UNIVERSIDADE FEDERAL DE SANTA CATARINA

DESENVOLVIMENTO DE UM GUIA PARA TESTES DE PENETRAÇÃO ATRAVÉS DE EXEMPLOS PRÁTICOS

JEAN PACHER

Florianópolis - SC

2014/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE UM GUIA PARA TESTES DE PENETRAÇÃO ATRAVÉS DE EXEMPLOS PRÁTICOS

Jean Pacher

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Florianópolis - SC

2014/2

Jean Pacher

DESENVOLVIMENTO DE UM GUIA PARA TESTES DE PENETRAÇÃO ATRAVÉS DE EXEMPLOS PRÁTICOS

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Orientadora: Prof. Dr. Carla Merkle Westphall

Banca Examinadora

Prof. Dr. João Bosco Mangueira Sobral

Me. Jorge Werner

Sumário

Lista de Figuras

Resumo

Em um cenário global onde o número de crimes cibernéticos crescem de maneira exponencial, apenas medidas de segurança tradicionais como o uso de anti-vírus, firewalls e outros sistemas de segurança não é o bastante. Testes de penetração são o complemento das medidas de segurança tradicionais, com eles uma organização consegue encontrar vulnerabilidades em suas defesas pela mesma ótica de um atacante. Esse trabalho apresenta os principais conceitos e ferramentas para testes de penetração através de exemplos práticos e mostra as fases de um processo de auditoria, desde o reconhecimento, até a tentativa de exploração, que corresponde a um teste da segurança da rede e de sistemas.

Palavras Chave: Teste de penetração, Segurança, OpenVAS, Nmap, Metasploit, Vulnerabilidade, *Exploits*.

Abstract

In a global scenery where the number of cyber crimes is rising exponentially, the simple use of traditional security measures like antiviruses, firewalls and other security systems is not enough. Penetration testing is the the complement of these traditional measures, with penetration testing an organization is able to find vulnerabilities on its defenses through the optics of an attacker. This work presents the main concepts and tools used in penetration testing through practical examples and also the phases involved in the auditing process, from reconnaissance to exploiting which corresponds to a security test for networks and other systems.

Keywords: Penetration Testing, Security, OpenVAS, Nmap, Metasploit, Vulnerabilities, Exploits.

1 Introdução

1.1 Motivação

Atualmente as organizações têm necessidade de proteger os seus sistemas computacionais já que o número de queixas de crimes cibernéticos aumentou mais do que 15 vezes desde o ano 2000 até o ano de 2013 (INTERNET CRIME COMPLAINT CENTER, 2013) e o custo estimado desse tipo de crime no Brasil foi de BRL 15,9 bilhões em 2012 e USD 388 bilhões no mundo (SYMANTEC, 2012).

Bechtsoudis e Sklavos (2012) argumentam que muitos administradores de rede acreditam estar protegidos uma vez que seus sistemas possuem softwares e hardwares atualizados, entretanto isso não basta uma vez que erros de configuração e falhas de projeto podem estar presentes. Nesse contexto encontram-se os testes de penetração, que são atividades na qual a segurança de um sistema computacional é examinada. O método em que a segurança é examinada durante um teste de penetração é similar ao método que um *black hat hacker* empregaria para quebrar a segurança de um sistema (NORTHCUTT ET AL., 2006).

A motivação para este trabalho é realizar uma revisão sobre os principais aspectos de um teste de penetração. O trabalho desenvolverá as fases mais importantes de um teste de penetração, as ferramentas utilizadas para se conduzir um teste e conterá exemplos práticos de um teste de penetração. E como resultado, é esperado que esse trabalho sirva como um guia introdutório sobre testes de penetração para outros estudantes da área que se interessem pelo assunto.

1.2 Objetivo Geral

Realizar uma revisão sobre os principais aspectos de um teste de penetração e realizar um teste de penetração real para assim criar um material de referência sobre testes de penetração. Esse material deverá servir como ponto de partida para um iniciante aprender os conceitos básicos, as ferramentas utilizadas e como conduzir um teste de penetração a partir da fase de reconhecimento até a fase de documentação. Este guia pretende ser claro e

objetivo, utilizando exemplos quando possível, para melhor ilustrar os conceitos e o uso das ferramentas apresentadas.

1.3 Objetivos Específicos

Os objetivos específicos desse trabalho são:

- Explicar conceitos básicos de segurança computacional;
- Explicar o conceito de teste de penetração, a maneira como um teste é composto e possíveis modalidades;
- Exibir conjuntos de ferramentas que podem ser utilizados em cada uma das etapas de um teste de penetração;
- Exibir a realização de um teste de penetração tendo como cliente a rede do Departamento de Informática e Estatística da UFSC;
- Explicar o processo de documentação de um teste de penetração.

1.4 Estrutura do Trabalho

Este trabalho está dividido em seis capítulos: (1) Introdução, (2) Conceitos Básicos, (3) Ferramentas Para testes de Penetração, (4) Exemplo Real de um Teste de Penetração, (5) Conclusão e Trabalhos Futuros e (6) Referências.

O capítulo 1 descreve a motivação para o trabalho e apresenta os objetivos do mesmo e a sua estrutura.

O capítulo 2 explica os conceitos básicos referentes à segurança, vulnerabilidades e ataques computacionais. O capítulo 2 também define o que são testes de penetração, suas modalidades e quais são as principais fases de um teste.

O capítulo 3 mostra ferramentas que podem ser utilizadas a cada fase de um teste de penetração e dá alguns exemplos de uso.

O capítulo 4 mostra como foi executado um teste de penetração realizado no Departamento de Informática e Estatística, além de explicar melhor algumas das ferramentas.

O capítulo 5 contém as conclusões do trabalho e sugestões para trabalhos futuros.

O capítulo 6 cobre as referências dos materiais utilizados como embasamento para o texto.

2 Conceitos Básicos

2.1 Segurança

Uma definição clássica para segurança computacional é que ela é a proteção dada a um sistema automatizado com o objetivo de preservar três características: a integridade, disponibilidade e a confidencialidade dos recursos do sistema, sendo que os recursos do sistema incluem *hardware*, *software*, *firmware* e dados (NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, 1995). Das características apresentadas, a integridade prevê que nenhum recurso do sistema é modificado ou removido sem autorização. A disponibilidade por sua vez, garante que o sistema funcione com prontidão e que nenhum recurso seja negado aos usuários com as devidas permissões. Por fim temos a confidencialidade, que assegura a revelação de informações apenas para usuários autenticados e autorizados.

Segundo Stallings (2007), entretanto, a definição de segurança computacional deve estender a anterior e incluir outras duas características, a autenticidade (que assegura a autenticidade tanto de recursos do sistema quanto a dos usuários) e a responsabilidade (prevê que toda ação feita no sistema possa ser rastreada a uma origem única, usuário ou programa).

Nesse trabalho será adotada a definição de segurança dada por Stallings (2007) e será tratado como sistema computacional qualquer sistema que envolva um ou mais computadores.

2.1.1 Problemas de Segurança ou Ameaças

De acordo com o NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (2006), uma ameaça é qualquer circunstância ou evento que possua o potencial de impactar uma organização de maneira negativa ou então o potencial para uma atacante explorar com êxito uma vulnerabilidade de um determinado sistema. Segundo a ISO 27005 (2008) ameaças podem ser classificadas como intencionais (e.g. furto de mídias ou documentos), acidentais (e.g. erro de uso) ou então naturais (e.g. fenômeno climático ou então a simples corrosão de algum componente). Já vulnerabilidade, ainda de acordo com a ISO 27005 (2008), é qualquer fraqueza em um ativo ou grupo de

ativos que possa ser explorada por uma ou mais ameaças. O foco do presente trabalho são testes de penetração, de tal modo que o mesmo abordará exclusivamente ameaças intencionais, ou seja, ataques à segurança.

2.1.2 Ataques à Segurança

Segundo Shirey (2000), um ataque é um conjunto de ações contra a segurança de um sistema. Shirey (2000) define que ataques podem ser classificados como passivos ou ativos. Ataques passivos visam aprender ou fazer uso de informações do sistema sem afetar os recursos do mesmo, já um ataques ativos, são ataques que podem possuir o mesmo objetivo dos ataques passivos, mas que de alguma forma alteram recursos do sistema ou afetam a sua operação. Além disso, também é possível classificá-los como ataques internos, quando são realizados por entidades que detém acesso autorizado ao sistema, mas que utilizam esse acesso de maneira não aprovada, ou então como ataques externos, que são executados por indivíduos não autorizados a usar o sistema.

Complementando a definição dada por Shirey (2000) para ataques passivos, Stallings (2007) diz que ataques passivos têm como natureza a espionagem ou o monitoramento de tráfego. Alguns exemplos de ataques passivos são:

Network Sniffing: segundo Sabeel, Chandrashekar e Rajeev (2002) esse ataque consiste em usar um *sniffer* (software de captura de pacotes) para analisar o tráfego em uma rede. Essa técnica funciona em redes cabeadas onde exista a replicação de pacotes e em redes wireless. Para esse ataque é necessário que a máquina atacante possua uma interface de rede que aceite operar em modo promíscuo (modo em que a interface aceita qualquer pacote que ela recebe, independente do mesmo ser destinado a ela ou não), e que a máquina esteja em um segmento da rede para onde os pacotes do alvo também sejam transmitidos. Com o ataque de *network sniffing*, é possível ter uma idéia geral dos hábitos de navegação do usuário pelos IPs acessados e o número de mensagens trocadas.

Ataques passivos são difíceis de serem detectados uma vez que eles não alteram os dados sendo transmitidos e recebidos pelo alvo. Para Stallings

(2007) a melhor maneira de se lidar com esse tipo de ataque é tentar prevenir que eles tenham sucesso, normalmente isso é feito utilizando uma solução criptográfica, um exemplo seria o uso de SSL (criptografia para comunicação com servidores web) apoiado por uma PKI (Estrutura Pública de Chaves).

Já sobre ataques ativos, esses são ataques que de algum modo alteram recursos de um sistema ou afetam sua operação, podem ser citados:

Man-in-the-middle: é outra forma de interceptação de pacotes. Apesar do objetivo similar à *network sniffing* a técnica do MITM (*man-in-the-midlle*) é bastante diferente. A técnica de ataque MITM é baseada na premissa que o atacante vai conseguir atuar como ambos os pontos de uma comunicação, receptor quando os dados forem enviados ao alvo e remetente quando o alvo enviar dados para fora, exercendo assim o papel de um *proxy* na comunicação (OWASP, 2009). Para que a técnica funcione por completo, é necessário que o MITM consiga burlar também os sistemas de criptografia, como por exemplo, o SSL, fazendo desse um ataque mais sofisticado que a técnica de *network sniffing*.

Injeção de SQL: é uma das formas mais comuns de se atacar uma aplicação web, é também a vulnerabilidade de segurança mais crítica (OWASP, 2014). A injeção de SQL consiste em enviar consultas SQL para a aplicação através dos formulários providos pela mesma. As consultas enviadas através de formulários para uma aplicação só serão executadas se essa aplicação não tratar corretamente os dados enviados a ela (nesse caso não filtrar adequadamente os caracteres de escape SQL). O resultado desse tipo de ataque pode ser variado. Com ele é possível conseguir diversos dados contidos no banco, como credenciais de acesso a aplicação (FRIEDL, 2007), endereço físico (MAC) e o *hostname* do servidor que hospeda o banco de dados (WEBSEC, 2013).

Negação de Serviço: do inglês *denial-of-service*, é um ataque cujo objetivo é fazer que uma aplicação, rede ou sistema computacional fique indisponível para os seus usuários. Uma técnica comum para se realizar esse ataque é sobrecarregar o alvo com comunicação externa de modo que o

mesmo fique ocupado em demasia, e não consiga atender as solicitações legítimas de seus usuários.

2.2 Testes de Penetração

De acordo com Engebretson (2011), um teste de penetração é uma atividade conduzida legalmente e com as devidas autorizações, cujo objetivo é identificar e explorar vulnerabilidades em sistemas computacionais, para que este possa ter a sua segurança ampliada. Testes de penetração também são conhecidos como *pen testing*, *ethical hacking* e *white hat hacking*. Engebretson (2011) diz que é necessário não confundir testes de penetração com avaliação de vulnerabilidades (*vulnerability assessment*), uma vez essa atividade busca apenas encontrar possíveis vulnerabilidades. Testes de penetração, como dito anteriormente, objetivam além de encontrar vulnerabilidades, explorá-las. É importante definir também que o profissional que executa um teste de penetração é conhecido como *white hat hacker* ou então hacker ético e quem tenta invadir sistemas computacionais com motivos maliciosos é nomeado *black hat hacker*. Além da razão que os impulsiona outra grande diferença entre o hacker ético e o *black hat hacker* é a autorização. Hackers éticos só fazem aquilo definido no escopo do contrato e tem autorização para isso, já os *black hats* não limitam seu foco de ação e muito menos possuem autorização para agir.

Engebretson (2011) alerta para a necessidade de usar uma metodologia para realizar testes de penetração, uma vez que é um processo bastante complexo. O uso de uma metodologia ajuda a dividir o teste de penetração em atividades menores e mais gerenciáveis. O autor subdivide os testes de penetração em cinco fases: reconhecimento, varredura, exploração, manutenção de acesso ao sistema e apagando rastros. Para Engebretson (2011) o resultado de cada etapa influencia diretamente o resultado da próxima, e a tendência é que ao longo de um teste de penetração a informação gerada em cada fase fique cada vez mais específica.

Outros autores também recomendam o uso de uma metodologia para executar um teste de penetração. No geral as fases contidas nas metodologias desses autores têm nomes diferentes das fases apresentadas por

Engebretson, e são mais numerosas, porém o conjunto final das atividades é bastante similar. Harper et al. (2011), por exemplo, divide as fases de um teste de penetração em: formação de times, estabelecimento de regras (escopo e legalidade do contrato), varredura passiva (equivalente a fase de reconhecimento descrita por Engebretson (2011)), varredura de portas, enumeração de serviço, coleta de impressões ou *fingerprinting* (essas três últimas fases correspondem à fase de varredura proposta por Engebretson (2011)), escolha de alvos, exploração de vulnerabilidades, ganho de privilégios dentro do sistema (essas três últimas fases são correspondentes a fase de exploração definida por Engebretson (2011)) e documentação do teste.

No escopo desse trabalho serão abordadas apenas as primeiras três fases descritas por Engebretson (2011) (reconhecimento, varredura e exploração) e a fase de documentação proposta por Harper et al. (2011). A fase de manutenção de acesso e remoção de traços não será abordada no escopo desse trabalho, pois o objetivo do mesmo não é realizar um ataque de verdade, e sim um teste de penetração. Uma vez que um teste de penetração é uma operação legítima, não há a necessidade de manter acesso ao sistema computacional do alvo ou de apagar rastros que mostrem que o teste ocorreu.

2.3 Modalidades de Teste

Harper et al. (2011) define que testes de penetração podem ser classificados em três tipos: caixa preta, branca e cinza. A diferença entre um e outro está basicamente relacionada à quantidade de informação disponível para quem realiza o teste. Em testes do tipo caixa preta normalmente a única coisa posta à disposição do hacker ético é o nome da empresa, uma lista de IPs e parâmetros para delimitar o escopo das suas ações, esse tipo de teste é mais demorado e custoso, porém simula de melhor maneira a ação de um grupo de hackers por exemplo. Já testes do tipo caixa branca são o oposto dos de caixa preta, nessa modalidade o hacker ético tem acesso a diagramas de rede, lista de ativos entre outras informações úteis. Harper et al. (2011) alerta que testes de caixa branca são menos realistas, porém eles são válidos quando a disponibilidade de tempo do alvo é pequena e quando o alvo não está disposto a gastar muito. O último tipo de teste é o chamado caixa cinza, essa modalidade é um meio termo entre as duas previamente descritas, nela o

hacker ético começa com a mesma quantidade de informação que começaria em um teste de caixa preta, entretanto à medida que ele avança no teste, algumas informações de acesso são requisitadas ao alvo como forma de acelerar o processo.

2.4 Fases de um Teste de Penetração

2.4.1 Reconhecimento

O propósito da fase de reconhecimento é a aquisição de informações sobre o alvo do teste de penetração, essa fase também é conhecida como a fase de coleta de informações. Para Engebretson (2011) a fase de reconhecimento é a mais importante em um teste de penetração, porque quanto mais tempo é gasto coletando informações a respeito de um alvo, maior é a chance de sucesso nas próximas fases.

A fase de reconhecimento se distingue bem das outras fases porque além de não possuir uma ferramenta que unifique o processo, também não existem regras gerais de como executá-la, isso é, não existe um conjunto fixo de passos nessa fase ou mesmo uma ordem entre os possíveis passos. A fase de reconhecimento não se limita apenas a tecnologia, ela também abrange o aspecto humano como, por exemplo, na forma de engenharia social, devido a isso Engebretson (2011) afirma que um bom coletor de informações tem características de hacker, engenheiro social e detetive.

Devido à amplitude dessa fase, é extremamente importante que o hacker ético não extrapole o escopo definido para o teste de penetração. Caso durante a fase de reconhecimento seja encontrado um sistema vulnerável relacionado ao alvo esse não pode ser explorado se não estiver no escopo do teste. Informações pertinentes a sistemas relacionados ao alvo, porém fora do escopo, podem ser catalogadas para serem repassadas ao cliente depois, entretanto nunca devem ser utilizadas nas fases posteriores, a menos que haja uma reformulação de contrato.

Apesar de não haver regras para a fase de reconhecimento, Engebretson (2011) afirma que para ter sucesso é preciso que quem esteja conduzindo essa fase tenha uma estratégia que englobe tanto técnicas de

reconhecimento ativo quanto passivo. As técnicas de reconhecimento ativo recebem esse nome, pois interagem diretamente com o alvo, como por exemplo, fazer uma cópia do website do alvo ou então executar alguma atividade de engenharia social. Já as técnicas de reconhecimento passivo fazem uso de informações disponíveis na web ou outros meios (e.g. jornais) e não deixam rastros nos recursos do alvo. São exemplos de técnicas de reconhecimento passivo: acessar o website do alvo através do cachê do Google e procurar por notícias referentes ao alvo em outras fontes não pertencentes ao mesmo.

Estratégias de reconhecimento ativo são mais arriscadas que as de reconhecimento passivo por deixar rastros em recursos do alvo, entretanto isso só é um problema se o cliente estipulou que a sua própria equipe de T.I. não está informada a respeito do teste. Esse tipo de cenário é possível, pois o teste de penetração pode servir não apenas para encontrar e explorar vulnerabilidades nos sistemas computacionais do cliente, mas também para testar se a equipe de T.I. do cliente está preparada para perceber e tomar medidas em relação a possíveis ataques.

Ao longo da fase de reconhecimento a quantidade de informação capturada é bastante grande e diversa fazendo com que ela deva ser catalogada de modo que ela esteja em mãos e organizada para as fases posteriores. Normalmente as informações mais comuns em relação ao alvo são os IPs de máquinas conectadas de algum modo a Internet, o conteúdo dos seus websites e outras informações diversas, como por exemplo: o nome dos seus funcionários, tecnologias empregadas por ele, notícias sobre o estado da sua organização, etc.

2.4.2 Varredura

De acordo com Engebretson (2011) a fase de varredura é subdividida em três atividades: checagem de *hosts* vivos, varredura de portas e varredura de vulnerabilidades.

A primeira atividade que é a checagem de *hosts* vivos é bastante simples, nela basta verificar quais máquinas do alvo estão ligadas e são capazes de se comunicar com a partir da rede externa ou das diversas redes internas do alvo. Esse passo serviria para reduzir a lista de IPs que precisam ser checados posteriormente, entretanto como os resultados dessa etapa não são plenamente confiáveis (e.g. alguma máquina poderia estar em manutenção) os outros passos devem ser realizados em todos os hosts encontrados até agora.

