Sistema de Detecção de Backdoors e Canais Dissimulados

Carlos Henrique P C Chaves[1] **, Antonio Montes**[2]

[1]Lab. Associado de Computação e Matemática Aplicada
Instituto Nacional de Pesquisas Espaciais
12227-010 São José dos Campos - SP

[2]Centro de Pesquisas Renato Archer
Rodovia Dom Pedro I, km 143,6
13082-120 Campinas - SP

cae@lac.inpe.br, antonio.montes@cenpra.gov.br

Abstract. *This paper presents the concepts of backdoors and covert channels, some examples of public available tools and a detection methodology. The intrusion detection system presented is an hybrid system, which uses self-organizing maps as anomaly detection technique and pattern recognition as abuse detection technique. The system aims to contribute with the development of tools to detect backdoors and covert channels, in order to improve network security.*

Resumo. *Este artigo apresenta os conceitos de backdoors e canais dissimulados, alguns exemplos de ferramentas disponíveis publicamente e uma metodologia para detectá-los. O sistema de detecção de intrusão apresentado é um sistema híbrido, que utiliza mapas auto-organizáveis como técnica de detecção por anomalia e reconhecimento de padrões como técnica de detecção por abuso. O sistema tem como objetivo contribuir com toda comunidade de administração e segurança de redes, utilizando-se das vantagens das estratégias de detecção.*

1 Introdução

No mundo atual, a maioria das empresas e corporações possui uma rede local com acesso à Internet e a utiliza para expor seu produtos (através de páginas em servidores HTTP), para entrar em contato com os clientes (através de troca de correios eletrônicos) ou mesmo para pedir aos fornecedores renovação de estoque (através da comunicação entre seus computadores). Essa situação gera uma preocupação, por parte da empresa, com a segurança de seus dados e informações sigilosas. Para tentar resolver este problema, são adotadas tecnologias de segurança da informação, como *firewalls*, antivírus e sistemas de detecção de intrusão.

Existe uma tendência de crescente preocupação com a segurança de organizações em relação a ataques vindos da Internet que possam tornar indisponíveis os servidores da empresa ou que possibilitem o roubo de dados de caráter confidencial. Porém, a ameaça interna é real e iminente. Funcionários insatisfeitos ou que, por ventura, tenham sido demitidos podem representar um grande perigo à empresa. Não seria uma tarefa muito difícil para um destes funcionários instalar um *backdoor* ou um *rootkit* na máquina que ele utiliza (ou utilizava), que permita acessá-la com o intuito de prejudicar a organização, ou mesmo obter dados que possam render algum benefício com organizações concorrentes. *Backdoors* são programas executados em computadores com o objetivo de prover acesso aos mesmos sem que haja a necessidade de exploração de alguma vulnerabilidade. Eles

podem ser detectados por sistemas antivírus, anti-*spyware* e de detecção de intrusão, uma vez que o tráfego destinado a eles normalmente contém padrões conhecidos.

Um outro aspecto importante consiste no fato de que os administradores de *firewall* normalmente definem regras para permitir o acesso, a partir da Internet, a apenas certas aplicações servidoras, como HTTP(S), SMTP, DNS, entre outros. As barreiras criadas por essas regras podem ser violadas com o auxílio de ferramentas que utilizam canais dissimulados (*covert channels*). Essas ferramentas tem como objetivo tentar transmitir informações de maneira que mecanismos de controle de acesso a redes e detecção de intrusão não consigam identificá-las.

Este artigo tem como objetivo apresentar um sistema de detecção de *backdoors* e canais dissimulados, que implementa técnicas de detecção de intrusão por anomalia e abuso para classificar um tráfego como pertencente a um certo *backdoor* ou ferramenta que implementa canais dissimulados. Além disso, o sistema visa superar os possíveis falso-negativos de um sistema de detecção por abuso, causados por eventuais alterações nos padrões das ferramentas de ataque, que são identificadas pelas regras de detecção. As ferramentas podem então ser detectadas por anomalia, uma vez que elas serão identificadas pelos seus comportamentos, e não por assinaturas em seus conteúdos. Este sistema também poderá ser utilizado para descobrir novos ataques, previamente não detectados por um sistema de detecção baseado somente em assinaturas.

O restante deste artigo está organizado da seguinte forma: a seção 2 conceitua *backdoors* e apresenta dois exemplos; a seção 3 define um canal dissimulado e discute as características de quatro ferramentas que implementam tais canais; a seção 4 conceitua sistemas de detecção de intrusão e apresenta técnicas de detecção; a seção 5 apresenta as metodologias e ferramentas utilizadas pelo sistema desenvolvido; a seção 6 descreve o sistema de detecção de *backdoors* e canais dissimulados; a seção 7 mostra os resultados obtidos e, finalmente, a seção 8 apresenta as conclusões sobre o assunto abordado e os próximos trabalhos a serem desenvolvidos.

2 Backdoors

Os atacantes, após terem comprometido um sistema, normalmente utilizam um mecanismo para conseguir acesso a esse sem que uma vulnerabilidade em um software tenha que ser explorada. Este mecanismo é chamado de *backdoor*. Esse é freqüentemente instalado com a intenção de facilitar o retorno do atacante, bem como dificultar a sua detecção.

Um *backdoor* é normalmente instalado através da adição de um novo serviço ao sistema ou em substituição de um serviço legítimo por outro que atenda às necessidades do atacante.

Um *backdoor* simples pode ser implementado com o auxílio do programa *netcat*[1]. Para criar o *backdoor*, o atacante deve executar o *netcat* com as seguintes opções: *nc -l -p <porta> -e /bin/bash*. A opção "-l" significa modo *listen* (servidor), esperando conexões. A opção "-p" informa em qual porta o servidor estará escutando e "-e" informa o programa a ser executado após uma conexão ter sido estabelecida. Agora, o atacante deve apenas executar, a partir de outro computador, *nc <endereço IP> <porta>* para obter um *shell*, com o privilégio do usuário que iniciou o *netcat* no computador alvo.

