes gerados na instituição. Atualmente, o CERT.Bahia gera algumas estatísticas sobre os incidentes por instituição, porém, considerando o uso de NAT, é interessante que a instituição mantenha estatísticas internas e mais detalhas sobre as notificações. Essas estatísticas internas são importantes para direcionar a equipe de segurança a atuar em setores específicos da instituição: as estatísticas devem ajudar a detectar a VLAN mais problemática, se existem casos de reincidência, etc.;

- Documentar a utilização do TRAIRA incluindo as boas práticas que ajudam no tratamento das notificações de incidentes. Por exemplo, gerar ou disponibilizar documentação sobre boas práticas de configuração de *firewall*, sistemas de *logging* remoto e gerenciamento desses *logs*, dentre outros.

Atualmente, o TRAIRA está sendo utilizado pela equipe de segurança da UFBA, recebendo todas as notificações de incidentes de segurança reportados à instituição e tratando de forma totalmente automatizada àquelas do tipo *host infectado com vírus/worm*. A única etapa executada manualmente no tratamento desses incidentes é justamente na etapa de mitigação e recuperação, ficando a cargo da equipe de segurança definir as ações a serem tomadas. Ainda, já é possível extrair uma série de estatísticas sobre os incidentes reportados e usá-las para direcionar medidas de mitigação. Por exemplo, é possível saber quais os segmentos de rede (VLANs) que mais geram incidentes, se existe reincidência de máquinas infectadas na rede, dentre outros.

1.3 ESTRUTURA DA DISSERTAÇÃO

O restante deste trabalho está estruturado da seguinte maneira. O Capítulo 2 aborda os principais conceitos e dificuldades envolvidos no tratamento de incidentes de segurança, além de alguns trabalhos correlatos. O Capítulo 3 aborda todos os detalhes envolvidos na implementação da ferramenta de Tratamento de Incidentes de Rede Automatizado, o TRAIRA, proposta deste trabalho, e avalia a eficácia de sua utilização através de experimentos controlados e de um estudo de caso na UFBA. Já o Capítulo 4 apresenta o NFCT-SNATLOG, uma aplicação para registro (*logging*) das traduções NAT do *IPTables/Netfilter*, e avalia o desempenho de sua

utilização através de medição. Por fim, o Capítulo 5 relaciona as conclusões obtidas a partir da produção deste trabalho e lista possíveis trabalhos futuros.

2 FUNDAMENTOS DO TRATAMENTO DE INCIDENTES DE SEGURANÇA

Este capítulo aborda os principais conceitos envolvidos no tratamento de incidentes de segurança, objeto de estudo deste trabalho. Será apresentada a diferença entre evento e incidente de segurança; abordada questões relacionadas às notificações e a importância em respondê-las, listando alguns grupos de segurança que comumente reportam incidentes; apresenta-se um exemplo de processo de tratamento de incidentes, comumente referenciado na literatura, e que serve de base para a ferramenta desenvolvida; lista-se as principais dificuldades encontradas nas instituições de ensino e pesquisa para o tratamento dos incidentes; e, por fim, relaciona-se alguns trabalhos correlatos.

2.1 EVENTOS E INCIDENTES DE SEGURANÇA

Segundo (SCARFONE; GRANCE; MASONE, 2008), um *evento* é qualquer ocorrência observável em um sistema ou na rede. Por exemplo, um usuário que acessa o servidor de arquivos, um servidor web que recebe uma requisição HTTP, um usuário que inicia uma sessão SMTP para envio de e-mail, ou mesmo o *firewall* que bloqueia uma tentativa de conexão. Por outro lado, um *evento adverso* é aquele que tem consequência negativa para a instituição, por exemplo, um sistema em colapso, inundação (*flooding*) de pacotes na rede, uso não autorizado de sistemas, acesso não autorizado à dados sensíveis, execução de código malicioso para destruição de dados, dentre outros. Vale salientar que alguns eventos adversos podem ocorrer por desastres naturais ou mesmo falhas de energia, porém este trabalho se preocupa apenas com eventos adversos relacionados com a segurança do software dos computadores.

Um *incidente de segurança* pode ser definido como qualquer evento adverso, confirmado ou sob suspeita, relacionado à segurança de sistemas de computação ou de redes de computadores (CERT.BR, 2006a). Alguns exemplos de incidentes de segurança são:

- Negação de Serviço. Neste tipo de incidente o atacante torna, ou visa tornar, indisponível um serviço ou recurso. Exemplos de incidentes de negação de serviço incluem: um atacante envia pacotes com conteúdo especial a um servidor web, causando colapso no servidor e indisponibilizando o serviço; centenas de computadores infectados, distribuídos pela Internet, enviam tantas mensagens *ICMP echo-request* quanto puderem para a rede de uma determinada instituição, congestionando e inviabilizando o uso da rede; dentre outros.

- Código Malicioso. O atacante visa executar código malicioso no *host* remoto para obter controle sobre o sistema ou propagar *vírus/worm*. Por exemplo, *worms* que usam sistemas de compartilhamento de arquivos abertos para infectar as estações de trabalho de uma organização. Tal código malicioso é disseminado de diversas maneiras, incluindo, por exemplo, arquivos PDF.

- Acesso não autorizado. Um atacante executa um *exploit*[1] para ganhar acesso a um servidor ou aumentar seu nível de acesso ao sistema (para o nível de administrador, por exemplo).

- Uso inapropriado. Um usuário disponibiliza cópias ilegais de software ou outros recursos digitais para outros usuários através de compartilhamento *peer-to-peer*, por exemplo. Outro exemplo comum é o envio de e-mail em massa não solicitado (para propaganda comercial ou disseminação de código malicioso) ou e-mails falsos para obter dados confidenciais dos usuários[2].

Mais genericamente, um *incidente de segurança* é uma violação, ou suspeita de violação, da política de segurança da informação, política de uso aceitável ou padrão de práticas de segurança de uma instituição (SCARFONE; GRANCE; MASONE, 2008). Por isso, é importante que a instituição tenha esses documentos bem definidos e disponíveis a seus usuários, principalmente a política de uso aceitável, para deixar claro o que é permitido e o que não é no uso da rede e dos recursos computacionais da instituição. Maiores informações sobre tais documentos podem ser obtidas em (CERT.BR, 2006a).

[1] Um *exploit* é um programa, porção de dados ou sequência de comandos que explora vulnerabilidades conhecidas de um sistema computacional.

[2] No geral esse tipo de incidente recebe uma classificação especial, por exemplo, *envio de spam*.

2.2 NOTIFICAÇÕES DE INCIDENTES DE SEGURANÇA

A ocorrência de ataques contra sistemas computacionais (incidente de segurança) geralmente está atrelada a duas situações principais: ataques automatizados feitos por programas maliciosos que executam em sistemas comprometidos (por exemplo, um *bot* ou um *worm*); pessoas mal intencionadas que podem ou não fazer uso de ferramentas que automatizam ataques. Em ambos os casos é importante alertar o(s) responsável(eis) pela segurança da instituição envolvida no incidente para que possa-se tomar as medidas de detecção e resolução do problema (no caso de ataques realizados por *bot* ou *worm*), evidenciar o mau comportamento de um usuário ou uma invasão de um sistema que ainda não havia sido detectada (no caso de ataques realizados por pessoas). Esses alertas são mais conhecidos por *notificações de incidentes de segurança*.

Além de enviar as notificações para os responsáveis pela máquina que originou o incidente, é importante também enviar uma cópia da notificação para os grupos de resposta a incidentes de segurança (CSIRT, do inglês *Computer Security Incident Response Teams*) das redes envolvidas (tanto da rede que gerou quanto da qual se está conectado, caso existam), de forma que eles possam gerar estatísticas sobre os incidentes e trabalhar estratégias de mitigação para problemas comuns.

A notificação de incidente de segurança deve incluir dados suficientes para que seja possível ao administrador da rede detectar a origem exata da atividade maliciosa, permitindo finalmente tomar as providências cabíveis. Abaixo uma lista dos dados essenciais que devem ser incluídos em uma notificação (CERT.BR, 2006a):

- *logs* completos. Devem ser incluídas todas as mensagens de *logs* que evidenciam a ocorrência do incidente sempre que possível ou com as informações permitidas;
- data, horário e *timezone* (fuso horário) dos *logs* ou da ocorrência sendo notificada;
- endereço de origem do ataque, incluindo IP e porta da conexão que causou o incidente.

Acerca dos grupos de resposta a incidentes de segurança, eles podem atuar também como organizações que reportam incidentes, quando i) possuem sensores para análise do tráfego de rede e descoberta automatizada de incidentes, gerando, dessa forma, notificações aos responsáveis pelas redes de forma automática ou simplesmente ii) encaminham as notificações recebidas aos responsáveis de fato pela rede em questão. No caso dos sensores, eles podem ser baseados na análise de fluxos da rede ou em sistemas de *honeypot*[3].

[3]Segundo um dos líderes do *Projeto Honeynet*, Lance Spitzner, *um honeypot é um recurso computacional de*

Nas subseções seguintes serão apresentados alguns desses grupos que atuam no Brasil e seus escopos de atuação.

2.2.1 CERT.BR

O *Grupo de Resposta a Incidentes de Segurança no Brasil* (CERT.br), mantido pelo *Comitê Gestor da Internet no Brasil* (CGI.br), é responsável por tratar incidentes de segurança em computadores que envolvam redes conectadas à Internet no Brasil (CERT.BR, 2010).

Dentre as atribuições do CERT.br estão:

- ser um ponto central para notificações de incidentes de segurança no Brasil, provendo a coordenação e o apoio no processo de resposta a incidentes, colocando as partes envolvidas em contato, quando necessário;
- manter estatísticas sobre os incidentes a ele reportados;
- desenvolver documentação de apoio para usuários e administradores de redes Internet.

Dessa forma, o CERT.br atende a qualquer rede brasileira conectada à Internet, recebendo, encaminhando e gerando notificações de incidentes de segurança aos responsáveis pela rede em questão. Em especial, no que tange à geração de notificações de incidentes, dois projetos do CERT.br têm um destaque especial:

- *Honeynet.BR*: rede brasileira de sensores distribuídos baseados em *honeypots* com o objetivo de aumentar a capacidade de detecção de incidentes, correlação de eventos e determinação de tendências de ataques no espaço Internet brasileiro (HONEYNET.BR, 2010);
- *Spampots*: o objetivo deste projeto é obter, através de *honeypots* de baixa interatividade, dados relativos ao abuso da infra-estrutura de Internet para o envio de *spam* (CERT.BR, 2007).

2.2.2 CAIS/RNP

O *Centro de Atendimento a Incidentes de Segurança* (CAIS), mantido pela *Rede Nacional de Ensino e Pesquisa* (RNP), atua na detecção, resolução e prevenção de incidentes de segurança

segurança, cujo valor reside em ser sondado, atacado e comprometido. A ideia básica de um sistema de *honeypot* é simular alguns serviços, porém sem divulgá-los em local algum. Com isso, todo acesso a ele é suspeito e potencialmente malicioso. Geralmente esses acessos são registrados em *logs* e usados para o envio de notificações às instituições de origem.

na rede acadêmica brasileira, além de elaborar, promover e disseminar práticas de segurança em redes (CAIS, 2010). Algumas das atribuições do CAIS são:

- Atendimento a incidentes de segurança;
- Coordenação com grupos de segurança já existentes;
- Fomento à criação de novos grupos de segurança no país;
- Disseminação de informações na área de segurança em redes;
- Divulgação de recomendações e alertas;
- Testes e recomendação de ferramentas de segurança;
- Gerar recomendações de políticas para a RNP, para os Pontos de Presença (PoPs) da RNP e para o backbone da RNP.

O CAIS atende à todas as instituições que se conectam à RNP no Brasil, recebendo e encaminhando notificações de incidentes relacionados a tais instituições. O CAIS também possui uma série de sensores para geração de notificações, além de parcerias nacionais e internacionais para coleta de evidências de incidentes. Não obstante, ele incentiva e apoia a criação de grupos de segurança brasileiros acadêmicos, tanto nas instituições usuárias da RNP quanto nos PoPs.

2.2.3 CERT.BAHIA

Grupo de Resposta a Incidentes de Segurança da Bahia/Brasil (CERT.Bahia) é o grupo responsável pelo tratamento e resposta aos incidentes de segurança relacionados à comunidade baiana conectada à RNP, mantido pelo *Ponto de Presença da RNP na Bahia* (PoP-BA/RNP) em parceria com a *Universidade Federal da Bahia* (UFBA) (CERT.BAHIA, 2010a). O objetivo do CERT.Bahia é ajudar e guiar as organizações na prevenção, detecção e tratamento dos incidentes de segurança, bem como criar e disseminar práticas para uso e administração seguros das Tecnologias de Informação (TI).

O CERT.Bahia hospeda um dos sensores do projeto *Honeynet.BR* e utiliza os dados coletados no sensor para geração de estatísticas (CERT.BAHIA, 2010b) e para notificação de incidentes de segurança cuja origem é um cliente da RNP.

Além de enviar notificações às instituições, o CERT.Bahia acompanha e apoia o processo de resposta aos incidentes, fornecendo treinamento, documentação e dicas de boas práticas,

além de desenvolver (ou ajudar no desenvolvimento) ferramentas para atender a demandas de automatização de subprocessos do tratamento dos incidentes de segurança. Em particular, este trabalho é fruto de uma demanda observada pelo CERT.Bahia no tratamento dos incidentes pelas instituições clientes do PoP-BA/RNP (descrito na Seção 2.4).

2.3 TRATAMENTO DE INCIDENTES DE SEGURANÇA

O tratamento de incidentes de segurança envolve três funções: notificação do incidente, análise do incidente e resposta ao incidente (CERT/CC, 2010). Conforme discutido na Seção 2.2, a notificação do incidente é importante para que um CSIRT possa centralizar e correlacionar os incidentes de segurança a fim de determinar tendências e padrões das atividades maliciosas, gerando recomendações de estratégias preventivas às instituições envolvidas. A análise do incidente, além da correlação dos eventos, envolve também estudar a notificação e as atividades do incidente para determinar seu escopo, prioridade e ameaças, pesquisando por respostas a incidentes anteriores que sejam semelhantes ao atual e montando estratégias de mitigação. A resposta ao incidente consiste de uma metodologia organizada para gerir as consequências de uma violação de segurança da informação, onde um CSIRT pode enviar recomendações para recuperação, contenção e prevenção, à instituição envolvida no incidente.

Nesse sentido, faz-se necessária a definição de um *processo de resposta a incidentes de segurança*, cujo objetivo é minimizar o impacto de um incidente e permitir o restabelecimento dos sistemas o mais rápido possível (CERON et al., 2009). A definição de tal processo deve observar alguns princípios que norteiam a concepção de um sistema de tratamento de incidentes de segurança. Segundo (SCARFONE; GRANCE; MASONE, 2008), o processo de resposta a incidentes é composto principalmente de quatro fases, conforme ilustrado na Figura 2.1 e descrito a seguir:

Figura 2.1: Ciclo de vida da resposta a incidentes - adaptado de (SCARFONE; GRANCE; MASONE, 2008)

- **Preparação**: esta fase inicial envolve o estabelecimento e treinamento de um grupo de resposta a incidentes, além da aquisição de ferramentas e recursos necessários. Durante a

preparação, a organização também tenta limitar o número de incidentes que irão ocorrer, selecionando e implementando um conjunto de controles baseados no resultado de uma análise de riscos na organização. No entanto, alguns riscos inevitavelmente persistirão mesmo depois dos controles serem implementados, ou seja, a possibilidade de ocorrência de incidentes permanecerá. Algumas medidas são essenciais de serem adotadas nessa fase para o sucesso do tratamento do incidente nas fases seguintes, dentre elas: armazenamento seguro dos *logs* dos sistemas; atualização dos sistemas operacionais e aplicações nos computadores da organização (por exemplo, atualização das assinaturas de anti-vírus, aplicação de *patches* de segurança, etc);

- **Detecção e Análise**: nesta etapa deve-se detectar ou identificar de fato a existência de um incidente. Os incidentes podem ser detectados por diferentes maneiras, com níveis variados de detalhes e fidelidade ao fato real. A detecção automatizada inclui *sistemas de detecção e prevenção de intrusão* (IDPS) baseados na rede ou no *host*, softwares anti-vírus e analisadores de *logs*. A detecção manual geralmente ocorre através do recebimento de notificações de incidentes enviadas por outros usuários. A equipe de resposta a incidentes deve então concentrar-se em identificar os sintomas do ataque e suas características, observando a severidade do incidente, ou seja, o quanto a estrutura de negócios da instituição foi afetada. Ainda, é importante que o time de resposta a incidentes mantenha uma base de conhecimento sobre os incidentes reportados no passado. Essa base pode ser confrontada com dados de novos incidentes, a fim de levantar informações como sintomas e características do ataque;

- **Contenção, Mitigação e Recuperação**: assim que o incidente é detectado e analisado, é fundamental iniciar mecanismos de contenção para evitar que ele se propague ou afete outros recursos da rede (o isolamento pode ser feito, por exemplo, desligando-se o sistema, desconectando-o da rede, desabilitando certas funções, bloqueando sua porta em um equipamento de rede mais próximo, etc). Inicia-se então o trabalho para mitigação e recuperação dos sistemas afetados. Por exemplo, para máquinas infectadas com vírus/worm, deve-se executar os softwares anti-vírus ou mesmo reinstalar o sistema. Para a recuperação, é importante que a organização tenha uma política de backup eficaz, pois a depender do incidente pode ser necessário refazer todo o sistema afetado;

- **Ações Pós-Incidente**: esta etapa consiste em avaliar o processo de tratamento de incidentes e verificar a eficácia das soluções adotadas. Deve-se relacionar as falhas e os recursos que faltaram ou não foram suficientes, para que possam ser providenciados nas próximas ocasiões. As *lições aprendidas* devem ser propagadas para toda a equipe, descrevendo

formas de obter melhores resultados e até mesmo recomendações aos usuários.

Finalizado o tratamento do incidente, é importante responder à notificação enviada, para que a organização que reportou o incidente possa atualizar seus registros locais. Essa resposta é especialmente importante quando trata-se de CSIRTs que reportam incidentes com frequência à sua rede, pois geralmente eles geram estatísticas sobre tais incidentes e usam essas estatísticas para criar ou direcionar atividades de apoio às instituições envolvidas.

2.4 PRINCIPAIS DIFICULDADES DO TRATAMENTO DE INCIDENTES

O processo de tratamento de incidentes, descrito na seção anterior, envolve a tomada de uma série de ações que dependem da fase do tratamento em questão. Por exemplo, na fase de detecção e análise, deve-se listar quais os recursos que foram afetados (no caso de incidentes contra a organização) ou qual foi a origem do incidente (no caso de incidentes originados na organização); na fase de contenção e mitigação deve-se isolar os sistemas diretamente relacionados ao incidente e efetuar o tratamento do recurso em questão (desinfecção de uma máquina contaminada com vírus/worm, remoção de um artefato malicioso, recuperação de uma página web modificada, etc).

Este trabalho se preocupa principalmente com a fase de detecção/análise para incidentes de segurança que são gerados na organização. Essa fase apresenta uma série de dificuldades que serão apresentadas nas subseções seguintes. As principais dificuldades aqui listadas estão relacionadas à utilização de NAT e DHCP na rede das instituições e dificuldade na análise de *logs* que contenham informações sobre a origem real de um incidente (a máquina da rede interna que gerou ou foi alvo do incidente).

2.4.1 NAT E O IMPACTO DE "ESCONDER" AS MÁQUINAS DA REDE INTERNA

A técnica de tradução de endereços de rede (NAT) foi desenvolvida pelo *IETF* (Internet Engineering Task Force) como uma medida paliativa para resolver o problema do esgotamento de endereços IPv4. Definida na RFC 1631 (EGEVANG; FRANCIS, 1994) (atualizada pela RFC 3022), tem como ideia básica permitir que, com um único IP roteável na Internet, ou um pequeno conjunto deles, vários *hosts* possam trafegar na Internet. Na rede interna, cada *host* recebe um endereço IP privado não roteável na Internet, definido na RFC 1918 (REKHTER et al.,

1996), que é utilizado apenas no roteamento interno. Quando o pacote precisa ser roteado para fora da rede interna, uma tradução de endereço precisa ser realizada, convertendo os endereços IP privados em endereços IP públicos roteáveis globalmente.