A segunda etapa, varredura de portas, tem o objetivo de identificar quais portas estão abertas nos hosts do alvo e quais são os serviços que estão disponíveis nessas portas. Muitos dos serviços mais comuns rodam em portas específicas, por exemplo, servidores HTTP rodam na porta 80, servidores de email usam o SMTP na porta 25. O fato de existir uma padronização no uso de portas facilita bastante para identificar o propósito de cada host que o alvo utiliza.

As portas que serão varridas nos hosts dos alvos podem ser tanto UDP quanto TCP, e as ferramentas para varredura normalmente subdividem a sua operação por esse critério.

O protocolo TCP que é usado em alguns tipos de varredura é um protocolo orientado a conexão, ou seja, quando há uma solicitação haverá uma resposta. Essa característica do TCP faz com que varreduras de portas que utilizam esse protocolo sejam no geral mais frutíferas do que as que utilizam o protocolo UDP que não é orientado a conexão.

Uma conexão TCP é iniciada com um “aperto de mão” conhecido como *Three-Way Handshake*. Brevemente, o aperto de mão TCP pode ser descrito da seguinte maneira: existe uma máquina esperando por conexões (isso é feito executando as primitivas LISTEN e ACCEPT) e quando outra máquina deseja

se comunicar com ela, esta executa a primitiva CONNECT que envia para aquela um segmento TCP com o bit SYN ligado. Se a máquina que estava esperando por conexões aceitar a solicitação de conexão, ela responde com um segmento com os bits SYN e ACK ligados. Por fim a máquina que solicitou a conexão responde com um segmento com o bit *ACK* ligado (TANENBAUM, WETHERALL, 2011).

Normalmente varreduras de TCP não completam a conexão, nesses casos quando a máquina requisitada responde com o segmento com os bits SYN e ACK ligados, a máquina requisitante envia um segmento com o bit RST ligado, o que faz com que a conexão não seja estabelecida. O propósito de se fazer varreduras dessa maneira, é o de evitar que a máquina requisitada sofra negação de serviço ou até mesmo trave, e também de diminuir o tempo para se completar uma varredura. Varreduras SYN são mais rápidas porque o comportamento padrão em TCP é que depois de estabelecida uma conexão é normal iniciar uma sessão, e isso não ocorre em uma varredura SYN. Além dos fatores apresentados anteriormente, parte dos sistemas de logs não registram conexões que não foram completadas, o que acrescenta a esse tipo de varredura a possibilidade de passar despercebida (MESSER, 2005).

Existem três outras maneiras de se fazer uma varredura TCP, elas utilizam pacotes especialmente feitos para serem anomalias que não deveriam ocorrer, são elas as varreduras do tipo Xmas Tree, FIN e Null. Essas varreduras enviam apenas um pacote por porta do alvo e seu objetivo é descobrir se uma porta está “aberta | filtrada”, fechada ou apenas filtrada. A base dessas varreduras são os pacotes não usuais que elas utilizam, os sistemas operacionais que implementam o protocolo TCP seguindo o RFC 793 retornam um pacote RST quando recebem esse tipo de pacote em uma porta fechada. Novamente esse tipo de varredura só funciona em sistemas que implementam o TCP de acordo com a RFC 793 o que limita a sua funcionalidade a sistemas do similares ao UNIX (MESSER, 2005).

Já o protocolo UDP, que também é utilizado para varreduras, não é um protocolo orientado a conexão, sendo assim os serviços que utilizam esse protocolo não respondem quando recebem um pacote qualquer, e quando os

pacotes chegam a uma porta fechada nem sempre é obtido um pacote ICMP de resposta. Isso acontece porque muitos sistemas operacionais têm buffers limitados para pacote ICMP, fazendo com que parte das requisições para portas fechadas, (caso sejam muitas em um tempo limitado), sejam simplesmente rejeitadas. Esse comportamento faz com que o uma varredura completa UDP seja bastante demorada.

Entretanto Engebretson (2011) alerta para a importância de varreduras UDP para que o teste de penetração não seja incompleto. O autor atenta para a existência de diversos protocolos importantes que utilizam UDP, como por exemplo, SNMP, DNS, DHCP e TFTP.

A última atividade da fase de varredura, é a busca por vulnerabilidades nos serviços disponíveis nas máquinas do alvo. O objetivo de se procurar por vulnerabilidades é que elas são possíveis pontos de acesso ao sistema se exploradas corretamente. Muitas vulnerabilidades conhecidas possuem exploits prontos para uso em diversas ferramentas, como por exemplo, o Metasploit (ENGEBRETSON, 2011).

2.4.3 Exploração

De acordo com Engebretson (2011), a etapa de exploração tem como objetivo tomar o controle de um sistema, para o autor esse processo pode ter várias formas, porém de maneira geral ele define como ter acesso administrativo ao(s) computador(es) do alvo. O autor também define exploração como o processo de lançar um *exploit* e ter sucesso ao fazê-lo. Um *exploit* é um software feito para explorar uma vulnerabilidade.

2.4.4 Documentação

A última etapa de um teste de penetração é a documentação dos resultados encontrados. Para Engebretson (2011) um teste de penetração é composto por 3 documentos, um sumário executivo, um relatório detalhado e os dados extraídos pelas ferramentas durante a execução do teste de penetração. Já Harper et al. (2011) sugere um relatório único contendo um índice, sumário executivo, metodologia usada, relatório detalhado do que foi encontrado e outras informações pertinentes. Apesar de Harper et al. (2011)

propor um relatório único o seu conteúdo é bastante similar ao dos 3 relatórios propostos por Engebretson (2011), porém um conjunto de relatórios voltados para cada uma das áreas da organização cliente parece mais interessante, sendo assim o modelo proposto por Engebretson (2011) foi adotado na execução desse trabalho.

Mais especificamente o conteúdo de cada um dos relatórios propostos por Engebretson (2011) é o seguinte:

2.4.4.1 Sumário Executivo

Esse relatório tem como propósito explicar de maneira geral o resultado dos testes e a gravidade dos problemas encontrados para os executivos da organização cliente. De acordo com o autor ele não deve exceder duas páginas e o seu conteúdo não deve conter terminologias muito tecnológicas, uma vez que os executivos de uma organização muitas vezes não estão familiarizados com eles.

Se vulnerabilidades e *exploits* foram encontrados durante a realização do teste, então este relatório deve focar em explicar como essas vulnerabilidades e *exploits* podem afetar a organização.

Por fim um sumário executivo deve conter *links* para o relatório detalhado.

2.4.4.2 Relatório Detalhado

O relatório detalhado deve conter uma lista de todos os achados (informações chaves que foram encontradas na fase de reconhecimento, vulnerabilidades dos sistemas, *exploits*, etc.). De acordo com o autor, o público alvo do relatório detalhado é composto por: gerentes de tecnologia da informação, administradores de rede e qualquer indivíduo da organização cliente que possua o conhecimento técnico e tenha permissão para ler o relatório. O objetivo do relatório detalhado é ajudar a equipe técnica do cliente a remediar os problemas encontrados.

Engebretson (2011) ainda diz que é muito importante que o relatório técnico seja direto e honesto, e que coloque ênfase nos problemas de

segurança mais graves. O autor adverte que muitas vezes quem conduz um teste de penetração pode ficar tentado a dar foco em seu relatório aos hosts que ele conseguiu comprometer, mesmo que esses sejam menos importantes. O autor cita um exemplo de um teste de penetração que conseguiu acesso completo a uma máquina que não possui informações importantes e a partir da qual não é possível atacar o resto da organização, e ao mesmo tempo o teste também descobre uma vulnerabilidade severa em um dos roteadores que conecta toda a rede do alvo, porém não foi possível explorar essa vulnerabilidade. Nesse caso apesar das duas coisas deverem ser reportadas, o que mais merece destaque, não é comprometimento da máquina, e sim a vulnerabilidade no roteador, uma vez que a partir dele é possível comprometer o resto da rede.

O autor também alerta que caso uma vulnerabilidade que já foi explorada em outros testes seja encontrada e não explorada, não é possível dizer para o cliente que ela foi explorada com sucesso.

Por fim o autor recomenda que sempre que possível é interessante escrever no relatório detalhado, soluções para mitigar os problemas encontrados.

2.4.4.3 Dados Extraídos de Ferramentas

Como o título sugere, esse relatório é composto pelo conjunto de dados retirados das ferramentas usadas no teste. É comum entregar aqui a saída de ferramentas como o Nmap, Nessus e OpenVAS.

3 Ferramentas para Testes de Penetração

Se atualmente testes de penetração não requerem um alto orçamento, isso é porque existe uma vasta gama de excelentes ferramentas gratuitas para realizar tal tarefa. De acordo com Espenschied (2008), algumas das ferramentas mais efetivas usadas por consultores de segurança, indústria e por governos são gratuitas. Nesse capítulo serão mostradas algumas dessas ferramentas e como elas se encaixam nas principais fases de um teste de penetração.

3.1 Reconhecimento

3.1.1 Website Downloaders

Uma das atividades na fase de reconhecimento é a de análise do *website* do alvo. Diversas informações pertinentes, como por exemplo, nome de funcionários e notícias sobre os últimos acontecimentos, podem ser encontradas no *website* do alvo. Uma vez que o acesso ao *website* do alvo é uma atividade que deixa rastros, pode ser interessante tê-lo disponível off-line fazendo apenas “uma visita” ao mesmo. Algumas ferramentas como o Wget e o HTTrack são capazes de baixar um *website* inteiro e deixá-lo organizado para navegação off-line.

Publicado pela primeira vez em 1996, o GNU Wget é programa escrito em C e facilmente portável para vários sistemas operacionais, atualmente se encontra na versão 1.15. O Wget original funciona por linha de comando e é capaz de fazer transferência de arquivos através de diversos protocolos, entre eles HTTP, HTTPS e FTP, além disso, o programa é não interativo (i.e. o usuário não precisa interferir na sua operação uma vez que o programa é iniciado) e não precisa bloquear um terminal enquanto executa.

Similar ao Wget, porém com menor suporte a algumas funcionalidades, o HTTrack é um excelente cliente para download de websites, por padrão ele possui builds gráficas para Windows, Solaris, Linux e outros sistemas do tipo UNIX. O HTTrack também possui *wizards* para auxiliar os usuários menos experientes e diferente do Wget possui suporte a conexões múltiplas.

3.1.2 Motores de Busca Google e Versões Anteriores de Websites

Mesmo que quem realize o teste de penetração possua o *website* do alvo salvo off-line, é possível (e provável) que essa pessoa possa ter deixado de lado alguma informação, ou melhor, não tenha encontrado tal informação em sua cópia. O motor de busca Google entra em cena aqui, pois ele possui cópias das páginas que indexa. Normalmente quando uma página deixe de existir, a cópia feita pelo Google ainda irá persistir por algum tempo.

O valor da cópia feita pelo Google (que é conhecida como cachê) é encontrado em situações, como por exemplo, a de o alvo ter publicado alguma informação que não devia. Nesse caso se o Google indexou a página a informação que o usuário não deveria ter publicado agora faz parte do cachê. Outro aspecto positivo do cachê é que quando ele é acessado, nem sequer um pacote chega aos servidores do alvo.

Existe outra ferramenta online que serve para o propósito de visitar versões anteriores de *websites*, conhecida como Way Back Machine, essa ferramenta criada pela organização sem fins lucrativos Internet Archive, possui 411 bilhões de páginas salvas em seus arquivos. Diferente do cachê do Google, a Way Back Machine possui mais de uma cópia para a mesma página, um exemplo é a página do próprio Google, que já foi salva mais do que 34 mil vezes.

Além de servir como cachê da Internet, o motor de busca Google oferece poderosas diretivas de busca, com elas é possível refinar os resultados de uma pesquisa a apenas um *website*, ou então buscar páginas pelo seu título ou conteúdo da URL que leva até elas. Esse poder de expressão é útil, pois possibilita realizar pesquisas de granularidade alta. Servidores web, ou então aplicativos que rodam sobre eles, normalmente possuem páginas padrão (como a página “It works!” do servidor Apache ou então a página de *login* da interface administrativa de bancos de dado MySQL phpMyAdmin). Encontrar uma página desse tipo é um sinal de que possivelmente os administradores do sistema alvo estão deixando os serviços instalados com algumas configurações padrões.

Como prova prática, suponha que um hacker ético saiba que todos os membros da organização alvo possuam um espaço para colocar sua própria página web em uma máquina Linux rodando o servidor Apache. Uma maneira de disponibilizar esse espaço para os usuário é usando o módulo *userdir* do servidor Apache. Esse módulo faz com que o conteúdo salvo na pasta /home/NOMEDOUSUARIO/public_html fique disponível para acesso no endereço http://dominio_alvo/~NOMEDOUSUARIO/. Agora suponha que esse hacker precise ter acesso a uma lista (mesmo que não completa) de *logins* dos membros dessa organização, uma consulta ao Google usando os termos **intitle:"Index of /" site:inf.ufsc.br** retornaria uma lista de links que contém *logins* de usuários válidos no sistema alvo.

Na consulta o termo **intitle: "Index of /"** especifica o buscador a trazer como resultado apenas as páginas que tenham no título a string **“Index of /”**, essa página é a padrão gerada pelo apache quando o diretório do usuário não possui nenhum arquivo *index* (que é um arquivo de indexa o conteúdo do diretório). Essa *string* de busca foi escolhida, pois basta um diretório sem o arquivo *index* para que a página *index* do Apache seja indexada pelo Google. Já a segunda parte da consulta especifica que os resultados devem pertencer apenas ao site inf.ufsc.br, que nesse caso serviu de alvo.

Essa consulta teve 6.590 resultados, é claro que muitos usuários foram repetidos neles, porém bastaria um script para gerar uma lista única de usuários do sistema. Como resultado não esperado nessa busca, também foi descoberta qual a versão do servidor Apache, o sistema operacional no qual ele é executado, a versão do PHP disponível no servidor e uma lista com alguns dos módulos habilitados.

3.1.3 Email

Endereços de email pertencentes ao domínio do alvo são úteis, pois é comum que organizações utilizem a primeira parte do email como *login* de acesso ao sistema, e.g. o Departamento de Informática e Estatística da UFSC.

Uma das maneiras de se localizar os endereços de email pertencentes ao domínio do alvo é o uso da ferramenta The Harvester (KATHYAT, 2013). Essa ferramenta escrita por Christian Martorella é capaz de buscar endereços

de email pertencentes ao domínio do alvo em motores de busca como Google e o Bing e bases de chave PGP. Além de ser um coletor de emails o The Harvester também tenta encontrar subdomínios, hosts e até mesmo *banners* (arquivo que comprova o acesso a uma máquina, e.g. o título exibido pelo SSH no início de uma sessão) e portas abertas através da base de dados do SHODAN. O SHODAN é um motor de busca de equipamentos conectados a Internet, permite encontrar máquinas por localização, IP, latitude, longitude, entre outros (SHODAN, 2014).

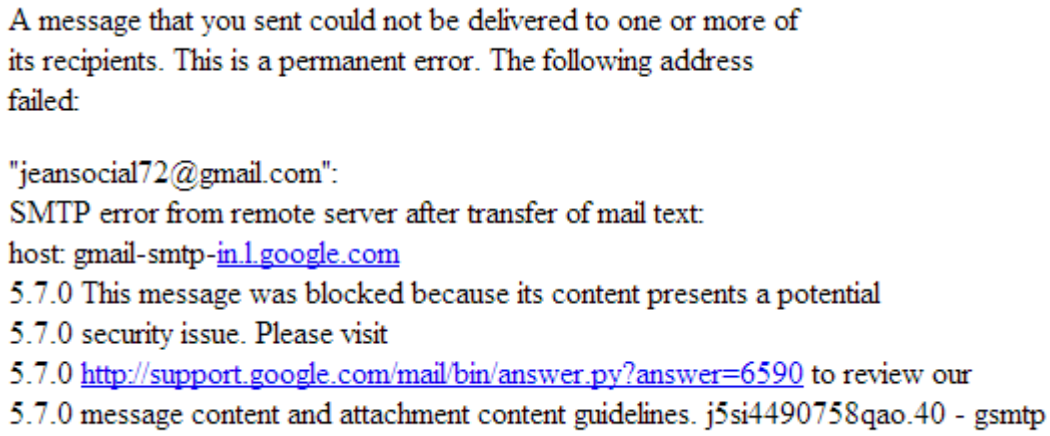
Para se buscar emails associados ao domínio inf.ufsc.br por exemplo, basta iniciar a ferramenta com o comando **./theharvester -d inf.ufsc.br -b google -l 100** . Esse comando vai fazer com que a ferramenta realize uma busca por resultados associados ao subdomínio inf.ufsc.br, no motor de busca Google, e essa busca está limitada a cem resultados. O resultado parcial da execução do programa pode ser visto na Figura 1.

FIGURA 1: EXEMPLO DE USO DA FERRAMENTA THE HARVESTER

A partir de um dos endereços de email do alvo é possível explorarmos outro aspecto referente ao correio eletrônico, o servidor de emails. Na fase de reconhecimento um hacker ético pode explorar servidores de e-mail através de, por exemplo, envio de emails com anexos de conteúdo duvidoso, como arquivos com as extensões “bat” e “exe”. Emails com anexos suspeitos são enviados com um único objetivo: o de serem rejeitados, para que o servidor de emails do retorne uma mensagem de erro. Segundo Engebretson (2011), em

muitos casos o corpo dessa mensagem de erro irá incluir um texto padrão falando que o servidor não aceitou a mensagem explicando o motivo. No exemplo dado, onde há o envio de e-mails com conteúdo suspeito, a mensagem de erro também poderá conter o nome do antivírus utilizado para varrer as mensagens recebidas e do seu fabricante, o que é uma informação valiosa para as fases futuras. Além disso, o hacker ético pode extrair informações úteis a partir do cabeçalho da mensagem enviada pelo servidor de email do alvo, detalhes como o IP desse servidor e muito possivelmente a versão e o nome do *software* que o compõe.

Como demonstração desse conceito, foi enviado um email para um endereço do Gmail, essa mensagem possuía como anexo um arquivo de texto plano chamado email_teste cuja extensão foi mudada para bat. O resultado foi o esperado, a mensagem foi recusada pelo servidor do Gmail. A resposta enviada para o remetente é exibida na Figura 2.


A message that you sent could not be delivered to one or more of
its recipients. This is a permanent error. The following address
failed:

"jeansocial72@gmail.com":
SMTP error from remote server after transfer of mail text:
host: gmail-smtp-in.l.google.com
5.7.0 This message was blocked because its content presents a potential
5.7.0 security issue. Please visit
5.7.0 http://support.google.com/mail/bin/answer.py?answer=6590 to review our
5.7.0 message content and attachment content guidelines. j5si4490758qao.40 - gsmtp

FIGURA 2: CORPO DA RESPOSTA AO EMAIL COM ANEXOS SUSPEITOS

Já pelo cabeçalho do email recebido (Figura 3), não é possível saber com qual antivírus o Gmail protege seus servidores de email. Também é possível concluir pelas informações dadas no cabeçalho do email que o email de resposta foi recebido a partir do endereço mout-bounce.gmx.com e ele possui o IP 74.208.4.220, todavia o uso da ferramenta *host* para o mesmo domínio, mostra que o mesmo pode apontar para vários outros IPs, isso quer dizer que existe mais de uma máquina responsável em cuidar da troca de emails para o Gmail.