Outro *backdoor* interessante é o *rwwwshell* [Hauser 2002]. Este é composto por um *script* desenvolvido na linguagem *Perl* que simula um servidor *Web* e um cliente que

[1] http://netcat.sourceforge.net/

deve se conectar ao servidor e permitir que comandos sejam executados pelo servidor no *host* do cliente. Essa técnica é chamada de *shell* reverso e será explicada com mais detalhes na seção 3.

O *rwwwshell* normalmente opera através de qualquer filtro de pacotes que permita que os usuários acessem servidores *Web* na Internet de forma irrestrita.

Um programa, que deve ser executado em um computador na rede interna (escravo), inicia um *shell* local, se conecta ao servidor *Web* controlado pelo atacante (mestre) e envia um sinal informando que está pronto para receber os comandos. Para o sistema de controle de acesso da rede, esse tráfego tem as mesmas características do tráfego de um usuário navegando na Internet. A resposta do servidor contém o comando a ser executado no *shell* aberto pelo escravo. Os comandos e respostas são codificados e passados como uma *cgi-string*, dificultando a sua identificação. Um exemplo de comunicação entre o escravo e o mestre é apresentado a seguir:

Escravo: GET /cgi-bin/order?M5mAejTgZdgYOdgIO0BqFfVYTgjFLdgxEdb1He7krj HTTP/1.0

Mestre: g5mAlfbknz

A requisição *GET* do computador na rede interna é um *prompt* de comando do *shell* e a resposta é um comando "ls" codificado vindo do servidor externo.

O escravo pode ser programado para tentar, em uma hora específica, uma conexão com o mestre. Caso um administrador veja uma conexão ao servidor *Web* controlado pelo atacante e tente uma conexão, ele terá a impressão de que o servidor está parado, pois deve existir uma senha codificada na requisição *GET* [Hauser 1999].

3 Canais Dissimulados

Existem várias definições para um canal dissimulado (*covert channel*). Lampson conceitua canais dissimulados como sendo canais usados para troca de informações que não são normalmente utilizados para comunicação e que não são protegidos por mecanismos de controle de acessos, ou seja, uma comunicação ilícita através de um canal de troca de informações legítimo. Ele também chama os canais dissimulados de "*leakage paths*" [Lampson 1973]. Segundo os critérios de avaliação de um sistema computacional confiável, do Departamento de Defesa dos Estados Unidos, um canal dissimulado é qualquer canal de comunicação que pode ser explorado por um processo para transmitir informações, de maneira que a política de segurança de um sistema seja violada [U.S. Department of Defense 1985]. Mudge o define como sendo o processo no qual um canal de comunicação é encapsulado por outro [Mudge 2003].

Os protocolos da pilha TCP/IP podem ser utilizados para o estabelecimento de canais dissimulados não apenas com o objetivo de burlar as barreiras impostas por filtros na rede, mas também com a intenção de esconder o conteúdo de uma sessão de uma observação casual, ou até mesmo de um sistema de detecção de intrusão [Mudge 2003]. Assim esse conhecimento possibilitou a criação de várias ferramentas que implementam canais dissimulados sobre alguns protocolos, como o ICMP [daemon9 and alhambra 1996, daemon9 1997], o HTTP [Mudge 2003, Farrow 2004] e o DNS [Farrow 2004].

Uma vez que o protocolo HTTP é muito utilizado nas redes das corporações e normalmente não é filtrado por filtro de pacotes, este artigo irá considerar os canais dissimulados sobre o protocolo HTTP, bem como as ferramentas que os implementam. É importante salientar que o simples fato de utilizar as portas associadas ao HTTP para

trafegar outras informações não caracteriza o uso de um canal dissimulado. Para isso, é necessário que outras informações estejam encapsuladas no protocolo HTTP. Seguem as descrições de funciomento de quatro ferramentas que implementam canais dissimulados.

O *Covert Channel Tunneling Tool - cctt* [Castro 2003a] é uma ferramenta que implementa canais dissimulados sobre o protocolo HTTP. Além disso, o *cctt* permite que um canal de comunicação seja estabelecido entre um cliente e um servidor, usando qualquer porta dos protocolos UDP e TCP (porém nesse caso a ferramenta funciona apenas como um *backdoor*). No primeiro caso, um canal é estabelecido entre o cliente e o servidor simulando o protocolo HTTP através dos pedidos *POST* e respostas. A ferramenta também pode utilizar *proxies* para conexão entre um cliente *cctt* e o servidor. Uma funcionalidade interessante é a de *shell* reverso, na qual o cliente inicia um *shell* local, se conecta a um servidor e permite que este execute comandos no cliente. Nesse caso, o que parece ser um usuário navegando na *Web*, visitando um servidor específico, é na verdade o cliente se tornando servidor do atacante externo e permitindo que este tenha controle de um *host* na rede interna. Segundo Mudge [Mudge 2003], este tipo de canal dissimulado permite a camuflagem da conexão e permite que os filtros externos sejam driblados por atacantes. Para dificultar a detecção da ferramenta por sistemas que procuram por assinaturas no conteúdo dos pacotes, o *cctt* possui a característica de codificar a comunicação de forma a dificultar este tipo de detecção.

O *Wsh* [Dyatlov and Castro 2003] é um *shell* UNIX/WINDOWS remoto, que utiliza os protocolos HTTP/HTTPS para transmitir os comandos e obter as respostas. O pacote contém um *script* desenvolvido em *Perl* para o cliente e um para o servidor. Além disso, o servidor também foi desenvolvido na linguagem C, podendo ser utilizado ao invés do *script* desenvolvido em *Perl*.

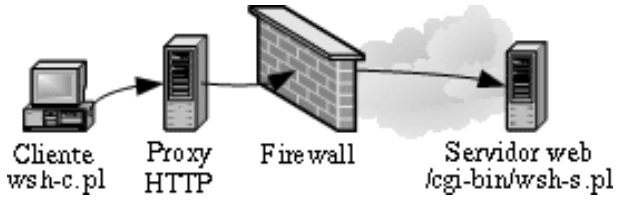

Figura 1: *Wsh - "Web Shell"*. Fonte: Adaptada de [Dyatlov and Castro 2003].