Na verdade, a ideia anterior descreve um dos vários tipos de NAT possíveis, o SNAT (*source NAT*, também conhecido como mascaramento, do inglês *masquerading*), que efetua a tradução da **origem** do pacote. Uma outra possibilidade é traduzir o **destino** do pacote, usando o DNAT (*destination NAT*), útil para implantação de estratégias de balanceamento de carga ou para esconder a rede dos servidores de uma instituição, por exemplo. Ainda sobre o SNAT, para realmente permitir que vários *hosts* compartilhem um único endereço IP, é preciso diferenciar os pacotes baseado no número da porta TCP/UDP. Essa variação do NAT é conhecida por *Tradução de Endereços de Rede e de Portas* (NAPT, do inglês *Network Address Port Translation*), porém ao longo deste documento NAT e NAPT terão o mesmo significado. Por exemplo, suponha que os *hosts* da rede interna 192.168.0.2 e 192.168.0.3 enviam pacotes a partir da porta de origem 1108. Um dispositivo SNAT poderá traduzir estes pacotes para um único endereço IP público, digamos 200.128.0.1, e duas portas de origem diferentes, digamos 61001 e 61002. O tráfego de resposta recebido para a porta 61001 será roteado de volta para 192.168.0.2:1108, enquanto o tráfego da porta 61002 será roteado de volta para 192.168.0.3:1108. Caso a porta original esteja disponível no dispositivo de NAT, a tradução é desnecessária, utilizando-se a mesma porta de origem que foi enviada pelo *host* da rede interna.

A utilização de NAT mostrou-se eficiente no que diz respeito à economia de endereços IP, além de apresentar alguns aspectos positivos, como facilitar a numeração interna das redes, ocultar a topologia das redes e só permitir a entrada de pacotes gerados em resposta a um pedido da rede. Não obstante, a utilização de NAT também traz uma série de inconvenientes que, para alguns, anulam suas vantagens.

No que tange ao tratamento de incidentes de segurança, a principal dificuldade está em determinar com precisão o endereço IP interno que foi traduzido no endereço externo. Esse mapeamento (aqui, o mapeamento refere-se ao inverso da tradução original) depende primeiramente do suporte do dispositivo de NAT à realização de registro (*logging*) das traduções e de uma busca nos arquivos de *logs* para correlacionar o IP, porta e data/horário com os dados da mensagem de *log*.

2.4.2 ATRIBUIÇÃO DINÂMICA DE ENDEREÇOS IP VIA DHCP

O *Protocolo de Configuração Dinâmica de Hosts* (DHCP, do inglês *Dynamic Host Configuration Protocol*) permite a um *host* obter um endereço IP automaticamente, além de outros

parâmetros de configuração da rede (máscara de sub-rede, roteador padrão, servidor DNS, etc). O DHCP foi definido na RFC 2131 (DROMS, 1997) pelo IETF, também como solução paliativa à problemática do esgotamento dos endereços IP versão 4.

O DHCP tem sido muito usado pelos provedores de acesso para permitir a atribuição de endereços temporários a seus clientes, eliminando, dessa forma, a necessidade de mapeamento um para um entre os IPs roteáveis e os clientes do provedor (essa tática tem como hipótese que nem todos os clientes acessarão a rede ao mesmo tempo). Outra utilização, ainda mais comum que a anterior, é para a configuração automática de máquinas na rede interna das instituições. Nesse caso, ao conectar um novo dispositivo à rede, lhe será automaticamente atribuído o próximo endereço IP disponível no servidor DHCP, juntamente com alguns parâmetros de configuração da rede.

Como consequência desta última utilização do DHCP, é possível que um mesmo dispositivo possua diferentes endereços IP ao longo do dia, da semana ou do mês, a depender do tempo de concessão (*lease time*). Esse parâmetro é configurável no servidor DHCP e depende de cada ambiente, variando de acordo com as demandas da rede em questão.

Com isso, é importante perceber que na fase de detecção do tratamento de um incidente de segurança, pode não ser suficiente saber o IP interno que gerou a notificação. Faz-se necessário um endereço que identifique unicamente o *host* em questão na rede. Esse identificador será o endereço *MAC* (Media Access Control), um endereço de 48 bits escrito na forma de 12 dígitos hexadecimais agrupados dois a dois (os grupos são separados por dois pontos). Metade do endereço é utilizado para identificar o fabricante do dispositivo de rede em questão e o restante é fornecido pelo fabricante, de forma que não devem existir dois dispositivos com o mesmo endereço MAC.

2.4.3 FALTA DE GERENCIAMENTO DOS *LOGS* DE DISPOSITIVOS

Um *log* é um registro dos eventos que ocorrem nos sistemas ou na rede de uma organização (SOUPPAYA; KENT, 2006). Esses registros, gerados pelos sistemas operacionais, serviços, aplicações ou dispositivos de rede, geralmente são de grande valor quando um incidente ocorre, pois muitos deles incluem dados relacionados à segurança dos sistemas (contas que acessaram o sistema, ações executadas, fluxos de rede, dentre outros).

Não obstante, a quantidade, volume e variedade dos *logs* de segurança dos sistemas têm crescido bastante, evidenciando a necessidade pelo seu gerenciamento – um processo que envolve a geração, transmissão, armazenamento, análise e descarte dos *logs* (SOUPPAYA; KENT,

2006). O correto gerenciamento dos *logs* é fundamental para a eficácia do tratamento de incidentes. Infelizmente, em muitos incidentes, os *logs* não contêm as evidências necessárias, pois o *logging* do sistema em questão ou estava desabilitado ou configurado inadequadamente. Assim, ainda na fase de preparação do processo de tratamento de incidentes, a instituição deve atentar-se à configuração do *logging* de seus equipamentos e sistemas, enviando-os para um servidor remoto que centralize essas informações. Para mais informações e recomendações sobre o gerenciamento dos *logs* de segurança, aconselha-se a leitura de *Guide to Computer Security Log Management*, um documento de recomendações do NIST (SOUPPAYA; KENT, 2006).

No caso dos incidentes de segurança que são gerados por uma instituição, seu tratamento geralmente inclui buscar nos *logs* do dispositivo de NAT por uma ocorrência do IP e porta listados na notificação e cuja data e hora estejam em concordância com os dados da notificação (levando-se em consideração a eventual falta de sincronismo entre os horários da notificação e dos *logs*). Os sistemas operacionais incluem uma série de ferramentas que ajudam nesse desafio (no GNU/Linux, por exemplo, existem as ferramentas *grep*, *awk*, dentre outras), porém ainda assim esse é um processo bastante trabalhoso e, em muitos casos, inviável de ser feito manualmente (principalmente considerando o volume de *logs* gerado e a insuficiência de recursos de segurança nas instituições – pessoal e ferramentas – já discutidos anteriormente).

Todavia, o tratamento dos *logs* é uma etapa totalmente passível de automatização desde que o sistema de *logging* remoto da instituição esteja bem configurado.

2.5 TRABALHOS CORRELATOS

Os incidentes de segurança trazem prejuízos financeiros e administrativos às instituições envolvidas. No caso dos incidentes gerados por uma instituição, por exemplo, além do mau uso dos recursos computacionais para realização de atividades maliciosas, a rede da instituição pode sofrer sanções em outras redes através de bloqueios e filtragem dos pacotes originados naquela instituição. Atividades preventivas são importantes para evitar os incidentes de segurança, porém, nem todos eles podem ser evitados. Assim, a capacidade de tratar e responder aos incidentes reportados a uma instituição é fundamental para detectar e mitigar as atividades maliciosas, reduzir os gastos e perdas, e prover um retorno à organização que reportou o incidente de que aquelas atividades estão sendo monitoradas e trabalhadas (evitando, assim, que a instituição sofra sanções pela pouca importância dada à notificação ou recorrência da mesma).

No entanto, dado o volume de notificações recebidas por uma instituição e a falta de recursos humanos para tratá-las (em tempo hábil), faz-se necessária a utilização de ferramentas

que automatizem ao máximo o tratamento dos incidentes de segurança, ou, no mínimo, aqueles mais simples. A literatura apresenta uma série de trabalhos sobre a definição de políticas de segurança e tratamento dos incidentes (CERON et al., 2009; SCARFONE; GRANCE; MASONE, 2008; WERLINGER; BOTTA; BEZNOSOV, 2007; LUNDELL, 2009), porém poucos deles têm se preocupado com a automatização do tratamento dos incidentes.

(SCARFONE; GRANCE; MASONE, 2008) fornece um guia completo para o tratamento de incidentes de segurança, direcionado a times de segurança novos e já estabelecidos. Ele apresenta diretivas para organização da capacidade de resposta a incidentes de segurança, incluindo a definição das políticas e procedimentos necessários ao processo de tratamento dos incidentes, informações para estruturação de um time de resposta a incidentes, ações envolvidas nas fases de tratamento do incidente (desde a preparação até a fase de lições aprendidas) e instruções para tratamento de tipos específicos de incidentes (DoS, código malicioso, acesso não autorizado, dentre outros). Esse guia é importante no estabelecimento e manutenção da equipe de segurança de uma instituição, fornecendo uma fundamentação essencial para eficiência e eficácia do tratamento de incidentes. Os autores citam a necessidade de automatização da fase de análise dos incidentes, recomendando softwares de correlação de eventos e *logging* remoto, porém não abordam a utilização de uma única ferramenta para este fim.

Em (KAISER et al., 2006) os autores propõem um gerenciamento semi-automatizado dos incidentes de segurança, onde aqueles menos importantes devem ser tratados pelo próprio usuário envolvido, ao passo que os incidentes mais sérios deveriam ser encaminhados para uma equipe de segurança qualificada. Nesse sentido, para possibilitar ao usuário não especializado tratar um incidente, a instituição deve prover documentação suficiente e de forma compreensível abordando as questões técnicas e organizacionais relacionadas. Neste trabalho, os autores propõem um sistema de gerenciamento de incidentes, o PRISM (*Portal for Reporting Incidents and Solution Management*), que consiste dos seguintes componentes: um componente que recebe as notificações no formato IDMEF[4], um componente contendo a lógica para tratamento de incidentes e medidas corretivas relacionadas, e um componente para geração de páginas web dinâmicas apresentando ao usuário as soluções indicadas para o incidente reportado. A principal desvantagem dessa solução é exatamente no componente que recebe as notificações, pois ele está preparado para receber apenas notificações internas, não preocupando-se, por exemplo, com as questões de mapeamento entre os NATs e as máquinas internas de uma instituição.

[4]Motivado pela necessidade de se definir um padrão de arquitetura e comunicação para *Sistemas de Detecção de Intrusão* (IDS, do inglês *Intrusion Detection System*), o IETF, através do grupo de trabalho IDWG (*Intrusion Detection Working Group*) especificou um protocolo, o IDXP (*Intrusion Detection Exchange Protocol*) (FEINSTEIN; MATTHEWS, 2007), e um formato para troca de alertas entre IDSs, o IDMEF (*Intrusion Detection Message Exchange Format*) (DEBAR; CURRY; FEINSTEIN, 2007). Apesar de originalmente pensados para sistemas IDS, esses padrões também são bastante utilizados para notificações de incidentes de segurança.

Já (FARNHAM, 2009) apresenta uma solução proprietária da Cisco, o *Cisco Security Agent* (CSA) e seu impacto nas várias fases do tratamento de incidentes. O CSA é um sistema de prevenção de intrusão baseado em *hosts* (HIPS, do inglês *Host Intrusion Prevention System*), que pode ser usado tanto em servidores quanto em máquinas clientes. A ideia do CSA é ter um agente em cada *host* e um centro de gerenciamento, que define as políticas de segurança do *host* e centraliza o registro de eventos (*logging*). O software é baseado em regras, examinando as atividades do sistema (no agente) e o tráfego de rede a fim de determinar quais comportamentos são normais e quais podem indicar um ataque. No artigo, o autor analisa a adequação do CSA nas etapas de tratamento de um incidente, usando como estudo de caso o software malicioso *Conficker*[5]. Alguns pontos negativos dessa solução estão relacionados principalmente ao custo de implantação e de sua adequação à ambientes heterogêneos, como no caso das instituições de ensino e pesquisa.

A maioria dos CSIRTs usa sistemas de *helpdesk* (também conhecidos como sistemas de chamados) para tratar seus incidentes, alguns deles com modificações para melhor atender às demandas do processo de tratamento de incidentes (KAISER et al., 2006). Dois sistemas bem conhecidos são o *Request Tracker* (RT) (BESTPRACTICAL, 2010a) e o *Open Source Ticket Request System* (OTRS) (OTRS, 2010). Existe ainda uma extensão para o RT chamada *Request Tracker for Incident Response* (RTIR) (BESTPRACTICAL, 2010b), que se concentra no tratamento de incidentes de segurança. Uma outra alternativa para o gerenciamento de incidentes é o SIRIOS (KLINGMÜLLER, 2005), que inclui algumas funcionalidades interessantes, como a de gerenciamento de contatos de segurança de uma instituição e a possibilidade de troca de informações de segurança com outros CSIRTs. Nenhum deles, no entanto, se preocupa especificamente com a automatização do processo de tratamento e resposta a incidentes.

[5]O *Conficker*, também conhecido como *Downadup* ou *Kido*, é um *worm* que ficou bastante conhecido pelo número de sistemas que conseguiu infectar ao redor do mundo. Ele explora uma vulnerabilidade conhecida do MS Windows Server (MS08-067) e pode comprometer uma série de versões do Windows (CWG, 2010).

3 TRAIRA: UMA FERRAMENTA PARA TRATAMENTO DE INCIDENTES DE REDE AUTOMATIZADO

Neste capítulo serão apresentados os detalhes envolvidos na implementação da ferramenta de Tratamento de Incidentes de Rede Automatizado, o TRAIRA, proposta deste trabalho. Para isso, inicialmente, será mostrada uma visão geral sobre a ferramenta, listando seus objetivos e principais funcionalidades; em seguida, apresenta-se a arquitetura do TRAIRA e seu fluxo de funcionamento, analisando cada uma das etapas do tratamento de incidentes que é executada automaticamente na ferramenta; apresenta-se uma avaliação da eficácia do TRAIRA, mostrando, através de experimentos controlados, a capacidade do software em tratar e responder aos incidentes reportados; e, por fim, serão listados alguns requisitos para a implantação do TRAIRA em uma organização, abordando desde os recursos necessários (servidor de *logs* centralizado, base de informações sobre os *hosts* da rede incluindo seus endereços MAC, etc) até sua instalação propriamente dita.

3.1 VISÃO GERAL DO TRAIRA

O TRAIRA (*Tratamento de Incidentes de Rede Automatizado*) é um software que atua nas duas primeiras fases do tratamento de incidentes de segurança (SCARFONE; GRANCE; MASONE, 2008), a saber, na fase de *preparação* e na fase de *detecção e análise*, de forma a automatizar as atividades trabalhosas que são executadas nessas fases. Na fase de preparação destacam-se duas principais recomendações de boas práticas: i) a configuração do serviço de *logging* remoto do equipamento de NAT e ii) a utilização de um sistema de registro sobre a atribuição de IPs, associando-os aos endereços físicos dos dispositivos de rede: os endereços MAC.

Já na fase de detecção e análise, o TRAIRA utiliza os recursos configurados anteriormente para automaticamente extrair as informações relevantes de uma notificação; buscar por evidências nos *logs* do dispositivo de NAT que informem o(s) IP(s) interno(s) responsável(veis) pela notificação recebida; associar o endereço IP interno à um endereço de MAC da máquina, de forma que sua identificação seja única; gerar relatórios e estatísticas sobre os incidentes recebidos; e responder à organização que reportou o incidente. Ainda, sua arquitetura de funcionamento prevê uma extensão desse processo até a fase de contenção, pois permite que a máquina (representada pelo seu endereço MAC) seja bloqueada no *switch* gerenciável (ou roteador) mais próximo. Essa extensão, no entanto, não está disponível na primeira versão do TRAIRA, pois no processo de desenvolvimento foram encontrados alguns desafios em sua modelagem que inviabilizaram sua implementação nesse primeiro momento (uma discussão detalhada sobre isso será fornecida na Subseção 3.2.1 seguinte). Ao final do tratamento de uma notificação, o TRAIRA gera uma resposta automática à organização que reportou o incidente e também um relatório detalhado para a equipe de apoio a fim de que as medidas cabíveis possam ser aplicadas.

Durante seu desenvolvimento, o TRAIRA foi idealizado para integrar-se a alguma ferramenta que já fosse utilizada nas instituições, ao invés de ser mais uma ferramenta e um serviço a ser mantido (evitando todas as implicações que isso traz – sistema de autenticação, backup, segurança da própria aplicação, atualização, etc). Obviamente, é muito difícil encontrar uma ferramenta que seja comumente utilizada pelas instituições e ainda permita a integração com um sistema externo, como o TRAIRA. No caso das instituições de ensino e pesquisa clientes da RNP, uma ferramenta bastante utilizada é o *Request Tracker* (RT), como sistema de *helpdesk*. O RT permite a adição de novas funcionalidades através de *extensões*, disponibilizando, inclusive, uma extensão para o gerenciamento dos incidentes de segurança, o *Request Tracker for Incident Response* (RTIR) – discutido em mais detalhes na Subseção 3.2.3 a seguir.

Na seção seguinte, a arquitetura de funcionamento do TRAIRA será apresentada em maiores detalhes.

3.2 ARQUITETURA DO TRAIRA

A arquitetura do TRAIRA é apresentada na Figura 3.1. Nessa figura, os componentes em formato de elipse (na cor azul) representam os módulos que foram desenvolvidos como parte do TRAIRA e os componentes em formato de retângulo (na cor cinza) representam softwares ou recursos externos necessários ao funcionamento do TRAIRA. O software é desenvolvido como uma extensão do RT, permitindo que o tratamento dos incidentes de segurança seja feito tanto

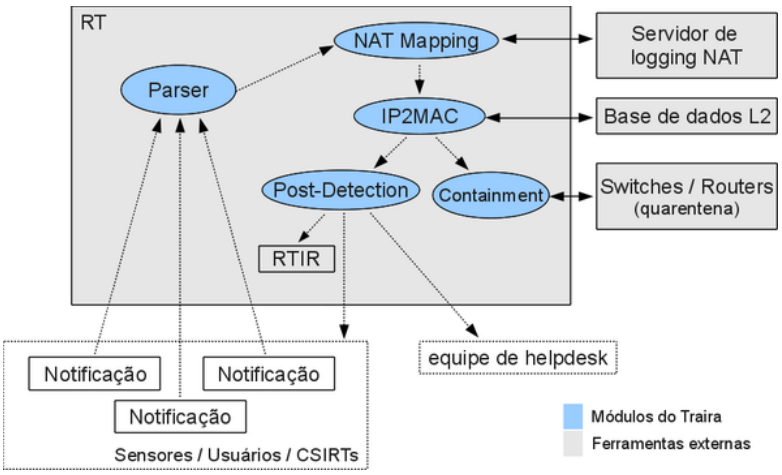

Figura 3.1: Visão geral da arquitetura do TRAIRA

pela interface web, onde o usuário fornece manualmente a notificação do incidente, quanto via e-mail quando a organização que reporta o incidente envia uma mensagem para um endereço de e-mail especialmente designado para este fim. A linguagem de programação utilizada é o *Perl*[1], com a qual o próprio RT é escrito. Em sua primeira versão, possui aproximadamente 2.500 linhas de código entre interfaces de usuário, núcleo da aplicação, módulos de interface com recursos externos (*logs*, tabela de endereços MACi, etc) e demais componentes, sendo implementado por 01 desenvolvedor em cerca de 30 dias. O TRAIRA é distribuído sob a licença GPLv2 ou superior[2] e encontra-se disponível para download em (TRAIRA, 2010).

O TRAIRA é composto por cinco módulos (objetos em formato de elipse com preenchimento azul na Figura 3.1), conforme listado abaixo e detalhado a seguir (na Subseção 3.2.1):

- *Parser:* é o módulo responsável pelo recebimento da notificação e pela extração das informações essenciais ao tratamento do incidente: endereço IP e porta de origem, data e horário.

- *NAT Mapping:* este módulo utiliza as informações extraídas pelo *parser* e realiza uma busca nos *logs* do dispositivo de NAT para associar a tupla `<data, hora, IP, porta>` a um endereço IP interno, responsável de fato pelo incidente.

- *IP2MAC:* aqui é feita a associação do IP interno ao endereço MAC da máquina. Esse passo é importante em instituições que utilizam o DHCP, pois um IP pode ter sido usado por mais de uma máquina ao longo do tempo.

