Ferramenta de Injeção de Falhas para Avaliação de Sistemas de Segurança em Rede

Paulo César Herrmann Wanner, Raul Fernando Weber

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil
{herrmann,weber}@inf.ufrgs.br

Abstract. *A great number of new security tools have been developed to solve the problems of IP networks but some doubts were raised about the their efficiency. There are many techniques used to evaluate the vunerabilites of a network, but their results are limited and few tools aim to facilitate and automate the evaluation of security network programs. This article presents a network fault injection system created to test network security equipment, aiming to fill the gap between packet injectors and vulnerability scanners.*

Resumo. *O surgimento de diversos programas na área de segurança trouxe inúmeros benefícios aos problemas de insegurança inerente às redes IP, mas também algumas dúvidas sobre a sua eficácia. Existem várias técnicas desenvolvidas para avaliar as vulnerabilidades de uma rede, mas suas avaliações são limitadas e poucas são as ferramentas desenvolvidas visando facilitar e automatizar o teste de equipamentos de segurança em rede. Este artigo apresenta um sistema de injeção de falhas específico para a avaliação desses equipamentos com o objetivo de preencher a lacuna existente entre os injetores de pacotes e os programas de varredura de vulnerabilidades.*

1. Introdução

Nos últimos anos, com a crescente popularização das redes de computadores baseadas no protocolo IP, desde pequenas redes até as redes metropolitanas começaram a se agrupar e a fazer parte do que hoje se conhece como a rede mundial de computadores. Apesar dos benefícios de comunicação e troca de informação da Internet, esse fenômeno global também trouxe problemas de segurança, pois a origem e estrutura dos protocolos utilizados na comunicação entre as diversas máquinas limitam as possibilidades de prevenir, identificar ou detectar possíveis ataques ou intrusos.

Assim, várias ferramentas surgiram para prevenir e auxiliar na tarefa de identificar problemas de segurança nas redes como *firewalls*, *sniffers* [Chapman 2000] e *intrusion detection systems* (IDS) [Bace 99]. Apesar dos benefícios trazidos por essas novas tecnologias, muitas dúvidas referentes à segurança que esses programas proporcionam surgiram. Afinal, esses sistemas são muito complexos de serem desenvolvidos e muitas vezes são alvos de ataques. O resultado disso, não raramente, é uma falsa noção de segurança devido à utilização inadequada desses equipamentos, o que é, normalmente, mais prejudicial que a simples inexistência de segurança em uma organização, mas cuja insegurança é conhecida por seus administradores.

Existem diversas técnicas para verificar a segurança de uma rede, mas ainda não há uma metodologia simples e eficaz que seja capaz de avaliar com precisão o nível de

vulnerabilidade de uma organização. Atualmente, pode-se identificar vulnerabilidades em um sistema e verificar a tolerância a ataques que uma rede suporta, mas, infelizmente, não se pode afirmar que uma rede é totalmente segura ou qual o nível de sua segurança.

Além de técnicas para a avaliação de segurança, é muito importante possuir ferramentas para a realização desses testes. Assim, é possível automatizar a realização de testes e verificar com maior facilidade a existência de problemas em uma rede. A existência de ferramentas que testem sistemas de segurança é extremamente importante e necessária, pois, afinal, a segurança de toda uma rede pode depender fortemente de algum desses sistemas.

2. Avaliação de Segurança

A razão principal da realização de testes de segurança em sistemas computacionais é a de identificar vulnerabilidades e conseqüentemente repará-las. O número de vulnerabilidades encontradas vem crescendo a cada ano e elas são normalmente utilizadas por atacantes que exploram suas fragilidades quando essas não são corrigidas por empresas ou organizações, segundo dados do CERT/CC.