O *script* cliente é usado através da linha de comando do *shell* e o servidor deve ser executado como *script* CGI no computador destino. O arquivo *wsh-c.pl* provê uma linha de comando. Cada comando dado pelo cliente é encapsulado em um pedido HTTP POST e enviado ao *script* wsh-s.pl (ou wsh-s.c) no servidor *web* destino ou através de um servidor *proxy* HTTP (Figura 1). O wsh-s.pl extrai, executa o comando e retorna seu resultado como uma mensagem de resposta HTTP.

As próximas ferramentas descritas nesta seção tem o mesmo objetivo: transmitir sobre o HTTP o tráfego de qualquer protocolo da camada de aplicação que utilize TCP ou UDP.

O *firepass* [Dyatlov 2003] cria canais dissimulados, encapsulando a comunicação em requisições *POST* do HTTP, com o objetivo de burlar as restrições impostas por um *firewall*. A ferramenta é composta por dois *scripts* escritos na linguagem *Perl*, um para o cliente e outro para executar como um *script* CGI em um servidor *Web*. No cliente é definido o protocolo (TCP ou UDP) e a porta que será aberta na máquina do cliente para servir como entrada do canal. Além disso, também é definido no cliente qual o endereço IP, porta e protocolo usado no qual o servidor do *firepass* deverá se conectar. Um exemplo prático, conforme ilustrado na Figura 2, seria a utilização da ferramenta para realizar uma conexão SSH entre o *host* interno A e o externo C. Para isso, no cliente do *firepass*, é

definido que a porta 22/tcp deverá ser aberta localmente e o *host* destino será o C na porta 22/tcp. Assim, quando o usuário interno executar o SSH na porta 22 do seu *host*, todo tráfego será encapsulado em requisições POST do HTTP, enviadas para o servidor *Web* B (que possui o *script* servidor do *firepass*), e este irá extrair a informação do canal dissimulado e transmitir para o servidor C na porta 22/tcp. Do ponto de vista do *firewall*, houve apenas uma conexão entre o cliente A e o servidor *Web* B, como se o usuário estivesse navegando em páginas HTML na Internet.

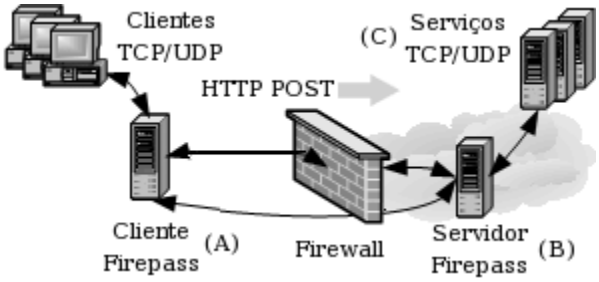

Figura 2: *Firepass*. Fonte: Adaptada de [Dyatlov 2003].

O *httptunnel* [Brinkhoff 2004] cria canais dissimulados sobre o protocolo HTTP entre duas máquinas, permitindo o encapsulamento de qualquer tipo de tráfego. O lado cliente da ferramenta converte as requisições do cliente interno em requisições *POST* do HTTP, e o lado servidor converte as requisições novamente em pacotes IP (contendo o protocolo de aplicação original) para serem enviadas ao destino final do canal. De maneira análoga, o lado servidor converte as respostas de volta para o protocolo HTTP antes de enviá-las ao cliente. Dessa maneira, o *httptunnel* permite que um usuário interno se comunique com um servidor IMAP ou SSH externo, mesmo que no *firewall* existam regras para impedir este tipo de comunicação. A principal diferença entre o *httptunnel* e o *firepass* é que o primeiro não precisa de um servidor *Web* para executar o lado servidor da ferramenta.

4 Sistemas de Detecção de Intrusão

Um sistema completamente seguro é praticamente impossível de se obter. Quando esse sistema é conectado a uma rede de computadores, o seu nível de segurança cai significativamente. Os filtros de pacotes e de aplicações desempenham um papel importante, se configurados corretamente, de modo a permitir que apenas o tráfego destinado aos serviços necessários passem por eles. Porém, estes filtros nada representam se o ataque estiver contido em pacotes que se enquadram nas suas exigências. Estes ataques então devem ser detectados de maneira que os danos causados possam ser rapidamente reparados.

A detecção pode ser feita por meio de sistemas de detecção de intrusão. Anderson [Anderson 1980], ao introduzir o conceito de detecção de intrusão, definiu uma tentativa de intrusão ou ameaça como sendo a possibilidade de ocorrência de uma tentativa deliberada e não autorizada para:

- acessar informações;
- manipular informações;
- tornar um sistema não confiável ou indisponível.

Desta maneira, os sistemas de detecção de intrusão podem ser definidos como mecanismos que permitem detectar a exploração de falhas de segurança, ou seja, detectar ataques.

Segundo Sundaram [Sundaram 2000], as técnicas de detecção de intrusão são divididas em duas categorias principais: detecção de intrusão por anomalia e detecção de intrusão por abuso. Estas serão discutidas na próximas subseções.

4.1 Detecção de Intrusão por Anomalia

As técnicas de detecção por anomalia assumem que todas as atividades intrusivas são necessariamente anômalas. Isto significa que se um perfil de uma atividade normal puder ser criado, é possível (em teoria) que todos os eventos que variam de forma estatisticamente significativa desse perfil sejam classificados como tentativas de intrusão. Entretanto, se for considerado que o conjunto de atividades intrusivas apenas intercepte o conjunto de atividades anômalas ao invés de ser exatamente o mesmo, então existem algumas possibilidades: atividades anômalas que não são intrusivas e são classificadas como intrusivas (falso-positivos); atividades intrusivas que não são anômalas e não são classificadas como intrusivas (falso-negativos) [Sundaram 2000].