Received	from mda by mogmxus002.server.lan id 0LhfR1-1WW94y3EVF-00mqFK Wed, 04 Jun 2014 184249 +0200
Received	from mout-bounce.gmx.com ([74.208.4.220]) by mx-ha.gmx.net (mxgmxus003) with ESMTPS (Nemesis)
Date	Wed, 04 Jun 2014 184249 +0200
From	MAILER-DAEMON@mail.gmx.com
To	adamsmithrocks@mail.com
Subject	Mail delivery failed returning message to sender
MIME-Version	1.0
Content-Type	text/plain; charset=utf-8
Content-Transfer-Encoding	8bit
Envelope-To	<adamsmithrocks@mail.com>
X-GMX-Antispam	0 (Mail was not recognized as spam); Detail=V3;
X-GMX-Antivirus	0 (no virus found)

FIGURA 3: CABEÇALHO DA RESPOSTA AO EMAIL COM ANEXOS SUSPEITOS

3.1.4 WHOIS

Segundo Daigle (2004), WHOIS é um protocolo baseado em TCP, no modelo de consulta e resposta e cliente servidor, que é amplamente utilizado para providenciar informações para os usuários da Internet. O serviço permite que seus usuários, e no caso desse trabalho o hacker ético que conduz o teste de penetração, tenha acesso a informações sobre o alvo, incluindo detalhes sobre endereços de IP e *hostnames* dos seus servidores de DNS e também nome e informações de contato do responsável pelo registro. De acordo com Engebretson (2011) é importante examinar o resultado exibido pelo cliente WHOIS após uma consulta, para ver se é possível encontrar o campo "*refer*", pois a URL apresentada nele indica outro servidor WHOIS que tem informações mais relevantes sobre o alvo. Uma consulta a informações sobre o domínio ufsc.br no servido whois.iana.org retorna informações sobre o Comitê Gestor da Internet no Brasil, porém ele também retorna como referência a URL do servidor whois.registro.br, que por sua vez se consultado trará informações sobre o registro da UFSC (Figura 4).

```
root@kali:~# whois inf.ufsc.br

% Copyright (c) Nic.br
%  The use of the data below is only permitted as described in
%  full by the terms of use (http://registro.br/termo/en.html),
%  being prohibited its distribution, comercialization or
%  reproduction, in particular, to use it for advertising or
%  any similar purpose.
%  2014-07-17 08:15:19 (BRT -03:00)

domain:      ufsc.br
owner:       Universidade Federal de Santa Catarina
ownerid:     083.899.526/0001-82
responsible: Edison Tadeu Lopes Melo
country:     BR
owner-c:     ETM
admin-c:     ETM
tech-c:      ETM
billing-c:   ETM
nserver:     adufsc.ufsc.br 150.162.1.69 2801:84:0:1001::69
nsstat:      20140716 AA
nslastaa:    20140716
nserver:     adns1.pop-sc.rnp.br
nsstat:      20140716 AA
nslastaa:    20140716
nserver:     ns67.ufsc.br 150.162.1.67
nsstat:      20140716 AA
nslastaa:    20140716
nserver:     ns1.sc.gov.br
nsstat:      20140716 AA
nslastaa:    20140716
nserver:     adufsc04.ufsc.br 150.162.1.66
nsstat:      20140716 AA
nslastaa:    20140716
nserver:     adns2.pop-sc.rnp.br
nsstat:      20140716 AA
nslastaa:    20140716
created:     before 19950101
changed:     20140213
status:      published

nic-hdl-br:  ETM
person:      Edison Tadeu Lopes Melo
e-mail:      edison.melo@ufsc.br
created:     19971218
changed:     20140213
status:      published

nic-hdl-br:  ETM
person:      Edison Tadeu Lopes Melo
e-mail:      edison.melo@ufsc.br
created:     19971218
changed:     20120229

% Security and mail abuse issues should also be addressed to
% cert.br, http://www.cert.br/, respectivelly to cert@cert.br
% and mail-abuse@cert.br
%
% whois.registro.br accepts only direct match queries. Types
% of queries are: domain (.br), registrant (tax ID), ticket,
% provider, contact handle (ID), CIDR block, IP and ASN.

root@kali:~#
```

FIGURA 4: EXEMPLO DE USO DO CLIENTE WHOIS LINUX

Um utilitário para quando houver a necessidade de converter um *hostname* para endereço IP, como por exemplo, os obtidos se utilizando WHOIS, ou então quando se quer fazer o contrário, converter um IP em um *hostname*, é a ferramenta *host*. Essa ferramenta é embutida nas mais diversas distribuições Linux, é ideal para função. Basta usar o comando *host* "*hostname*" para que haja a conversão. De acordo com o seu manual, *host* é uma simples ferramenta de consulta a servidores DNS. A Figura 5 exibe um exemplo de uso da ferramenta *host*.

```
root@kali:~# host inf.ufsc.br
inf.ufsc.br has address 150.162.60.21
inf.ufsc.br mail is handled by 10 mx2.inf.ufsc.br.
root@kali:~#
```

FIGURA 5: EXEMPLO DE USO DA FERRAMENTA HOST

3.1.5 DNS

Domain Name System ou DNS é um sistema distribuído e hierárquico que serve para atribuir nomes a recursos conectados a uma rede de computadores. Sua principal função é traduzir *hostnames* para endereços IP, além disso, servidores DNS também podem manter registros de diversos outros recursos, como por exemplo, uma lista de outros servidores DNS, o endereço do servidor responsável por fazer a troca de emails para um determinado domínio e até mesmo campos de texto plano que podem ser utilizados para os mais diversos propósitos.

De acordo com Engebretson (2011), é importante explorar os servidores de DNS do alvo. Uma vez que para funcionar adequadamente eles precisam conter a lista de todos os *hosts* nomeados e seus respectivos IPs dentro do sistema, sendo essa uma informação valiosa para se obter durante a fase de reconhecimento. Engebretson (2011) também afirma que em muitos casos os servidores de DNS tendem a ser negligenciados por administradores sistema inexperientes. Esse fato faz com que servidores DNS estejam mais facilmente desatualizados ou mal configurados, o que facilita o trabalho de quem conduz um teste de penetração.

Existem algumas ferramentas para se explorar um servidor de DNS, entre elas estão o NS Lookup e o Dig.

O NS Lookup está disponível em diversas distribuições Linux e também no Windows. Essa ferramenta opera de maneira similar nos diversos sistemas operacionais em que ela está disponível, entretanto pode haver algumas diferenças na utilização das diversas versões. Quando a ferramenta é executada sem parâmetros ela entra em modo interativo e exibe o caractere “>” no console, a partir daqui é possível definir parâmetros como o IP do servidor de DNS que será consultado e o tipo de informação que se busca.

Um exemplo de uso é mostrado na Figura 6, nela, o comando ***> server 150.162.1.69*** faz a definição do servidor DNS que será utilizado como sendo o servidor localizado no endereço 150.162.1.69, o primeiro que foi obtido no exemplo da ferramenta WHOIS na Figura 4. Posteriormente o comando `> set type=any` define que devem ser buscados todos os registros possíveis para o domínio que será especificado em seguida. Os tipos de registro mais comuns em servidores DNS são:

A: endereços de *hosts* do IP do tipo IPv4.

AAAA: endereços de *hosts* de IP do tipo IPv6.

NS: outros servidores DNS.

MX: servidores de troca de email.

Após definido os tipos de registro, o comando `> inf.ufsc.br` define o *hostname* do alvo para que então a ferramenta faça a consulta ao servidor. A Figura 6 também contém o resultado da execução da ferramenta.

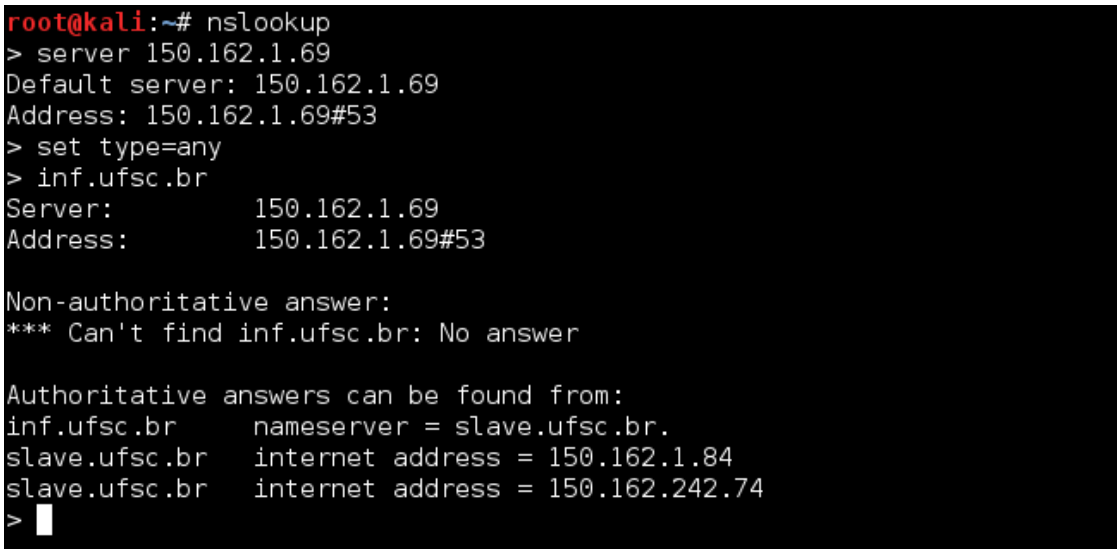


FIGURA 6: EXEMPLO DE USO DA FERRAMENTA NS LOOKUP (A)

O resultado da execução presente na Figura 6 indica que o servidor de DNS que foi escolhido para realizar a busca não tinha informações sobre o *hostname* especificado, porém ele informa qual servidor possui a informação, nesse caso é o servidor slave.ufsc.br. A partir do resultado podemos executar o comando `> Server: 150.162.1.84` para definir o novo servidor de DNS a ser utilizado, após isso basta inserir novamente o endereço inf.ufsc.br para realizar uma nova consulta, o resultado pode ser visto na Figura 7.

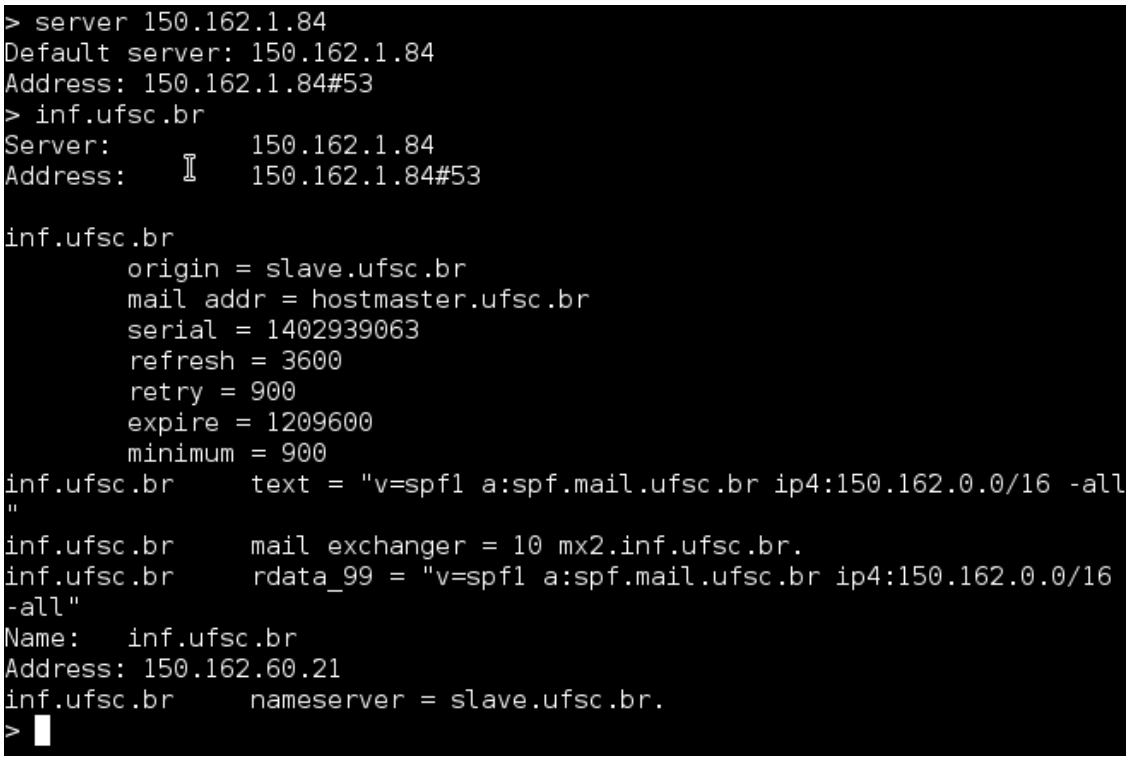


FIGURA 7: EXEMPLO DE USO DA FERRAMENTA NS LOOKUP (B)

A partir dos resultados apresentado na Figura 7 é possível saber que o IP do *host* inf.ufsc.br é 150.162.60.21, o *hostname* da máquina responsável pela troca de emails é mx2.inf.ufsc.br (que com o programa *host* foi descoberto possuir IP o 150.162.2.76), além de detalhes referentes ao servidor de DNS.

Outra ferramenta para explorar servidores de DNS é o Dig, ela funciona de maneira similar ao NS Lookup, porém ela classifica e formata melhor os resultados obtidos nas consultas. Por exemplo, a mesma consulta feita ao servidor 150.162.1.84 pode ser vista sendo efetuada pela ferramenta Dig na Figura 8.

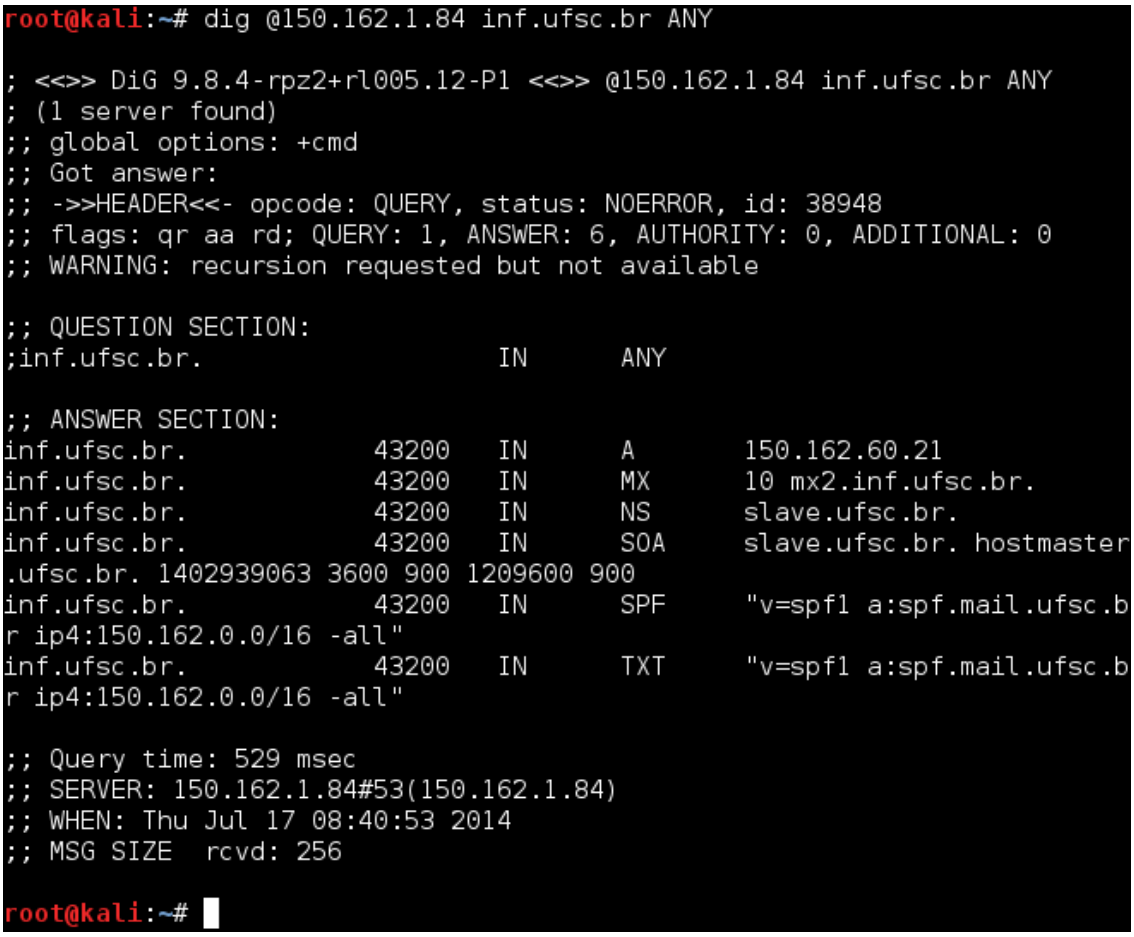


FIGURA 8: EXEMPLO DE USO DA FERRAMENTA DIG

Uma das diferenças na apresentação do resultado é que a ferramenta Dig exibe o tipo dos registros que foram retornados, ele podem ser vistos na “ANSWER SECTION” da Figura 8, representados pelas letras “A”,”MX”,”NS”,etc. em cada um dos resultados, enquanto o NS Lookup não o faz. Com a ferramenta Dig também é possível a realização de transferência de zonas, que É a obtenção de uma cópia fiel do conteúdo de um servidor DNS. Essa transferência seria realizada com o comando

dig @endereço_do_servidor_DNS domínio_do_alvo –t AXFR

Se for possível a transferência de zona, o hacker ético obtém um mapa dos *hosts* do alvo mapeados pelo servidor DNS em questão.

3.2 Varredura

3.2.1 Varredura de Portas Com o Nmap

Uma ferramenta muito utilizada para realização de varredura de portas é o Nmap. Originalmente escrito por Gordon Lyon em 1997 e mantido até os dias de hoje, o Nmap ou *Network Mapper* (OREBAUGH, PINKARD, 2008), é uma ferramenta de código aberto que serve para explorar e realizar auditorias de segurança em redes.

Seu projeto original tinha como objetivo a varredura de grandes redes de computadores, entretanto a ferramenta funciona bem até mesmo quando utilizada apenas em um *host*. A ferramenta funciona basicamente através do envio de pacotes, para com isso descobrir quais *host*s estão disponíveis em uma rede, quais serviços esses *hosts* oferecem a rede, quais sistemas operacionais estão rodando, que tipo de filtro de pacotes ou *firewall* estão sendo utilizados entre outras características (LYON, 2009).

3.2.1.1 Especificando o Alvo

Sempre que executado em linha de comando, tudo que não for uma opção o Nmap irá tratar como um alvo. Como dito anteriormente, o Nmap tinha como objetivo inicial fazer varredura de portas em redes grandes, sendo assim é possível especificar para a ferramenta um bloco de endereços de IP através da notação CIDR, e.g. se especificado como alvo 192.168.1.0/24 todos os 256 endereços de IP possíveis no último octeto serão varridos. Além da notação CIDR a ferramenta também aceita uma notação própria, onde é possível especificar uma faixa de números e/ou valores específicos para cada octeto do endereço de IP, por exemplo, se definido como alvo o valor 10.0.0.1-254, todos os IPs da faixa 10.0.0.1 até 10.0.0.254 serão varridos, enquanto os endereços 10.0.0.0 e 10.0.0.255, respectivamente utilizados como endereço da rede e de *broadcast*, serão ignorados. Outro exemplo é o valor “192.168.1.2-9,34”, cujo resultado seria uma varredura dos endereços 192.169.1.2 até o endereço 192.168.1.9 e excepcionalmente o endereço 192.168.1.34. Para IPv6 entretanto, a ferramenta atualmente suporta apenas a especificação de endereços completos.