[1] *Perl* é uma linguagem de programação interpretada, multiplataforma e bastante usada no processamento de cadeias de caracteres (*strings*), manipulação de texto e casamento de padrões (através de expressões regulares) (PERL.ORG, 2010).

[2] GPL é uma sigla usada para *GNU Public License*, uma licença de software livre especificada pela *Free Software Foundation*.

- *Post-Detection:* Este módulo é responsável pela extração de dados da notificação e do tratamento realizado a fim de gerar estatísticas sobre os incidentes, gerar relatórios à equipe de *helpdesk* (para, por exemplo, efetuar o isolamento e desinfecção da máquina) e responder à instituição que reportou o incidente.
- *Containment:* Neste módulo é feito o isolamento do *host* que causou o incidente para evitar que ele continue com a atividade maliciosa na rede ou afete outros recursos.

Dessa forma, o tratamento de incidentes que podem ser automatizados pelo TRAIRA segue o seguinte fluxo de trabalho:

1. Uma entidade submete uma notificação ao TRAIRA reportando um problema de segurança. Essa notificação deve conter evidências suficientes para comprovar a atividade maliciosa, incluindo, no mínimo, o endereço IP e porta de origem, data e hora que ocorreu o incidente (além do *timezone*). A entidade que reporta incidentes pode ser materializada nos CSIRTs (CAIS, CERT.Bahia, CERT.br, etc), em sensores de monitoramento de atividades maliciosas (IDSs, *honeypots*, etc) ou até mesmo em usuários que submetem incidentes através da interface web;
2. O TRAIRA verifica se existe um *parser* definido para aquela notificação e, caso exista, extrai os dados importantes da notificação e passa para a etapa de detecção da máquina interna. Caso não exista um *parser* definido, a notificação permanecerá em aberto no RT aguardando pelo tratamento manual da equipe de segurança;
3. Usando os dados extraídos da notificação (tupla `<data, hora, IP, porta>`) é feita uma busca nos *logs* do dispositivo de NAT para determinar qual o IP interno utilizou o IP e porta reportados;
4. Uma nova busca é feita; agora para mapear o IP interno em um endereço MAC da máquina que causou o incidente;
5. De posse do endereço MAC, o TRAIRA notifica a equipe de *helpdesk* para que possa tomar as medidas cabíveis;
6. Uma resposta automática é enviada à instituição que reportou o incidente informando que a máquina foi detectada e que os procedimentos de desinfecção já foram iniciados (no caso de máquinas infectadas com *vírus/worm*, por exemplo).

Como se pode perceber, o TRAIRA automatiza todo o processo inicial de tratamento de incidentes de segurança. Não obstante, o tratamento de um incidente ainda não está completo

com os passos acima, e o TRAIRA deixa a cargo do administrador definir a próxima providência a ser tomada (por exemplo, desinfectar uma máquina contaminada com *vírus/worm* ou aplicar as medidas administrativas cabíveis à uma violação de *copyright*). Vale destacar, no entanto, que, dado o volume de notificações que uma instituição recebe e a falta de equipe suficiente para responder às notificações, essa etapa de detecção muitas vezes nem chega a ser iniciada. Assim, o TRAIRA proporciona uma importante contribuição para o processo de tratamento e resposta aos incidentes de segurança de uma instituição.

As etapas descritas acima são executadas de forma *on-line*, ou seja, assim que um incidente é reportado ao TRAIRA, o tratamento é iniciado. A duração do tratamento vai depender da capacidade computacional (processador e acesso a disco) do servidor em que o TRAIRA esteja instalado.

Na subseção seguinte, serão apresentados os módulos do TRAIRA listados anteriormente.

3.2.1 MÓDULOS DO TRAIRA

Nesta subseção serão apresentados em detalhes os módulos do TRAIRA, representados na Figura 3.1 pelos componentes em formato de elipse (na cor azul), e o papel de cada um deles no tratamento dos incidentes de segurança.

PARSER

O *parser* é o módulo responsável pelo recebimento da notificação e pela extração das informações essenciais ao tratamento do incidente: endereço IP e porta de origem, data e horário. Este módulo pode ser iniciado de duas formas: quando da chegada de uma nova notificação de incidente de segurança no RT (via e-mail ou pela interface web) ou quando um usuário privilegiado do TRAIRA fornece manualmente a notificação. Caso a notificação seja fornecida manualmente, o usuário deverá informar também qual *parser* deverá ser usado; caso a notificação tenha sido criada no RT via e-mail ou pela interface web, o TRAIRA selecionará automaticamente um dos *parsers* previamente cadastrados para tratá-la.

A escolha do *parser* relativo à uma notificação é feita a partir de uma conjunção considerando-se os campos remetente (*From*) e assunto (*Subject*) da notificação. Assim, caso a notificação tenha sido reportada por um certo e-mail e siga um certo padrão no assunto, o *parser* será aplicado sobre essa notificação. Caso contrário, o processo de tratamento automatizado do incidente é abortado e a notificação fica em um estado *pendente* no RT, para tratamento manual da equipe de segurança. O cadastro de novos *parsers* e a definição do código de extração dos

dados da notificação são feitos na própria interface do TRAIRA (RT), e devem ser escritas na linguagem *Perl*.

Como exemplo de utilização desse módulo, considere uma notificação de incidente de segurança reportada pelo CERT.Bahia à instituição *InstituicaoTeste*. Essa notificação, enviada por e-mail, segue um certo padrão de formatação por parte da equipe do CERT.Bahia, que pode ser observado na Listagem 3.1.

Listagem 3.1: Exemplo de notificação enviada pelo CERT.Bahia para *InstituicaoTeste*

```
Subject: Aviso de deteccao de Virus/Worm com origem em 200.128.99.1 (InstituicaoTeste)
From: PoP-BA Security Team <incidentes@pop-ba.rnp.br>
To: Instituicao Teste CSIRT <security@instituicaoteste.edu.br>

Prezado(a) responsavel pela InstituicaoTeste,

O endereco IP 200.128.99.1 foi detectado como possivelmente
infectado e propagando virus. Abaixo seguem algumas evidencias coletadas:

53164 | 200.128.99.1 | 2010-03-31 22:50:20 srcport 51774 | (GMT-3) | PoP-BA/RNP
53164 | 200.128.99.1 | 2010-04-01 10:38:11 srcport 59441 | (GMT-3) | PoP-BA/RNP
53164 | 200.128.99.1 | 2010-04-01 10:58:00 srcport 59441 | (GMT-3) | PoP-BA/RNP
53164 | 200.128.99.1 | 2010-04-01 13:10:30 srcport 24475 | (GMT-3) | PoP-BA/RNP

Pedimos a gentileza de verificar. Caso tenha alguma duvida, favor entrar em contato.

Atenciosamente,
--
CERT.Bahia :: Grupo de Resposta a Incidentes de Seguranca da Bahia/Brasil
PoP-BA/RNP :: Ponto de Presenca da RNP na Bahia
Tel.: +55 71 3283-6098
```

A Tabela 3.1 mostra um exemplo de definição de *parser* que pode ser utilizado para as notificações que atendem àquele padrão. Na criação do chamado no RT para a notificação acima, o módulo `Traira::Parser` tentará fazer a correspondência dos campos *From* e *Subject* com o definido no *parser*. Note que os valores definidos no *parser* são, na verdade, expressões regulares e portanto toleram pequenas variações entre as notificações (endereço IP no *Subject*, por exemplo). Em seguida, para cada linha da notificação, o *código do parser* é executado, retornando a tupla `<data, hora, IP, porta>` que será usada na próxima etapa da notificação.

Note ainda a utilização do método `Traira::Parser->AdjustTimezone`, cujo objetivo é ajustar a data e hora enviados na notificação com aquelas configuradas no sistema. Por exemplo, no caso das notificações enviadas pelo CERT.Bahia, o *timezone* usado é o *GMT-3* (ou *BRT*). Suponha agora que a instituição que recebeu o incidente armazena os *logs* do dispositivo de NAT no formato *GMT*. Nessa situação, a data/horário reportados na notificação deverão ser somados de três horas para que os registros de eventos estejam sincronizados. O *timezone*

Tabela 3.1: *Parser para notificações enviadas pelo CERT.Bahia*

From	`incidentes@pop-ba.rnp.br`
Subject	`Aviso de deteccao de Virus/Worm com origem em [0-9.]{7,15} \(InstituicaoTeste\)`
Código do *parser*	`my $IP  = '[0-9.]{7,15}';` `my $DATE = '[0-9-]{10}';` `my $TIME = '[0-9:]{8}';` `my $PORT = '[0-9]+';` `if ($line =~ /^53164 | ($IP) \| ($DATE) ($TIME) srcport ($PORT)\b.*$/) {` `  my ($date, $time) = $self->AdjustTimezone($2, $3, '-0300');` `  return ($date, $time, $1, $4);` `}` `return undef;`

utilizado pela instituição é configurável pela interface web do TRAIRA (RT).

O resultado da execução do módulo `Traira::Parser` sobre aquela notificação (Listagem 3.1) é mostrado na Tabela 3.2. Observe, por exemplo, os ajustes de horários que foram necessários devido à diferença de *timezone* utilizada, inclusive modificando o dia da ocorrência de um dos eventos.

Tabela 3.2: *Resultado da execução do módulo Traira::Parser sobre a notificação da Listagem 3.1*

#	Data	Hora	IP externo	Porta externa
1	2010-04-01	01:50:20	200.128.99.1	51774
2	2010-04-01	13:38:11	200.128.99.1	59441
3	2010-04-01	13:48:00	200.128.99.1	59441
4	2010-04-01	16:10:30	200.128.99.1	24475

A definição de *parsers* especializados permite tratar um grande número de incidentes de forma automatizada, pois a maioria deles inclui as informações necessárias já listadas. Nesse sentido, pode-se afirmar que o TRAIRA é adaptável para novos tipos de incidentes, sendo necessário, apenas, definir um novo *parser* para a notificação recebida.

NAT MAPPING

Apesar de suas desvantagens (EGEVANG; FRANCIS, 1994, Seção 4), o NAT é uma técnica bastante utilizada pelas instituições de ensino e pesquisa conectadas à Internet, principalmente pela possibilidade de economia de endereços IPv4 e ocultação da topologia da rede interna. Por outro lado, sua utilização traz implicações no tratamento de incidentes de segurança, uma vez que o endereço listado na notificação não representa diretamente a máquina da rede interna que realmente causou o incidente. Nesse caso, faz-se necessário um mapeamento entre o IP e porta

listados na notificação e o IP interno que causou o incidente. Para esse mapeamento, o módulo `Traira::NATMapping` utiliza as informações extraídas pelo *parser* e as correlaciona aos *logs* do(s) dispositivo(s) de NAT, retornando o(s) IP(s) internos responsáveis pela notificação.

O processo de correlacionar essas duas entradas, no entanto, não é uma tarefa simples, pois deve-se levar em consideração a grande diversidade de dispositivos de NAT disponíveis (e cada um deles gera os *logs* de forma particular), o grande volume de dados a serem processados e, ainda pior, a correspondência entre a data/horário que é listado na notificação e aqueles listados nos *logs*. No caso da data/horário, dois pontos exigem maior atenção: i) é difícil, na prática, manter relógios sincronizados (principalmente porque a entidade que reporta incidentes não tem essa obrigação); ii) uma tradução NAT possui duração temporal e seu registro nos *logs* do servidor não representa diretamente o momento em que ela ocorreu – sabe-se, todavia, que foi no instante em que o evento foi registrado no *log* menos a duração do NAT (por exemplo, se o *log* foi registrado às `16:31:11` e o NAT possui duração de 120 segundos, então a tradução ocorreu às `16:29:11`, considerando o horário do servidor de *logs*).

Listagem 3.2: Exemplos de *logs* de traduções NAT no *firewall* ASA da Cisco

```
1 Apr  1 01:50:54 172.16.254.1 %ASA-0-305012: Teardown dynamic UDP translation from
      int_in:10.1.0.8/51386 to int_out:200.128.99.1/51774 duration 0:00:30
2 Apr  1 13:39:56 172.16.254.1 %ASA-0-305012: Teardown dynamic TCP translation from
      int_in:192.168.0.37/60523 to int_out:200.128.99.1/59441 duration 0:02:30
3 Apr  1 13:50:57 172.16.254.1 %ASA-0-305012: Teardown dynamic TCP translation from
      int_in:10.2.0.44/50071 to int_out:200.128.99.1/59441 duration 0:03:00
4 Apr  1 16:12:28 172.16.254.1 %ASA-0-305012: Teardown dynamic UDP translation from
      int_in:10.1.10.9/60818 to int_out:200.128.99.1/24475 duration 0:02:21
```

Como exemplo para o problema supracitado, considere os *logs* exibidos na Listagem 3.2 (com *timezone* em *GMT*) extraídos de um ambiente experimental da instituição *Instituicao-Teste*, que utiliza o *firewall* ASA da Cisco (CISCO, 2005). Nesse ambiente, todas as máquinas da rede interna acessam a Internet através de um único IP de NAT (`200.128.99.1`). Após receber uma notificação de incidente de segurança do CERT.Bahia, Listagem 3.1, e extrair as informações mais importantes desta notificação (através do *parser*), Tabela 3.2, é necessário buscar nos arquivos de *logs* pelo endereço IP interno que gerou cada um dos incidentes. Nesse exemplo, fica claro que não é suficiente filtrar pelo endereço IP e porta nos *logs*, pois o endereço `200.128.99.1:59441`, por exemplo, aparece listado duas vezes tanto na notificação quanto nos *logs*, inviabilizando a distinção entre os eventos apenas com esse filtro simples. É preciso considerar também a data e horário em que a conexão aconteceu. Analisando, por exemplo, a tupla 3 da Tabela 3.2, os *logs* apresentam dois candidatos a responsável por aquele incidente: uma tradução que terminou às `13:39:56` e teve duração de `00:02:30`, ou seja, iniciou às `13:37:26` e terminou às `13:39:56`, e uma segunda tradução que reaproveitou a porta de origem ante-

rior porém finalizou às 13:50:57, com duração de 00:03:00, ou seja, iniciou às 13:47:57 e terminou às 13:50:57. A partir do cálculo e análise desses intervalos, é possível identificar 10.2.0.44 como o endereço IP interno responsável pelo incidente.

Ainda sobre o exemplo anterior, encontra-se nele uma situação em que a falta de sincronismo entre os relógios pode prejudicar o tratamento de incidentes. Tomando a tupla 1 da Tabela 3.2 e buscando por ocorrências daquele IP/porta nos *logs* a única entrada retornada será referente a uma conexão NAT que iniciou-se à 01:50:24 e terminou à 01:50:54. Contudo, o horário reportado na notificação informa que o incidente ocorreu à 01:50:20. Nesse caso, não se pode simplesmente invalidar a ocorrência do incidente, deve-se levar em consideração a falta de sincronismo entre os relógios. Para resolver esse problema, é preciso adicionar uma tolerância temporal ao intervalo de duração do NAT listado nos *logs*. O Traira::NATMapping atualmente utiliza uma tolerância de 10 segundos para mais ou para menos.

A definição do valor de tolerância temporal mais adequado para uma instituição vai depender das características da rede. Um fator obrigatório a ser observado nessa definição é a relação de máquinas da rede interna por IP de NAT: quanto mais máquinas são associadas a um único NAT, maior será a taxa de reaproveitamento de portas e, consequentemente, menor poderá ser a tolerância à diferença nos relógios.

Para tratar da diversidade na utilização de dispositivos de NAT nas instituições, e até mesmo internamente à uma instituição (com diferentes dispositivos de NAT por segmento de rede), o módulo Traira::NATMapping foi desenvolvido de forma que seja possível definir um dispositivo de NAT para cada segmento de rede (levando-se em consideração a sobreposição entre segmentos) e um dispositivo padrão a ser usado caso não haja uma definição específica para determinado segmento de rede. Por exemplo, o administrador pode definir que a rede 200.128.99.0/24 utiliza o *ASA/Cisco*, já a rede 200.128.196.0/23 utiliza *IPTables/Netfilter* com exceção da sub-rede 200.128.197.0/28 que também utiliza *ASA/Cisco* e, finalmente, a rede 200.128.199.0/24 não utiliza NAT. Note que o mapeamento acima é sobre uma visão externa ou, mais especificamente, considerando os dados da notificação. Essa flexibilidade de configuração permite, por exemplo, definir as redes privadas (REKHTER et al., 1996) como sem NAT, o que viabiliza também o tratamento de incidentes internos (sensores na rede interna).

O tratamento dos *logs* de NAT de cada dispositivo no Traira::NATMapping é feito através de *handlers* (manipuladores), que são sub-módulos específicos por dispositivo. Os *handlers* são responsáveis pelas operações de busca nos *logs* e pela definição do formato de alguns campos de um registro de *log*. Por exemplo, o formato do par IP/porta externo (ou traduzido) no *ASA/Cisco* é IP/porta, ao passo que no *NFCT-SNATLOG/IPTables* (Capítulo 4) é

`t-src=IP t-spt=porta`.

Na configuração do TRAIRA, o administrador deverá informar, para cada segmento de rede que está sujeito a receber notificação (redes externas para sensores externos e redes internas para sensores internos), o dispositivo de NAT daquela rede e a localização do arquivo de *log* no sistema de arquivos. Por exemplo, retomando o cenário citado acima, teríamos a configuração mostrada na Tabela 3.3. Os caracteres especiais `%Y`, `%m` e `%d` na definição do arquivo de *log* serão expandidos para o ano (formato YYYY), mês (01..12) e dia (01..31) do incidente reportado, respectivamente.

Tabela 3.3: *Exemplo de configuração do módulo NATMapping do TRAIRA*

Rede	NATMapping	Arquivo de *log*
200.128.99.0/24	`Traira::NATMapping::asa_cisco`	`/var/log/local4-%Y%m%d.log`
200.128.196.0/23	`Traira::NATMapping::iptables`	`/var/log/nfct-snatlog-%Y-%m-%d.log`
200.128.197.0/28	`Traira::NATMapping::asa_cisco`	`/var/log/local4-%Y%m%d.log`
200.128.199.0/24	`Traira::NATMapping::none`	–

O resultado da execução do módulo `Traira::NATMapping` é mostrado na Tabela 3.4, que consiste dos dados iniciais extraídos pelo *parser* substituindo-se o par IP-externo/porta-externa pelo respectivo IP interno associado à cada incidente.

Tabela 3.4: *Resultado da execução do módulo Traira::NATMapping sobre os incidentes da Tabela 3.2*

#	Data	Hora	IP interno
1	2010-04-01	01:50:20	10.1.0.8
2	2010-04-01	13:38:11	192.168.0.37
3	2010-04-01	13:48:00	10.2.0.44
4	2010-04-01	16:10:30	10.1.10.9

IP2MAC

O protocolo DHCP permite a atribuição de endereços IP dinamicamente às máquinas da rede. Ao receber tal endereço, o *host* recebe também um parâmetro de tempo, definindo o período de validade daquele endereço (tempo de concessão, do inglês, *lease time*). Vencido o *lease time*, o *host* deve inutilizar o endereço ou renová-lo junto ao servidor DHCP. Como consequência, é possível que um mesmo *host* possua diferentes endereços IP ao longo do dia, da semana ou do mês, dependendo do *lease time* configurado. Como essa é uma funcionalidade bastante útil e comum nas instituições, usar somente o IP para identificar um *host* pode produzir resultados inconsistentes. Faz-se necessário um mecanismo único de identificação do dispositivo, no

mínimo, no contexto da rede interna da instituição. A identificação utilizada pelo TRAIRA é o endereço MAC, escolhido pela unicidade teórica que oferece e pela facilidade de mapeamento entre endereços IP e MAC (PLUMMER, 1982).

O mecanismo utilizado atualmente para esse mapeamento entre endereços IP e MAC é através do *Address Resolution Protocol* (ARP) (PLUMMER, 1982). Em linhas gerais, o funcionamento do protocolo ARP baseia-se no envio de requisições em *broadcast* contendo o endereço IP do *host* alvo e resposta em *unicast* com o respectivo endereço MAC. Para diminuir o número de mensagens desse tipo na rede, utiliza-se um mecanismo de cache, conhecido como *cache da tabela ARP*. Assim, bastaria consultar a tabela ARP dos equipamentos de roteamento da rede para saber o endereço MAC associado a cada endereço IP em uso. Entretanto, a tabela ARP é dinâmica e, dessa forma, não mantém o histórico de utilização da rede. Faz-se necessário, então, um software que mantenha o histórico sobre a resolução de endereços IP para MAC da rede, a fim de salvaguardar a informação do endereço IP em uso por determinado endereço MAC em determinado momento.