As questões primordiais em sistemas de detecção por anomalia estão na seleção dos valores dos limiares de maneira que o número de falso-positivos e falso-negativos sejam mínimos, e também na escolha das características a serem monitoradas. Outra questão que deve ser mencionada se refere ao custo desses sistemas. Por utilizarem perfis e tendo que mantê-los atualizados e sob vigilância constante, esses sistemas normalmente têm o custo bastante elevado.

Os sistemas que utilizam o método de detecção por anomalia são desenvolvidos segundo algumas abordagens, como a abordagem estatística ou utilização de redes neurais [Sundaram 2000].

4.2 Detecção de Intrusão por Abuso

As técnicas de detecção por abuso se baseiam na afirmativa de que existem meios de representar um ataque utilizando padrões ou assinaturas, de forma que variações do mesmo ataque possam ser detectadas.

As questões primordiais em sistemas de detecção por abuso estão em como escrever uma assinatura que cerque todas as variações de um determinado ataque, e também em como escrever assinaturas que não casem com atividades não intrusivas. Uma limitação desse método consiste no fato de que ele busca por vulnerabilidades conhecidas, sendo que novas vulnerabilidades não são reconhecidas como intrusões [Sundaram 2000].

Os sistemas que utilizam o método de detecção por abuso são desenvolvidos segundo algumas abordagens, como a utilização de sistemas especialistas e a detecção por reconhecimento de padrões.

4.3 Sistemas Híbridos

Os sistemas de detecção de intrusão híbridos combinam as técnicas de detecção por anomalia e por abuso, com o objetivo de resolver ou atenuar as deficiências apresentadas por cada uma delas. Eles utilizam os mecanismos de detecção de forma paralela ou serial, com o objetivo de combinar o resultado da análise feita por cada método. A modelagem dos sistemas híbridos é complexa, contendo várias considerações, restrições ou limitações relacionadas a sua implantação. Esses sistemas, geralmente, apresentam bons resultados.

O sistema de detecção de *backdoors* e canais dissimulados, que será descrito na seção 6, é um sistema de deteção de intrusão híbrido, realizando primeiro a detecção por anomalia e depois por abuso.

5 Metodologias e Ferramentas

Esta seção contém as metodologias e ferramentas utilizadas pelo sistema de detecção de *backdoors* e canais dissimulados. A subseção 5.1 apresenta o sistema que serve de base para obtenção das informações sobre as sessões TCP/IP contidas no tráfego analisado, e a subseção 5.2 apresenta a rede neural utilizada como mecanismo de detecção de intrusão por anomalia do sistema desenvolvido.

5.1 Sistema de Reconstrução de Sessões TCP/IP

O sistema de reconstrução de sessões TCP/IP - Recon [Chaves 2002] - reconstrói e rastreia o estado das sessões TCP/IP, baseando-se em um modelo gerado a partir de dados extraídos dos cabeçalhos dos pacotes da pilha de protocolos TCP/IP. O modelo é formado por um grafo, onde os vértices representam todos os endereços IP do tráfego de rede, e cada aresta representa a comunicação entre dois endereços IP do conjunto de vértices. A cada vértice é associado uma lista de vértices adjacentes, contendo todos os endereços IP com os quais este IP se comunicou.

O Recon se restringe a analisar apenas os pacotes dos protocolos *Ethernet*, IP, ICMP, TCP e UDP. À medida que o tráfego é lido de um arquivo em formato *tcpdump* [Jacobson et al. 2004], o Recon reconstrói as sessões ICMP, TCP e UDP entre os pares de endereços IP.

O sistema de reconstrução de sessões TCP/IP foi subdividido em módulos, onde cada módulo possui sua função específica. Após todos os pacotes do tráfego de rede terem sido avaliados pelos módulos do Recon, tem-se como resultado uma árvore binária balanceada com todos os IPs do tráfego. Para cada par de IPs interconectados, seus respectivos nós na árvore terão em suas listas de adjacências um elemento referente ao outro IP em questão. Esses elementos, pertencentes às listas de adjacências de dois IPs, apontam para um bloco de sessões TCP/IP em comum. Esse bloco possui referências para a primeira e a última sessão TCP, UDP e ICMP de informação, e para o primeiro e último pacote ICMP de erro. Essas sessões e os pacotes são elementos de listas duplamente encadeadas [Chaves 2003].

Dessa forma, o Recon organiza as sessões TCP/IP de maneira que as informações de cada sessão podem ser facilmente obtidas, mostrando-se como uma ferramenta interessante para servir como base para um sistema de detecção de intrusão.

5.2 Mapas Auto-Organizáveis

Os mapas auto-organizáveis (SOM) são um tipo de rede neural artificial (RNA) desenvolvida por Kohonen [Kohonen 1989] na década de 1980. As RNAs auto-organizáveis possuem um amplo domínio de aplicações, principalmente em problemas de reconhecimento de padrões e categorização de dados em que as classes não são conhecidas anteriormente. Um dos princípios em que se baseiam os modelos auto-organizáveis é o de que padrões que compartilham características comuns devem ser agrupados, com cada grupo de padrões representando uma e apenas uma classe (embora uma mesma classe possa ser representada por mais de um agrupamento) [Braga et al. 2002].

A rede SOM é utilizada em duas fases: na primeira ocorre o treinamento da rede para organizar os dados, de forma que os mais parecidos fiquem próximos entre si. Para isso, quando um padrão de entrada **p** é apresentado, a rede procura a unidade mais parecida com **p**. Assim, a rede constrói um mapa topológico, onde os nós que estão topologicamente próximos respondem de forma semelhante a padrões de entrada semelhantes. A segunda fase é a de classificação, onde a rede SOM utiliza o mapa organizado para identificar a classe mais próxima à entrada.

6 Sistema de Detecção de *Backdoors* e Canais Dissimulados

A idéia inicial do sistema de detecção de *backdoors* e canais dissimulados surgiu com a criação de uma metodologia de detecção proposta por Chaves e Montes [Chaves and Montes 2004], baseada na análise das sessões TCP/IP do tráfego de uma rede.