Por fim o Nmap também permite que *host*s sejam escolhidos aleatoriamente através da opção **–iR <número de hosts>**, especificados por arquivo de entrada (desde que separados por espaços em branco, tabs ou nova linha) pela opção **–iL <caminho para o arquivo>**, excluídos através da linha de comando com a opção **--exclude <host1>[,<host2>[,...]]** , ou então excluídos da varredura através de um arquivo de entrada, especificado na forma **--exclude-file <caminho para o arquivo>** e formatados da mesma maneira que o arquivo da opção **–iL**.

3.2.1.2 Descoberta de Hosts

Em situações onde o número de IPs a ser varrido é muito grande, e.g. uma varredura de uma rede privada onde o único endereço disponível é o endereço da sub-rede, é interessante reduzir a quantidade de endereços para aqueles que efetivamente estão sendo utilizados. Isso é importante, pois uma varredura de todos os possíveis endereços de uma rede pode ser bastante demorada e desnecessária caso não haja um *host* correspondente ao IP.

Lyon (2009), diz que para um administrador de redes, basta utilizar pacotes ICMP (como no comando *ping*) para descobrir quais *host*s estão vivos, entretanto, em um teste de penetração, a tarefa de descoberta de *host*s vai além, utilizando diversos tipos técnicas de sondagem para, por exemplo, tentar evadir restrições de firewall.

O Nmap oferece diversas opções para customizar como é feita a descoberta de *host*s, que como dito anteriormente pode ir muito além do simples comando *ping* (que pode até mesmo ser totalmente descartado). Se o Nmap for utilizando sem definir opções de varredura, ele irá realiza a descoberta de *host*s enviando a seguinte informações (LYON, 2009):

- Um pacote de sincronização (SYN) TCP para a porta 443.
- Uma requisição de timestamp ICMP.
- Um pacote de reconhecimento (ACK) TCP para a porta 80.
- Uma requisição echo ICMP.

Além disso, se executado localmente, o Nmap também realiza descoberta através do protocolo ARP em redes IPv4 e o protocolo de

descoberta de vizinhos para IPv6. É importante ressaltar que para realizar todas essas requisições de maneira padrão e tantas outras com pacotes mais complexos, a ferramenta precisa estar sendo executada com privilégios de administrador, pois ela necessita de acesso à interface de rede para criar pacotes mais específicos. O Nmap permite especificar o modo qual é realizada a descoberta de *host*s através das opções **–P***.

3.2.1.3 Estados de uma Porta

Lyon (2009) afirma que muitos varredores de porta têm classificado as mesmas apenas como abertas ou fechadas, enquanto o Nmap tem maior granularidade, podendo uma porta estar: aberta, fechada, filtrada, não filtrada, aberta | filtrada ou então fechada | filtrada. Sendo que esses possíveis estados não são realmente inerentes às portas, mas a maneira que a ferramenta as detectou, sendo assim uma questão de perspectiva, e.g. a porta 80 de uma determinada máquina pode estar aberta para um computador pertencente à mesma LAN que ela, e fechada para computadores fora dessa rede.

Os seis estados possíveis para uma porta segundo o Nmap são: aberta, fechada, filtrada, não filtrada, “aberta | filtrada” e “fechada | filtrada”.

Aberta

Uma porta esta aberta se ela aceita conexões TCP, datagramas UDP, ou associações SCTP. Em um teste de penetração o melhor tipo de porta que se pode encontrar, é uma porta aberta, porque cada porta aberta possui por trás algum serviço rodando o que possibilita um novo vetor de ataque.

Fechada

Nesse estado uma porta é dita acessível, uma vez que responde aos pacotes do Nmap, entretanto uma porta fechada não possui nenhuma aplicação rodando e associada a ela. De acordo com Lyon (2009), portas nesse estado são úteis para realizar a detecção de sistemas operacionais e devem ser varridas outra vez posteriormente para checar se elas continuam sem aplicações associadas a elas.

Filtrada

Portas que estão filtradas são portas cujos pacotes de sondagem do Nmap não conseguiram chegar até o *host* de destino. Isso pode acontecer por diversos motivos: os pacotes podem ter sidos bloqueados por uma máquina firewall no caminho até o *host* ou pelo próprio firewall do *host* em questão, por regras de roteamento ou por simples congestionamento de dados. Às vezes os pacotes de sondagem são respondidos com pacotes de ICMP de erro referentes a “destino inalcançável”, mas de acordo como Lyon (2009), esse não é o caso comum, e os pacotes de sondagem são apenas bloqueados sem receberem resposta. O autor ainda diz que esse tipo de porta é bastante “frustrante”, pois quase não provê informação.

Não Filtrada

Esse estado só é dado a uma porta em varreduras do tipo ACK, que são varreduras utilizadas para determinar as regras de um firewall, outros tipos de varredura podem ajudar a melhor determinar qual o estado de uma porta não filtrada. Esse estado indica que uma porta é acessível, porém o Nmap não consegue definir se ela está aberta ou fechada.

Aberta | Filtrada

Nesse estado não é possível determinar se uma porta está aberta ou filtrada. Isso ocorre em varreduras em que as portas não dão nenhuma resposta, pois esse é o seu comportamento padrão ou porque os pacotes de sondagem estão sendo bloqueados. Esse estado é comum em varreduras UDP (como explicado anteriormente) e a varreduras TCP especiais como a FIN e XMas.

Fechada | Filtrada

Estado utilizado quando o Nmap não consegue determinar se uma porta está fechada ou sendo filtrada.

3.2.1.4 Técnicas de Varredura de Portas

Existem diversos tipos de varreduras possíveis com o Nmap, da varredura TCP SYN, até mesmo varreduras TCP customizadas pelo usuário. A ferramenta trabalha normalmente apenas com um tipo de varredura a cada execução, a exceção à regra é que é possível combinar um tipo de varredura TCP com uma de SCTP mais a varredura UDP em uma mesma execução. O Nmap também permite especificar quais portas devem ser varridas através da opção **-p <faixa de portas no formato número-número>**. O padrão da ferramenta quando executada é realizar apenas uma varredura to tipo TCP SYN (que pode ser especificada pela opção **-sS**), outros tipos possíveis de varredura de portas são:

-sT

Varredura do tipo TCP connect, esse tipo de varredura é inferior a TCP SYN, pois leva mais tempo na sua execução e é mais plausível de deixar rastros, pois ela realiza uma conexão TCP completa com cada porta varrida. Lyon (2009) afirma que ela só deve ser utilizada em situações em que não se é possível utilizar uma varredura TCP SYN, esse é o caso quando a ferramenta é executada sem privilégios administrativos em sistemas *NIX.

-sU

Como explicado anteriormente é uma varredura bastante demorada, por isso se o número de portas for delimitado as mais relevantes que utilizam o protocolo UDP, a ferramenta irá varrer essas portas com pacotes específicos para cada serviço comum que roda nelas, entretanto para portas incomuns o pacote enviado é normalmente vazio.

-sY

Conhecida como SCTP INIT, esse tipo de varredura opera com o protocolo SCTP que segundo R. Stewart (2007), é um protocolo confiável que opera sobre uma rede de pacotes como a IP, e funciona de maneira similar a uma TCP SYN, sendo rápida, não obstrutiva para o sistema alvo e capaz de diferenciar as portas entre abertas, fechadas ou filtradas.

-sF, -sN e -sX

Chamadas respectivamente de varreduras FIN, NULL e Xmas, elas funcionam enviando pacotes TCP incomuns: NULL não seta nenhum bit em seu cabeçalho, FIN seta apenas o bit FIN (finalizar conexão) e o Xmas seta os bits FIN, PSH e URG. Esse tipo de varredura só funcionam em sistemas que implementam a RFC 793 (MESSER, 2005) e classificam as portas analisadas em fechadas caso o alvo responda com um pacote RST, filtradas caso a resposta seja um pacote ICMP do tipo 3 (destino inalcançável) ou então aberta | fechada se não houver resposta alguma.

-sA

A varredura TCP ACK, serve para determinar quais são acessíveis ou não, as portas que estão acessíveis devem responder com um pacote RST, entretanto não é possível diferenciar portas abertas e fechadas de modo que elas são marcadas como não filtradas. Portas inacessíveis são aquelas que respondem com um pacote ICMP tipo 3 ou então não respondem, elas são marcadas como filtradas.

-sW

Os pacotes enviados são iguais aos do TCP ACK, porém o TCP Window se diferencia como varredura, pois tenta explorar alguns detalhes de implementação em alguns sistemas específicos, uma vez que alguns deles, o campo Window do pacote RST de resposta tem tamanho zero para portas fechadas e tamanhas positivo para portas abertas, sendo essa a sua vantagem sobre a varredura **-sA**. Em sistemas que não possuem esse detalhe a varredura TCP Window ainda é útil. Em situações onde a maior parte das portas é marcada como fechada, porém portas padrão como a 22 são marcadas como filtradas é possível que essas portas estejam abertas. Em uma situação diferente, onde a grande maioria das portas é detectada como aberta pelo Nmap, e apenas algumas fechadas ou filtradas, é possível que as portas marcadas como fechadas ou filtradas é que sejam as portas que estão verdadeiramente abertas.

--scanflags

Esse tipo de varredura TCP permite especificar quais bits do pacote TCP estarão ligados. É possível especificar esses bits através de valores numéricos ou então com nomes simbólicos, e.g. --scanflags SYNFIN liga os bits SYN e FIN deixando os outros zerados. Além disso, também é possível especificar como os resultados serão interpretados setando o tipo base da varredura. Se a varredura dada como exemplo fosse efetuada com o tipo base **–sA** (varedura TCP ACK), a falta de resposta iria causar uma porta a ser marcada como filtrada.

-sM

Varredura similar as FIN, NULL e Xmas, entretanto implementações que seguem a RFC 793 devem responder com um pacote RST independente de a porta estar aberta ou fechada. Alguns sistemas derivados do BSD bloqueiam a pacote caso a porta esteja aberta (Lyon, 2009).

3.2.1.5 Detectando Versões de Serviços

De maneira padrão o Nmap lista qual o serviço ele detecta estar escutando em portas abertas, entretanto apesar de possuir um banco de dados com aproximadamente 2200 serviços conhecidos que são associados a possíveis portas, a ferramenta nem sempre acerta, além do que os resultados são um pouco superficiais, de modo que não dizem nome nem versão do serviço associado a porta. Por tal motivo esse comportamento de uma varredura padrão não é o mais adequado a um teste de penetração, e a ferramenta nos oferece a possibilidade de fazer uma sondagem mais especifica a cada porta que foi encontrada aberta em uma varredura.

O serviço de detecção de serviços do Nmap possui um banco de dados que contém formatos específicos de requisições de sondagem que servem para tentar determinar o protocolo específico, nome, número de versão, tipo de dispositivo, e para qual sistema operacional o serviço rodando em uma porta aberta foi produzido e quando possível o CPE (*Common Plataform Enumeration*) associado. O CPE é esquema estruturado para nomear sistemas, softwares e pacotes (NATIONAL INSTITUTE OF TECHNOLOGY).

Alguns outros detalhes extras também podem surgir como, por exemplo, a versão do protocolo SSH usado pela máquina ou se a mesma possui um servidor X aberto a conexões.

Portas (UDP e TCP) que na varredura foram marcadas como “abertas | filtradas” também são utilizadas na fase de detecção de versão, algumas vezes com consultas mais específicas é possível ver que o estado verdadeiro dessas, era aberta.

Algumas opções da detecção de versão são:

-sV

Para que a detecção de versão ocorra é necessário que essa opção ou a opção **-A** seja ativada. A diferença é que a opção **-A** além de acionar a detecção de versão também aciona a detecção de sistema operacional e a ferramenta de traceroute (serve para identificar qual o caminho percorrido até o *host* alvo).

--version-intensity <intensidade>

Cada tipo de sondagem feita pelo Nmap para tentar detectar a versão de um serviço recebe um valor associado à quão comum ou rara ela é de acordo com o Nmap. Os valores vão de zero a nove, sendo que quanto maior o valor, mais rara é a sondagem sendo feita e mais dificilmente ela vai funcionar, e quanto menor o valor, mais comum e genérica é o tipo da sondagem e mais comumente ela possui algum resultado. Entretanto quanto mais rara uma sondagem, é mais provável que quando ela obtenha algum resultado, este seja mais preciso.

3.2.1.6 Reconhecimento (Fingerprinting) de Sistemas Operacionais

De acordo com Lyon (2009) um dos recursos mais conhecidos do Nmap é a sua habilidade de fazer remotamente o reconhecimento de sistemas operacionais. A ferramenta realiza isso enviando uma série de pacotes TCP e UDP para o *host* alvo e examina praticamente todos os bits nos pacotes de resposta e com esses resultados realiza uma série de testes, Lyon cita:

amostragem TCP ISN, ordenação de opções TCP, amostragem de IP ID, checagem do tamanho de TCP Window inicial.

Com o resultado dos teste o Nmap compara a impressão (*fingerprint*) gerada com o seu banco de dados de impressões de sistemas operacionais (que contém mais de 2600 itens), se houver uma combinação o sistema operacional é identificado. Cada impressão no banco de dados possui uma descrição do desenvolvedor do sistema, qual o nome do sistema operacional, a geração do mesmo, e o tipo de dispositivo para qual o mesmo foi compila (roteadores, impressoras, computadores, etc.), Lyon também diz que a maior parte dos sistemas operacionais também possui uma representação do CPE. Na fase de reconhecimento de sistema operacional também é possível tentar descobrir qual o *uptime* da máquina alvo.

Para habilitar o reconhecimento de sistemas operacionais basta adicionar a opção -O na linha de comando.

3.2.2 Varredura de Vulnerabilidades com OpenVAS

O *Open Vulnerability Assessment System* (Sistema Aberto de Avaliação de Vulnerabilidade) ou OpenVAS, é um projeto derivado da ferramenta (anteriormente de código aberto) Nessus, cujo conjunto de softwares livres desenvolvidos forma um framework (e WebService) para realização e gerenciamento de soluções para varreduras de vulnerabilidades . A escolha da realização desse trabalho com o OpenVAS e não com o Nessus é baseada no fato de que o OpenVAS é uma ferramenta de código aberto e gratuita, fazendo com que seja ideal para um iniciante no assunto (o público alvo desse trabalho). Ela é ideal para um iniciante pois o mesmo não vai precisar ter um custo inicial alto com um varredor de vulnerabilidades, e caso seja do seu interesse ele pode até mesmo alterar o modo que a ferramenta funciona (código aberto), possibilitando que o mesmo aprofunde o seu conhecimento, uma situação oposta a qual ocorreria com um produto fechado como o Nessus.

3.2.2.1 Arquitetura do Sistema

A principal parte do projeto é openVAS Scanner, que é quem faz a varredura no OpenVas, e quem executa os testes de vulnerabilidade conhecidos como NVTs (*Network Vulnerability Tests*), os quais são atualizados diariamente. NVTs são os testes que a ferramenta executa para determinar se um sistema possui ou não uma vulnerabilidade. A arquitetura do OpenVAS é apresentada na Figura 9 e explicada no texto a seguir.

O openVAS Scanner é seguido do gerenciador, que é o serviço central que controla o *scanner* e centraliza toda a inteligência, de modo que todos os clientes para o opensVAS podem ser implementados de maneira consistente em relação ao modo como filtram e organizam os resultados das varreduras de vulnerabilidades. O gerenciador também controlar um banco de dados sqlite que contém todas as configurações do sistema, resultados de varreduras, lista de alvos e varreduras cadastradas, informações de usuários cadastrados no sistema, etc.

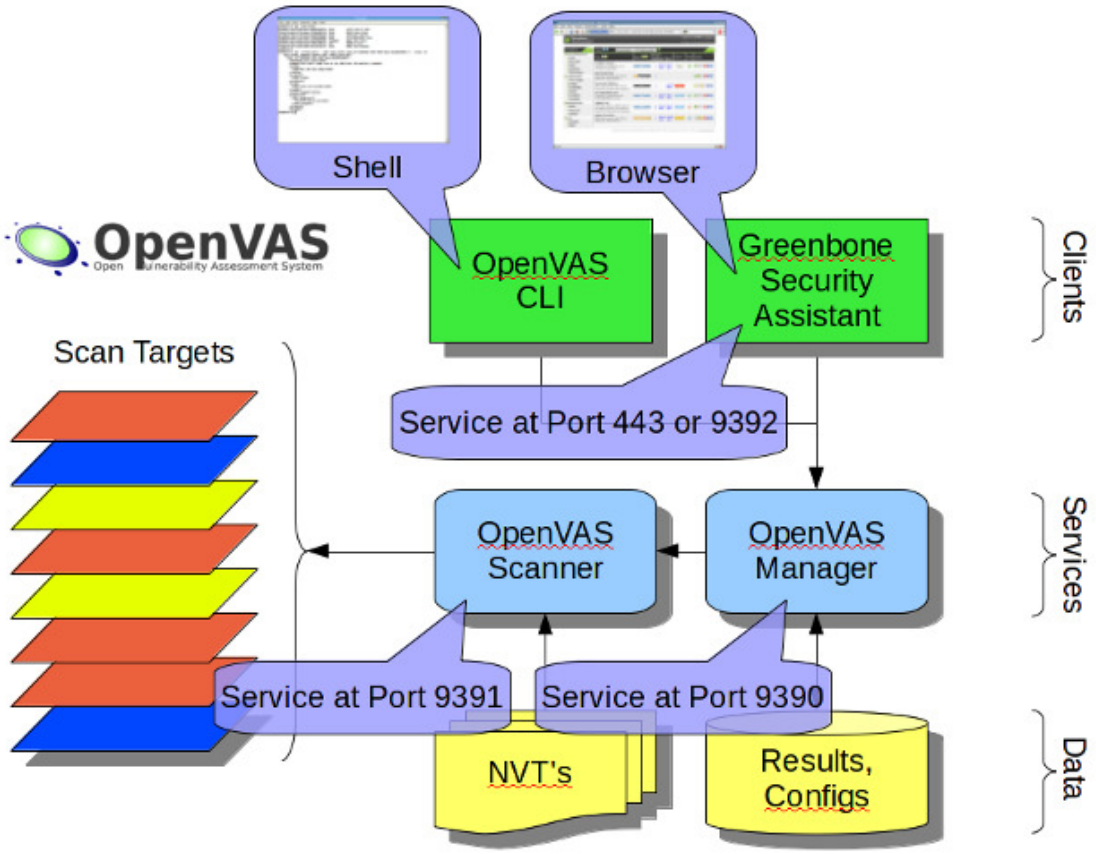


FIGURA 9: ARQUITETURA DO OPENVAS

Existem alguns clientes *front-end* para o gerenciador OpenVAS disponíveis. Dois deles são a interface de linha de comando e o Greenbone Security Assistant, que é um *webservice* com interface para navegadores web. O Greenbone tem como vantagem sobre a interface de linha de comandos a facilidade de uso em todas as tarefas, seja o cadastro de um alvo, de uma varredura ou a exportação os resultados. Para exportação de resultados a ferramenta oferece diversas opções de formato, incluindo HTML, XML, PDF, LaTex e texto plano. Toda a arquitetura do OpenVAS está presente na figura 9, nela é mostrada como os diferentes componentes do sistema se encaixam.

3.2.3 Bruteforce Exploit Detector (BED)

Bruteforce Exploit Detector ou simplesmente BED é um ferramenta desenvolvida com o propósito de testar a segurança de implementações de uma gama de protocolos baseados em texto. Os protocolos cuja ferramenta é capaz de analisar são: FTP, SMTP, POP, HTTP, IRC, IMAP, PJL, LDP, FINGER, SOCKS4 e SOCKS5.