A realização de testes para verificação da segurança de uma organização é uma atividade fundamental a fim de alcançar um ambiente operacional seguro e de verificar a correta aplicação dos requisitos de segurança de uma organização. O uso de testes permite a uma empresa verificar com precisão a postura de segurança utilizada em seus sistemas ao mesmo tempo em que permite visualizar a sua rede da mesma maneira que um atacante a visualizaria. Pode-se dizer que essa é a grande vantagem da realização de teste de segurança em relação aos outros mecanismos de verificação de segurança. Ao visualizar a rede conforme um atacante, é possível verificar com que facilidade obtém-se informações da rede, quais suas fragilidades e a dificuldade que se tem para invadi-la. Assim, obtém-se uma visão mais realista da segurança de uma organização.

Existem diferentes tipos de testes de segurança que podem ser realizados em uma organização. Alguns são predominantemente manuais e requerem intervenção humana para serem conduzidos, enquanto que outros são altamente automatizados e necessitam pouca interação humana. Indiferentemente ao tipo de teste de segurança, todos necessitam ser aplicados por pessoas especializadas na área de segurança para melhor acompanhar os testes e interpretar os resultados obtidos. Os tipos de testes de segurança mais comuns existentes são [Wack 2002]: *Network Mapping*, *Vulnerability Scanning*, *Penetration Testing*, *Security Test and Evaluation*, *Password Cracking*, *Log Review*, *Integrity Checkers*, *Virus Detection*, *War Dialing*. É importante salientar também que nenhum dos testes listados tem a capacidade, se aplicado sozinho, de prover uma visão completa da rede testada ou de sua segurança.

3. Ferramentas para Avaliação de Segurança de Rede

Dos métodos enumerados anteriormente, normalmente três são utilizados diretamente para avaliar mecanismos de segurança da rede: *Network Mapping*, *Vulnerability Scanning* e *Penetration Testing*.

Cada um possui suas características, vantagens e desvantagens, contudo todos possuem um problema em comum: a falta de programas especificamente desenvolvidos para injeção de falhas em sistemas de segurança de rede. A maioria das ferramentas

encontrada é inadequada para a realização desses testes, pois foram projetadas com outras finalidades [McHugh 2000]. Uma pesquisa sobre ferramentas para injeção de falhas em sistemas de segurança em rede observou a existência de basicamente dois tipos de programas para esse fim. Existem as ferramentas para geração de pacotes e os programas de varredura de vulnerabilidades.

Os aplicativos para geração de pacotes, também conhecidos como injetores de pacotes, são ferramentas que permitem a criação e envio de pacotes de protocolos da pilha TCP/IP para uma determinada estação. Dessa forma, é possível criar qualquer tipo de pacote, forjando-se comunicações e origens de dados com a finalidade de testar-se programas como *firewalls* e sistemas de detecção de intrusão. Examinou-se algumas das ferramentas para a geração e envio de pacotes de protocolos da pilha TCP/IP como hping [Sanfilippo 2001], nemesis [Grimes 2000] e sendip [Ricketts 2000]. O problema, atualmente, em utilizar esse tipo de software para injetar falhas em sistemas de segurança em rede é que eles não fornecem as características necessárias para tais fins. As ferramentas de injeção de pacotes de modo geral: são em modo texto; não enviam seqüências de pacotes; possuem poucos recursos para monitorar respostas da máquina destino; e não permitem a implementação de testes sofisticados.

Outras ferramentas encontradas para injeção de falhas são os *scanners* de vulnerabilidades. Essas ferramentas são desenvolvidas com o objetivo de automatizar a procura por falhas em serviços e máquinas existentes em uma determinada rede. Apesar de não serem aplicativos que realizem testes em rede, são bastante utilizados para testar os mecanismos de detecção de um IDS por serem usados por intrusos. Entre os aplicativos encontrados na Internet para esse fim pode-se citar: Nessus [Deraison 2000], SAINT [World Wide Digital Security Inc. 2000] e SARA [Todd 1999]. São ferramentas de mais alto nível que possuem uma série de facilidades, contudo a maioria delas não possui um mecanismo de monitoração das respostas do destino acessível aos usuários, ou seja, o usuário não tem acesso direto as respostas. O único contato é através do relatório de vulnerabilidades, após as realizações dos testes. Elas também não permitem ao usuário criar um teste a ser executado, com exceção do Nessus [Deraison 2000], devido a sua linguagem *script* NASL.