A metodologia é dividida em três fases (como mostra a Figura 3): a reconstrução das sessões TCP/IP, a análise e classificação, e a geração do resultado.

Figura 3: Fases da detecção de *backdoors* e canais dissimulados

A primeira fase propõe a utilização do Recon [Chaves 2002] para efetuar a reconstrução das sessões TCP/IP, e, além disso, a extensão do Recon, de modo que ele passe a analisar uma quantidade configurável de dados, para tornar possível a busca por assinaturas no conteúdo dos pacotes. Na segunda fase, uma vez reconstruída, a sessão é analisada em busca de características que a classifiquem como pertencente a um *backdoor* ou canal dissimulado. Primeiramente é feita uma análise de comportamento do protocolo utilizado na sessão. Depois, procura-se no conteúdo do pacote por assinaturas conhecidas dos *backdoors* e das ferramentas que implementam canais dissimulados, de forma a identificá-los. A terceira fase da metodologia gera um relatório contendo o resultado da análise e classificação feita na segunda fase e as informações das sessões TCP/IP [Chaves and Montes 2004].

Essa metodologia leva à criação de um sistema de detecção de intrusão híbrido (subseção 4.3), sendo feita primeiramente a detecção por anomalia e posteriormente por abuso.

Algumas modificações foram feitas no Recon, sendo a primeira a substituição das rotinas que implementam a árvore de endereços IP por rotinas de uma biblioteca otimizada chamada *GNU libavl*[2], desenvolvida por Ben Pfaff. Essa modificação foi realizada com o intuito de diminuir o tempo de criação das estruturas, uma vez que novas funcionalidades foram incorporadas e passou-se a gastar mais tempo nessa fase.

Neste artigo, serão discutidas apenas as sessões que utilizam o protocolo TCP na camada de transporte (porém as idéias podem ser facilmente implementadas para outros protocolos). Assim, para possibilitar a reconstrução do conteúdo das sessões TCP, a máquina de estados TCP do Recon foi atualizada, incluindo-se um controle sobre o número de seqüência e confirmação dos pacotes, para fazer com que apenas os dados que foram realmente recebidos pela outra parte de uma conexão sejam reconstruídos.

Uma vez que a base do sistema está formada, o próximo passo caracteriza-se no desenvolvimento do mecanismo de detecção de *backdoors* e canais dissimulados. O princípio básico para detecção de um *backdoor* está em encontrar características que indiquem a atividade de interesse. Entre os candidatos para essas características, estão: as assinaturas nos dados, o tamanho e a taxa de transmissão dos pacotes e o intervalo de tempo entre os pacotes de uma dada sessão [Zhang and Paxon 2000]. Em relação a detecção de canais dissimulados, mais especificamente os que utilizam o protocolo HTTP, alguns autores concordam que a definição de limiares para certas características das sessões HTTP são suficientes para detecção de canais dissimulados. Além

[2]http://www.stanford.edu/ blp/avl/

disso, eles concordam que essa idéia é aplicável a outros protocolos da pilha TCP/IP [Mudge 2003, Castro 2003b, Pack et al. 2002].

As idéias pesquisadas, em adição ao objetivo de desenvolver um sistema de detecção de intrusão por anomalia, levaram à busca por trabalhos com objetivos semelhantes ao desse. Conforme dito na subseção 4.1, a detecção de intrusão por anomalia se baseia na definição de um perfil normal e na comparação das informações dos eventos com o normal. Sendo assim, a técnica difere entre os sistemas na maneira de representar o perfil e em como calcular a diferença entre o normal e os eventos observados. Para isso, podem ser utilizadas técnicas probabilísticas [Ye et al. 2001], técnicas baseadas em análise de agrupamentos (*clustering*) [Portnoy et al. 2001], técnicas de *Data Mining* [Dokas et al. 2002, Ertoz et al. 2003, Barbara et al. 2001, Lee and Stolfo 1998] ou redes neurais artificiais [Rhodes et al. 2000, Ramadas et al. 2003].

Para representar uma sessão, nove características foram selecionadas, sendo estas:

1. Tamanho médio dos pacotes recebidos pelo cliente;
2. Tamanho médio dos pacotes recebidos pelo servidor;
3. Número de pacotes recebidos pelo cliente;
4. Número de pacotes recebidos pelo servidor;
5. Porcentagem de pacotes pequenos;
6. Direção do tráfego;
7. Total de dados recebidos pelo cliente;
8. Total de dados recebidos pelo servidor;
9. Duração da sessão.

Para cada sessão, essas características são extraídas de forma a gerar perfis normais do tráfego. Uma vez que cada protocolo pode gerar vários perfis normais com certa diferença entre eles, o mecanismo de detecção de *backdoors* e canais dissimulados irá utilizar, para representar esses perfis e detectar sessões anômalas, mapas auto-organizáveis (SOM) [Kohonen 1989].

Uma vez que as características utilizadas estão em ordem de grandeza e unidades diferentes, é feita uma normalização dos dados. Este processo é dividido em duas etapas, sendo que a primeira consiste no cálculo da média (μ) e do desvio padrão (σ) para cada uma das características. Na segunda etapa, cada vetor de nove posições $< c1, c2, c3, c4, c5, c6, c7, c8, c9 >$, contendo as características das sessões, é normalizado para $< n1, n2, n3, n4, n5, n6, n7, n8, n9 >$, utilizando-se a seguinte equação:

$$n_i = \frac{c_i - \mu_i}{\sigma_i} \tag{1}$$

O SOM utilizado no sistema é bidimensional, e cada neurônio é representado como um vetor de nove dimensões, formados pelas características das sessões de forma normalizada. Conforme dito na subseção 5.2, o mapa precisa ser primeiramente treinado, como qualquer rede neural artificial. Para isso, um tráfego legítimo (sem nenhum pacote ou sessão maliciosa) deve ser utilizado.