A ferramenta busca encontrar potenciais estouros de *buffer*, erros de formatação de *string*, estouro de inteiros, condições de negação de serviço, condições de corrida entre outros. Os testes realizados por essa ferramenta são baseados no envio de combinações de diversos comandos conhecidos associados a *strings* consideradas problemáticas pelos desenvolvedores da ferramenta (ALLEN, HERIYANTO, ALI, 2014).

3.3 Exploração

3.3.1 Metasploit Framework

A parte código aberto do projeto Metasploit, o Metasploit Framework é uma ferramenta que possibilita o desenvolvimento, teste e a execução de exploits. Criado em 2003 por HD Moore, o projeto Metasploit pertence atualmente a companhia Rapid7 (ROUSE, 2011) e é uma das ferramentas mais bem colocadas no quadro de ferramentas da página SecTools. O SecTools é um projeto mantido pela equipe do Nmap, cujo objetivo é o de catalogar as ferramentas preferidas pelos profissionais da área de segurança computacional (SECTOOLS, 2014).

De acordo com Jaswal (2014) a estrutura do Metasploit Framework pode ser representada pelo diagrama da Figura 10, nesse diagrama a biblioteca REX é responsável por gerenciar as funções mais internas ao programa, como por exemplo, estabelecer *sockets* para conexões. A bilioteca MSF CORE é responsável pela API de nível mais baixo do *framework,* além disso, ela também é o núcleo do framework. A última biblioteca no diagrama é a MSF BASE que oferece uma API de nível um pouco mais alto, essa biblioteca é utilizada pelos módulos e pelas várias interfaces oferecidas para o *framework.*

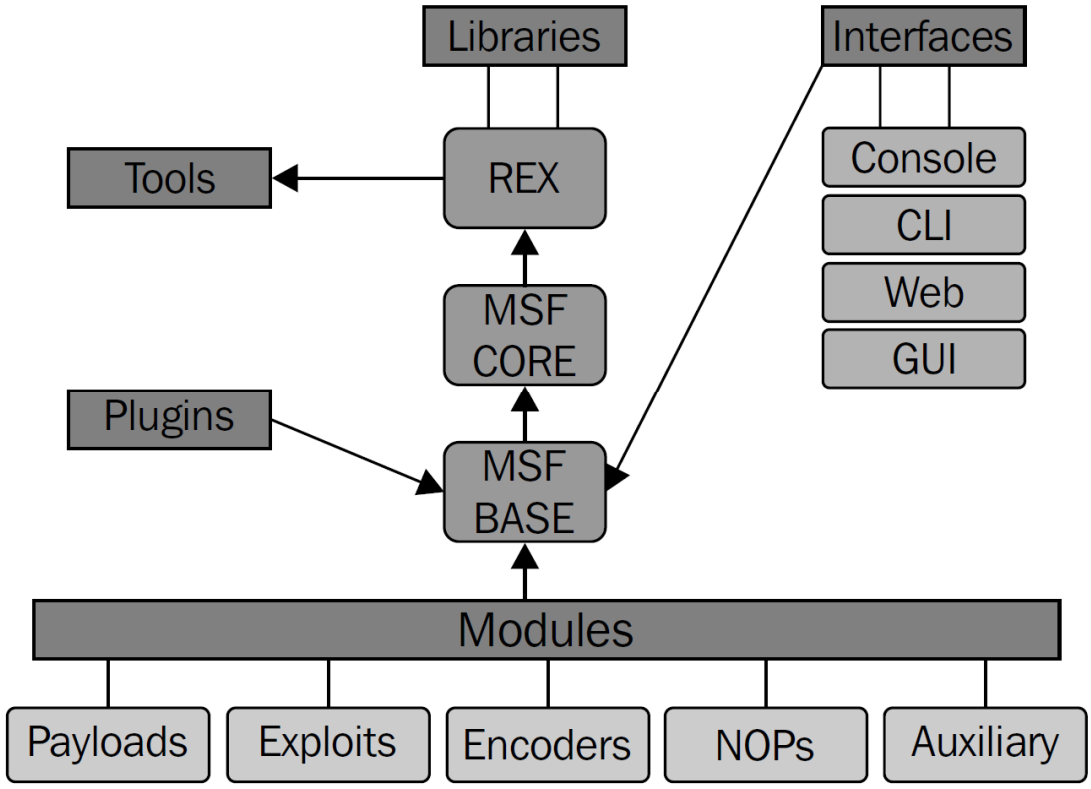


FIGURA 10: DIAGRAMA DA ESTRUTURA DO METASPLOIT FRAMEWORK

Ainda segundo Jaswal (2014), os módulos do framework são organizados de acordo com a sua funcionalidade. Como o framework é feito em código aberto, os usuários mais avançados podem usar a API disponível na biblioteca MSF BASE para criar os seus próprios módulos. A classificação dos módulos é a seguinte:

- Payloads: são módulos usados para realizar operações dentro de sistemas que foram explorados com sucesso, são exemplos módulos

que estabelecem uma conexão com o atacante, instalam serviços na máquina atacada, abrem uma porta específica no firewall, e outros.

- Exploit: módulos que exploram alguma vulnerabilidade de um sistema.
- Encoders: módulos de codificação que servem para codificar o conteúdo de um payload para que o mesmo não seja pego por um antivírus ou um sistema de detecção de intrusão.
- NOPs: esses módulos servem para fazer com que o tamanho de um payload seja consistente. NOPs são instruções vazias para um processador, quando executadas apenas avançam o programa para a próxima instrução.
- Auxiliary: módulos nessa categoria realizam tarefas específicas não necessariamente relacionadas à exploração do sistema. Módulos auxiliares executam tarefas como varredura de portas, enumeração de serviços, *fingerprinting* de banco de dados.

De acordo com Kennedy et al. (2011), as três principais interfaces para o Metasploit Framework são: MSFconsole, MSFcli e Armitage. Segundo esses autores, a interface MSFconsole é a mais popular interface de acesso ao *framework,* essa interface funciona através de um console interativo e provê acesso a todas funcionalidades do mesmo. A interface MSFcli também executa na linha de comando, entretanto ela não é interativa como o MSFconsole. Para Kennedy et al. (2011), o foco dessa interface é o teste de scripts uma vez que o resultado da execução pode ser passado diretamente para outras ferramentas da linha de comando. Por último temos o Armitage, que é um interface gráfica para o *framework* , os autores afirmam que é bastante completa, entretanto deveria ser deixada para usuários mais experientes do *framework,* uma que iniciantes teriam maior facilidade em apreender a usar a ferramenta através do MSFconsole.

4 Exemplo Real de um Teste de Penetração

Para demonstrar os conceitos apresentados até então nesse trabalho, foi efetuado um teste de penetração com a colaboração do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina, o INE.

Durante o teste diversas ferramentas foram utilizadas. Mais especificamente foram utilizadas as seguintes ferramentas para cada fase do teste de penetração:

Fase de Varredura: Nmap e OpenVAS, Greenbone Security Assistant.

Fase de Exploração: Metasploit Framework.

Fase de Documentação: Word, PowerPoint, Excel e Bullzip PDF Printer.

Além das ferramentas específicas por fase do projeto, também foram utilizadas algumas outras que são mais gerais, como por exemplo, o Kali Linux e editores de texto diversos.

O teste abrangeu os *host*s de uma sub-rede do departamento que é dedicada à hospedagem de projetos dos seus professores. O processo para realização desse teste foi dividido em:

1. Obtenção da permissão do chefe do departamento para realização do experimento.
2. Definição do escopo dos testes junto ao gerente da rede.
3. Realização da varredura de portas.
4. Realização da varredura de vulnerabilidades.
5. Realização da fase de exploração.
6. Documentação dos Resultados.

O pedido de permissão para a realização foi basicamente um e-mail que foi enviado ao gerente da rede e posteriormente encaminhado ao chefe do departamento. O texto original do pedido é o seguinte:

Pedido de Permissão para a Realização de Testes de Penetração em Máquinas da Rede INE como Parte do Trabalho de Conclusão de Curso de Sistemas de Informação

Venho por meio desse documento, pedir permissão para realização de testes de penetração em algumas máquinas da rede INF. Um teste de penetração é uma atividadeque consiste em buscar vulnerabilidades e tentar explorá-las para ganhar acesso a umdeterminado ambiente computacional. Testes de penetração são importantes para qualquer organização preocupada com a sua segurança computacional. Eles oferecem a chance de analisar os mecanismos de segurança empregados pela organização a partir da ótica de umatacante.

O TCC que estou desenvolvendo sob a orientação da professora Carla Merkle Westphall tem como objetivo criar um guia sobre testes de penetração para estudantes da área da informática. Nele são abordados aspectos teóricos e práticos referentes à segurança e testes de penetração. É necessário, no entanto, que para a melhor realização desse trabalho, a parte prática seja desenvolvida em um ambiente real. A parte prática do trabalho visa à demonstração de conceitos e do uso de ferramentas para testes de penetração. Os testes de penetração que desejo realizar também servirão para ajudar a consertar eventuais falhas que sejam encontradas na segurança do ambiente computacional da rede INF.

Os testes de penetração que proponho são constituídos basicamente por varreduras de portas, análise dos serviços encontrados nessas portas e a tentativa de explorer potenciais vulnerabilidades neles. Esses testes não serão obstrutivos e não possuem o objetivo de gerar instabilidade dentro da rede INF.

As máquinas que serão testadas serão delimitadas pelo servidor Alexandre Sardin (o qual me orientou a escrever esse pedido formal), além disso, o horário bem como o ponto de origem para a realização dos testes poderão ser especificado de maneira a não interferir com o funcionamento normal da rede. Todo resultado do teste será entregue aos responsáveis pela

rede INE antes da publicação do trabalho, e se for necessário poderá ser editado para que não haja impacto na segurança da rede. Solicito assim, a permissão necessária para realizar esse trabalho.

Atenciosamente,

Jean Pacher

Graduando em Sistemas de Informação

7.Com a aprovação do chefe do departamento o gerente da rede liberou a realização do teste na rede dedicada aos servidores dos professores. Essa escolha foi realizada pelo gerente, pois as máquinas que se encontram nessa rede são responsabilidade dos próprios professores, ou seja, são configuradas e atualizadas apenas pelos critérios dos professores. As outras etapas do processo são **detalhadas** nos capítulos a seguir.

As etapas de varredura de portas e varredura de vulnerabilidades foram executadas cada uma 3 vezes a partir de 3 diferentes pontos. O primeiro conjunto de varreduras (portas e vulnerabilidades) foi executado a partir de um computador interno a rede alvo, o segundo a partir de um computador localizado fora da rede da universidade, o último conjunto de varreduras foi realizado a partir da rede do Laboratório de Redes e Gerência (LRG), a rede do laboratório não faz parte da rede alvo, entretanto ela fica dentro da universidade.

A realização das varreduras de portas e de vulnerabilidades a partir de diferentes localizações agrega valor ao teste, pois isso fornece ao cliente uma visão mais ampla sobre os vetores de ataque que podem ser usados contra ele, dependendo de onde um atacante inicia o seu ataque. Apesar disso essas duas varreduras não faziam parte do roteiro inicial de testes, de forma que foram adotadas opções menos detalhadas para a varredura de portas por se tratar de um processo que consume muito tempo.

4.1 Varredura de Portas

Como explicado anteriormente, a varredura de portas é um processo no qual *host*s são sondados com o objetivo de descobrir informações sobre suas portas, quais serviços estão rodando, qual o sistema operacional de cada um, entre outras informações. Na execução desse trabalho foi usada a ferramenta Nmap para a realização dessa etapa.

Apesar de Engebretson (2011) afirmar que a varredura de portas deve ser realizada até mesmo em IPs cujos *host*s pareçam estar desligados, no desenvolvimento desse trabalho foi optado ignorar essa recomendação. A varredura foi realizada apenas em *host*s que estivessem claramente ligados para economizar tempo.

Na detecção de quais IPs possuiam um *host* associado, foi utilizada a ferramenta fping, com ela é possível sondar uma faixa de endereços IP com apenas um comando, presente na primeira linha da Figura 11.

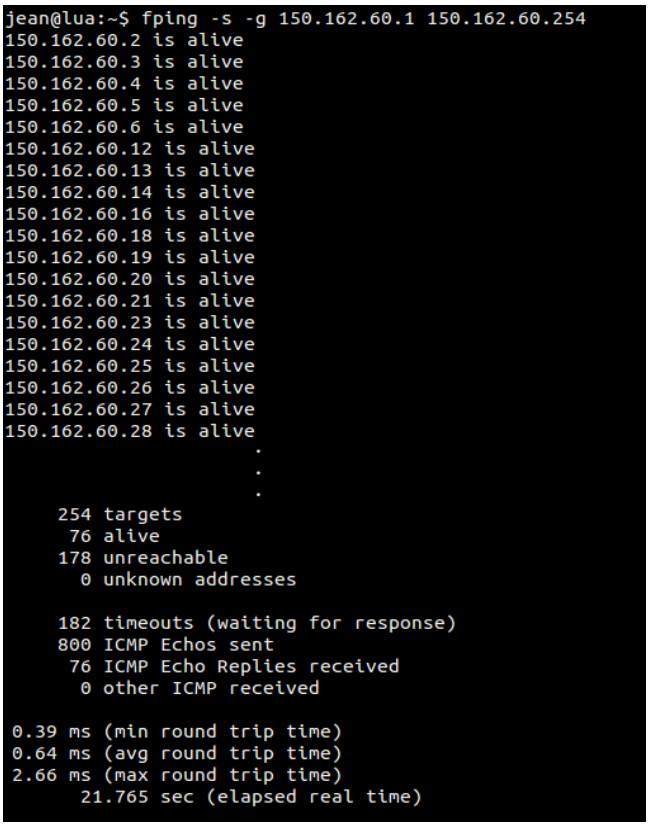


FIGURA 11: EXEMPLO DE USO DA FERRAMENTA FPING

No comando da figura 11 o operador “-s” especifica que um resumo dos resultados deve ser apresentado após a execução do programa, já o operado “-g” serve para especificar a faixa de IPs que devem ser pingados. Foram encontrados 76 *host*s ativos que são:

- 150.162.60.2
- 150.162.60.3
- 150.162.60.4
- 150.162.60.5
- 150.162.60.6
- 150.162.60.12
- 150.162.60.13
- 150.162.60.14
- 150.162.60.16
- 150.162.60.18
- 150.162.60.19
- 150.162.60.20
- 150.162.60.21
- 150.162.60.23
- 150.162.60.24
- 150.162.60.25
- 150.162.60.26
- 150.162.60.27
- 150.162.60.28
- 150.162.60.29
- 150.162.60.30
- 150.162.60.31
- 150.162.60.34
- 150.162.60.36
- 150.162.60.38
- 150.162.60.40
- 150.162.60.41
- 150.162.60.42
- 150.162.60.43
- 150.162.60.60
- 150.162.60.62
- 150.162.60.65
- 150.162.60.66
- 150.162.60.67
- 150.162.60.94
- 150.162.60.103
- 150.162.60.106
- 150.162.60.118
- 150.162.60.119
- 150.162.60.120
- 150.162.60.121
- 150.162.60.122
- 150.162.60.123
- 150.162.60.126
- 150.162.60.129
- 150.162.60.133
- 150.162.60.134
- 150.162.60.140
- 150.162.60.141
- 150.162.60.144
- 150.162.60.147
- 150.162.60.148
- 150.162.60.151
- 150.162.60.152
- 150.162.60.154
- 150.162.60.155
- 150.162.60.157
- 150.162.60.158
- 150.162.60.161
- 150.162.60.164
- 150.162.60.165
- 150.162.60.166
- 150.162.60.168
- 150.162.60.172
- 150.162.60.173
- 150.162.60.178
- 150.162.60.179
- 150.162.60.180
- 150.162.60.181
- 150.162.60.182
- 150.162.60.183
- 150.162.60.184
- 150.162.60.185
- 150.162.60.186
- 150.162.60.187
- 150.162.60.254

Essa lista de hosts obtida antes do primeiro conjunto de varreduras foi utilizada nas varreduras posteriores para manter a consistência entre os resultados.

A primeira varredura de portas (realizada internamente a rede alvo) foi feita com cinco operadores: -v (operador “detalhamento”, exibe detalhes da varredura em tempo de execução), -sV (detecção de versão dos serviços encontrados), -O (detecção de sistema operacional), -sS(varredura TCP SYN), -sU (varredura UDP), –iL (operador para leitura de alvos a partir de arquivo) e –oG (específica que o resultado da varredura deve ser salvo em um arquivo de texto próprio para buscas com a ferramenta “grep”, presente em distribuições

GNU/Linux). O comando utilizado para iniciar a varredura na rede do Departamento de Informática está na primeira linha da Figura 12.

```
root@lua:/home/jean# nmap -v -sV -O -sS -sU -iL hosts.txt -oG nmap_main_scan

Starting Nmap 6.40 ( http://nmap.org ) at 2014-11-03 19:26 BRST
NSE: Loaded 23 scripts for scanning.
Initiating Ping Scan at 19:26
Scanning 76 hosts [4 ports/host]
```

FIGURA 12: COMANDO UTILIZADO PARA FAZER A VARREDURA DE PORTAS NA REDE DO INE

O resultado salvo em arquivo pelo nmap começa descrevendo qual foi o comando usado (primeira linha da Figura 12) e segue com a listagem de todas as portas que foram varridas pela aplicação, um parcial da listagem de portas varridas (tanto TCP quanto UDP) é exibido na Figura 13.

```
# Ports scanned: TCP(1000;1,3-4,6-7,9,13,17,19-26,30,32-33,37,42-
43,49,53,70,79-85,88-90,99-100,106,109-111,113,119,125,135,139,143-
144,146,161,163,179,199,211-212,222,254-
256,259,264,280,301,306,311,340,366,389,406-407,416-417,425,427,443
-445,458,464-465,481,497,500,512-515,524,541,543-545,548,554-
555,563,587,593,616-617,625,631,636,646,648,666-
668,683,687,691,700,705,711,714,720,722,726,749,765,777,783,787,800
-801,808,843,873,880,888,898,900-903,911-912,981,987,990,992-
993,995,999-1002,1007,1009-1011,1021-1100,1102,1104-1108,1110-
                              .
                              .
                              .
1900,62078,63331,64623,64680,65000,65129,65389) UDP(1000;2-
3,7,9,13,17,19-23,37-38,42,49,53,67-69,80,88,111-113,120,123,135-
139,158,161-162,177,192,199,207,217,363,389,402,407,427,434,443,445
,464,497,500,502,512-515,517-
518,520,539,559,593,623,626,631,639,643,657,664,682-689,764,767,772
-776,780-782,786,789,800,814,826,829,838,902-
903,944,959,965,983,989-990,996-1001,1007-1008,1012-1014,1019-
1051,1053-1060,1064-1070,1072,1080-1081,1087-1088,1090,1100-
```

FIGURA 13: LISTAGEM PARCIAL DAS PORTAS VARRIDAS PELO NMAP

Após a listagem de portas varridas o Nmap lista no arquivo os resultados específicos de cada um dos *host*s que foi varrido.

O formato que foi escolhido para armazenar os resultados da varredura é ideal para trabalhar com a ferramenta grep do GNU/Linux, sendo que o resultado de cada um dos *host*s varridos é colocado uma única linha. Para facilitar a compreensão, os resultados que serão mostrados a seguir foram

formatados manualmente. Nessa formatação é pulada uma linha para separar os resultados de diferentes *host*s e o sumário dos serviços encontrados em cada porta é separado por novas linhas.

Os resultados obtidos na primeira varredura de portas (interna a rede alvo) foram bastante variados, no total foi identificado o estado de 298 portas, porém a ferramenta não conseguiu identificar o serviço associado a 14 delas.