Uma análise e comparação mais detalhadas desses dois tipos de ferramentas podem ser encontradas em [Wanner 2001].

4. Modelo de um Sistema de Injeção de Falhas em Redes

Como pôde ser visto na seção anterior, existem dois tipos de ferramentas que podem ser utilizadas para injeção de falhas em segurança de rede. A ferramenta ideal para injetar falhas em equipamentos de segurança de rede está em um meio termo entre esses dois tipos de aplicativos. É um injetor de falhas do nível de rede do mesmo modo que os geradores de pacotes, mas com uma maior flexibilidade de utilização como os *scanners* de vulnerabilidades. Nesse caso, deve ser capaz de gerar e enviar pacotes de protocolos do nível de rede, transporte e aplicação. Além disso, deve possuir uma linguagem de programação que permita enviar seqüência de pacotes conforme as respostas monitoradas da máquina alvo.

Segundo Hsueh [Hsueh 1997], um sistema de injeção de falhas deve possuir os seguintes elementos: injetor de falhas, biblioteca de falhas, gerador de *workload*, biblioteca de *workload*, controlador, monitor, coletor de dados e analisador de dados.

O sistema proposto, por ser um injetor de falhas, deveria realizar todas as atividades de um sistema de injeção de falhas, contudo como realiza testes em equipamentos de segurança de rede algumas dessas atribuições podem ser simplificadas. O modelo proposto visa implementar esses elementos através de três módulos, figura 1:

- módulo de gerência;
- módulo de injeção e monitoração;
- módulo de teste.

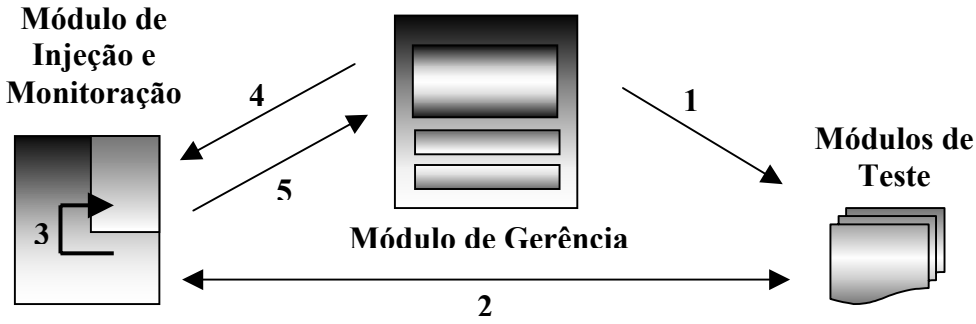

Figura 1 - Arquitetura proposta

O módulo de gerência é responsável pela inicialização, finalização (figura 1, (1)) e monitoração (figura 1, (4, 5)) dos testes em execução e pela interface gráfica com o usuário do sistema; implementando, assim, algumas das funções destinadas ao controlador e ao monitor de um sistema de injeção de falhas. Outra funcionalidade importante desempenhada pelo módulo de gerência é a de edição de pacotes. O módulo de gerência, através de sua interface gráfica, permite que pacotes sejam criados e editados para serem utilizados na avaliação de sistemas de segurança em rede.

O módulo de injeção e monitoração, como o próprio nome sugere, é responsável pelo injetor de falhas e pelo coletor de dados. O injetor de falhas propriamente dito é representado pela geração e envio de pacotes de protocolos da pilha TCP/IP, enquanto o coletor de dados é representado pela monitoração da rede e observação das respostas geradas pela estação sendo testada. Essas duas funções são disponibilizadas para os usuários do sistema a fim de que estes as utilizem para o desenvolvimento dos seus testes (*plug-ins*).