A fase de aprendizagem (treinamento) começa com a inicialização do SOM com valores aleatórios entre 0 e 1. Depois, para cada vetor (instância) das sessões, um algoritmo competitivo é utilizado para encontrar o nó vencedor na rede (aquele que possui menor distância euclidiana em relação a instância avaliada). Ao ser encontrado, o nó vencedor e seus vizinhos dentro de um certo raio ou vizinhança atualizam seus pesos (através de uma função de aprendizagem) para representar a classe do padrão de entrada. Ao final

dessa fase, o SOM está treinado e pronto para ser utilizado para o reconhecimento de padrões (fase de operação), ou seja, para detecção de sessões anômalas.

A fase de operação do SOM acontece da seguinte forma: para cada sessão extraída pelo Recon, é criado um vetor (instância) contendo as nove características utilizadas. Esse vetor é normalizado utilizando-se a equação 1, e depois o nó vencedor (com menor distância euclidiana) é encontrado. A sessão será classificada como normal se ela estiver suficientemente perto do nó vencedor, e anômala caso a distância para o nó vencedor seja maior que um limiar pré-definido. A definição deste limiar é feita por tentativa e erro, de modo que o número de falso-positivos e falso-negativos sejam mínimos. Ao final dessa fase, todas as sessões TCP foram classificadas como normais ou anômalas.

Após a detecção por anomalia, é feita a detecção por abuso, onde o conteúdo das sessões classificadas como anômalas são analisados em busca de assinaturas de ataques conhecidos. O mecanismo de detecção utiliza regras baseadas nas utilizadas pelo sistema de detecção de intrusão *Snort*[3]. Essa estratégia visa a identificação do *backdoor* ou ferramenta de canal dissimulado utilizada no ataque. Um exemplo de regras de detecção é mostrado na Figura 4.

```
Covert channel tool httptunnel Usage#POST /index.html?crap=#3000001
Covert channel tool firepass Usage#POST /cgi-bin/fpserver.cgi#3000002
Covert channel tool wsh Usage#POST /cgi-bin/wsh-s#3000003
Backdoor rwwwshell Usage#POST /cgi-bin/orderform#3000004
ATTACK-RESPONSES id check returned root#uid=0(root)#498
```

Figura 4: Exemplo de regras do Sistema de Detecção de *Backdoors* e Canais Dissimulados.

As regras possuem apenas três campos separados por #, sendo o primeiro a mensagem do alerta a ser gerado, o segundo a cadeia de caracteres a ser encontrada no conteúdo da sessão, e o terceiro é um identificador da regra (igual ao SID das regras do *Snort*).

Finalmente, o sistema de detecção de *backdoors* e canais dissimulados gera um relatório detalhado contendo todos os dados da sessões TCP/IP, incluindo a sua classificação (normal ou anômala) e os alertas gerados pelo mecanismo de detecção por abuso. Este relatório deverá ser examinado por um analista para uma estudo mais detalhado sobre as sessões maliciosas.

7 Resultados

O sistema de detecção de *backdoors* e canais dissimulados foi testado utilizando tráfego do protocolo HTTP da rede do Laboratório Associado de Computação a Matemática Aplicada (LAC) do INPE. Para sua coleta, um sensor foi posicionado no mesmo segmento da interface interna do *firewall*, e foi capturado o tráfego de uma semana. O perfil do tráfego pode variar normalmente, de acordo com o dia da semana e o período de cada dia, ou seja, o perfil do tráfego da madrugada de domingo não deve ser parecido com o perfil do horário comercial de um dia útil. Sendo assim, foram gerados quatro arquivos por dia, de acordo com os horários de trabalho no LAC (0:00-7:59, 8:00-12:59, 13:00-17:59 e 18:00-23:59). Na fase de teste, o perfil referente ao dia e período correto deve ser utilizado, ou seja, se o tráfego analisado for de segunda-feira, entre 10:00 e 11:00 horas, então o perfil utilizado será de segunda-feira, de 8:00 às 12:59.

[3]http://www.snort.org

Para testar o sistema, foi utilizado como estação de análise um computador com processador Intel Pentium 4 de 2.66GHz, 512Kb de memória cache, 512Mb de memória RAM e o sistema operacional Slackware Linux 10.1. Um fator importante a ser colocado é o fato de toda a reconstrução e análise das sessões TCP/IP ser feita em memória. Assim, o desempenho do sistema é dependente do total de memória RAM livre no momento da sua execução. Se o arquivo contendo as sessões TCP/IP ultrapassar a quantidade de memória disponível, fatalmente a área de *swap* será utilizada, e o desempenho do sistema será afetado. Sendo assim, existe uma limitação em relação ao tamanho do arquivo a ser analisado.

O primeiro teste realizado no sistema foi em busca de falso-positivos, utilizando como entrada os arquivos usados para gerar os perfis do tráfego. Além de informar o número de falso-positivos, este teste valida o treinamento da rede neural. O sistema apresentou uma taxa média de 0,22% de sessões anômalas, ou seja, em um tráfego contendo apenas sessões lícitas, 0,22% delas são classificadas como anômalas pelo sistema de detecção, resultando em uma taxa de 0,22% de falso-positivos.

Para testar a detecção de *backdoors* e canais dissimulados, foram utilizadas duas máquinas: uma atuando como cliente, localizada na rede interna do LAC, e outra atuando como servidora, localizada fora da rede do LAC. O tráfego entre as máquinas foi capturado durante a execução das ferramentas, e analisado pelo sistema de detecção de *backdoors* e canais dissimulados.

O *netcat* foi utilizado para criar um *backdoor* no servidor, na porta 80/tcp. O cliente, na máquina da rede interna, executou vários comandos no servidor. O sistema reconstruiu uma sessão do *netcat*, sendo essa classificada como anômala. Como não foi definida uma assinatura que identifica o *netcat*, a ferramenta não foi identificada, porém um alerta foi disparado informando que o comando *id* retornou *root*, ou seja, o atacante tinha privilégio de super-usuário. Uma vez que comando *id* foi executado pelo cliente, e o *backdoor* executado com privilégio de súper-usuário, o resultado do comando foi *uid=0(root)*. Essa cadeia casou com a cadeia de caracteres procurada por uma regra de detecção: "*uid=0(root)*".