O número de serviços por *host* também foi bastante variado, em alguns *host*s não foi possível detectar sequer um serviço ativo, nem mesmo identificar qual o sistema operacional era presente nele, como por exemplo, os *host*s 150.162.60.34, 150.162.60.38 e 150.162.60.41 cujo resultado da varredura é exibido na Figura 14. A única coisa que se foi possível constatar nos *host*s da Figura 14, é que estavam ativos e quais eram os seus respectivos domínios.

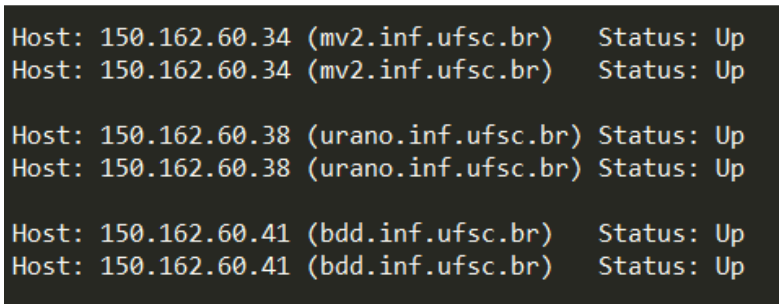


FIGURA 14: EXEMPLOS DE HOSTS SEM RESULTADOS NA VARREDURA

Já em diversos outros *host*s foram identificados vários serviços junto as suas versões a ainda o sistema operacional rodando em cada um. São exemplos desse caso os *host*s 150.162.60.129 e 150.162.60.140 cujos resultados são exibidos na Figura 15. Nesses 2 *host*s o resultado de cada porta segue o formato: nº da porta / estado da porta / protocolo de transporte associado a porta // tipo do serviço encontrado // nome do software que executa o serviço <versão do serviço>.

```
Host: 150.162.60.129 (inca.inf.ufsc.br) Status: Up
Host: 150.162.60.129 (inca.inf.ufsc.br)
Ports:
 22/open/tcp//ssh//(protocol 2.0)/
 8009/open/tcp//ajp13//Apache Jserv (Protocol v1.3)/
 8080/open/tcp//http//Apache Tomcat|Coyote JSP engine 1.1/
 8443/open/tcp//ssl|http//Apache Tomcat|Coyote JSP engine 1.1/
 68/open|filtered/udp//dhcpc///
 631/open|filtered/udp//ipp///
 5353/open/udp//mdns//DNS-based service discovery/  Ignored State: closed (1993)
OS: Linux 3.11 - 3.14   Seq Index: 261  IP ID Seq: All zeros

Host: 150.162.60.140 (pegasus.inf.ufsc.br)  Status: Up
Host: 150.162.60.140 (pegasus.inf.ufsc.br)
Ports:
 22/open/tcp//ssh//OpenSSH 5.5p1 Debian 6+squeeze3 (protocol 2.0)/
 80/open/tcp//http//Apache httpd 2.2.16 ((Debian))/
 111/filtered/tcp//rpcbind///
 139/filtered/tcp//netbios-ssn///
 445/open/tcp//netbios-ssn//Samba smbd 3.X (workgroup: LAPS)/
 548/open/tcp//afp//Netatalk 2 (name: pegasus; protocol 3.2)/
 901/open/tcp//http//Samba SWAT administration server/
 2049/open/tcp//nfs//2-4 (RPC #100003)/
 3389/filtered/tcp//ms-wbt-server///
 67/open|filtered/udp//dhcps///
 111/open/udp//rpcbind//2 (RPC #100000)/
 123/open/udp//ntp//NTP v4 (unsynchronized)/
 631/open|filtered/udp//ipp///
 2049/open/udp//nfs//2-4 (RPC #100003)/
 5353/open/udp//mdns//DNS-based service discovery/  Ignored State: closed (1985)
OS: Linux 2.6.32    Seq Index: 257  IP ID Seq: All zeros
```

FIGURA 15: EXEMPLO DE HOSTS COM VÁRIAS INFORMAÇÕES IDENTIFICADAS NA VARREDURA

Os serviços mais comuns nos *host*s são listados no gráfico de barras da Figura 16, nele são listados os serviços e o número de ocorrências de cada um.

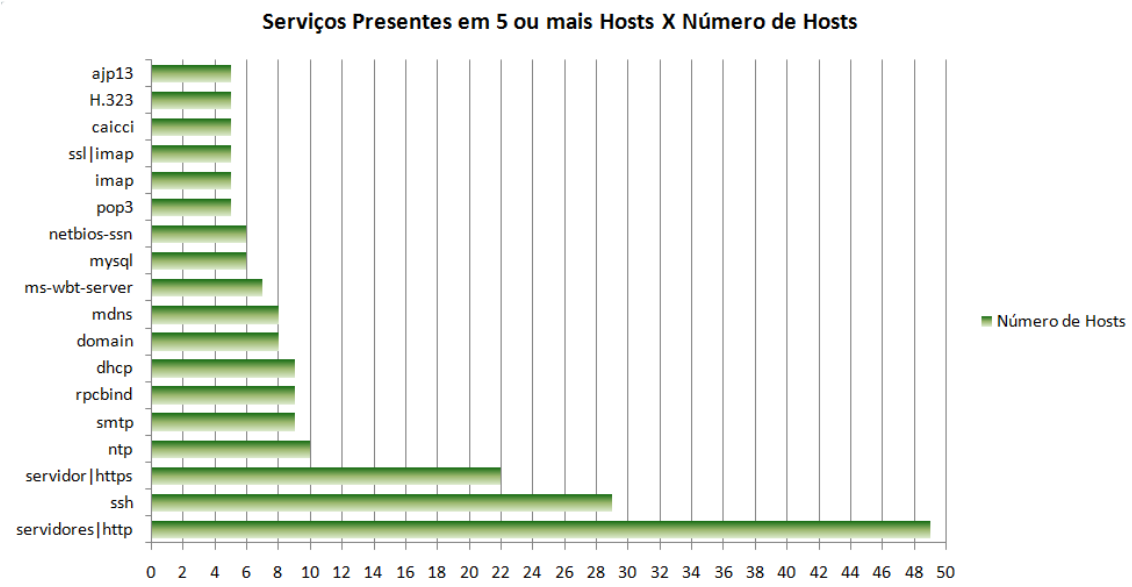


FIGURA 16: GRÁFICO DOS SERVIÇOS MAIS COMUNS ENTRE OS HOSTS

Com os resultados da varredura também foi possível ter uma idéia do propósito de cada *host* que foi sondado. Isso pode ser observado pelo nome de cada um dos *host*s e os serviços que rodam nele, por exemplo, o *host* mail2.inf.ufsc.br hospeda serviços de SMTP, pop3 e imap, tendo como possível propósito o de servir como servidor de emails.

Além de uma enumeração dos serviços encontrados a ferramenta conseguiu identificar quais os sistemas operacionais em 62 dos 76 *host*s varridos (Figura 17).

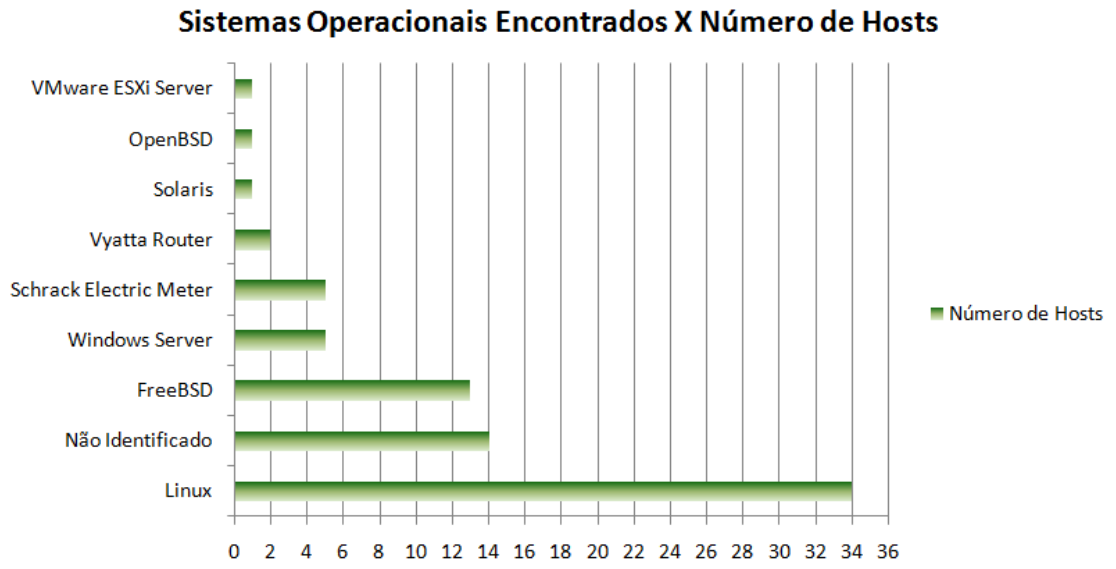


FIGURA 17: SISTEMAS OPERACIONAIS IDENTIFICADOS

Nas duas outras varreduras de portas, foi utilizado um outro conjunto de operadores: -v (operador “detalhamento”), -sS(varredura TCP SYN), -sU (varredura UDP), –iL (operador para leitura de alvos a partir de arquivo) e –oG (específica que o resultado da varredura deve ser salvo em um arquivo de texto próprio para buscas com a ferramenta “grep”).

Em comparação com a varredura feita internamente a rede alvo, a varredura externa conseguiu varrer apenas 67 dos 76 *hosts* especificados. Os *hosts* que não foram identificados na varredura externa (e na varredura feita a partir do LRG) se dividem em três categorias: roteadores, máquinas restritas a rede interna e máquinas que estavam em manutenção ou foram removidas.

Comparando apenas os hosts em comum nas duas varreduras, a externa conseguiu identificar o estado de 2 portas a mais do que a interna (261 portas identificadas na interna e 263 na externa), a causa dessa situação não foi encontrada, entretanto é possível que um novo serviço tenha sido instalado em um dos *hosts*, que um serviço estivesse desabilitado durante a varredura inicial (interna) ou então que houvesse alguma regra de firewall que restringia internamente o acesso a certas portas.

O resultado aponta que a quantidade de portas abertas para o exterior é quase a metade do número de portas abertas para a rede interna (220 portas

abertas identificadas na varredura interna contra 112 na externa). O número de portas fechadas para a rede exterior é superior a 7 vezes o total de portas fechadas para a rede interna (13 portas identificadas fechadas na varredura interna e 93 na externa). E o total de portas identificadas como filtradas (portas as quais o nmap não consegue determinar se estão abertas devido a filtragem de pacotes realizada pelo alvo) passou de 3 portas na varredura interna para 58 na varredura externa.

Uma possível interpretação para o aumento de número de portas fechadas (portas para as quais o nmap estipula que não existem serviços associados) quando comparadas as varreduras interna e externa, é que o número de serviços rodando nos *hosts* varridos teria diminuído. Entretanto quando o resultado da varredura feita a partir do LRG é analisado, é constatado que número de portas fechadas encontradas nessa varredura é similar ao número de portas fechadas encontradas na varredura interna (contando apenas *hosts* em comum), o que pode sugerir que algumas dos serviços hospedados na rede alvo são configurados para descartar requisições vindas de fora da rede, ou que o firewall se comporta de modo a dar impressão que certas portas estão fechadas, ou até mesmo que houve alguma falha na entrega de pacotes ao Nmap durante a varredura.

Assim como a varredura realizada externamente a rede da UFSC, a varredura feita a partir do Laboratório de Redes e Gerência (LRG), conseguiu varrer apenas 67 do 76 *hosts* especificados. No entanto de maneira inesperada, a varredura feita a partir do LRG identificou o estado de apenas 186 portas (quase 80 a menos que a varredura externa). O número mais baixo de portas identificadas, foi provavelmente ocasionado por mudanças feitas na rede alvo após a varredura externa, e não por outro fator. Apesar da redução do número de portas identificadas, o número de portas abertas identificadas na rede alvo é maior na varredura feita a partir do LRG do que o número de portas abertas identificadas na varredura externa, além disso o número de portas fechadas também caiu bastante se comparada também a varredura externa.

A Figura 18 contém o resumo do resultado das 3 varreduras de portas realizadas, nela é exibido para cada uma das varreduras e para um subconjunto dos resultados da varredura interna que contém apenas o *hosts* em comum com as outras duas, o número total de portas e o número total de acordo com a classificação de portas dada pelo Nmap.

Varredura	Abertas	Abertas|Filtradas	Fechadas	Filtradas	Total de Portas
Interna a rede INE (completa)	256	26	13	3	298
Interna a rede INE (hosts varredura externa e LRG)	220	25	13	3	261
Externa (fora da rede UFSC)	112	0	93	58	263
Laboratório de Redes e Gerência (LRG)	160	1	14	11	186

FIGURA 18: RESUMO DOS RESULTADOSS DAS VARREDURAS DE PORTAS

4.2 Varredura de Vulnerabilidades

Após cada uma das varreduras de portas foi seguida de uma varredura de vulnerabilidades. Para realização de cada uma dessas varreduras de vulnerabilidades foi utilizada a ferramenta openVAS juntamente com a sua interface web o Greenbone Security Assistant. O processo apresentado a seguir é comum em todas as varreduras, porém ele é demonstrado com a primeira delas (varredura interna a rede alvo).

O primeiro passo na execução da varredura de vulnerabilidades através do openVAS foi a criação de uma lista de alvos. Para criar uma lista de alvos através do Greenbone é necessário navegar até a aba *Configuration* e clicar na opção *Target*, uma vez dentro da página de alvos é possível visualizar os alvos cadastrados no sistema, editá-los ou cadastrar novos. Para cadastrar um alvo novo basta clicar no ícone de estrela (em vermelho na Figura 19) que irá

redirecionar o usuário para a página de cadastro.

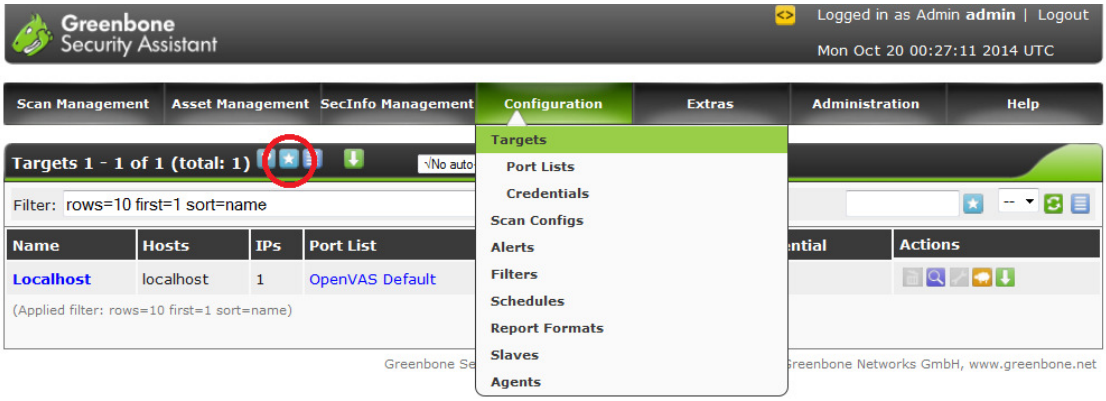


FIGURA 19: PÁGINA DE GERENCIAMENTO DE ALVOS

Na página de cadastro é possível especificar os *host*s manualmente ou a partir de um arquivo de texto. No teste, os alvos foram importados a partir de um arquivo de texto, o formato do arquivo é simples, devem-se listar cada um dos *host*s que serão varridos separados por novas linhas, esse formato é exibido na Figura 20, que contém parte do arquivo usado para o cadastro de alvos.

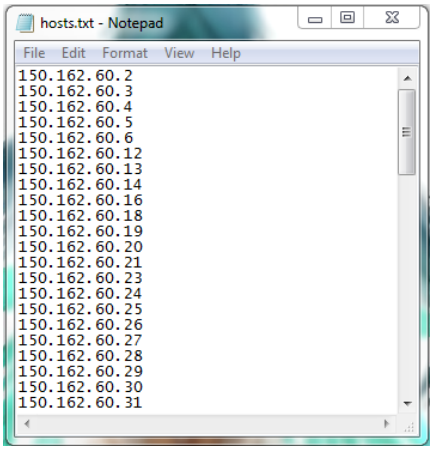


FIGURA 20: ARQUIVO COM OS HOSTS ALVO

Depois de selecionado o arquivo com os alvos é possível escolher um nome para essa lista de alvos, adicionar um comentário, escolher quais serão as portas varridas, e adicionar credenciais de acesso SSH e SMB. Todas essas informações são mostradas na Figura 21. No desenvolvimento desse experimento as opções de adicionar credenciais de acesso SSH e SMB não

foram usadas, pois o teste que foi feito é do tipo caixa preta, e não se possuia informações prévias sobre os alvos.

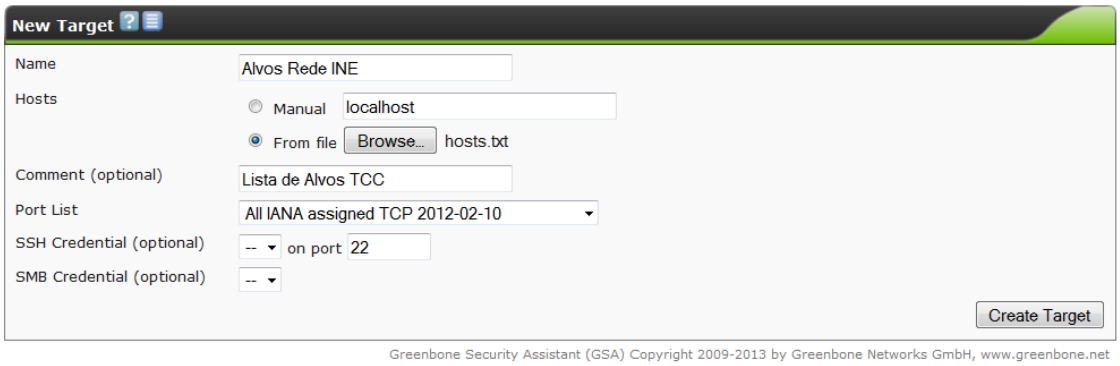


FIGURA 21: TELA DE CADASTRO DE ALVOS

Durante o processo de varreduras de vulnerabilidades o openVAS realiza a sua própria varredura de portas com o Nmap, por isso que há uma opção de escolha de portas quando se adiciona um novo alvo. As possíveis opções de porta são mostradas na Figura 22.

Nesse experimento os alvos foram cadastrados com a opção grifada na Figura 22, “All IANA assigned TCP and UDP 2012-02-10”, essa opção compreende todas as portas registradas na IANA e que possuam algum serviço associado que opere sobre os protocolos TCP e UDP. O registro completo pode ser encontrado no site da IANA (IANA, 2014).

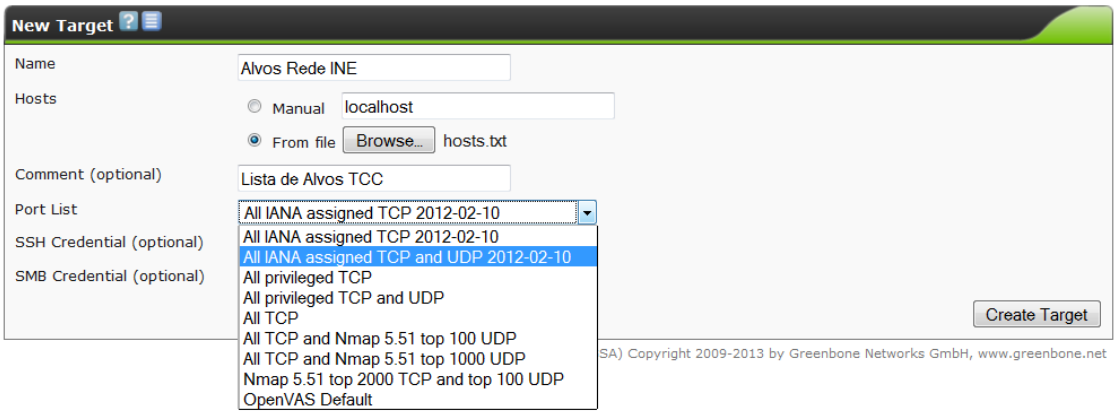


FIGURA 22: CADASTRO DE ALVOS, OPÇÕES DE PORTAS PARA VARREDURA

Após o cadastro de alvos, a próxima etapa da varredura de vulnerabilidades com o openVAS foi a criação de uma tarefa. A página de criação de tarefas pode ser acessada através do ícone de estrela na página inicial (Figura 23).