O módulo de injeção e monitoração deve fornecer as funções básicas para a injeção de falhas como funções de envio de pacotes *ethernet*, ARP, IP, UDP, TCP e ICMP, funções para monitoração da rede e funções avançadas para teste de segurança tais como *dnsspoof*, *tcpkill*, *arpspoof* e *scanning*. Desta forma, visa fornecer ao usuário uma base para a realização de testes simples através das funções de envio de pacotes da pilha TCP/IP e também uma base mínima para a realização de testes mais sofisticados através das funções de ataque como *dnsspoof*, *tcpkill*, etc.

Além das funções de teste, esse módulo implementa algumas funções específicas para o módulo de gerenciamento: comunicação, gerenciamento e estatísticas, figura 1 (3, 4, 5). Essas funções são importantes para a troca de informações entre o módulo de gerência e os módulos de testes, além de permitir ao usuário ter informações sobre os testes durante a sua execução e obter um resumo de cada teste ao seu final.

Uma característica importante em uma ferramenta para injeção de falhas em sistema de segurança em rede é que os usuários sejam capazes de desenvolver seus

próprios testes, não sendo obrigados a depender dos testes fornecidos juntos com a ferramenta. Por isso, a existência de algum tipo de linguagem para a criação de testes é necessária. As características desejáveis em uma linguagem deste tipo são: simplicidade, eficiência e funcionalidade. Deve ser eficiente no sentido de realizar os testes sem que a tradução ou compilação da linguagem represente muito *overhead* na execução dos testes.

Assim, a biblioteca de falhas, o gerador de *workload* e algumas funções do controlador e do monitor do sistema de injeção de falhas são de responsabilidade dos módulos de teste em execução, ou seja, quem deve criar e determinar a lógica do teste ou do *workload* é o usuário da ferramenta, conforme o tipo de falhas ou carga que deseja injetar. Um conjunto de modelos de ataques, *workloads* e pacotes pode ser definido e fornecido junto com a ferramenta, contudo o usuário também é capaz de criar os seus próprios ataques e pacotes de testes.

Através do módulo de testes e de sua linguagem para especificação de testes, permite-se que esse modelo proposto atinja uma maior flexibilidade e portabilidade que as ferramentas de geração de pacotes, aproximando-se das funcionalidades de um *scanner* de vulnerabilidades. Toda a comunicação entre os módulos de teste e a ferramenta é feita através das funções do módulo de injeção e monitoração disponibilizado ao usuário, figura 1 (2, 3, 5). A única interação entre os testes e o módulo de gerenciamento é no momento de sua inicialização e finalização, figura 1 (1).

O coletor de dados e o analisador teriam suas funções compartilhadas entre o injetor de falhas e o sistema de segurança de rede sendo testado. Isso se deve ao fato que um dos objetivos da injeção de falhas é verificar a capacidade de detecção, do correto registro das atividades e da correta ativação de alarmes e contramedidas do sistema de segurança em rede sendo testado. Assim, a coleta de dados do injetor de falhas restringe-se as respostas do equipamento testado e do tráfego existente na rede, enquanto que a análise é feita pelo usuário do injetor através das funções fornecidas pelo módulo de injeção e monitoração. Todos os demais dados devem ser coletados e analisados pelo equipamento testado e a correta funcionalidade dele depende dos testes sendo feitos, da sua política de segurança e da sua correta configuração.

5. Implementação

A ferramenta de injeção de falhas descrita é orientada para o teste de sistemas de segurança em rede, preenchendo a lacuna existente entre as ferramentas de injeção de pacotes e os *scanners* de vulnerabilidades. Suas características permitem a realização de testes tanto em *firewall* como em IDS. Um protótipo dessa ferramenta foi desenvolvido em ANSI/C para a plataforma GNU/Linux. O módulo de gerenciamento da ferramenta possui uma interface gráfica, desenvolvida utilizando-se a biblioteca GTK, através da qual pode-se criar, editar e enviar seqüências de pacotes da pilha TCP/IP, figura 2 (1).