O esquema de resolução de IPs para MACs utilizado pelo TRAIRA vem de um software desenvolvido pela UFBA, em parceria com o CERT.Bahia, para o gerenciamento e documentação dos endereços de camada 2: o *Layer 2 Manager* (L2M) (BEZERRA, 2009). Ele permite, através de uma interface web, realizar uma série de consultas para um *host* (tanto via endereço IP quanto via endereço MAC) e obter o histórico de acesso desse *host* na rede, com horário de entrada e saída. Além disso, pode-se consultar quantas máquinas utilizam a rede em determinado momento ou, por exemplo, nos últimos 15 dias. É possível também realizar consultas diretamente ao banco de dados do L2M, com algumas funções SQL disponibilizadas com o software.

Dessa forma, o módulo `Traira::IP2MAC` recebe uma lista de endereços IP internos com data e horário de acesso à rede e retorna o endereço MAC que estava associado àquele endereço IP naquele momento. Para isso, o `Traira::IP2MAC` consulta o banco de dados do L2M ou verifica uma tabela estática de IPs/MACs configurada no próprio TRAIRA. Recomenda-se, no entanto, a utilização do L2M nesse processo, pois o valor agregado com aquela ferramenta é bem maior, além da dinamicidade que esse mapeamento pode exigir (entrada de novos *hosts* na rede) e que é coberto pelo L2M.

Voltando ao exemplo de notificação de incidente de segurança usado anteriormente, após a execução do módulo `Traira::IP2MAC`, obtém-se o resultado apresentado na Tabela 3.5, que contém agora o endereço MAC da máquina que causou cada incidente e a VLAN a que ela pertence (caso esta informação esteja disponível).

Tabela 3.5: *Resultado da execução do módulo Traira::IP2MAC sobre os endereços internos listados na Tabela 3.4*

#	Data	Hora	IP interno	MAC	VLAN
1	2010-04-01	01:50:20	10.1.0.8	00:16:3e:ef:dc:6b	Rede_Lab1
2	2010-04-01	13:38:11	192.168.0.37	00:16:3e:ad:1c:6e	Rede_InstA
3	2010-04-01	13:48:00	10.2.0.44	00:16:3e:bb:2a:3b	Rede_Wireless
4	2010-04-01	16:10:30	10.1.10.9	00:16:3e:fa:ca:1a	Rede_Lab2

POST-DETECTION

Após o processo de detecção e análise do incidente de segurança, detalhado nas subseções anteriores, a etapa seguinte consiste em extrair dados da notificação e do tratamento que facilitem gerar de estatísticas, gerar relatórios para a equipe de *helpdesk* que será responsável, por exemplo, pelo processo de isolamento e desinfecção das máquinas, e responder à organização que reportou o incidente, atualizando-a sobre o sucesso ou falha no tratamento da notificação enviada.

A Listagem 3.3 ilustra um exemplo de e-mail enviado à equipe de *helpdesk* sobre um dos incidentes dos exemplos anteriores (tupla 1 da Tabela 3.5). De posse dessa notificação, a equipe poderá efetuar o bloqueio do MAC no *switch* gerenciável ou roteador mais próximo do *host* e proceder com sua desinfecção.

Listagem 3.3: Exemplo de e-mail enviado à equipe de *helpdesk* para bloqueio/desinfecção do *host*

```
1  Subject: Solicitacao de bloqueio/desinfeccao de maquina com virus/worm
2  From: Instituicao Teste CSIRT <security@instituicaoteste.edu.br>
3  To: Helpdesk Instituicao Teste <helpdesk@instituicaoteste.edu.br>
4
5  Prezados,
6
7  Segue abaixo uma relacao <IP | MAC | VLAN> das maquinas detectadas como
8  possivelmente comprometidas com virus/worm. Favor realizar o tratamento
9  das maquinas (ex: anti-virus, etc.).
10
11 10.1.0.8 | 00:16:3e:ef:dc:6b | Rede_Lab1
12
13 Atenciosamente,
14 --
15 TRAIRA :: Tratamento de Incidentes de Rede Automatizado.
16 Instituicao Teste CSIRT
```

Em seguida, o TRAIRA atualiza o chamado de incidente de segurança que foi aberto no RT, informando à organização que o reportou sobre o estado atual do tratamento da notificação. Esse

passo finaliza a interação com a organização que reportou o incidente; as ações seguintes terão interesse apenas para o controle interno da instituição (medidas tomadas, lições aprendidas, etc).

CONTAINMENT

Este módulo prevê que seja feito um isolamento do *host* infectado para evitar que ele continue com a atividade maliciosa na rede ou comprometa outros recursos. O isolamento pode se dar de várias maneiras: desligamento da máquina, remoção do cabo de rede, desabilitando certos recursos do sistema, ou ainda bloqueando-a no dispositivo de rede gerenciável mais próximo. Essa última alternativa é mais interessante do ponto de vista de automatização do processo.

Uma outra abordagem, ainda mais atrativa, proposta em (KAISER et al., 2006), é, em vez de simplesmente bloquear a máquina no *switch* ou roteador da rede, direcioná-la à uma VLAN de quarentena, onde todas as requisições do usuário seriam direcionadas para uma página web com uma nota de que a máquina está possivelmente comprometida e apresentando recomendações para desinfecção.

Durante o desenvolvimento do TRAIRA, optou-se inicialmente apenas por bloquear o *host* no equipamento de rede gerenciável mais próximo, porém essa abordagem apresentou uma série de desafios, o principal deles foi no que tange à diversidade de equipamentos encontrados na UFBA, instituição usada como piloto para a implantação. A diversidade é concretizada, por exemplo, nos comandos a serem executados no *switch* para bloquear um MAC. Essa diversidade, no entanto, é uma característica que provavelmente será encontrada em outras instituições que utilizarem o TRAIRA, portanto, faz-se necessária uma solução que seja adaptável e extensível a novos ambientes.

Por esse motivo, definiu-se como trabalho futuro utilizar uma linguagem de comandos de *switches* que abstraia a sintaxe específica da cada equipamento, usando-a para efetuar o isolamento dos *hosts*.

3.2.2 GERAÇÃO DE ESTATÍSTICAS

Um recurso fundamental aos grupos de resposta a incidentes de segurança (CSIRTs) são as estatísticas (ARVIDSSON et al., 2001). Elas ajudam os CSIRTs a detectar tendências, prever futuros ataques em grande escala, direcionar atividades, dentre outros.

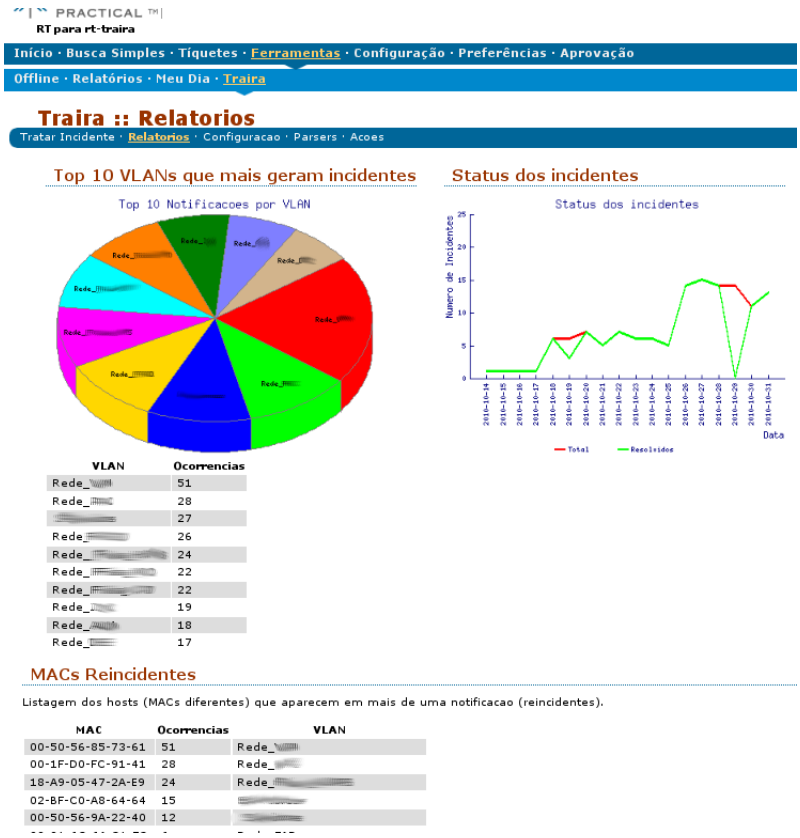

Figura 3.2: Tela do TRAIRA para exibição de relatórios/estatísticas

A implementação atual do TRAIRA fornece algumas estatísticas geradas automaticamente a partir de informações retiradas da notificação recebida e do tratamento efetuado. A Figura 3.2 mostra a tela do TRAIRA para exibição de estatísticas (relatórios), baseadas em dados experimentais. Naquela figura tem-se as seguintes estatísticas:

- *Gráfico de incidentes por VLAN.* Esse gráfico ressalta as VLANs que mais impactam na geração de incidentes de segurança, o que permite direcionar medidas de prevenção específicas;

- *Quantidade de incidentes por dia.* Esse é um gráfico quantitativo que pode ser usado para medir a efetividade do tratamento de incidentes de segurança na instituição. Ele lista os incidentes que são reportados *versus* aqueles que foram resolvidos. O esperado é que a linha de incidentes resolvidos se aproxime da linha dos incidentes reportados e elas tendam a cair (a menos que haja implantação de novos sensores);

- *MACs reincidentes.* Esta estatística pode ser usada como indicador qualitativo do tratamento de incidentes, quando observa-se reincidência na geração de incidentes em determinado *host*. A interpretação desse dado pode levar a uma série de hipóteses, por exemplo: a fase de isolamento e desinfecção não está sendo eficaz; no caso dos inciden-

tes de *vírus/worm* pode indicar inexperiência do usuário no uso do recurso, propiciando novas infecções com facilidade; dentre outros.

3.2.3 INTEGRAÇÃO COM O RT

O *Request Tracker* (RT) é um sistema de chamados, de distribuição livre, que permite interação via e-mail e via interface web. Bastante utilizado para gerência de problemas (funcionando como um sistema de gestão de chamados para *helpdesk*) ou para gerência de *bugs*, o RT também possui uma extensão para gerência de incidentes de segurança: o *Request Tracker for Incident Response* (RTIR). Além das funcionalidades básicas de controle de acesso e agrupamento de chamados por categorias através de filas, o RT possibilita definir *campos personalizados* que são úteis para indexar informações mais importantes de um chamado, facilitando a geração de buscas para composição de estatísticas. Essas características, em conjunto com outras não listadas aqui, foram decisivas na escolha do RT como ferramenta base para o TRAIRA.

Para habilitar o TRAIRA no RT basta, a partir de uma instalação funcional do RT, instalar a extensão `RT::Extension::Traira` (disponível em (TRAIRA, 2010)) e configurá-lo via interface web. A configuração consiste basicamente em definir uma fila do RT que será usada especificamente para os incidentes de segurança (o que viabiliza utilizar o RT não somente para o tratamento de incidentes de segurança, mas também para outras categorias, ou mesmo aproveitar um ambiente já em produção da instituição); definir a configuração do dispositivo NAT (qual dispositivo de NAT é usado por rede e a localização do arquivo de *log* daquele dispositivo); definir o *timezone* que é usado no sistema; definir a forma de mapeamento dos IPs e MACs (e os parâmetros de acesso ao L2M, se for o caso); registrar os *parsers*; dentre outras configurações opcionais. Finalizada a configuração, o TRAIRA estará preparado para efetuar o tratamento automático dos incidentes de segurança criados na fila definida anteriormente e que sejam atendidos por um determinado *parser*.

Todavia, o tratamento de um incidente de segurança não se restringe às etapas cobertas pelo TRAIRA (detecção e análise, e isolamento): é necessário ainda um *gerenciamento* sobre tais incidentes. Gerenciar um incidente, nesse contexto, está atrelado à todo o fluxo de trabalho e processo de resposta a incidente que os CSIRTs devem seguir (SCARFONE; GRANCE; MASONE, 2008). Nesse sentido, acredita-se que o TRAIRA pode integrar-se facilmente ao RTIR, utilizando as recursos desse último após o tratamento inicial que o TRAIRA realiza. Um trabalho futuro consiste, portanto, em avaliar a integração entre essas duas ferramentas e as vantagens dessa abordagem.

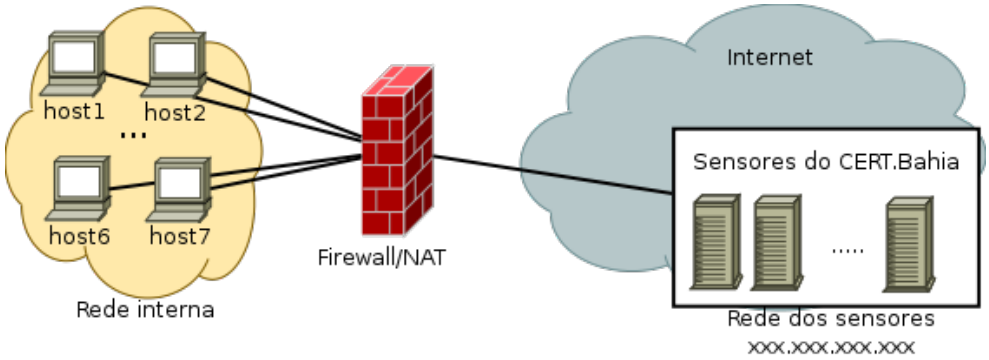

Figura 3.3: Cenário utilizado nos experimentos do TRAIRA

3.3 AVALIAÇÃO EXPERIMENTAL

Para medir a eficácia da proposta, foram realizados alguns experimentos na rede do Departamento de Ciência da Computação da UFBA (DCC/UFBA). O objetivo dos experimentos foi gerar uma série de incidentes de segurança em um ambiente controlado e verificar se eles seriam corretamente tratados pelo TRAIRA. Dessa forma, tratando-se de um ambiente controlado, saberia-se a priori as máquinas que deveriam ser listadas em cada incidente que fosse tratado.

Para realização dos testes, fez-se uma parceria com o CERT.Bahia para atacar propositalmente alguns de seus sensores e receber as notificações de incidentes, nesse caso simulando máquinas infectadas com *vírus/worm*. As notificações no CERT.Bahia são agrupadas por endereço IP de origem. Portanto, para que fosse possível gerar mais de uma notificação, o endereço IP do NAT variou entre dez endereços possíveis. A Figura 3.3 ilustra o cenário utilizado nos experimentos. Naquela figura, cada uma das máquinas da rede interna (sete máquinas: *host* 1..*host* 7) "ataca" as portas 445 (Compartilhamento do Windows), 139 (NetBIOS) e 22 (SSH) dos sensores (na Internet). A escolha das portas foi baseada naquelas mais utilizadas por atacantes, segundo estatísticas do CERT.Bahia (CERT.BAHIA, 2010b). O "ataque" foi feito utilizando o utilitário *nmap*, através do comando `nmap -p 445,139,22 xxx.xxx.xxx.xxx` (`xxx.xxx.xxx.xxx` são os IPs dos sensores, omitidos por questões de segurança). Como resultado do suposto ataque, dez notificações de incidentes de segurança (uma para cada IP de NAT) foram enviados pelo CERT.Bahia para o e-mail do responsável pela rede em questão (`security@dcc.ufba.br`).

O e-mail `security@dcc.ufba.br` foi configurado para redirecionar as mensagens ao RT, a fim de que o tratamento dos incidentes de segurança fossem realizados pelo TRAIRA.

Analisando os resultados do tratamento efetuado pelo TRAIRA, ele foi capaz de detectar,

em cada notificação recebida, sete máquinas da rede interna como responsáveis pelos incidentes e identificar cada uma dessas máquinas. Foram enviados dez e-mails ao CERT.Bahia, em resposta a cada uma das notificações, e dez e-mails para a equipe de operação, solicitando a desinfecção de sete máquinas (cada um dos e-mails continha as mesmas máquinas, gerando duplicidade nesse caso específico).

Assim, pode-se considerar que o TRAIRA foi eficaz no tratamento dos incidentes de segurança reportados. Entretanto, caso o TRAIRA não fosse capaz de tratar um determinado incidente de forma automática, a equipe de segurança não teria perda alguma: a notificação continuaria pendente de tratamento manual no RT.

Além dos experimentos realizados, o TRAIRA encontra-se em fase de homologação na UFBA, onde será usado como base para o processo de tratamento de incidentes de segurança da instituição. Na verdade, antes da adoção do TRAIRA, a UFBA já utilizava uma série de *scripts* que tratavam os incidentes de *vírus/worm* enviados pelo CERT.Bahia, porém estendê-los para tratar outros tipos de incidentes (ou incidentes em outro formato) não era uma tarefa simples e, ainda, não existia geração de relatórios automatizada. Atualmente, para tratar outros incidentes, basta cadastrar um novo *parser* no TRAIRA. Sobre os relatórios, a interface do TRAIRA já apresenta alguns relatórios pré-configurados, mas também é possível definir novos gráficos a partir de buscas configuradas no RT.

3.4 REQUISITOS PARA IMPLANTAÇÃO DO TRAIRA

Como já foi discutido nas seções anteriores, o TRAIRA necessita que alguns recursos sejam configurados na rede antes do início do tratamento de incidentes de segurança, ainda na fase de preparação. Não obstante, alguns desses recursos já deveriam fazer parte da infraestrutura de segurança da instituição ou, do contrário, seria impossível tratar os incidentes reportados (com exceção de alguns casos específicos e incomuns – como, por exemplo, da não utilização de NAT e distribuição estática de endereços na rede interna).

Abaixo uma listagem dos recursos necessários ao correto funcionamento do TRAIRA.

- **Registro remoto dos eventos de tradução de NAT (SNAT).** Para as instituições que utilizam NAT para tradução de endereços da rede interna no acesso à Internet (SNAT), é necessário que o dispositivo de NAT faça o registro (*logging*) das traduções incluindo informações sobre: endereços IP e portas de origem originais (antes da tradução), endereços IP e portas de origem traduzidos (após a tradução) e duração do SNAT (mais

detalhes sobre a importância desses campos podem ser obtidos nas Subseções 2.4.1 e 3.2.1, tópico *NAT Mapping*). Alguns dispositivos de NAT suportam o registro das traduções de NAT (por exemplo, o *ASA/Cisco* (CISCO, 2005)), sendo necessário apenas habilitá-lo no dispositivo e enviar os *logs* a um servidor de *logging* remoto (através do protocolo *syslog* (GERHARDS, 2009), por exemplo), onde será instalado o TRAIRA. Outros dispositivos não suportam nativamente esse recurso, a exemplo do *Netfilter/IPTables* (*firewall* usado em sistemas GNU/Linux). Tendo em vista que o *iptables* é usado em muitas instituições clientes da RNP na Bahia, este trabalho propõe também uma solução integrada que será discutida no Capítulo 4. Em qualquer caso, uma questão que deve chamar a atenção dos administradores consiste no espaço em disco ocupado pelos *logs* do NAT: o tamanho do arquivo de *logs* será proporcional ao fluxo de rede afetado pelas traduções NAT. Recomenda-se, assim, a definição de uma política de gerenciamento dos *logs* (SOUPPAYA; KENT, 2006) que, no mínimo, deve especificar por quanto tempo o *log* será mantido;

- **Histórico sobre a associação entre endereços IP e MAC dos *hosts*.** Sempre que houver atribuição dinâmica de endereços IP na rede (DHCP), será necessário manter um histórico do mapeamento entre esses endereços e o endereço MAC de cada *host* (mais informações sobre esse problema podem ser obtidas nas Subseções 2.4.2 e 3.2.1, tópico *IP2MAC*). Nesse trabalho, recomenda-se a utilização do L2M, um software desenvolvido pela UFBA em parceria com o CERT.Bahia para o gerenciamento de recursos da camada 2 (modelo TCP/IP) de uma instituição;

- **Request Tracker (RT).** O TRAIRA foi desenvolvido como um módulo do RT, demandando uma instalação mínima do RT para sua utilização. Contudo, o TRAIRA não requer exclusividade na utilização do RT, o que possibilita adoção de uma instância do RT que já esteja em utilização pela instituição;

- **Banco de dados.** Tanto o RT quanto o L2M necessitam de um banco de dados para registro de suas informações. O RT suporta uma série de sistemas de gerenciamento de banco de dados, porém o L2M suporta somente MySQL. Assim, pode-se usar uma instalação MySQL compartilhada para as duas ferramentas.