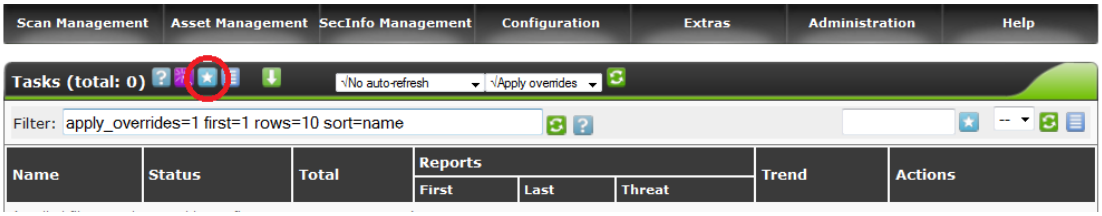


FIGURA 23: ÍCONE DE ESTRELA, PÁGINA INICIAL

Uma tarefa no openVAS é a definição de uma varredura de vulnerabilidades, nela é possível:

- Escolher qual lista de alvos será varrida.
- Definir a configuração da varredura (*full and fast, full and fast ultimate, full and very deep* e *full and very deep ultimate*). As configurações de varredura *fast* e *deep* se diferenciam pelo fato de que a *fast* usa informações previamente coletadas em outras varreduras enquanto a *deep* sempre faz uma varredura a partir do zero. Já as varreduras *ultimate* se diferenciam das varreduras normais porque realizam testes que podem afetar o funcionamento dos serviços que são varridos.
- Associar alertas a varredura. Alertas são ações que são acionadas quando ocorre algum evento durante a varredura (e.g. o estado da varredura passa para "varredura completa"). De acordo com o glossário interno da Greenbone, alertas normalmente são utilizados para enviar emails para os interessados em uma determinada tarefa.
- Escolher um cronograma, para a tarefa. Cronogramas definem horário e data para que uma tarefa seja iniciada automaticamente, qual o período em que a tarefa deve ser executada (e.g. 3 semanas) e qual a duração máxima para a tarefa.
- Adicionar um *slave,* que é outro servidor do openVAS que também pode ajudar a executar a tarefa.

- Definir observadores, que são usuários que podem acompanhar a tarefa.
- Escolher o número de testes que podem ser aplicados a cada *host* paralelamente e o número máximo de *host*s que pode ser varrido simultaneamente.
- Dar um nome a tarefa e adicionar comentários.

Além das funcionalidades previamente descritas a página de criação de tarefas também permite a criação de um contêiner de atividades, que serve como um agrupador de atividades. A tela de cadastro de tarefas pode ser vista na Figura 24.

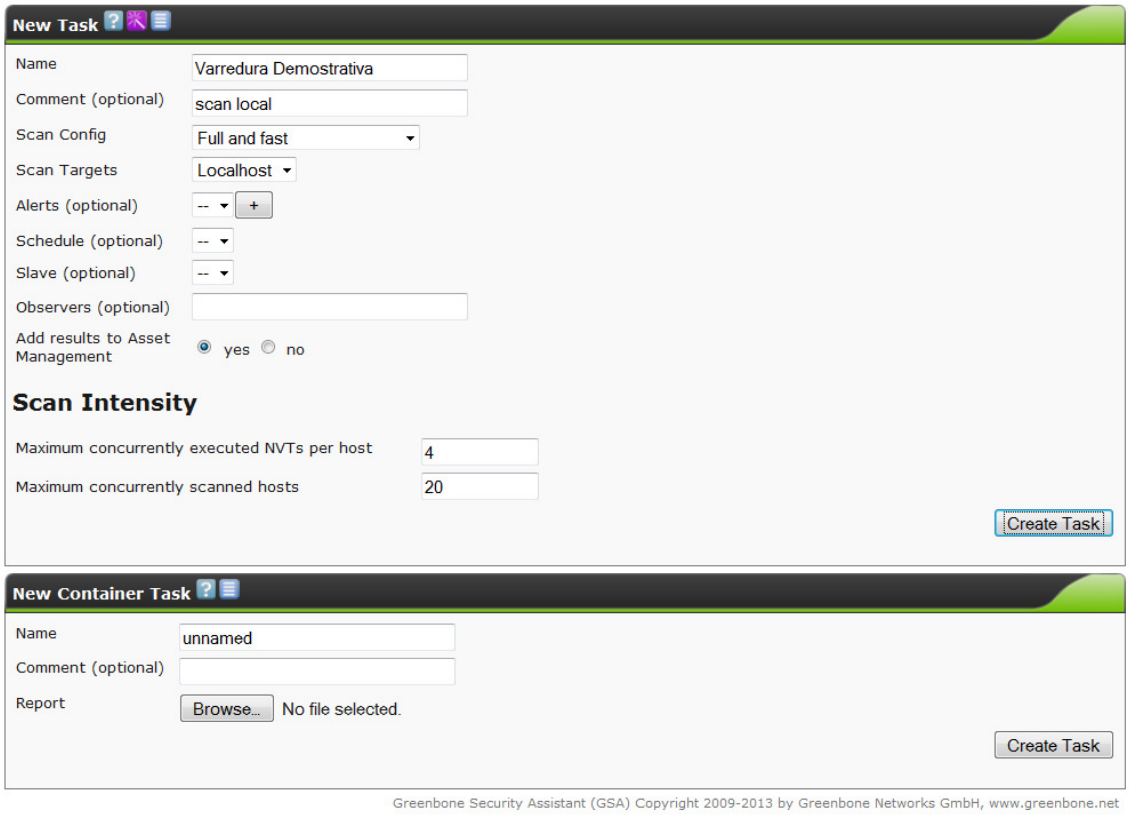


FIGURA 24: TELAS DE CADASTRO DE VARREDURA

A tarefa cadastrada no experimento feito no Departamento de Informática teve como tipo de configuração adota "*deep and fast*". Não se adotou a configuração *ultimate,* uma vez que a configuração *ultimate* utilza NVTs que podem interromper o serviço sendo testado e uma das condições

para a execução do teste era que o mesmo não causasse instabilidade na rede. Nas opções de intensidade foi escolhido rodar no máximo 6 NVTs simultaneamente em cada *host*, e varrer no máximo 20 *host*s por vez. Não foram adotados alertas, *slaves*, cronogramas ou observadores nessa tarefa.

O passo que se segue a criação de uma tarefa é a sua execução. Como explicado anteriormente, uma tarefa pode ser iniciada automaticamente se for cadastrada junto com um cronograma.

Entretanto caso uma tarefa não seja cadastrada com um cronograma, assim como a tarefa de varredura do teste executado na rede do Departamento, ela pode ser iniciada a partir da página inicial do sistema, clicando no ícone de "*play*" (exibido na) associado a tarefa. Algumas outras funcionalidades de gerenciamento estão disponíveis na mesma barra do menu que contém o botão play. São elas na Figura 25 da esquerda para a direita: botões de pausar, resumir, parar ou excluir uma tarefa e atalhos para visualização de detalhes, edição de configurações, cópia ou exportação de uma tarefa.

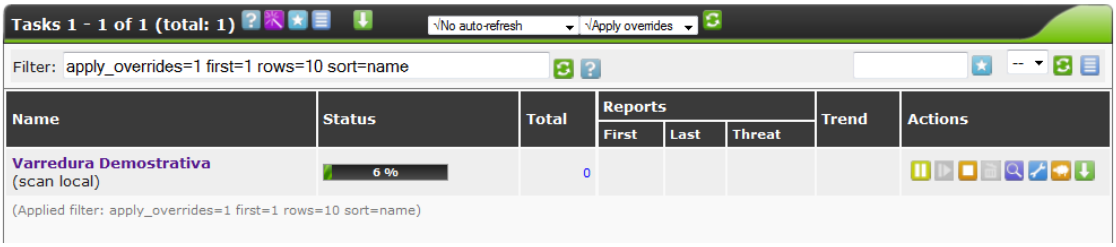


FIGURA 25: PÁGINA INICIAL, MENU DE GERENCIAMENTO DE TAREFA

A primeira varredura de vulnerabilidade levou aproximadamente 20 horas.

Para visualizar os resultados de uma tarefa que foi executada pelo openVAS através do Greenbone é necessário acessar a página de visualização de detalhes, isso pode ser feito clicando no botão de lupa ou no nome da tarefa a partir da página inicial, como visto na Figura 25. A página de visualização de detalhes provê diversas informações úteis para o usuário, como por exemplo, a configuração escolhida para a tarefa, seu alvo, em qual estado ela se encontra

e no final da página uma lista dos relatórios disponíveis. A lista de relatórios (canto inferior Figura 26) possui um resumo numérico da quantidade de vulnerabilidades encontradas por categoria de severidade (alta, média, baixa, log e falso positivo), além de uma classificação geral para a severidade da tarefa como um todo (nesse caso apenas alta, média ou baixa).

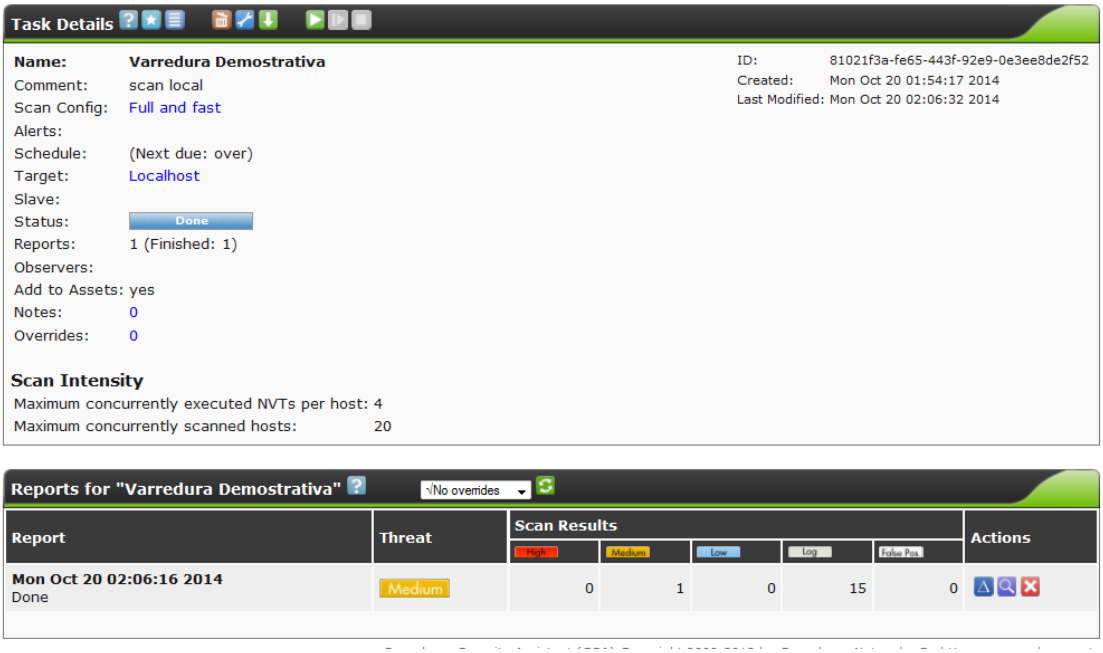


FIGURA 26: PÁGINA DE VISUALIZAÇÃO DE DETALHES DE UMA TAREFA COMPLETA

Apesar de útil o resumo do resultado da varredura oferece apenas uma visão superficial dos resultados, para analisar cada uma das vulnerabilidades encontradas em cada um dos *host*s é necessário ir à página de visualização completa do relatório, essa página pode ser acessada através do ícone de lupa exibido na coluna "*actions*" da tabela de relatórios, canto inferior direito da Figura 26.

A página de visualização completa do relatório apresenta ao usuário 3 grandes seções: resumo do relatório, filtragem de resultados e resultados filtrados.

A primeira seção, resumo do relatório, é similar a tabela de relatórios vista na Figura 27, entretanto na página de visualização completa do relatório possui em seu resumo numérico das vulnerabilidades a possibilidade de

exportação de resultados. Além de possibilitar a exportação completa dos resultados essa seção permite exportar os resultados de acordo com o filtro aplicado na segunda seção (filtragem de resultados) e a exportação apenas dos resultados que estão sendo exibidos na paginação atual da terceira seção (resultados filtrados). O Greenbone possui 9 formatos nativos para exportação de resultados: ARF, CPE, HTML, ITG, LaTeX, NBE, PDF, TXT e XML. O usuário também tem a possibilidade de exportar o relatório em formatos definidos por ele em XML.

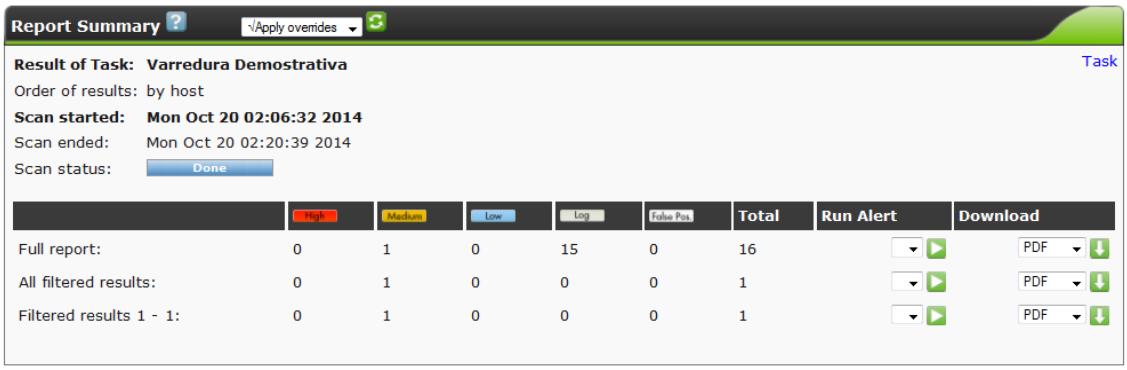


FIGURA 27: SEÇÃO DE RESUMO DO RELATÓRIO

A segunda seção exibida na Figura 28, é responsável pela filtragem de relatórios, permite filtrar e ordenar os dados do relatório. As opções de ordenação dos resultados são: a numeração da portas vulneráveis (ascendente ou descendente) ou o nível de severidade de uma vulnerabilidade (nível ascendente ou descendente). Entretanto essa ordenação não é absoluta, pois as vulnerabilidades são agrupadas por *host*. A filtragem dos resultados possui mais opções, é possível filtrar os resultados pelo:

- Nível de severidade de cada uma das vulnerabilidades encontradas (e.g. mostrar todas as vulnerabilidade de nível médio e alto, enquanto oculta as demais).
- Número de resultados exibidos.
- Texto de descrição das vulnerabilidades.
- Nível de correspondência com qualquer identificador CVE, os níveis são: parcial, total ou ambos. CVE é sigla para *Common Vulnerabilities and Exposures* (vulnerabilidades e exposições

comuns). O CVE, de acordo com a sua página oficial (CVE,2014), é um dicionário internacional, gratuito e de uso público que contém vulnerabilidades de conhecimento público. Um identificador CVE é um código que identifica unicamente uma vulnerabilidade.

- Valor do CVSS associado a uma vulnerabilidade. CVSS ou *Common Vulnerability Scoring System* (sistema comum para pontuação de vulnerabilidades), é um sistema de pontuação desenvolvido com o objetivo de disponibilizar um método aberto e padronizado para atribuir uma pontuação para vulnerabilidades de acordo com a sua severidade, a pontuação CVSS vai de 0 até 10. O CVSS é responsabilidade do NIST (NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, 2014), e todas as vulnerabilidades catalogadas com CVE possuem uma pontuação CVSS.
- "Super Filtro", que é um campo onde palavras chave podem ser usadas tanto para filtrar quanto para ordenar os resultados, e.g. se o texto "rows=10 first=1 sort=name" fosse digitado no campo do "super filtro", seriam exibidos 10 resultados, começando a partir do primeiro e ordenados pela coluna nome.

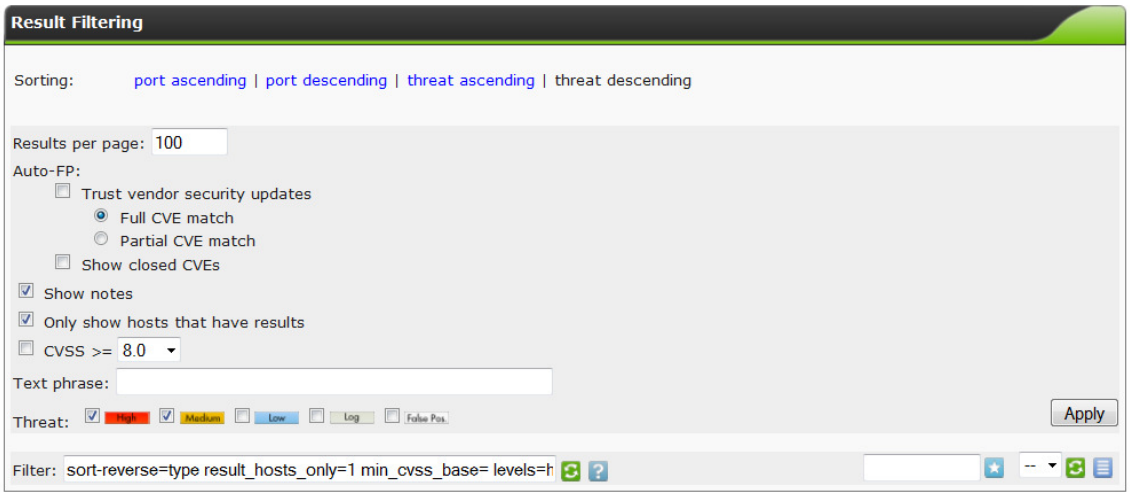
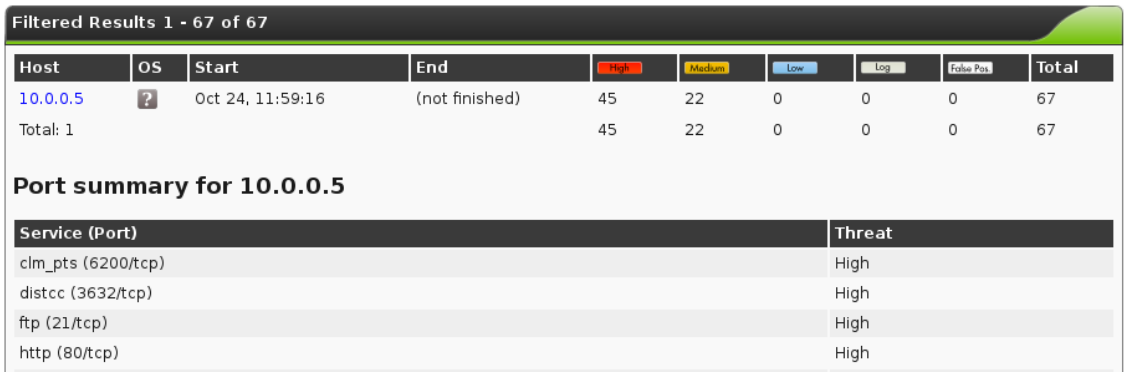


FIGURA 28: SEÇÃO DE FILTRAGEM DO RELATÓRIO

A terceira e última seção da página (Figura 29) de visualização completa do relatório, é a seção de resultados filtrados. Essa seção serve apenas para exibir os resultados da varredura. Assim como na primeira seção, a seção de visualização de dados também mostra uma tabela com o resumo numérico das vulnerabilidades encontradas, porém esse resumo é tabelado por *host*s da página sendo exibida (os resultados são paginados, sendo que cada vulnerabilidade encontrada conta como um resultado). Os resultados são agrupados por *host* e para cada *host* é exibido um sumário das portas varridas, esse sumário contém o número da porta, o serviço associado a ela e a severidade da vulnerabilidade encontrada nela.