O *rwwwshell* foi utilizado para simular um ataque de *shell* reverso, onde o atacante, no servidor externo, executa comandos na máquina localizada na rede interna. O sistema reconstruiu oito sessões, sendo apenas uma classificada como anômala. A assinatura que identifica a ferramenta foi encontrada na sessão anômala (*POST /cgi-bin/orderform*). Sete sessões referentes a comandos executados pelo atacante foram classificadas como normais, resultando em uma taxa de 87,5% de falso-negativos para o teste com essa ferramenta. Porém, este valor alto não é um problema, visto que todas as sessões, tanto anômalas quanto normais, pertencem ao mesmo par de IPs. Assim, todas deverão ser posteriormente examinadas pelo analista.

O último *backdoor* testado foi o *cctt*, sendo realizado o mesmo tipo de teste do *netcat*. O sistema reconstruiu uma sessão do *cctt*, sendo essa classificada como anômala. Neste modo de operação, o *cctt* envia diretamente os comandos para o servidor, não existindo assinaturas específicas. Como no caso no *netcat*, um alerta foi disparado informando que o comando *id* retornou *root*, ou seja, o atacante tinha privilégio de super-usuário.

As ferramentas *httptunnel* e *firepass* foram utilizadas para encapsular o tráfego de SSH sobre o HTTP, simulando um cliente na rede interna utilizando um canal dissimulado para enviar dados para fora da organização (espionagem industrial). O sistema reconstruiu duas sessões do *httptunnel*, e ambas foram classificadas como anômalas. Além disso, uma assinatura que identifica a ferramenta foi encontrada na primeira sessão (*POST /index.html?crap=*). A respeito do *firepass*, três sessões foram reconstruídas, sendo todas

classificadas como anômalas. A assinatura que identifica a ferramenta foi encontrada em todas as sessões anômalas (*POST /cgi-bin/fpserver.cgi*).

A ferramenta *wsh* foi utilizada para executar comandos no servidor *Web* fora da rede do LAC. Além disso, o *wsh* foi utilizado para transferir um arquivo do cliente para o servidor, através do canal dissimulado, simulando o ato de um atacante enviar um programa à máquina invadida para, por exemplo, elevar seu privilégio de usuário comum (*apache*) para super-usuário (*root*). O sistema reconstruiu sete sessões, sendo duas classificadas como anômalas (sessões referentes a transferência de dois arquivos para o servidor). A assinatura que identifica a ferramenta foi encontrada em todas as sessões anômalas (*POST /cgi-bin/wsh-s*). As sessões referentes ao envio de comandos para o servidor e o recebimento das respostas foram classificadas como normais, indicando uma taxa de 71% de falso-negativos para o teste com essa ferramenta. Porém, como no caso do *rwwwshell*, este valor alto não é um problema, pois todas as sessões pertencentes as duas máquinas deverão ser examinadas pelo analista.

8 Conclusão e Trabalhos Futuros

Os *backdoors* e canais ocultos são problemas enfrentados por qualquer administrador de redes. O sistema desenvolvido para sua detecção pretende dar uma contribuição para toda comunidade de administração e segurança de redes.

A utilização de mapas auto-organizáveis (SOMs) como mecanismo de detecção por anomalia se mostrou eficiente para detecção de *backdoors* e canais dissimulados. A utilização deles em sistemas de detecção de intrusão não é inédita, porém a utilização em conjunto com um mecanismo de detecção por abuso se torna um diferencial entre os sistemas existentes. As assinaturas dos *backdoors* e canais dissimulados podem ser encontradas e assim, regras de detecção podem ser criadas. Porém, como é comum em todo sistema de detecção por abuso, as ferramentas podem ser alteradas para não serem mais identificadas pelas regras. Neste caso, as ferramentas podem ser identificadas pelo seu comportamento, e assim uma nova regra de detecção pode ser gerada de acordo com a nova assinatura.

É importante salientar que a criação de perfis para um determinado tráfego deve ser uma tarefa dinâmica. A característica do tráfego de uma rede pode mudar normalmente, sem que ocorram eventos maliciosos. Assim, os perfis devem ser ajustados a essa nova situação, a fim de reduzir o número de falso positivos.

O sistema considera apenas as sessões que utilizam o protocolo TCP na camada de transporte. Porém, ele foi desenvolvido de forma que a expansão para a análise de outros protocolos, como UDP e ICMP, pode ser feita com facilidade. Além disso, várias características observadas nas sessões TCP podem ser empregadas para a definição dos perfis normais de outros protocolos.

A técnica de detecção por abuso se baseia na busca de assinatura no conteúdo das sessões, e o mecanismo de detecção utiliza regras baseadas nas utilizadas pelo sistema de detecção de intrusão *Snort*. Este mecanismo foi desenvolvido, de maneira simples, para buscar por cadeias de caracteres no conteúdo das sessões. Para o próximo trabalho, é proposta a utilização de expressões regulares e busca por assinaturas em hexadecimal, como é feito no *Snort*.

O sistema apresentou um resultado satisfatório, pois pelo menos uma sessão referente a cada uma das ferramentas testadas foi detectada. Em algums casos houveram taxas elevadas de falso-negativos, onde apenas uma de várias sessões foram detectadas como anômalas. Porém, isto não representa um problema, pois o resultado do sistema

sempre deve ser avaliado por um analista. Assim, ao analisar a sessão que foi detectada como anômala, o analista deverá também analisar as outras sessões entre um mesmo par de IPs. Como continuação, é proposto o teste do sistema com tráfego gerado por outros *backdoors* e ferramentas que implementam canais dissimulados.