Como exposto acima, a implantação do TRAIRA requer baixo investimento e tempo de instalação. Alguns de seus pré-requisitos, inclusive, são facilmente encontrados em um ambiente de produção de uma instituição. Por conseguinte, acredita-se que é viável sua implantação em diversos tipos de instituições com baixo nível de investimento e espaço.

4 NFCT-SNATLOG: UMA FERRAMENTA PARA REGISTRO DAS TRADUÇÕES SNAT DO IPTABLES/NETFILTER

O *Netfilter* é um componente do *kernel* do Linux que permite a filtragem de pacotes, tradução de endereços de rede (e portas) (NAT/NAPT) e outras manipulações de pacotes (NETFILTER, 2010b). Ele está disponível como módulo do *kernel*. O *iptables* é uma aplicação cliente, que permite a criação de regras de *firewall* e NAT, manipulando alguns dos subsistemas do *Netfilter*.

Embora o *iptables* seja robusto e bastante utilizado nas instituições, ele não dispõe de funcionalidades que permitam o registro das traduções SNAT realizadas, o que pode comprometer o tratamento de um incidente de segurança gerado por uma máquina da rede interna.

Diante dessa importante limitação, este capítulo detalha o desenvolvimento de uma aplicação que permite monitorar conexões do tipo SNAT e registrar nos *logs* informações sobre IP e porta de origem originais, IP e porta de origem traduzidas e duração da conexão. A aplicação foi chamada NFCT-SNATLOG, uma abreviação para *Netfilter Connection Tracking SNAT Logging*.

O restante deste capítulo está estruturado da seguinte maneira. A Seção 4.1 mostra porque o *iptables* não está preparado nativamente para o registros das traduções de SNAT. Em seguida, na Seção 4.2, será apresentada a ferramenta desenvolvida para o registro das traduções de NAT de origem criadas pelo *IPTables/Netfilter*. Por fim, a Seção 4.3 apresenta alguns cenários de avaliação de desempenho do NFCT-SNATLOG, mostra os experimentos realizados e os resultados alcançados, e discute estes resultados a fim de propor soluções aos problemas encontrados ou simplesmente evidenciá-los.

4.1 O PROBLEMA DO REGISTRO DE TRADUÇÕES NAT NO IPTABLES/NETFILTER

O NAT de origem (*Source NAT*, ou simplesmente SNAT) ocorre quando o endereço de origem de um determinado pacote é alterado. Isso serve, por exemplo, para mascarar a topologia de rede interna de uma instituição. Um outro tipo de NAT é o NAT de destino (*Destination NAT*, ou simplesmente DNAT), onde o endereço de destino do pacote é alterado. Uma utilização do DNAT reside na construção de *proxy* transparente, por exemplo.

O *IPTables/Netfilter* não faz, nativamente, registro das traduções SNAT, ou pelo menos não de forma que se possa usar os *logs* gerados em auditorias do sistema. Para auditoria de um sistema, é importante que os *logs* apresentem tanto o par IP/porta de origem internos (aqueles antes da tradução SNAT, ou *originais*) quanto o par IP/porta de origem externos (aqueles resultantes da tradução, ou *traduzidos*). Tal característica é inerente à forma de funcionamento do *iptables* e será explicada e exemplificada nos parágrafos seguintes. Para entender melhor o problema, serão mostrados os passos de criação de regras SNAT no *Netfilter*.

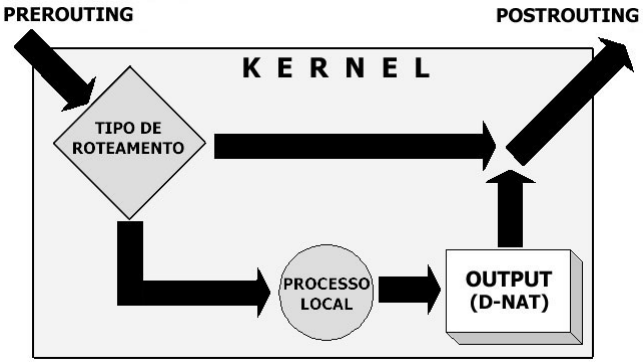

Figura 4.1: Esquema de *chains* da tabela NAT (MOTA FILHO, 2003)

O primeiro passo para criar regras de SNAT no *Netfilter* é informar ao *kernel* quais conexões serão traduzidas e como alterá-las. Para isso, utiliza-se o comando `iptables` com a opção `-t nat`, informando ao *kernel* que deseja-se alterar a tabela NAT do *Netfilter*. O *Netfilter* é divido em tabelas (`filter`, `nat`, `mangle` e `raw`) e cada tabela concentra funcionalidades específicas do sistema – a tabela `filter` concentra a funcionalidade de filtro de pacotes, ao passo que a tabela `nat` a tradução de endereços de rede e portas. Cada tabela contém filas, mais conhecidas como *chains*, que categorizam os pacotes (pacotes que chegam ao *firewall*, pacotes gerados internamente, pacotes prestes a deixar o *firewall*, pacotes sendo roteados, etc). A Figura 4.1 ilustra o esquema de *chains* para a tabela NAT. Aquela figura mostra qual *chain* se aplica a um pacote, baseado em seu fluxo no *kernel*: PREROUTING, quando o pacote está prestes à

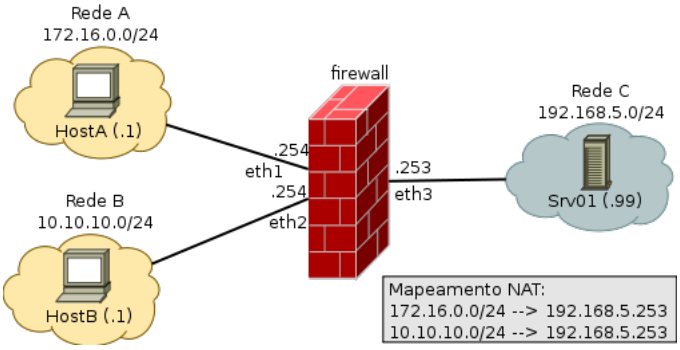

Figura 4.2: Cenário para utilização de SNAT no *IPTables/Netfilter*

iniciar o processo de roteamento do *kernel* (usado para DNAT); POSTROUTING, logo após a decisão de roteamento ter sido tomada (usado para SNAT) ou OUTPUT (para pacotes gerados localmente antes do roteamento (usado para DNAT).

Cada *chain* consiste de uma lista ordenada de regras. Uma regra, por sua vez, é composta de filtros e ações. Os filtros definem critérios a que um pacote deve corresponder (filtro por origem, destino, protocolo etc.). Ações definem o que fazer com o pacote (aceitar, descartar, etc). Cada regra na *chain* é examinada em ordem, até que uma delas corresponda ao pacote analisado, quando a ação definida na regra é aplicada. As ações possíveis para a tabela NAT são: modificar o endereço IP (e porta) de origem do pacote (SNAT) ou modificar o endereço IP (e porta) de destino do pacote (DNAT)[1].

A título de exemplo, considere a topologia de rede da Figura 4.2. Nesse cenário, deseja-se que as máquinas na rede A e na rede B acessem a rede C através do IP `192.168.5.253`. A Listagem 4.1 mostra os comandos executados na máquina `firewall` para configurar tal cenário (considerando que as configurações de rede já foram efetuadas). O parâmetro `-A POSTROUTING` adiciona uma regra ao final da *chain* POSTROUTING, o parâmetro `-o eth3` define um filtro para pacotes que sairão pela interface `eth3`, o parâmetro `-s REDE` filtra pela REDE de origem do pacote e, finalmente, o parâmetro `-j SNAT --to-source IP` define a ação SNAT modificando a origem dos pacotes para o IP especificado.

Listagem 4.1: *Comandos do iptables para implementar o cenário de SNAT da Figura 4.2*

```
1 iptables -t nat -A POSTROUTING -o eth3 -s 172.16.0.0/24 -j SNAT --to-source 192.168.5.253
2 iptables -t nat -A POSTROUTING -o eth3 -s 10.10.10.0/24 -j SNAT --to-source 192.168.5.253
```

Após a execução desses comandos, a *chain* POSTROUTING da tabela NAT do *iptables* encontra-se conforme mostrado na Tabela 4.1.

[1]Uma terceira ação possível na tabela NAT é a MASQUERADE, uma especialização da SNAT usado em conexões que atribuem dinamicamente o endereço IP.

Tabela 4.1: *Lista de regras instaladas na chain POSTROUTING da tabela NAT do iptables após a execução dos comandos da Listagem 4.1*

#Regra	Prot.	In	Out	Orig.	Dest.	Ação
1	all	*	eth3	172.16.0.0/24	0.0.0.0/0	SNAT (to:192.168.5.253)
2	all	*	eth3	10.10.10.0/24	0.0.0.0/0	SNAT (to:192.168.5.253)

Deseja-se agora registrar as traduções de SNAT realizadas na máquina `firewall` para fins de auditoria futura. Uma "solução" simplória encontrada em muitas listas de discussão para esse objetivo é adicionar uma regra imediatamente antes da regra de SNAT desejada, cuja ação seja fazer o registro dos pacotes (*logging*). Para tanto, utiliza-se a ação LOG (`-j LOG`) com os mesmos filtros da regra que deseja-se registrar[2]. Os comandos para implementar essa proposta são mostrados na Listagem 4.2.

Listagem 4.2: *Comandos do iptables para implementar um falso logging das regras SNAT definidas na Figura 4.2*

```
1 iptables -t nat -I POSTROUTING 1 -o eth3 -s 172.16.0.0/24 -j LOG --log-prefix 'SNAT '
2 iptables -t nat -I POSTROUTING 3 -o eth3 -s 10.10.10.0/24 -j LOG --log-prefix 'SNAT '
```

O resultado dos comandos da Listagem 4.2 é mostrado na Tabela 4.2.

Tabela 4.2: *Lista de regras instaladas na chain POSTROUTING da tabela NAT do iptables após a execução dos comandos da Listagem 4.2*

#Regra	Prot.	In	Out	Orig.	Dest.	Ação
1	all	*	eth3	172.16.0.0/24	0.0.0.0/0	LOG (prefixo 'SNAT ')
2	all	*	eth3	172.16.0.0/24	0.0.0.0/0	SNAT (to:192.168.5.253)
3	all	*	eth3	10.10.10.0/24	0.0.0.0/0	LOG (prefixo 'SNAT ')
4	all	*	eth3	10.10.10.0/24	0.0.0.0/0	SNAT (to:192.168.5.253)

A fim de verificar o funcionamento das regras definidas pelos comandos anteriores, usando o cenário da Figura 4.2, executa-se duas conexões simultâneas, uma da máquina *HostA* e outra da máquina *HostB*, para a máquina *Srv01* com porta de origem `40000/TCP` e porta de destino `80/TCP`. Para iniciar essas conexões, pode-se utilizar a ferramenta *netcat*[3] no *HostA* e no *HostB* da seguinte forma: `nc -p 40000 192.168.5.99 80`. Adicionalmente, na máquina *Srv01* inicia-se um monitoramento do tráfego na interface com o IP `192.168.5.99` (`tap0`) a fim de verificar os endereços que acessaram aquele *host*. Esse monitoramento é feito com o

[2]Diferente de outras ações (*targets*), como DNAT, SNAT etc., a ação LOG não é bloqueante. Ou seja, o *iptables* continuará processando a tabela de regras até encontrar outra que também corresponda ao pacote.

[3]Netcat é um utilitário de rede capaz de ler e escrever dados através de conexões de rede, usando o protocolo TCP/IP.

utilitário *tcpdump*[4] através do seguinte comando:

```
sudo tcpdump -i tap0 -n src host 192.168.5.253 and dst port 80
```

A mensagem de *log* enviada pelo *iptables* como consequência das conexões anteriormente estabelecidas é mostrada na Listagem 4.3. A saída do *tcpdump* é mostrada na Listagem 4.4.

Listagem 4.3: *Saída do log do iptables para as conexões iniciadas pelo comando nc (netcat) executado nas máquinas HostA e HostB.*

```
1 Nov 20 09:47:07 firewall kernel: [32385.753454] SNAT IN= OUT=eth3 SRC=172.16.0.1
    DST=192.168.5.99 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=52200 DF PROTO=TCP SPT=40000 DPT=80
    WINDOW=5840 RES=0x00 SYN URGP=0
2 Nov 20 09:47:08 firewall kernel: [32386.479462] SNAT IN= OUT=eth3 SRC=10.10.10.1
    DST=192.168.5.99 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=18925 DF PROTO=TCP SPT=40000 DPT=80
    WINDOW=5840 RES=0x00 SYN URGP=0
```

Listagem 4.4: *Saída do monitoramento com tcpdump em Srv01 para as conexões iniciadas com o netcat de HostA e HostB*

```
1 09:47:07.615051 IP 192.168.5.253.40000 > 192.168.5.99.80: Flags [S], seq 3428346057, win
    5840, options [mss 1460,sackOK,TS val 9478945 ecr 0,nop,wscale 2], length 0
2 09:47:07.627116 IP 192.168.5.253.40000 > 192.168.5.99.80: Flags [.], ack 128236544, win 1460,
    options [nop,nop,TS val 9478948 ecr 46727625], length 0
3 09:47:08.439314 IP 192.168.5.253.1024 > 192.168.5.99.80: Flags [S], seq 2237520910, win 5840,
    options [mss 1460,sackOK,TS val 9475555 ecr 0,nop,wscale 2], length 0
4 09:47:08.445493 IP 192.168.5.253.1024 > 192.168.5.99.80: Flags [.], ack 122234339, win 1460,
    options [nop,nop,TS val 9475555 ecr 46727831], length 0
5 ...
```

Cabe notar que as portas listadas na saída do *tcpdump* (`192.168.5.253.40000` e `192.168.5.253.1024`) são diferentes daquelas listadas nos *logs* do *iptables* (`SRC=172.16.0.1 ... SPT=40000` e `SRC=10.10.10.1 ... SPT=40000`), o que inviabilizaria o processo de auditoria para uma conexão de origem `192.168.5.253:1024`, por exemplo. Isso acontece pois quando a regra do LOG (regra 1 ou 3 da Tabela 4.2) é aplicada, a tradução ainda não aconteceu, daí o endereço registrado no *log* é o par IP/porta originais e não os traduzidos. Ainda, não há como adicionar uma regra logo após a ação SNAT (logo após 2 ou 4 na Tabela 4.2), pois essa regra nunca seria executada (lembre-se que a única ação não bloqueante é a LOG).

Destaca-se, novamente, que os resultados apresentados acima implicam que mesmo os administradores que acreditam terem habilitado o monitoramento sobre as traduções NAT no *iptables* (via, por exemplo, passos descritos na Listagem 4.2) serão impossibilitados de tratar alguns incidentes de segurança reportados à sua instituição. Mesmo considerando-se a criticidade

[4] O *tcpdump* é um utilitário de linha de comando para captura e análise de tráfego de rede.

desse resultado, não foi encontrado nenhum trabalho que aborde o problema com a importância merecida.

Faz-se necessária, assim, uma ferramenta complementar ao *iptables* que monitore as conexões de SNAT do *kernel* e informe os dados relevantes dessa conexão quando necessário. Nas pesquisas realizadas para esse trabalho, a ferramenta que mais se aproxima desses requisitos é o *conntrack-tools* (NETFILTER, 2010a) – um subprojeto do *Netfilter* cujo objetivo é prover uma interface em espaço de usuário para monitorar e gerenciar a tabela de rastreamento de conexões do *kernel* (*Connection Tracking System*, comumente abreviado para *conntrack*). Para o exemplo anterior, poder-se-ia iniciar o monitoramento das conexões de NAT com o seguinte comando: `conntrack -E -e NEW,DESTROY -n -o timestamp`. Esse comando inicia o monitoramento de eventos do tipo NEW e DESTROY (`-E -e NEW,DESTROY`, discutidos em mais detalhes na Seção 4.2), cuja conexão seja relacionada à um SNAT (`-n`) e listando o tempo em que o evento ocorreu (`-o timestamp`). O resultado do comando acima para a mesma conexão dos exemplos anteriores é mostrado na Listagem 4.5.

Listagem 4.5: *Saída do monitoramento com conntrack no firewall para as conexões iniciadas com o netcat de HostA e HostB*

```
1 [1290257227.849654]     [NEW] tcp      6 timeout=120 SYN_SENT src=172.16.0.1
      dst=192.168.5.99 sport=40000 dport=80 [UNREPLIED] src=192.168.5.99 dst=192.168.5.253
      sport=80 dport=40000
2 [1290257228.571181]     [NEW] tcp      6 timeout=120 SYN_SENT src=10.10.10.1
      dst=192.168.5.99 sport=40000 dport=80 [UNREPLIED] src=192.168.5.99 dst=192.168.5.253
      sport=80 dport=1024
3 [1290257357.613241] [DESTROY] tcp      6 src=172.16.0.1 dst=192.168.5.99 sport=40000
      dport=80 packets=4 bytes=228 src=192.168.5.99 dst=192.168.5.253 sport=80 dport=40000
      packets=3 bytes=164
4 [1290257358.505345] [DESTROY] tcp      6 src=10.10.10.1 dst=192.168.5.99 sport=40000
      dport=80 packets=4 bytes=228 src=192.168.5.99 dst=192.168.5.253 sport=80 dport=1024
      packets=3 bytes=164
```

A partir dos *logs* acima pode-se obter os tempos de início e término de uma conexão. Ainda, cada linha do *log* é composta por um par de campos do tipo `src, dst, sport, dport`, onde o primeiro deve condizer com o envio do pacote e o segundo com a resposta esperada pelo *kernel*. Em outras palavras, os valores da primeira ocorrência dos campos `src` e `sport` representam o par IP/porta originais, ao passo que os valores dos campos `dst` e `dport` na segunda ocorrência representam o par IP/porta traduzidos. Tais informações resolvem parcialmente o problema do *logging* do *iptables*.

Porém, mesmo essa abordagem, ainda não é a ideal para o cenário proposto. As principais desvantagens de utilizar o `conntrack` diretamente no *logging* das conexões SNAT do *iptables* são as seguintes:

- O escopo de atuação do `conntrack` é bem mais amplo que simplesmente fazer *logging* de conexões SNAT. Por isso, uma série de campos são registrados desnecessariamente junto ao *log* de uma conexão SNAT. Em ambientes com um grande número de *logs* sendo gravados no servidor, esses campos adicionais podem elevar a carga e os requisitos de armazenamento do sistema;

- O `conntrack` não gera saída via *syslog*, somente na saída padrão. Assim, para integrá-lo à um servidor de *logs* remoto seria necessário um terceiro processo que redirecionasse a saída padrão do `conntrack` para o servidor de *logs* remoto, gerando mais carga ao sistema;

- Inferir a duração de uma conexão a partir de duas mensagens de *log* é muito mais custoso que com apenas uma mensagem (tanto em termos de processamento e armazenamento na gravação, quanto de processamento na recuperação – sendo mais crítica a questão do armazenamento).

Os aspectos listados nessa seção motivaram a criação de uma ferramenta para o *logging* de conexões SNAT do *iptables*– o NFCT-SNATLOG – objeto de estudo da seção seguinte.

4.2 NFCT-SNATLOG: REGISTRO DAS TRADUÇÕES DE SNAT NO IPTABLES/NETFILTER

A Seção 4.1 descreveu as limitações do *IPTables/Netfilter* no registro das conexões de SNAT, enfatizando a insuficiência de informações para auditorias ou tratamento de incidentes de segurança no sistema. Além disso, uma segunda ferramenta que resolveria esse problema (o *conntrack-tools*) também apresenta desvantagens, principalmente por não ter foco específico no problema em questão. Esses fatores, atrelados à grande taxa de utilização do *iptables* nas instituições de ensino e pesquisa clientes da RNP, motivaram a criação de uma ferramenta para o *logging* das conexões SNAT do *iptables*, o *NFCT-SNATLOG*, que será detalhado nesta seção.