O injetor de falhas foi desenvolvido de modo a permitir o máximo de mobilidade e praticidade de uso ao usuário. Todos os pacotes de protocolos implementados estão disponíveis em uma estrutura de *tags* o que facilita o deslocamento entre os diversos protocolos, figura 2 (2). Não é feito nenhum tipo de consistência dos dados entrados pelo usuário, tanto de tipo de dados como de tamanho. Assim, qualquer que seja o dado entrado o programa tentará enviá-lo. Essa característica foi definida para garantir uma melhor flexibilidade de uso por parte do usuário, desta forma, é possível criar os mais

diversos tipos de pacotes mesmo que eles não estejam corretos e explorar possíveis vulnerabilidades em sistemas e protocolos. Todos os pacotes trabalhados pela interface são salvos no formato da Libpcap, assim permitindo uma maior compatibilidade entre sistemas e uma maior facilidade de uso. Além disso, uma série de funcionalidades para a edição de seqüências de pacotes também é disponibilizada, como: criação, inserção e exclusão de pacotes, figura 2 (1).

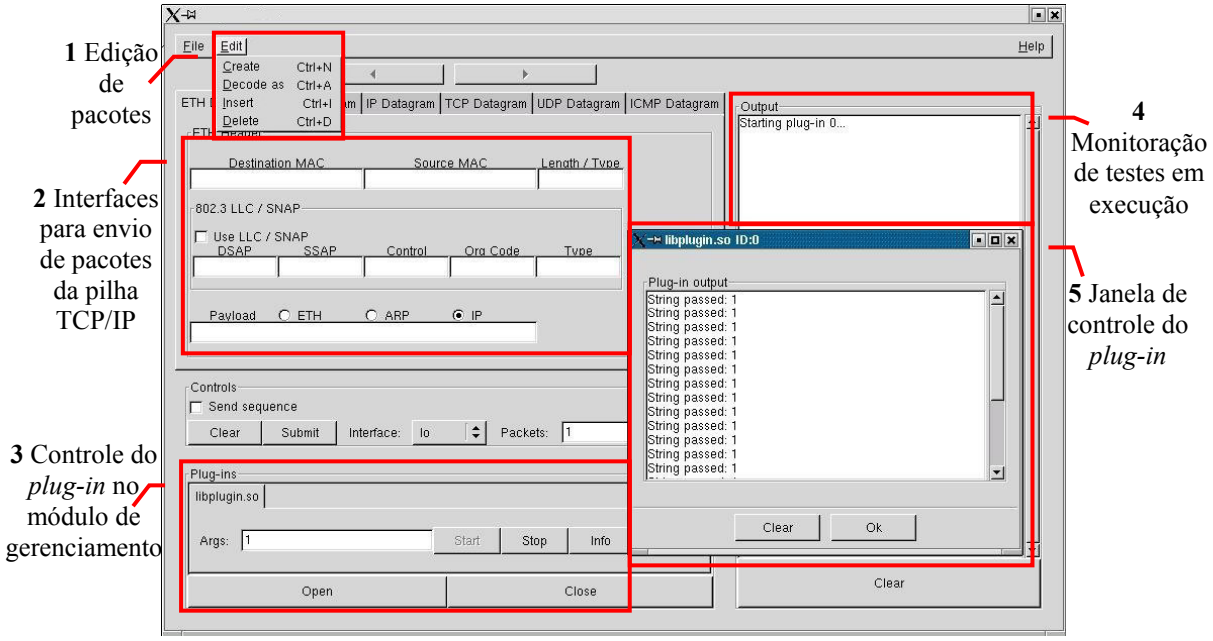

Figura 2. Interface do módulo de gerência

Através do módulo de gerenciamento são realizados também o controle e a monitoração dos testes em execução pelo usuário, seja através do próprio módulo de gerência, seja pela utilização de *plug-ins*, como pode ser visto na figura 2 (3). O módulo de injeção e monitoração fornece as funções básicas para injeção de pacotes e monitoração da rede, bem como funções mais sofisticadas para testes de equipamentos de segurança. Com o intuito de alcançar esse objetivo, esse módulo foi implementado em linguagem ANSI/C na forma de uma biblioteca a fim de ser facilmente utilizado tanto pelo módulo de gerenciamento como pelos módulos de teste desenvolvidos. O módulo de injeção e monitoração fornece atualmente uma API que permite realizar quatro tipos de operações: criar pacotes da pilha TCP/IP; enviar pacotes; monitorar pacotes na rede; enviar mensagens para o módulo de gerenciamento.