A última consideração sobre os trabalhos futuros consiste na modificação do sistema para trabalhar em tempo real, armazenando o tráfego de rede coletado e gerando alertas a medida que as sessões anômalas são identificadas.

Referências

Anderson, J. P. (1980). Computer security threat monitoring and surveillance. Technical Report, James P. Anderson Co. http://csrc.nist.gov/publications/history/ande80.pdf.

Barbara, D., Couto, J., Jajodia, S., Popyack, L., and Wu, N. (2001). Adam: Detecting intrusions by data mining. In *IEEE Workshop on Information Assurance and Security*. http://www.itoc.usma.edu/Workshop/2001/Authors/Submitted_Abstracts/paperT1A3(21).pdf.

Braga, A. P., Carvalho, A. C. P. L. F., and Ludemir, T. B. (2002). *Redes Neurais Artificiais Teoria e Aplicações*. LTC, 1 edition.

Brinkhoff, L. (2004). Httptunnel. NoCrew.org. http://www.nocrew.org/software/httptunnel.html.

Castro, S. (2003a). Cctt - Covert Channel Tunneling Tool. Gray World.net Team. http://www.gray-world.net/pr_cctt.shtml.

Castro, S. (2003b). Covert Channel and Tunneling over the HTTP protocol Detection: GW implementation theoretical design. Gray World.net Team. http://www.gray-world.net/projects/papers/html/cctde.html.

Chaves, C. H. P. C. (2003). Ferramenta de visualização gráfica do tráfego em redes tcp/ip - trafficshow. Relatório técnico 04/2003, Universidade Federal de Ouro Preto, Departamento de Computação.

Chaves, C. H. P. C. and Montes, A. (2004). Backdoors e Canais Dissimulados: uma metodologia para detecção. In *Anais do VI Simpósio sobre Segurança em Informática (SSI'2004)*, São José dos Campos, SP.

Chaves, M. H. P. C. (2002). Análise de estado de tráfego de redes tcp/ip para aplicação em detecção de intrusão. Dissertação de mestrado, Instituto Nacional de Pesquisas Espaciais, Laboratório Associado de Computação e Matemática Aplicada LAC.

daemon9 (1997). LOKI2 (the implementation). *Phrack Inc.*, 51(06). http://www.phrack.org/show.php?p=51\&a=6.

daemon9 and alhambra (1996). Project Loki. *Phrack Inc.*, 49(06). http://www.phrack.org/show.php?p=49\&a=6.

Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., and Tan, P. N. (2002). Data mining for network intrusion detection. In *NSF Workshop on Next Generation Data Mining*. http://www-users.cs.umn.edu/~aleks/MINDS/papers/nsf_ngdm_2002.pdf.

Dyatlov, A. (2003). Firepass. Gray World.net Team. http://www.gray-world.net/pr_firepass.shtml.

Dyatlov, A. and Castro, S. (2003). Wsh - Web Shell. Gray World.net Team. http://www.gray-world.net/pr_wsh.shtml.

Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P. N., Dokas, P., Kumar, V., and Srivastava, J. (2003). Detection and summarization of novel network attacks using data mining. Technical report, University of Minnesota, Computer Science Departament. http://www-users.cs.umn.edu/~aleks/MINDS/papers/raid03.pdf.

Farrow, R. (2004). Musings. *;login: The Usenix Magazine*, 29(08).

Hauser, V. (1999). Placing Backdoors Through Firewalls. The Hacker's Choice. http://www.thc.org/download.php?t=p&d=fw-backd.htm.

Hauser, V. (2002). The Reverse WWW Shell. The Hacker's Choice. http://www.thc.org/download.php?t=r&d=rwwwshell-2.0.pl.gz.

Jacobson, V., Leres, C., and McCanne, S. (2004). Tcpdump. TCPDUMP public repository. http://www.tcpdump.org/release/tcpdump-3.8.3.tar.gz.

Kohonen, T. (1989). *Self-Organizing and Associative Memory*. Springer-Verlag, 3 edition.

Lampson, B. W. (1973). A note on the confinement problem. *Communications of the ACM*, 16(10):613–615.

Lee, W. and Stolfo, S. (1998). Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*. http://citeseer.ist.psu.edu/article/lee98data.html.

Mudge (2003). Insider threat. *;login: The Usenix Magazine*, 28(06).

Pack, D. J., Streilein, W., Webster, S., and Cunningham, R. (2002). Detecting HTTP Tunneling Activities. In *In Proceedings of the 2002 IEEE Workshop on Information Assurance*. http://www.ll.mit.edu/IST/pubs/Pack-IEEE2002.pdf.

Portnoy, L., Eskin, E., and Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. In *ACM Workshop on Data Mining Applied to Security (DMSA 2001)*. http://citeseer.ist.psu.edu/article/portnoy01intrusion.html.

Ramadas, M., Ostermann, S., and Tjaden, B. C. (2003). Detecting anomalous network traffic with self-organizing maps. In *RAID*. http://www.cs.fit.edu/~pkc/id/related/ramadas03raid.ps.gz.

Rhodes, B. C., Mahaffey, J. A., and Cannady, J. D. (2000). Multiple self-organizing maps for intrusion detection. In *Proceedings of the 23rd National Information Systems Security Conference*. http://dbvis.fmi.uni-konstanz.de/members/panse/seminar_ws0203/pdf/045.pdf.

Sundaram, A. (2000). An introduction to intrusion detection. http://coast.cs.purdue.edu/pub/doc/intrusion_detection/Intrusion-Detection-Intro.ps.Z.

U.S. Department of Defense (1985). Trusted Computer System Evaluation. In *The Orange Book*, Washington, DC, USA. http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html.

Ye, N., Li, X., Chen, Q., Emran, S. M., and Xu, M. (2001). Probabilistic techniques for intrusion detection based on computer audit data. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(4).

Zhang, Y. and Paxon, V. (2000). Detecting Backdoors. In *Proceedings of the 9th Usenix Security Symposium*. http://www-cse.ucsd.edu/~savage/cse291/papers/Zhang00-1.pdf.