Os principais requisitos necessários à tal ferramenta incluem:

- Dada uma conexão de SNAT no *kernel* do Linux (*iptables*), os seguintes dados devem ser registrados sobre tal conexão (agrupados em uma única mensagem de *log*): endereço IP e porta de origem originais, endereço IP e porta de origem traduzidos, protocolo de transporte utilizado (TCP ou UDP), data e horário do fim da conexão, duração da conexão;

- Deve ser possível gravar as mensagens de *log* via protocolo *syslog*, permitindo o *logging* remoto da aplicação;

- Deve ser possível executar a aplicação como *daemon* do sistema. Um *daemon* (ou serviço) é um processo que executa em segundo plano (*background*) de forma autônoma do restante das aplicações e com pouca ou nenhuma interação com o usuário. A principal vantagem em executar uma aplicação como *daemon* está em torná-la independente de outros processos de usuário, liberando, assim, recursos do sistema que não serão necessários (por exemplo, descritores de entrada padrão, saída padrão, etc).

O *Netfilter Conntrack SNAT Logging* (NFCT-SNATLOG) é um software desenvolvido para a geração dos *logs* de SNAT do *iptables* atendendo aos requisitos acima. Baseado no *conntrack-tools*, o NFCT-SNATLOG foi programado usando a biblioteca `libnetfilter_conntrack` (NETFILTER, 2010c), que provê uma interface de programação (API) para acessar a tabela de rastreamento de conexões do *kernel* em espaço de usuário. Por meio dessa biblioteca, é possível monitorar a pilha de rede do *kernel* e registrar funções (*callbacks*) que serão chamadas na ocorrência de eventos específicos (AYUSO, 2006). Exemplos de eventos de interesse incluem: criação de novas conexões (NEW), atualização de uma conexão existente (UPDATE) ou ainda a destruição (DESTROY) de uma conexão cujo tempo limite (*timeout*) foi ultrapassado. Além disso, uma conexão pode ser filtrada pelo tipo de protocolo de transporte (TCP ou UDP), tradução NAT de origem ou destino, dentre outras.

O NFCT-SNATLOG foi desenvolvido em C, possui cerca de 500 linhas de código, é distribuído sob a licença GPLv2 ou superior[5] e encontra-se disponível para download em (NFCT-SNATLOG, 2010).

4.2.1 ARQUITETURA DO NFCT-SNATLOG

A Figura 4.3 mostra o fluxograma de execução do NFCT-SNATLOG. As principais etapas de sua execução podem ser sumarizadas nos seguintes itens:

1. Monitorar os eventos NEW e DESTROY das conexões no *kernel* do Linux;

2. Filtrar as conexões que sejam do tipo SNAT e cujo protocolo de transporte seja TCP ou UDP;

[5]GPL é uma sigla usada para *GNU Public License*, uma licença de software livre especificada pela *Free Software Foundation*.

3. Calcular a diferença de tempo entre os eventos NEW e DESTROY de determinada conexão;

4. Salvaguardar as informações relevantes dessa conexão no *log* do sistema;

5. Repassar a informação do estado da conexão ao `conntrack`, para que ele possa chamar a próxima *callback* registrada.

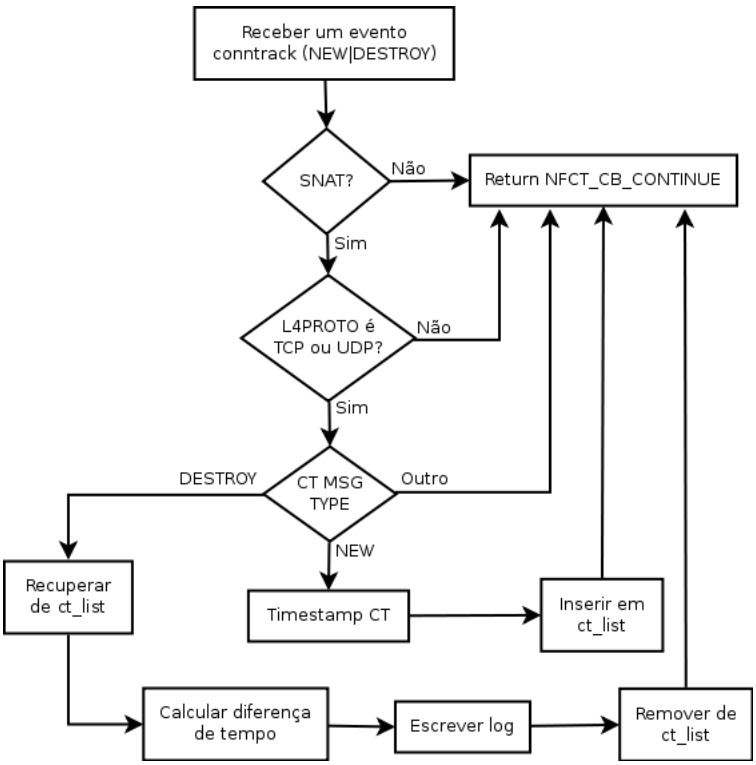

Figura 4.3: Visão geral de funcionamento do *NFCT-SNATLOG*

4.2.2 QUESTÕES DE IMPLEMENTAÇÃO

A Listagem 4.6 apresenta a principal função do NFCT-SNATLOG (`nfct_snatlog_cb`), responsável pelo tratamento dos eventos do `conntrack`.

Listagem 4.6: *Código da função do NFCT-SNATLOG responsável pelo tratamento dos eventos do conntrack*

```
1  static int nfct_snatlog_cb(enum nf_conntrack_msg_type type,
2          struct nf_conntrack *ct,
3          void *data) {
4    struct conntrack_list *no;
5    u_int8_t l4proto;
6
7    // we are interested only in SNAT connections
```

```
8     if (!nfct_getobjopt(ct, NFCT_GOPT_IS_SNAT))
9        return NFCT_CB_CONTINUE;
10
11    // We are interested only in TCP/UDP L4 protocols...
12    l4proto = nfct_get_attr_u8(ct,ATTR_ORIG_L4PROTO);
13    if (l4proto != IPPROTO_TCP && l4proto != IPPROTO_UDP)
14       return NFCT_CB_CONTINUE;
15
16    if (debug_flag) {
17       print_debug(ct, type, proto_str(l4proto));
18    }
19
20    switch(type) {
21       case NFCT_T_NEW:
22          no = (struct conntrack_list *)malloc(sizeof(struct conntrack_list));
23          no->id = nfct_get_attr_u32(ct,ATTR_ID);
24          no->orig_ipv4_src = nfct_get_attr_u32(ct,ATTR_ORIG_IPV4_SRC);
25          no->orig_port_src = nfct_get_attr_u16(ct,ATTR_ORIG_PORT_SRC);
26          time(&no->timestamp);
27          list_add(&ct_list, no);
28          break;
29       case NFCT_T_DESTROY:
30          no = list_find(ct_list,
31                nfct_get_attr_u32(ct,ATTR_ID),
32                nfct_get_attr_u32(ct,ATTR_ORIG_IPV4_SRC),
33                nfct_get_attr_u16(ct,ATTR_ORIG_PORT_SRC));
34          if (no) {
35             print_snatlog(ct, &no->timestamp, proto_str(l4proto));
36             list_del(&ct_list,no);
37          }
38          break;
39       default:
40          break;
41    }
42
43    return NFCT_CB_CONTINUE;
44 }
```

Essa função é registrada como *callback* para monitorar os eventos de criação de nova conexão (NEW) e encerramento de uma conexão existente (DESTROY). Na ocorrência de algum desses eventos, a função será chamada e receberá os seguintes parâmetros:

- `enum nf_conntrack_msg_type type`: uma constante que define o evento ocorrido (NEW, UPDATE, DESTROY, etc);
- `struct nf_conntrack *ct`: um apontador para a região de memória contendo a estrutura de dados do módulo `conntrack` que armazena informações sobre a conexão. As informações são armazenadas na forma de tuplas de *hash* (para otimizar a consulta) (AYUSO, 2006). Cada tupla contém o endereço IP de origem e destino, número da porta

de origem e destino, tipos dos protocolos, estado e *timeout*. Existem duas tuplas de *hash* para cada conexão (AYUSO, 2006): uma para a direção original do pacote (isto é, os pacotes vindos do ponto que iniciou a conexão) e uma para direção da resposta (isto é, pacotes de resposta indo para o ponto que iniciou a conexão);

- `void *data`: um apontador para uma região de memória que pode ser usada internamente pela *callback* (não será utilizada pelo NFCT-SNATLOG).

As linhas 7 a 14 da Listagem 4.6 são responsáveis pelo primeiro passo na execução do NFCT-SNATLOG: filtrar as conexões do tipo SNAT e cujo protocolo de transporte seja TCP ou UDP.

Em seguida, para calcular a diferença de tempo entre os eventos (duração do SNAT), foi necessário manter uma estrutura de dados interna que armazenasse o instante em que a conexão foi iniciada (evento NEW), pois o *kernel* não armazena tal informação na estrutura `conntrack`. A estrutura de dados utilizada foi uma lista duplamente encadeada, chamada `ct_list`, indexada pelo identificador da conexão (`nfct_get_attr_u32(ct,ATTR_ID)`), endereço IP de origem (`nfct_get_attr_u32(ct,ATTR_ORIG_IPV4_SRC)`) e porta de origem (`nfct_get_attr_u16(ct,ATTR_ORIG_PORT_SRC)`)[6].

Assim, caso trate-se de uma nova conexão, as linhas 22 a 27 serão responsáveis por armazenar o momento em que o evento ocorreu (adicionando um novo nó à lista `ct_list`). Já em se tratando do encerramento de uma conexão, recupera-se a informação sobre o instante de criação da conexão da lista `ct_list` (linhas 30 a 33), calcula-se a diferença de tempo e registra-se os dados da conexão (linha 35) e, finalmente, remove-se tal entrada da lista `ct_list` (linha 36).

Deve-se notar que, como a conexão não será modificada pelo NFCT-SNATLOG, a *callback* sempre retorna a constante `NFCT_CB_CONTINUE`, simplesmente informando ao módulo `conntrack` para chamar a próxima *callback* registrada.

Com relação à forma como os *logs* de SNAT são registrados no sistema, o NFCT-SNATLOG utiliza a biblioteca `syslog.h` que fornece funções para envio de mensagens ao *log* do sistema via protocolo *syslog* (GERHARDS, 2009). É possível também escrever os *logs* na saída padrão, porém o comportamento padrão é utilizar o *syslog*.

[6]Todos esses campos foram usados invés de simplesmente o identificador da conexão pois, segundo Pablo Neira Ayuso (um dos desenvolvedores do *Netfilter* e principal desenvolvedor do *conntrack-tools*), em algumas condições de corrida do *kernel*, o identificador pode não ser suficiente. Fonte: http://marc.info/?l=netfilter&m=128725256809325&w=2, último acesso em 27 de Novembro de 2010

4.2.3 EXEMPLO DE UTILIZAÇÃO

Para mostrar um exemplo de uso do *NFCT-SNATLOG* será retomado o mesmo cenário proposto na Seção 4.1. Antes de iniciar os testes é preciso instalar e executar o *NFCT-SNATLOG*. O *site* do projeto já fornece a documentação necessária para sua instalação e execução (NFCT-SNATLOG, 2010).

A Listagem 4.7 apresenta as mensagens extraídas dos *logs* do sistema na máquina *firewall* para as conexões dos *hosts HostA* e *HostB* ao *host Srv01*, considerando o monitoramento com o NFCT-SNATLOG (foram adotadas as mesmas configurações propostas na Seção 4.1).

Listagem 4.7: *Saída do monitoramento com NFCT-SNATLOG no firewall para as conexões iniciadas com o netcat de HostA e HostB, conforme descrito na Seção 4.1*

```
1 Nov 20 09:49:17 firewall nfct-snatlog: [SNAT_LOG] proto=tcp o-src=172.16.0.1 o-spt=40000
      t-src=192.168.5.253 t-spt=40000 duration=130s
2 Nov 20 09:49:18 firewall nfct-snatlog: [SNAT_LOG] proto=tcp o-src=10.10.10.1 o-spt=40000
      t-src=192.168.5.253 t-spt=1024 duration=130s
```

Como pode-se notar na Listagem 4.7, os *logs* enviados pelo NFCT-SNATLOG contêm as informações necessárias ao tratamento dos incidentes de segurança, e somente elas. Os campos `o-src` e `o-spt` armazenam o par IP/porta de origem originais, enquanto que os campos `t-src` e `t-spt` armazenam o par IP/porta de origem traduzidos; o campo `duration`, por sua vez, armazena o tempo em que a conexão permaneceu armazenada na tabela de conexões do *kernel*.

Dessa forma, pode-se considerar que o NFCT-SNATLOG atende aos requisitos propostos anteriormente e, portanto, pode ser usado em conjunto com o *iptables* para atender à demanda de *logging* das conexões SNAT não atendida pelo *iptables*.

Na seção seguinte, será apresentada uma análise de desempenho e impacto frente a um sistema em produção do NFCT-SNATLOG.

4.3 AVALIAÇÃO DE DESEMPENHO DA PROPOSTA

Esta seção apresenta uma avaliação de desempenho do NFCT-SNATLOG, a fim de verificar o impacto que sua utilização pode trazer a um sistema de *firewall* em produção. Vale ressaltar que já se espera alguma queda no desempenho do sistema, quando da utilização do NFCT-SNATLOG em relação ao ambiente original. Porém, o objetivo fundamental é medir a taxa de degradação que o sistema vai sofrer para aferir a viabilidade de ser usado em ambientes reais. Em outras palavras, o objetivo desses experimentos é apresentar resultados de desempenho de

um *firewall* que não faz *logging* das traduções SNAT, comparando-os com um ambiente em que essa funcionalidade é habilitada através do NFCT-SNATLOG. Por fim, apresentam-se algumas conclusões observadas durante a realização dos experimentos.

4.3.1 CENÁRIOS PARA A AVALIAÇÃO DE DESEMPENHO

Na realização de experimentos para avaliação de desempenho, pode-se utilizar uma das três técnicas seguintes: modelagem analítica, simulação ou medição (JAIN, 1991). A escolha por uma dessas técnicas deve levar em consideração o tempo necessário para realização dos experimentos em cada uma delas, a disponibilidade de ferramentas de suporte, precisão, custo e, principalmente, estágio de desenvolvimento da proposta. Assim, vários fatores devem ser levados em consideração na escolha da técnica de avaliação. Este tópico é discutido com mais detalhes em (JAIN, 1991, Seção 3.1). Neste trabalho optou-se por usar medição pois já dispõe-se do código da proposta implementado e tem-se acesso fácil a um ambiente de experimentação (cenário sem o *logging* das conexões SNAT e cenário com o *logging* de tais conexões através do NFCT-SNATLOG). Dessa forma, é possível a obtenção de resultados mais próximos da realidade.

A Figura 4.4 mostra a topologia de rede utilizada para realização dos experimentos de medição. Neste cenário, todas as máquinas executam o sistema Debian GNU/Linux (*kernel* 2.6.32). As máquinas clientes (rede A) e servidores (rede B) são do tipo *Intel Pentium Dual Core E2160* 1.800GHz, 1GB de memória DDR2 667MHz, HD Sata 160GB 5400rpm e placa de rede FastEthernet. A máquina do *firewall* é um *AMD Athlon 64 2800+ 1.800GHz*, 2 x 256MB de memória DDR400, HD IDE 80GB 5400rpm e 3 x placa de rede FastEthernet. A escolha das máquinas foi baseada na disponibilidade dos equipamentos no momento da execução dos experimentos. As máquinas na rede A e na rede B estão conectadas por dois *switches FastEthernet* (um *switch* em cada rede). Para o monitoramento do *firewall* optou-se pela utilização do software Zabbix[7], pela facilidade de instalação, configuração e coleta dos dados. Para evitar que o monitoramento influencie no tráfego de rede dos experimentos, a coleta era feita por um enlace ponto-a-ponto entre o *firewall* e o servidor zabbix, dedicado a este fim.

A escolha do intervalo de medição é outro fator importante no experimento de avaliação de desempenho, pois intervalos grandes podem causar a perda de eventos importantes, ao passo que intervalos muito curtos provocam ruído e carga excessiva, impactando diretamente nos resultados obtidos. O intervalo de coleta dos dados escolhido foi de 30 segundos, definido após

[7]Zabbix é um software para monitoramento de rede, que possui um agente na estação monitorada e um servidor responsável por coletar os dados dos agentes.

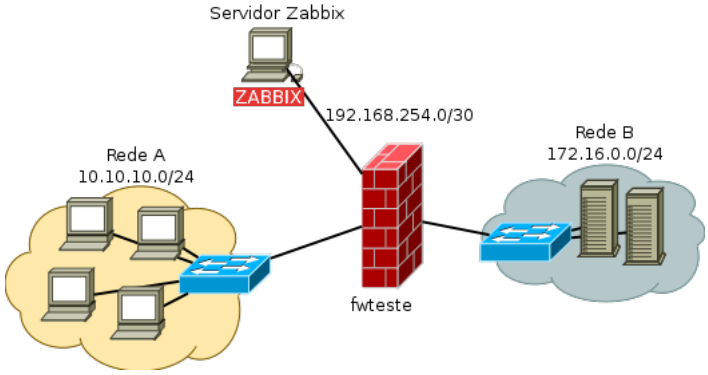

Figura 4.4: Cenário para experimentos de avaliação de desempenho do NFCT-SNATLOG

a realização de algumas medições e avaliação dos gráficos gerados.

Usando a topologia da Figura 4.4 como base dos experimentos, foram avaliadas três configurações diferentes no *firewall* (objeto de observação):

1. O *firewall* realiza traduções SNAT da rede A para a rede B sem o *logging* das traduções;
2. O *firewall* realiza traduções SNAT da rede A para a rede B com o *logging* das traduções através do NFCT-SNATLOG e escreve os *logs* localmente (syslog apenas local);
3. O *firewall* realiza traduções SNAT da rede A para a rede B com o *logging* das traduções através do NFCT-SNATLOG e escreve os *logs* remotamente (syslog apenas remoto, instalado na máquina servidor zabbix);

As métricas de desempenho observadas no experimento foram: utilização de CPU, utilização de memória e operações de entrada e saída no disco. A escolha dessas métricas está relacionada à natureza de funcionamento do NFCT-SNATLOG: utilização de CPU para o processamento da lista de conexões que ele gerencia e cálculo da diferença de tempos; utilização de memória para armazenamento da lista de conexões `ct_list`; operações de entrada e saída no disco para avaliar a escrita dos *logs*.

Ainda considerando a natureza de funcionamento do NFCT-SNATLOG, é importante perceber que a carga de trabalho adicionada ao *firewall* devido à sua utilização independe do tamanho dos pacotes transmitidos, sendo limitada, então, pela quantidade de pacotes por segundo que devem ser tratados pelo NFCT-SNATLOG. Nesse sentido, a carga de trabalho gerada nos experimentos visou aumentar o número de pacotes por segundo que o *firewall* deveria tratar ao longo do tempo, permitindo, inclusive, ter uma noção acerca da escalabilidade da ferramenta. A Tabela 4.3 mostra a carga de trabalho submetida ao *firewall* por período de tempo. A variação da carga por período de tempo (definidos em 10 minutos) é ocasionada exclusivamente

pelo aumento da taxa de envio de pacotes nas máquinas clientes (rede A). Os dados da tabela foram medidos diretamente no *firewall*, através da interface que o conecta à rede B. A escolha do valor inicial, incremento e final foram baseadas em experimentos anteriores realizados: a máquina utilizada era incapaz de tratar valores maiores.

Tabela 4.3: *Carga de trabalho gerada no firewall para experimentos de Avaliação de desempenho do NFCT-SNATLOG*

Intervalo (min)	**Tráfego gerado (Pkts/sec)**
0 - 10	840.47
10 - 20	1247.52
20 - 30	1623.06
30 - 40	1961.51
40 - 50	2313.77
50 - 60	2694.24

Para gerar o tráfego mostrado na Tabela 4.3 foram necessárias as configurações listadas a seguir. Cada um dos servidores (máquinas na rede B) executa o servidor `iperf`, aceitando conexões na porta 5001/TCP (os servidores foram iniciados a partir do comando `iperf -p 5001 -s`), e um servidor UDP (construído com a biblioteca `socket.h`[8]) que apenas aceita conexões na porta 6001, recebe os dados transferidos e aguarda por uma nova conexão. Cada máquina cliente (máquinas na rede A), inicia várias conexões em paralelo com as máquinas servidoras (dois clientes por servidor) para transferência de 2 KBytes via UDP (usando o utilitário *netcat*) e 64 KBytes via TCP (usando o *iperf* como cliente). As transferências são feitas de forma sucessiva e o número de conexões em paralelo inicia em 4 e é incrementado de 2 a cada 10 minutos. O código-fonte dos programas utilizados está disponível no Apêndice A.