A API disponibilizada permite a criação de qualquer tipo de pacote, contudo fornece uma maior facilidade para a criação de pacotes *ethernet*, ARP, IP, TCP, UDP e ICMP. Existem também quatro funções para o envio de pacotes, visando atingir diferentes níveis de detalhes são elas: sendip, sendarp, sendeth e sendfile. Assim a função a ser utilizada vai depender do nível de detalhe especificado em um pacote. Há também a opção de enviar pacotes ou seqüência de pacotes previamente salvos. Essa função simplesmente abre um arquivo no formato Libpcap e envia todos os pacotes existentes no mesmo.

Além das funções de criação e envio de pacotes existem mais duas classes de funções fornecidas pela API do módulo de injeção e monitoração que são uma função para monitoração de pacotes da rede e uma função para comunicação entre os módulos de testes e o módulo de gerenciamento. A função de monitoração é implementada

utilizando funções de captura da biblioteca Libpcap, fornecendo uma abstração de mais alto nível para o usuário da ferramenta. Esta abstração permite que o usuário não se preocupe com vários detalhes de implementação, no caso específico, o usuário apenas necessita chamar uma única função, enquanto que seria necessária a chamada de várias funções da Libpcap. Esta função é bloqueante e fica escutando a rede a procura de pacotes que satisfaçam o filtro especificado pelo usuário. Quando um pacote preenche os requisitos definidos, o mesmo é retornado ao usuário para verificação ou tratamento.

A função de comunicação, por sua vez, permite que o usuário da ferramenta envie mensagens a serem exibidas pelo módulo de gerenciamento, sejam elas para informar sobre a execução do módulo de teste ou sobre a ocorrência de eventos importantes. Ao ser chamada esta função escreve a mensagem recebida em um canal de comunicação criado entre o módulo de gerenciamento e o módulo de injeção e monitoração.

O módulo de teste é a parte mais importante do sistema, pois representa a inteligência dos testes sendo realizados. Sem a existência dos *plug-ins* o sistema não seria muito diferente de um injetor de pacotes, por isso é importante que a linguagem na qual ele seja desenvolvido seja simples e poderosa. Assim, a fim de se facilitar o uso e o desenvolvimento de testes de segurança, optou-se pelo uso da linguagem ANSI/C, ao invés de desenvolver-se uma linguagem própria. Desta forma, não é necessário que o sistema implemente algum tipo de linguagem para o desenvolvimento de teste, o que torna a ferramenta menos complexa, e permite que testes sofisticados possam ser desenvolvidos utilizando-se de toda a funcionalidade do ANSI/C padrão. Os *plug-ins* devem ser disponibilizados na forma de uma biblioteca dinâmica a ser carregada pela ferramenta, permitindo que o módulo de gerenciamento os carregue e execute como módulos autônomos do sistema. Para tanto, algumas funções padrões devem ser definidas nos módulos de teste para serem chamadas pelo módulo de gerenciamento no momento de sua execução: _start, _stop e _info.

A função _start é chamada pelo módulo de gerenciamento ao iniciar a execução de um módulo de teste. São passados como parâmetros um identificador para o módulo de teste e uma string de valores passados pelo usuário, figura 2 (3). Uma *thread* é criada especialmente para a chamada desta função, desta maneira o *plug-in* do usuário pode executar em paralelo ao módulo de gerenciamento permitindo também que mais de um teste seja executado simultaneamente. A função _stop, por sua vez, informa ao módulo de teste em execução que o mesmo deve ser encerrado. Neste caso, o *plug-in* deve tomar as atitudes necessárias para a correta finalização dos testes em andamento. E a última função que um teste deve implementar é a de informação (_info) que consiste em fornecer um resumo explicando o teste que o *plug-in* implementa.

Na figura 2 (5), é possível visualizar a execução de um simples módulo de teste. Esse módulo simplesmente recebe uma string como parâmetro de entrada e a escreve na janela de controle do teste. Ao inicializar um módulo de teste, é criada uma janela de controle onde são mostradas as saídas geradas pelo módulo, caso esta janela seja fechada, todas as saídas do módulo são redirecionadas para a janela de monitoração do módulo de gerenciamento, figura 2 (4). Os módulos de teste são independentes do módulo de gerenciamento, ou seja, o módulo de gerenciamento não possui controle sobre os testes em execução e sobre a correta implementação dos mesmos. A única forma de controle existente é sobre as *threads* criadas para a execução dos testes, neste

caso a única atitude possível é cancelar alguma *thread* criada, e conseqüentemente finalizar a execução de algum teste em andamento de forma abrupta.

Através da utilização de *plug-ins* o sistema de injeção de falhas torna-se bastante expansível e com funcionalidades similares a de um *scanner* de vulnerabilidade. O que torna a ferramenta ideal para a avaliação de sistemas de segurança em rede, visto que seus testes têm a granularidade de um injetor de pacotes com as características e funcionalidades de um *scanner* de vulnerabilidades.

6. Conclusão

Atualmente, a preocupação com segurança aumentou, visto que inúmeros dados sigilosos circulam pela rede, e novas tecnologias foram criadas para sanar as dificuldades enfrentadas pela Internet, tentando evitar possíveis invasões e ataques. O correto projeto e funcionamento desses novos sistemas é fundamental para a segurança de uma rede. Por isso, técnicas e ferramentas específicas para a avaliação de equipamentos de segurança em rede são extremamente importantes.

Este trabalho apresentou uma ferramenta específica para injetar falhas em sistemas de segurança em rede a fim de testá-los quanto a sua eficiência e o seu correto funcionamento. O injetor de falhas em sistemas de segurança em rede foi projetado com o objetivo de possuir uma granularidade de ataque pequena injetando ataques no nível de rede, similar aos injetores de pacotes, podendo assim testar problemas em protocolos e sistemas utilizados na rede. E, tendo recursos e funcionalidades similares aos programas de varredura de vulnerabilidades através da utilização de *plug-ins*, permitir que diversos tipos de ataque mais sofisticados possam ser realizados.

Referências

Bace, R. (1999) “An Introduction to Intrusion Detection & Assessment”, Scotts Valley: Infidel.

Chapman, D. B., Zwicky, E. D., Building Internet Firewalls. 2nd ed. O’Reilly, 2000.

Deraison, R. (2000) “Nessus Security Scanner”, http://www.nessus.org/, Dez. 2002.

Grimes, M (2000) “Nemesis Packet Injection” , http://www.packetfactory.net/Projects/nemesis/, Dez. 2002.

Hsueh, M. et al. (1997) “Fault Injection Techniques and Tools”, Computer, v. 30, n. 4, p.75-82.

McHugh, J., Christie, A., Allen, J. (2000) “Defending Yourself: The Role of Intrusion Detection Systems”, IEEE Software Computer, v. 17, n. 5, p. 42-51.

Ricketts M. (200) “SendIP”, http:// www.earth.li/projectpurple/progs/sendip.html, Dez. 2002.

Sanfilippo, S. (2001) “Hping”, http://www.hping.org, Dez. 2002.

Todd, B. (1999) “Security Auditor's Research Assistant”, http://www-arc.com/sara/index.shtml, Dez. 2002.

Wack, J.; Tracey, M. (2002) “Guideline on Network Security Testing (DRAFT)”, National Institute of Standards and Technology Special Publication 800-42.

Wanner, P. (2001) “Técnicas de Injeção de Falhas e Avaliação de Segurança de Rede”, Trabalho Individual - Instituto de Informática, UFRGS, Porto Alegre.

WORLD WIDE DIGITAL SECURITY, INC. (2000) “Security Administrator`s Integrated Network Tool”, http://www.wwdsi.com/saint/, Dez. 2002.