A seção seguinte apresenta os dados obtidos a partir das medições realizadas no cenário de experimentação proposto acima.

4.3.2 RESULTADOS OBTIDOS

As figuras 4.5, 4.6 e 4.7 apresentam os gráficos de utilização de CPU, memória e disco, respectivamente, para a primeira configuração do cenário de experimentos proposto na subseção anterior: o *firewall* realiza traduções SNAT para conexões da rede A para a rede B, sem fazer *log* das traduções. Como pode-se observar, a utilização de CPU (Figura 4.5) permaneceu baixa ao longo do tempo (lembrando que o tráfego aumenta com o tempo, conforme Tabela 4.3). A

[8]Optou-se por construir tal serviço ao invés de usar ferramentas prontas, pois as ferramentas testadas todas apresentaram problema com relação ao *timeout* das conexões abertas quando muitas delas eram iniciadas simultaneamente. O código-fonte do servidor UDP construído encontra-se disponível no Apêndice A.

memória livre (Figura 4.6) diminuiu cerca de 3.6MB (0.75% do total), dos quais cerca de 3MB (0.62% do total) foram destinadas aos *buffers* e aproximadamente 600KB (0.12% do total) às outras aplicações do sistema. As operações de entrada/saída no disco (Figura 4.7) também permaneceram em taxas baixas. Note que, apesar do experimento ter durado uma hora, o gráfico abrange um intervalo de uma hora e quatro minutos. Essa margem de tempo foi propositalmente inserida baseando-se nos seguintes critérios: um minuto antes e no fim do experimento para evidenciar a carga gerada no sistema e mais dois minutos no final, pois esse é o tempo limite que o *kernel* aguarda até declarar que uma conexão TCP deve ser destruída[9].

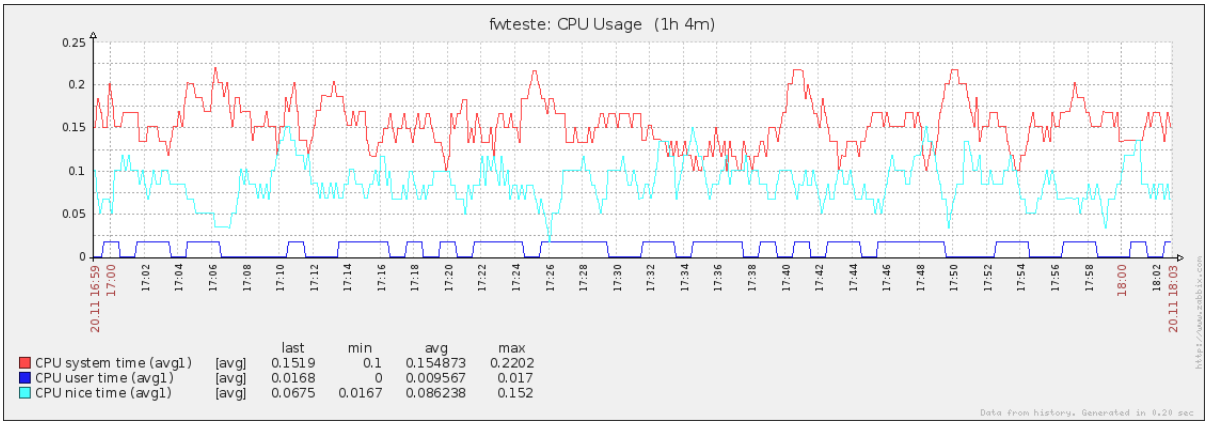

Figura 4.5: Gráficos de utilização de CPU do *firewall* com traduções SNAT sem *logging*.

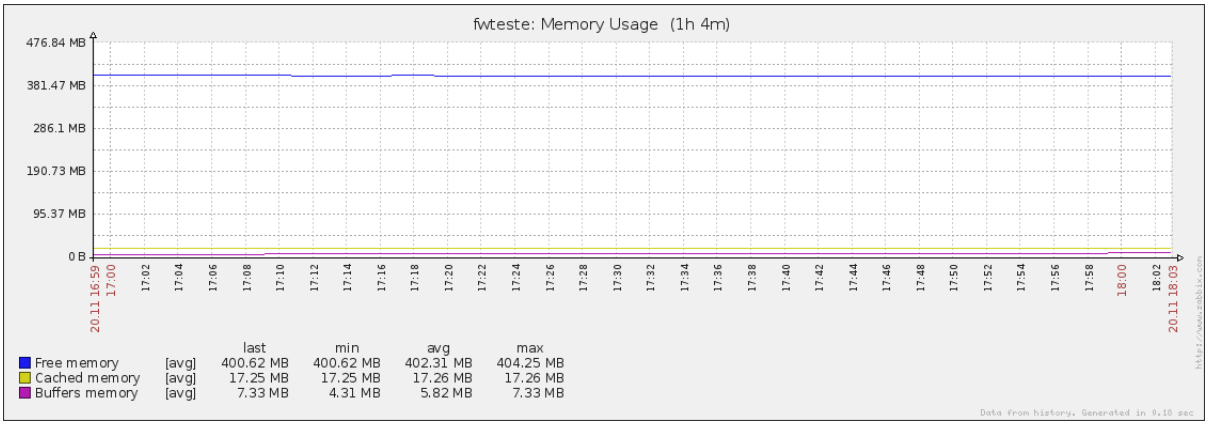

Figura 4.6: Gráficos de utilização de memória do *firewall* com traduções SNAT sem *logging*.

A próxima configuração avaliada foi com o *firewall* realizando traduções SNAT para conexões da rede A para a rede B, com o *logging* das traduções através do NFCT-SNATLOG e escrevendo os *logs* localmente (syslog apenas local). As figuras 4.8, 4.9 e 4.10 apresentam os gráficos de utilização de CPU, memória e disco, respectivamente, para esse cenário. Esses gráficos, se comparados aqueles produzidos para a primeira configuração, mantêm a característica

[9]Para conexões TCP esse tempo limite (*timeout*) é de 120 segundos, já para UDP o *timeout* é 30 segundos. Esses tempos podem ser ajustados através de parâmetros no /proc.

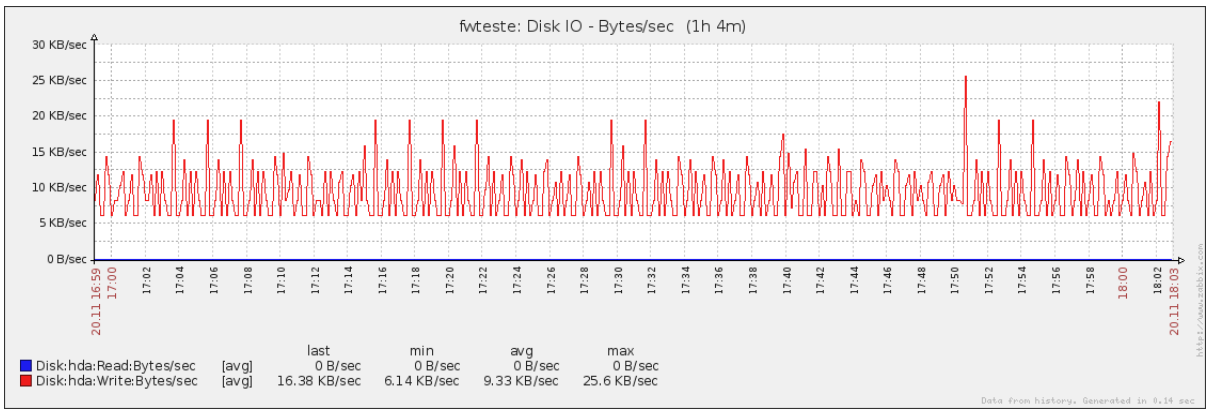

Figura 4.7: Gráficos de utilização de disco do *firewall* com traduções SNAT sem *logging*.

de baixo consumo de CPU porém evidenciam o consumo de memória e disco à medida que o tráfego aumenta. No caso da memória livre (Figura 4.9) houve uma diminuição de aproximadamente 36.5MB (7.65%), dos quais cerca de 27MB (5.66%) passaram a ser usados como *cache* e 5.8MB (1.22%) como *buffers*, contabilizando cerca de 3.7MB (0.78%) de pico no uso de memória. Comparando esse valor diretamente com o obtido no experimento anterior pode-se inferir um pico de consumo de 3MB memória pelo NFCT-SNATLOG e pelo serviço *syslog* (únicas aplicações não presentes na primeira configuração), um valor aceitável considerando-se a carga máxima que foi atribuída ao sistema. Obviamente, essa comparação simplória é imprecisa, porém acredita-se que a falta de rigor nesse caso não tem grande relevância. Acerca das operações de entrada e saída no disco (Figura 4.10), essa segunda configuração também evidencia uma utilização razoável do disco se comparado ao cenário anterior, porém, ainda assim em valores baixos.

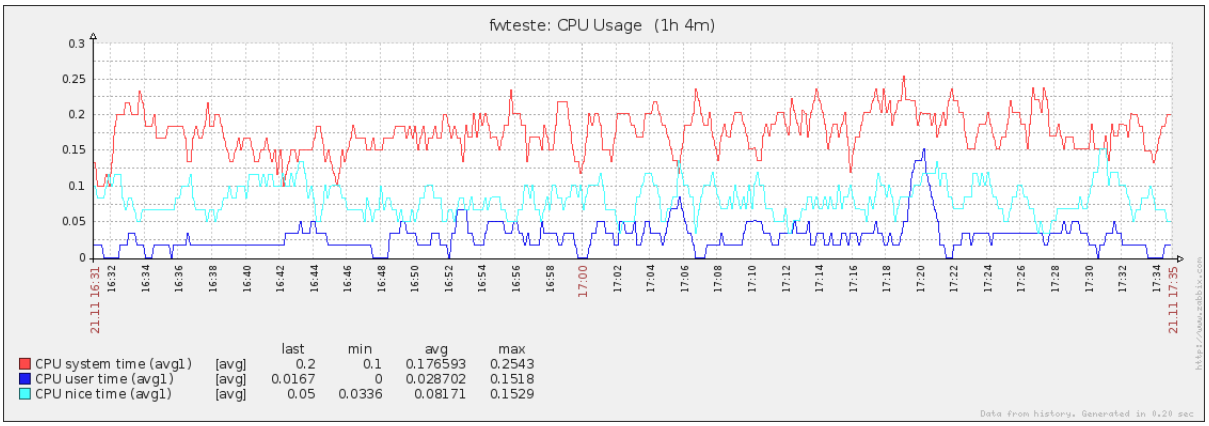

Figura 4.8: Gráficos de utilização de CPU do *firewall* com traduções SNAT e *logging* local.

Na terceira e última configuração do cenário de experimentos tem-se o *firewall* realizando traduções SNAT para conexões da rede A para a rede B, com o *logging* das traduções via NFCT-SNATLOG sendo enviado à um servidor de *logs* remoto (sem escrita no disco local). As figuras

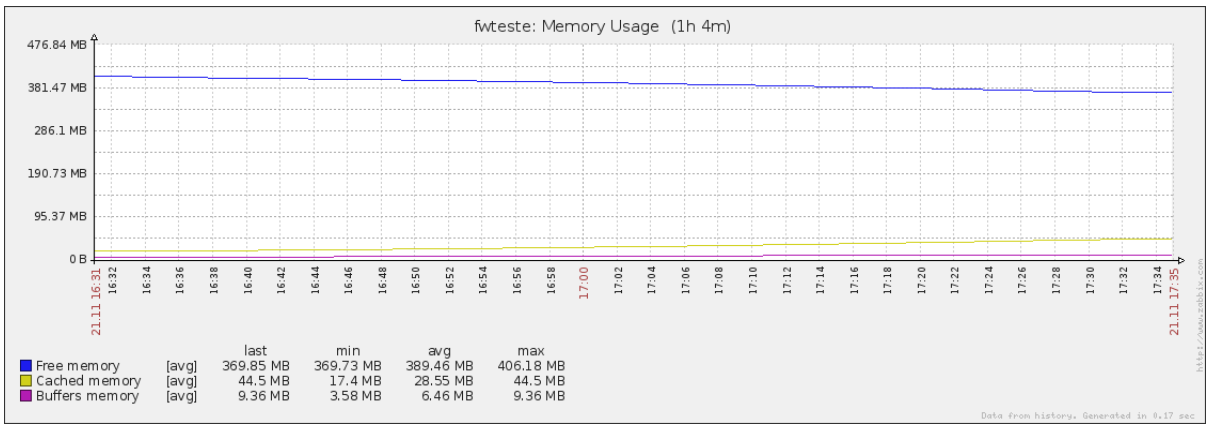

Figura 4.9: Gráficos de utilização de memória do *firewall* com traduções SNAT e *logging* local.

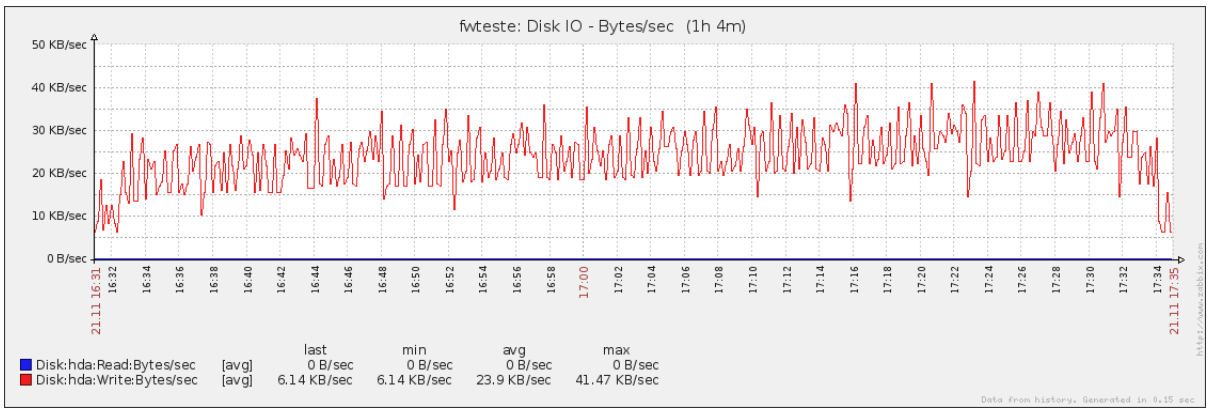

Figura 4.10: Gráficos de utilização de disco do *firewall* com traduções SNAT e *logging* local.

4.11, 4.12 e 4.13 apresentam os gráficos de utilização de CPU, memória e disco, respectivamente, nesse cenário. A utilização de CPU (Figura 4.11) continua praticamente desprezível ao longo do experimento, porém nota-se claramente uma diferença na utilização de memória e escrita em disco comparado ao ambiente com *logging* local (com os valores aproximando-se, inclusive, dos gráficos onde o *logging* das traduções estava desabilitado).

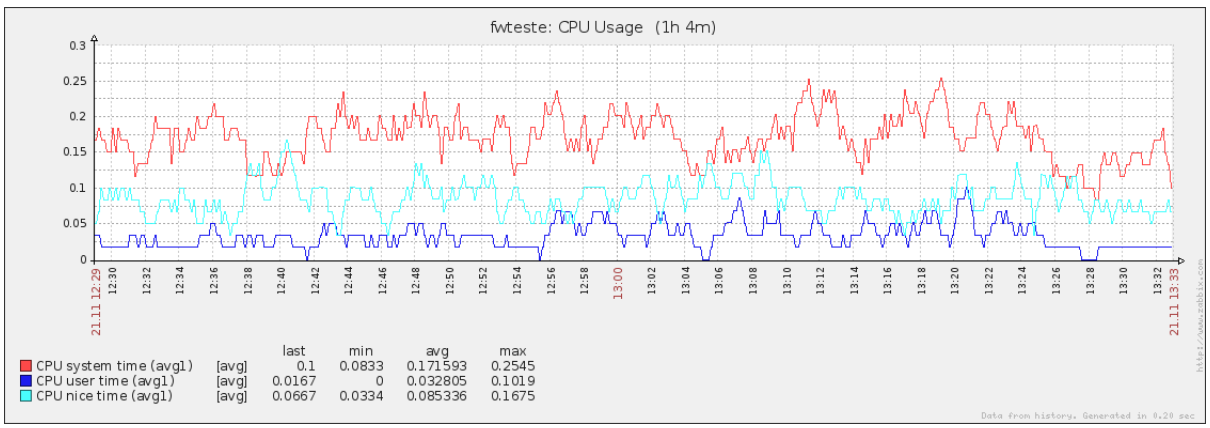

Figura 4.11: Gráficos de utilização de CPU do *firewall* com traduções SNAT e *logging* remoto.

A memória livre (Figura 4.12), diminuiu cerca de 4.4MB (0.92%), dos quais cerca de 3MB (0.63%) foram destinados aos *buffers* e os 1.4MB (0.29%) restantes às outras aplicações do sistema. Novamente comparando esses valores diretamente aos obtidos na primeira configuração, pode-se inferir um pico de consumo de 800KB (0.16%) de memória pelo NFCT-SNATLOG e pelo serviço *syslog* (únicas aplicações que não estavam executando na primeira configuração). Já se comparados diretamente aos obtidos na segunda configuração, pode-se inferir que provavelmente o serviço *syslog* teve uma forte contribuição para o consumo de memória do sistema no segundo cenário. Por conseguinte, observa-se bons níveis de consumo de memória do NFCT-SNATLOG em função da carga a que o sistema foi submetido. O gráfico de operações de entrada e saída no disco (Figura 4.13) apresenta praticamente o mesmo comportamento do observado no cenário sem *logging* das traduções, um resultado esperado.

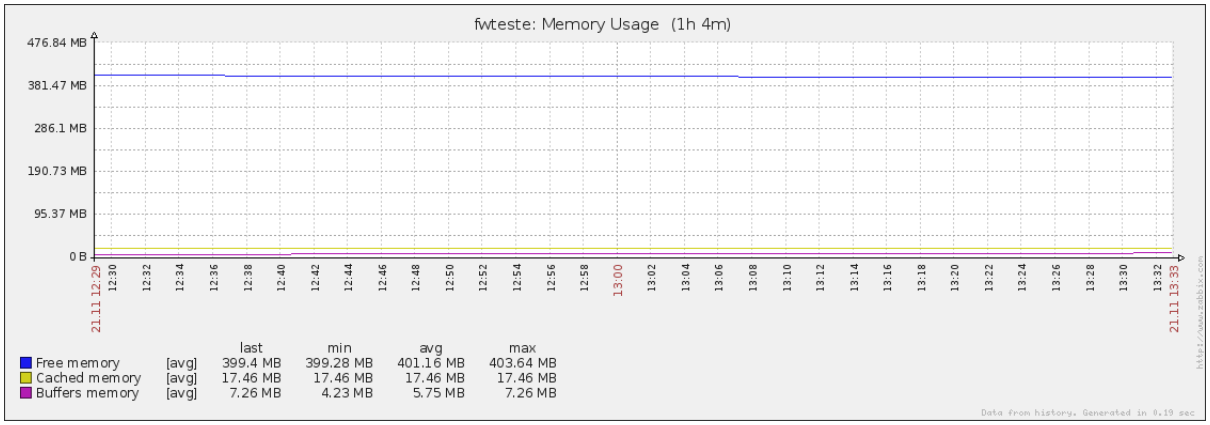

Figura 4.12: Gráficos de utilização de memória do *firewall* com traduções SNAT e *logging* remoto.

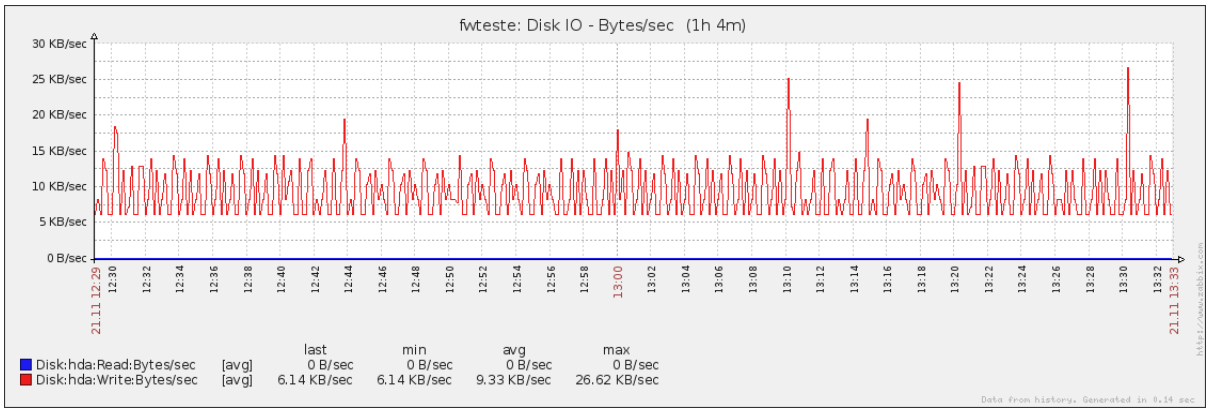

Figura 4.13: Gráficos de utilização de disco do *firewall* com traduções SNAT e *logging* remoto.

A Tabela 4.4 sumariza os resultados obtidos nos experimentos realizados. Nessa tabela as informações sobre utilização de CPU são omitidas, pois praticamente não houve variação nos experimentos realizados; pelo mesmo motivo, considera-se apenas a escrita em disco. As informações sobre memória são obtidas da comparação com o cenário sem *logging*.

Tabela 4.4: *Resumo dos resultados obtidos no experimento de avaliação de desempenho do NFCT-SNATLOG*

Cenário	*Memória*			*Escrita em disco*		
	Livre	*Buffers*	*Cache*	*Min*	*Média*	*Max*
Logging local	-7.65%	1.22%	5.66%	6.14 KB/s	23.9 KB/s	41.47 KB/s
Logging remoto	-0.92%	0.63%	–	6.14 KB/s	9.33 KB/s	26.62 KB/s

4.3.3 ANÁLISE DOS RESULTADOS

A partir da análise dos resultados para os experimentos apresentados nesta seção, alguns pontos importantes merecem destaque:

- O registro das traduções SNAT do *iptables* através do NFCT-SNATLOG mostra-se viável levando-se em consideração a possibilidade de auditoria do sistema quando de sua utilização. No que tange à carga inserida no sistema, acredita-se que ela não será o fator limitante da capacidade de processamento do *firewall*;

- Embora este trabalho não tenha avaliado diretamente as questões relativas ao espaço para armazenamento dos *logs* das traduções, os resultados dos experimentos com *logging* local e *logging* remoto apontam para maiores vantagens na adoção do segundo cenário, onde um terceiro servidor será responsável pelo armazenamento dos *logs* (permitindo, assim, planejar a utilização de espaço em disco nesse servidor);

- Utilizar um número grande de máquinas por endereço IP de NAT não é uma boa prática para o tratamento de incidentes de segurança (ou qualquer outro processo investigativo que necessite realizar buscas nos *logs* das traduções). Recomenda-se, então, aprimorar a distribuição de endereços internos da instituição para os endereços globais que serão usados no NAT. Por exemplo, para instituições que possuem um esquema de segmentação da rede em VLANs bem definido, uma possibilidade é atribuir um endereço IP de NAT por VLAN. No entanto, cada cenário necessita de avaliação particular.

5 CONCLUSÃO

O crescimento atual da Internet tem influenciado diretamente no aumento do número de incidentes de segurança. Como consequência direta desses incidentes pode-se listar as perdas financeiras decorrentes dos gastos com análise e remoção de *malwares*, perda de produtividade, queda de receita devido à indisponibilidade de sistemas, além da utilização indevida de recursos da instituição para atividade maliciosa. Uma vez que nem todos os incidentes podem ser prevenidos (SCARFONE; GRANCE; MASONE, 2008), faz-se necessário o desenvolvimento da capacidade de tratar e responder aos incidentes de segurança reportados à uma instituição.

Entretanto, a identificação e tratamento de um incidente de segurança não é uma tarefa simples. Pelo contrário, exige que uma série de medidas seja tomada para identificar, isolar e tratar a origem real do incidente. Em particular, nas instituições de ensino e pesquisa, o tratamento e resposta aos incidentes reportados são ainda mais complicados. Essa dificuldade é inerente à utilização de técnicas como NAT e DHCP, e ao grande volume de dados a serem manipulados no tratamento de uma notificação. Além disso, deve-se perceber que a maior parte dos incidentes reportados a uma instituição são causados por ferramentas que comprometem a máquina de forma automatizada; a própria notificação de incidente de segurança geralmente é automatizada. Dessa maneira, tratar manualmente um incidente que foi gerado e reportado de forma automatizada é inviável. Faz-se necessária a utilização de ferramentas que automatizem o processo de tratamento de incidentes, ou, pelo menos, parte dele. Nesse sentido, duas etapas com grande possibilidade de automatização são a detecção e isolamento da origem do incidente.

Este trabalho apresenta uma ferramenta para automatizar o processo de tratamento de incidentes de segurança, o TRAIRA (um acrônimo para *Tratamento de Incidentes de Rede Automatizado*), que atua nas etapas de detecção e isolamento da origem do incidente, deixando a cargo da equipe de segurança apenas as próximas medidas a serem tomadas. Com a utilização do TRAIRA, reserva-se o time de segurança da instituição (cujo custo de contratação é comumente alto) para o tratamento de incidentes mais importantes ou complexos e para as outras atividades de segurança, como atividades preventivas, por exemplo.

Desenvolvido como extensão do RT, o TRAIRA permite integração com outras ferramentas de resposta a incidentes, especialmente o RTIR (extensão do RT para Resposta a Incidentes), permitindo uma gerência completa sobre um incidente de segurança. Atualmente, o TRAIRA encontra-se em fase de homologação na Universidade Federal da Bahia (UFBA), já permitindo o tratamento automatizado de todos os incidentes do tipo *vírus/worm*.

Para seu correto funcionamento, o TRAIRA depende que alguns recursos básicos de segurança estejam habilitados na instituição, dentre os quais destaca-se o registro (*logging*) das traduções SNAT realizadas no acesso à Internet. Alguns dispositivos de NAT já possuem essa funcionalidade nativamente, como é o caso do *firewall* ASA da CISCO; outros não possuem. O *IPTables/Netfilter*, o *firewall* nativo do Linux, por exemplo, não é capaz de registrar as traduções NAT. Por ser uma ferramenta bastante utilizada, principalmente nas instituições de ensino e pesquisa, este trabalho propõe, implementa e avalia uma aplicação que adiciona tal funcionalidade ao *iptables*, o NFCT-SNATLOG.

O NFCT-SNATLOG é uma aplicação em espaço de usuário que monitora a tabela de rastreamento de conexões do *kernel* do Linux e registra em *log* (via *syslog*) as traduções do tipo SNAT (NAT de origem) ocorridas no sistema, incluindo apenas informações relevantes ao tratamento de incidentes de segurança (IP e porta de origem originais, IP e porta de origem traduzidos, protocolo e duração do NAT). O fato de registrar os *logs* via *syslog* é importante pois permite a implementação de um servidor de *logs* remoto, diminuindo o impacto que essa funcionalidade pode adicionar ao *firewall*. Apresenta-se também uma avaliação de desempenho sobre o NFCT-SNATLOG, mostrando que sua utilização afeta muito pouco o desempenho do *firewall* e, portanto, sua implantação é bastante viável considerando-se a capacidade de monitoramento das conexões SNAT que ela acrescenta ao *iptables*.

Por fim, acredita-se que os resultados obtidos com esse trabalho trazem uma contribuição importante no processo de tratamento de incidentes de segurança, principalmente no que tange às instituições de ensino e pesquisa e como elas têm impactado o cenário de (in)segurança da Internet.

Não obstante, alguns aspectos discutidos neste trabalho são passíveis de exploração futura, os quais são listados a seguir:

- Estudar e comparar técnicas de armazenamento dos *logs* em bancos de dados. A utilização de um banco de dados para armazenamento dos *logs* em detrimento de arquivos texto pode trazer ganhos significativos no que tange ao desempenho das consultas. Porém, deve-se levar em consideração a questão de compressão dos dados: usando-se arquivos

de texto do sistema de arquivos é muito fácil comprimi-los e gerenciar o espaço em disco ocupado pelos *logs*. Alguns bancos de dados também oferecem mecanismos para compressão dos dados. Assim, vale a pena iniciar um trabalho investigativo e comparativo entre as duas abordagens de armazenamento, a fim de recomendar a utilização de uma ou outra abordagem;

- Ainda sobre o armazenamento dos dados, uma abordagem que pode ser explorada é sobre os campos fundamentais ao tratamento dos incidentes de segurança (ou processos de auditoria, no geral): talvez seja possível, por exemplo, mapear o par IP/porta em um *hash* e, dessa forma, economizar espaço de armazenamento. No entanto, deve-se considerar a expressividade do mapeamento realizado, pois no futuro pode ser necessário obter exatamente aquele par IP/porta;

- Verificar de forma mais detalhada a integração do TRAIRA com o RTIR a fim de prover um ambiente mais completo de gerência de incidentes de segurança;

- Avaliar os impactos das características dessa nova Internet do futuro (já presente em alguns aspectos), como mobilidade, utilização de IPv6 e "fim teórico" da necessidade de NAT, dentre outros aspectos, no processo de tratamento de incidentes de segurança. No caso da mobilidade, por exemplo, detectar e isolar um dispositivo móvel na rede é muito mais complicado que uma estação de trabalho fixa de uma unidade. Com relação à utilização de IPv6, deve-se observar se o NAT não será mesmo utilizado como atualmente ou ainda se o mapeamento IPv6 global para máquina interna será na proporção um para um, como espera-se.

APÊNDICE A – FERRAMENTAS PARA AVALIAÇÃO DE DESEMPENHO DO NFCT-SNATLOG

Listagem A.1: *Servidor UDP para transferência de dados usado nos experimentos do NFCT-SNATLOG*

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <errno.h>
5  #include <string.h>
6  #include <sys/types.h>
7  #include <sys/socket.h>
8  #include <netinet/in.h>
9  #include <arpa/inet.h>
10 #define MAXBUFLEN 1024 // 1MB
11 int main(int argc, char **argv)
12 {
13     int sockfd;
14     struct sockaddr_in my_addr;
15     struct sockaddr_in their_addr;
16     int addr_len, numbytes;
17     char buf[MAXBUFLEN];
18     if (argc < 1) {
19         printf("Usage: %s <Port to listen>\n", argv[0]);
20         exit(1);
21     }
22     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
23         perror("socket");
24         exit(1);
25     }
26     my_addr.sin_family = AF_INET;
27     my_addr.sin_port = htons(atoi(argv[1]));
28     my_addr.sin_addr.s_addr = INADDR_ANY;
29     memset(&(my_addr.sin_zero), '\0', 8);
30     if (bind(sockfd, (struct sockaddr *)&my_addr,
31                                         sizeof(struct sockaddr)) == -1) {
32         perror("bind");
33         exit(1);
34     }
```

```
35      while(1) { // main accept() loop
36          addr_len = sizeof(struct sockaddr_in);
37          if ((numbytes=recvfrom(sockfd,buf, MAXBUFLEN-1, 0, (struct sockaddr *)&their_addr,
                &addr_len)) == -1) {
38              perror("recvfrom");
39              continue;
40          }
41          printf("got packet from
                %s:%d\n",inet_ntoa(their_addr.sin_addr),ntohs(their_addr.sin_port));
42      }
43    close(sockfd);
44    return 0;
45  }
```

Listagem A.2: *Aplicação para execução de múltiplicas conexões em paralelo, usando o netcat e iperf*

```
1   #include <pthread.h>
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <unistd.h>
5   #define MAX_THREADS  100
6
7   char comando_tcp[1024];
8   char comando_udp[1024];
9
10  void *run_thread(void *threadid)
11  {
12      if (system(comando_tcp)) {
13          perror("failed to run iperf TCP");
14      }
15      if (system(comando_udp)) {
16          perror("failed to run iperf UDP");
17      }
18      pthread_exit(NULL);
19  }
20
21  int main (int argc, char *argv[])
22  {
23      pthread_t threads[MAX_THREADS];
24      int rc;
25      long t;
26      if (argc < 2) {
27          printf("Usage: %s <Remote IP> <Number of Threads>\n", argv[0]);
28          exit(1);
29      }
30      sprintf(comando_tcp, "iperf -c %s -p 5001 -n 64K", argv[1]);
31      // for the UDP connection we transfer our own data! :)
32      sprintf(comando_udp, "nc -u %s 6001 -q 0 < %s", argv[1], argv[0]);
33      for(t=0; t<atoi(argv[2]); t++){
34          printf("In main: creating thread %ld\n", t);
35          rc = pthread_create(&threads[t], NULL, run_thread, (void *)t);
36          if (rc){
37              printf("ERROR; return code from pthread_create() is %d\n", rc);
```

```
38              exit(-1);
39          }
40      }
41      pthread_exit(NULL);
42  }
```

REFERÊNCIAS BIBLIOGRÁFICAS

ARVIDSSON, J. et al. *TERENA'S Incident Object Description and Exchange Format Requirements*. IETF, fev. 2001. RFC 3067 (Informational). (Request for Comments, 3067). Disponível em: <http://www.ietf.org/rfc/rfc3067.txt>.

AYUSO, P. Netfilter's connection tracking system. *LOGIN: The USENIX magazine*, v. 31, p. 34–39, 2006.

BESTPRACTICAL. *RT: Request Tracker*. 2010. Último acesso em 23 de Novembro de 2010. Disponível em: <http://www.bestpractical.com/rt/>.

BESTPRACTICAL. *RTIR: RT for Incident Response*. 2010. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://www.bestpractical.com/rtir/>.

BEZERRA, J. A. *Relatório de atividades - Operação e Segurança - parte II*. [S.l.], 2009.

CAIS. *CAIS - Centro de Atendimento a Incidentes de Segurança*. 2010. Último acesso em 14 de Novembro de 2010. Disponível em: <http://www.rnp.br/cais/>.

CERON, J. et al. O processo de tratamento de incidentes de segurança. *IV Workshop de TI das IFES*, 2009.

CERT.BAHIA. *CERT.Bahia - Grupo e Resposta a Incidentes de Segurança na Bahia/Brasil*. 2010. Último acesso em 14 de Novembro de 2010. Disponível em: <http://www.certbahia.pop-ba.rnp.br>.

CERT.BAHIA. *Estatísticas do CERT.Bahia*. 2010. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://www.certbahia.pop-ba.rnp.br/Estatisticas>.

CERT.BR. *Cartilha de Segurança para Internet. Parte VII: Incidentes de Segurança e Uso Abusivo da Rede*. 2006. Último acesso em 14 de Novembro de 2010. Disponível em: <http://cartilha.cert.br/download/cartilha-07-incidentes.pdf>.

CERT.BR. *Cartilha de Segurança para Internet. Parte VIII: Códigos Maliciosos (Malware)*. 2006. Último acesso em 14 de Novembro de 2010. Disponível em: <http://cartilha.cert.br/download/cartilha-08-malware.pdf>.

CERT.BR. *Resultados Preliminares do Projeto SpamPots*. 2007. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://www.cert.br/docs/whitepapers/spampots/>.

CERT.BR. *CERT.br - Grupo Resposta a Incidentes de Segurança no Brasil*. 2010. Último acesso em 14 de Novembro de 2010. Disponível em: <http://www.cert.br>.

CERT/CC. *Computer Segurity Incident Response Team FAQ*. 2010. Último acesso em 14 de Novembro de 2010. Disponível em: <http://www.cert.org/csirts/csirt_faq.html>.

CISCO, A. 5500 Series Adaptive Security Appliances. *Cisco Systems, Inc*, 2005.

CWG. *Conficker Working Group*. 2010. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://www.confickerworkinggroup.org/wiki/>.

DEBAR, H.; CURRY, D.; FEINSTEIN, B. *The Intrusion Detection Message Exchange Format (IDMEF)*. IETF, mar. 2007. RFC 4765 (Experimental). (Request for Comments, 4765). Disponível em: <http://www.ietf.org/rfc/rfc4765.txt>.

DROMS, R. *Dynamic Host Configuration Protocol*. IETF, mar. 1997. RFC 2131 (Draft Standard). (Request for Comments, 2131). Updated by RFCs 3396, 4361, 5494. Disponível em: <http://www.ietf.org/rfc/rfc2131.txt>.

EGEVANG, K.; FRANCIS, P. *The IP Network Address Translator (NAT)*. IETF, maio 1994. RFC 1631 (Informational). (Request for Comments, 1631). Obsoleted by RFC 3022. Disponível em: <http://www.ietf.org/rfc/rfc1631.txt>.

FARNHAM, G. Cisco Security Agent and Incident Handling. *SANS Institute InfoSec Reading Room*, 2009.

FEINSTEIN, B.; MATTHEWS, G. *The Intrusion Detection Exchange Protocol (IDXP)*. IETF, mar. 2007. RFC 4767 (Experimental). (Request for Comments, 4767). Disponível em: <http://www.ietf.org/rfc/rfc4767.txt>.

GERHARDS, R. *The Syslog Protocol*. IETF, mar. 2009. RFC 5424 (Proposed Standard). (Request for Comments, 5424). Disponível em: <http://www.ietf.org/rfc/rfc5424.txt>.

HONEYNET.BR. *Brazilian Honeypots Alliance*. 2010. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://www.honeypots-alliance.org.br/>.

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: Wiley New York, 1991.

KAISER, J. et al. Automated resolving of security incidents as a key mechanism to fight massive infections of malicious software. In: CITESEER. *GI SIDAR International Conference on IT-Incident Management and IT-Forensics (IMF 2006), volume LNI P-97*. [S.l.], 2006. p. 92–103.

KLINGMÜLLER, T. SIRIOS: A Framework for CERTs. *FIRST Conference on Computer Security Incident Handling*, 2005.

LUNDELL, M. Incident Handling as a Service. *SANS Institute InfoSec Reading Room*, 2009.

MOTA FILHO, J. *Firewall com iptables*. 2003. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://www.eriberto.pro.br/iptables/>.

NETFILTER. *conntrack-tools: Netfilter's connection tracking userspace tools*. 2010. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://conntrack-tools.netfilter.org/>.

NETFILTER. *Netfilter: firewalling, NAT and packet mangling for linux*. 2010. Último acesso em 27 de Novembro de 2010. Disponível em: <http://www.netfilter.org/>.

NETFILTER. *The netfilter.org "libnetfilter_conntrack" project*. 2010. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://www.netfilter.org/projects/libnetfilter_conntrack/index.html>.

NFCT-SNATLOG. *NFCT-SNATLOG - Netfilter Conntrack SNAT Logging Tool*. 2010. Último acesso em 27 de Novembro de 2010. Disponível em: <http://github.com/italovalcy/nfct-snatlog>.

OTRS. *Open Source Trouble Ticket System*. 2010. Último acesso em 23 de Novembro de 2010. Disponível em: <http://www.otrs.org/>.

PERL.ORG. *The Perl Programming Language*. 2010. Último acesso em 14 de Novembro de 2010. Disponível em: <http://www.perl.org>.

PLUMMER, D. *Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. IETF, nov. 1982. RFC 826 (Standard). (Request for Comments, 826). Updated by RFCs 5227, 5494. Disponível em: <http://www.ietf.org/rfc/rfc826.txt>.

REKHTER, Y. et al. *Address Allocation for Private Internets*. IETF, fev. 1996. RFC 1918 (Best Current Practice). (Request for Comments, 1918). Disponível em: <http://www.ietf.org/rfc/rfc1918.txt>.

RNP. *A rede Ipê*. 2007. Último acesso em 22 de Novembro de 2010. Disponível em: <http://www.rnp.br/ipe/>.

RNP. *Rede Ipê: Política de Uso*. 2007. Último acesso em 29 de Setembro de 2010. Disponível em: <http://www.rnp.br/conexao/>.

SCARFONE, K.; GRANCE, T.; MASONE, K. Computer Security Incident Handling Guide. *NIST Special Publication*, v. 800–61, 2008.

SOUPPAYA, M.; KENT, K. Guide to Computer Security Log Management. *NIST Special Publication*, p. 800–92, 2006.

TRAIRA. *TRAIRA - Tratamento de Incidentes de Rede Automatizado*. 2010. Último acesso em 03 de Dezembro de 2010. Disponível em: <http://www.pop-ba.rnp.br/~arquivos/RT-Extension-Traira.tgz>.

WERLINGER, R.; BOTTA, D.; BEZNOSOV, K. Detecting, analyzing and responding to security incidents: a qualitative analysis. In: ACM. *Proceedings of the 3rd symposium on Usable privacy and security*. [S.l.], 2007. p. 149–150.