Filtered Results 1 - 67 of 67

Host	OS	Start	End	High	Medium	Low	Log	False Pos.	Total
10.0.0.5		Oct 24, 11:59:16	(not finished)	45	22	0	0	0	67
Total: 1				45	22	0	0	0	67

Port summary for 10.0.0.5

Service (Port)	Threat
clm_pts (6200/tcp)	High
distcc (3632/tcp)	High
ftp (21/tcp)	High
http (80/tcp)	High

FIGURA 29: TABELA DE RESUMO E SUMÁRIO DE PORTAS

Abaixo do sumário de portas, são listadas as vulnerabilidades encontradas para o *host*. Cada uma das vulnerabilidades é exibida com a sua pontuação CVSS, descrição e referências. O campo de descrição é um campo de texto livre, que geralmente contém uma breve explicação da vulnerabilidade e o que pode ser feito para corrigi-la. O campo de referência normalmente contem o endereço de páginas onde a vulnerabilidade foi reportada, os identificadores CVE associados a vulnerabilidade e possivelmente o seu BID (identificador Bugtraq). O Bugtraq é uma lista de emails onde profissionais da área de segurança podem falar sobre vulnerabilidades, como explorá-las e se proteger delas (BUGTRAQ, 2006). Como exemplo, a Figura 30 e a Figura 31 contém a listagem de vulnerabilidades para um host, essa listagem foi retirada da varredura de vulnerabilidades feita externamente à rede da UFSC e contém apenas vulnerabilidades com nível de risco médio e alto.

2.[redacted] 150.162.60.[redacted]

Host scan start Sun Nov 9 03:42:13 2014 UTC
Host scan end Sun Nov 9 04:31:21 2014 UTC

Service (Port)	Threat Level
https (443/tcp)	High
general/tcp	Medium

2.[redacted].1 High https (443/tcp)

High (CVSS: 10.0)
NVT: Apache Multiple Security Vulnerabilities

```
Summary:
Apache is prone to multiple vulnerabilities.
These issues may lead to information disclosure or other attacks.
Apache versions prior to 2.2.15 are affected.
Solution:
Upgrade to  Apache 2.2.15 or Later.
```

OID of test routine: 1.3.6.1.4.1.25623.1.0.100514

References

```
CVE: CVE-2010-0425, CVE-2010-0434, CVE-2010-0408, CVE-2007-6750
BID:38494, 38491
Other:
  URL:http://www.securityfocus.com/bid/38494
   URL:http://httpd.apache.org/security/vulnerabilities_22.html
   URL:http://httpd.apache.org/
   URL:https://issues.apache.org/bugzilla/show_bug.cgi?id=48359
   URL:http://svn.apache.org/viewvc?view=revision&amp;revision=917870
```

High (CVSS: 6.8)
NVT: OpenSSL CCS Man in the Middle Security Bypass Vulnerability

OID of test routine: 1.3.6.1.4.1.25623.1.0.105042

References

```
CVE: CVE-2014-0224
BID:67899
Other:
  URL:http://www.securityfocus.com/bid/67899
   URL:http://openssl.org/
```

FIGURA 30: LISTAGEM DE VULNERABILIDADES PARA UM HOST 1

High (CVSS: 5.8)
NVT: http TRACE XSS attack

...continued from previous page ...

```
 Summary:
 Debugging functions are enabled on the remote HTTP server.
Description :
The remote webserver supports the TRACE and/or TRACK methods. TRACE and TRACK
are HTTP methods which are used to debug web server connections.
It has been shown that servers supporting this method are subject to
cross-site-scripting attacks, dubbed XST for Cross-Site-Tracing, when
used in conjunction with various weaknesses in browsers.
An attacker may use this flaw to trick your legitimate web users to give
him their credentials.
 Solution:
 Disable these methods.
Plugin output :
Solution:
Add the following lines for each virtual host in your configuration file :
    RewriteEngine on
    RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
    RewriteRule .* - [F]
See also http://httpd.apache.org/docs/current/de/mod/core.html#traceenable
```

OID of test routine: 1.3.6.1.4.1.25623.1.0.11213

References

```
CVE: CVE-2004-2320, CVE-2003-1567
BID:9506, 9561, 11604
Other:
  URL:http://www.kb.cert.org/vuls/id/867593
```

2.[redacted].2 Medium general/tcp

Medium (CVSS: 2.6)
NVT: TCP timestamps

```
It was detected that the host implements RFC1323.
The following timestamps were retrieved with a delay of 1 seconds in-between:
Paket 1: 573141994
Paket 2: 573142281
```

OID of test routine: 1.3.6.1.4.1.25623.1.0.80091

References

```
Other:
  URL:http://www.ietf.org/rfc/rfc1323.txt
```

FIGURA 31: LISTAGEM DE VULNERABILIDADES PARA UM HOST 2

4.2.1 Resultado da Varredura Interna a Rede Alvo

A primeira varredura de vulnerabilidades (interna a rede alvo) encontrou o total de 328 vulnerabilidades de alto risco, 275 de médio risco, 180 de baixo risco e 1675 *logs,* que são constatações feitas pelo sistema que podem ou não ser vulnerabilidades, no geral são de caráter apenas informacional. As vulnerabilidades encontradas estavam associadas a 238 identificadores CVE distintos que se repetiram num total de 2202 vezes. A Figura 32, contém uma visão geral dos resultados por *host* (alguns deles estão censurados a pedido do gerente da rede).

Host	Most Severe Result(s)	High	Medium	Low	Log	False Posit
150.162.60.2 (apolo.inf.ufsc.br)	Severity: Medium	0	1	1	12	0
150.162.60.3 (bdine.inf.ufsc.br)	Severity: Medium	0	1	1	10	0
150.162.60.4 (smtp2.inf.ufsc.br)	Severity: Medium	0	3	0	10	0
150.162.60.5 (ceres.inf.ufsc.br)	Severity: High	49	16	5	60	0
[redacted]						
150.162.60.12 (sol.inf.ufsc.br)	Severity: High	8	10	3	22	0
[redacted]						
150.162.60.14 (triton.inf.ufsc.br)	Severity: High	35	12	4	32	0
150.162.60.16 (classificados.inf.ufsc.br)	Severity: High	12	6	0	26	0
150.162.60.18 (webmail.inf.ufsc.br)	Severity: High	5	7	6	23	0
150.162.60.19 (wmail.inf.ufsc.br)	Severity: High	6	7	3	24	0
150.162.60.20 (java.inf.ufsc.br)	Severity: High	9	15	5	26	0
150.162.60.21 (www.inf.ufsc.br)	Severity: High	44	12	0	29	0
150.162.60.23 (euryale.inf.ufsc.br)	Severity: High	8	9	3	21	0
150.162.60.24 (svn.inf.ufsc.br)	Severity: High	6	7	4	23	0
150.162.60.25 (wwwexe.inf.ufsc.br)	Severity: High	15	19	3	26	0
150.162.60.26 (admrede1.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.27 (www2.inf.ufsc.br)	Severity: High	4	3	3	24	0
150.162.60.28 (admrede2.inf.ufsc.br)	Severity: High	8	10	4	34	0
150.162.60.29 (mail2.inf.ufsc.br)	Severity: High	2	1	1	18	0
150.162.60.30 (ldapn.inf.ufsc.br)	Severity: Medium	0	1	0	9	0
150.162.60.31 (rcmail.inf.ufsc.br)	Severity: High	4	5	3	23	0
150.162.60.34 (mv2.inf.ufsc.br)	Severity: Log	0	0	0	7	0
150.162.60.36 (admredeu.inf.ufsc.br)	Severity: Medium	0	2	2	14	0
[redacted]						
150.162.60.40 (maild.inf.ufsc.br)	Severity: Medium	0	1	4	16	0
150.162.60.41 (bdd.inf.ufsc.br)	Severity: Log	0	0	0	7	0
150.162.60.42 (admd.inf.ufsc.br)	Severity: High	1	4	5	25	0
150.162.60.43 (wwwd.inf.ufsc.br)	Severity: High	1	2	3	24	0
[redacted]						
150.162.60.65 (venus.inf.ufsc.br)	Severity: Log	0	0	0	4	0
150.162.60.66 (marte.inf.ufsc.br)	Severity: Log	0	0	0	4	0
150.162.60.67 (saturno.inf.ufsc.br)	Severity: Log	0	0	0	4	0
[redacted]						
150.162.60.129 (inca.inf.ufsc.br)	Severity: High	2	3	2	32	0
150.162.60.133 (tcc.inf.ufsc.br)	Severity: Medium	0	1	0	11	0
150.162.60.134 (plab.inf.ufsc.br)	Severity: Medium	0	2	3	21	0
150.162.60.140 (pegasus.inf.ufsc.br)	Severity: High	3	2	9	48	0
150.162.60.141 (gufsc.inf.ufsc.br)	Severity: High	16	2	2	26	0
150.162.60.144 (bixin.inf.ufsc.br)	Severity: Medium	0	1	1	14	0
150.162.60.147 (chuck.inf.ufsc.br)	Severity: High	1	2	2	32	0
150.162.60.148 (jigsaw.inf.ufsc.br)	Severity: Medium	0	3	2	22	0
150.162.60.151 (jason.inf.ufsc.br)	Severity: High	1	2	4	31	0
150.162.60.152 (liate.inf.ufsc.br)	Severity: Medium	0	3	3	25	0
150.162.60.154 (biblio.inf.ufsc.br)	Severity: High	2	7	9	58	0
150.162.60.155 (concerto.inf.ufsc.br)	Severity: High	3	14	2	32	0
150.162.60.157 (uzi.inf.ufsc.br)	Severity: High	2	2	2	37	0
150.162.60.158 (apilab.inf.ufsc.br)	Severity: Medium	0	5	4	25	0
150.162.60.161 (lisaserver.inf.ufsc.br)	Severity: High	5	14	9	68	0
150.162.60.164 (minicursos.inf.ufsc.br)	Severity: High	1	1	2	23	0
150.162.60.165 (servidor.iate.ufsc.br)	Severity: Medium	0	4	2	45	0
150.162.60.166 (saas2.inf.ufsc.br)	Severity: Medium	0	1	2	18	0
150.162.60.168 (saad.inf.ufsc.br)	Severity: High	5	5	6	51	0
150.162.60.172 (hannibal.inf.ufsc.br)	Severity: Medium	0	4	7	31	0
150.162.60.173 (dracula.inf.ufsc.br)	Severity: High	2	3	5	26	0
150.162.60.178 (servidort.iate.ufsc.br)	Severity: Medium	0	2	4	26	0
[redacted]						
150.162.60.180 (bksetic.inf.ufsc.br)	Severity: Medium	0	1	0	13	0
150.162.60.181 (saas.inf.ufsc.br)	Severity: Medium	0	1	2	25	0
150.162.60.182 (sestatnet.inf.ufsc.br)	Severity: High	1	1	1	24	0
150.162.60.183 (agil.inf.ufsc.br)	Severity: Medium	0	2	4	22	0
150.162.60.184 (ambiente.etec.ufsc.br)	Severity: High	2	2	3	24	0
150.162.60.185 (redmine.saas.etec.ufsc.br)	Severity: High	4	2	3	32	0
150.162.60.186 (projeto.etec.ufsc.br)	Severity: High	6	2	3	36	0
150.162.60.187 (tri.unasus.ufsc.br)	Severity: High	2	4	5	32	0
[redacted]						
Total: 76		328	275	180	1675	0

FIGURA 32: VISÃO GERAL DOS RESULTADOS DA VARREDURA DE VULNERABILIDADES INTERNA

Os 25 identificadores que mais apareceram durante a primeira varredura de vulnerabilidades estão exibidos no gráfico da Figura 33.

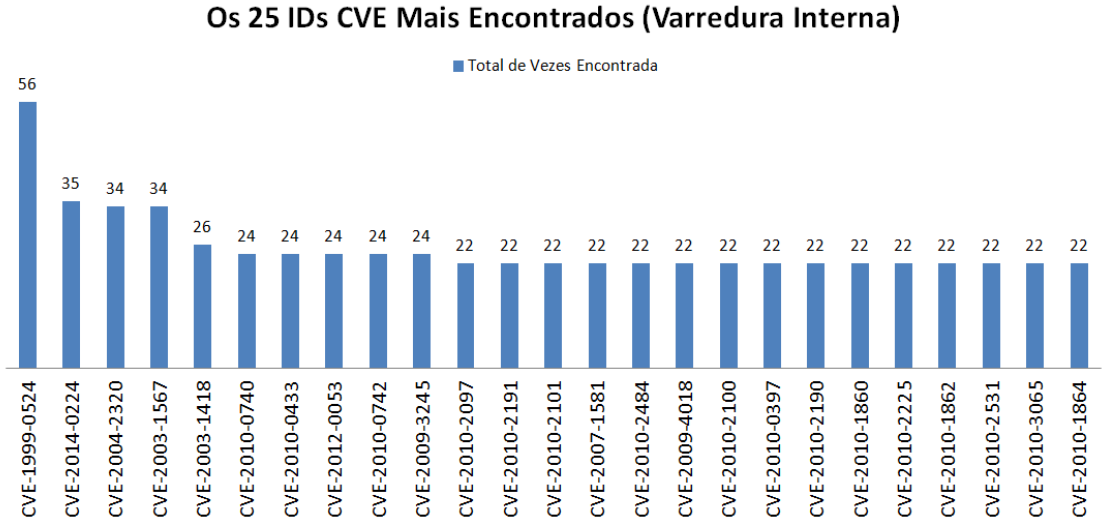


FIGURA 33: HISTROGRAMA DOS 25 IDENTIFICADORES CVE MAIS ENCONTRADOS (VARREDURA INTERNA)

Dos 25 identificadores CVE mais encontrados, 20 estão associados a vulnerabilidades de alta severidade alta, 4 a vulnerabilidades de média severidade e 1 deles (o que mais se repetiu) está associado a um resultado apenas informacional, uma possível vulnerabilidade. Uma explicação detalhada das vulnerabilidades associadas aos 25 identificadores CVE é dada logo a seguir.

CVE-1999-0524

- Pontuação CVSS: 0,0
- Classificação OpenVAS: Informativa

O identificador CVE mais encontrado durante a varredura está associado a uma possível vulnerabilidade de revelação de informação. Essa possível vulnerabilidade é devida ao fato do *host* varrido responder a requisições de ICMP de carimbos de tempo. A resposta para uma requisição de ICMP com carimbo de tempo contém o carimbo de tempo da requisição, um carimbo de tempo indicando quando a requisição foi recebida e um carimbo de tempo de quando a resposta foi enviada. Teoricamente seria possível explorar geradores de número aleatórios baseados em carimbo de tempo com essa informação e

assim explorar os serviços que utilizam esse tipo de gerador de número aleatório. A solução para essa vulnerabilidade é desabilitar a resposta para requisições ICMP de carimbo de tempo.

CVE-2014-0224

- Pontuação CVSS: 6,8
- Classificação OpenVAS: Severidade Alta

Servidores cuja versão da biblioteca OpenSSL é anterior a 1.0.1 e a 1.0.2-beta1 são vulneráveis a ataques do tipo *man-in-the-middle*. A vulnerabilidade encontrada nessa biblioteca pode ser explorada por um atacante através de uma solicitação de *handshake* desenvolvida com o objetivo de forçar que a chave simétrica gerada para a comunicação seja fraca e fácil de ser quebrada, o que possibilita o ataque *man-in-the-middle*. A solução para essa vulnerabilidade é atualização da biblioteca OpenSSL.

CVE-2004-2320 e CVE-2003-1567

- Pontuação CVSS: 5,8
- Classificação OpenVAS: Severidade Alta

De acordo com o relatório gerado pelo OpenVAS, servidores web que permitem o uso dos métodos TRACE ou TRACK estão sujeitos a ataques a ataques de XSS ou *cross-site-scripting*. Os métodos TRACE e TRACK são utilizados para testar conexões com servidores web, entretanto associados a vulnerabilidades encontradas em browsers é possível que um atacante realize um ataque XSS nesse caso chamado de XST (*cross-site-tracing*). Para solucionar essa vulnerabilidade é necessário desabilitar os métodos TRACE e TRACK nos servidores web.

CVE-2003-1418

- Pontuação CVSS: 4,3
- Classificação OpenVAS: Severidade Média

Um dos mecanismos para validação de cachê offline de páginas web é a ETag (*entity tag*). ETags tornam possível que clientes façam requisições

condicionais aos servidores para que esse retornem apenas conteúdo novo (CLAUSEN, 2004).

Apesar da economia de banda e entrega mais eficiente de conteúdo (uma vez que menos coisas precisam ser entregues ao cliente que possui cachê da página), servidores Apache possuem uma fraqueza associada ao uso de Etags. Quando servidores Apache utilizam Etags eles acabam disponibilizando informações sensíveis referentes aos arquivos hospedados no servidor. Mais especificamente, o cabeçalho das respostas ETag geradas pelo Apache contém o número inode do arquivo associado a ETag. Em versões do Apache iguais ou maiores a 1.3.27 ETags podem ser utilizadas de maneira segura se a diretiva "FileETag –Inode" for adicionada ao arquivo de configuração do servidor. Caso a versão do servidor seja inferior a 1.3.27 e o uso de ETags seja necessário, então uma atualização do servidor é recomendada (NOVELL, 2004).

CVE-2010-0740 e CVE-2010-0433

- Pontuação CVSS: 5,0
- Classificação OpenVAS: Severidade Média

Serviços que estabelecem conexões seguras utilizando a biblioteca OpenSSL podem sofrer ataques de negação de serviço por culpa de uma vulnerabilidade causada pela referência feita a um ponteiro que em determinado momento pode ser nulo. Se a versão da biblioteca estiver entre 0.9.8f e 0.9.8m então ela pode ser afetada por essa vulnerabilidade. De acordo com o site do OpenSSL a melhor solução para essa vulnerabilidade é a atualização da biblioteca, entretanto caso isso não seja possível, existe um *patch* que corrige o problema, esse *patch* é disponibilizado no site oficial da ferramenta (OPENSSL, 2010).

CVE-2012-0053

- Pontuação CVSS: 4,3
- Classificação OpenVAS: Severidade Média

De acordo com o NIST (2012), servidores Apache nas versões 2.2.x até a versão 2.2.21 não restringem de maneira correta as informações no cabeçalho de resposta durante a criação de documentos de erro de má requisição (código 400), o que permite a um atacante obter os valores de cookies to tipo HTTPOnly. Segundo a OWASP, cookies do tipo HTTPOnly são cookies que contém informações sensívies, como por exemplo as referentes a sessão, e que não podem ser acessados por scripts, mas apenas pelos browsers. Esse tipo de vulnerabilidade permite a realização de XSS. A solução do problema é a atualização do software para uma versão mais recente.

CVE-2010-0742

- Pontuação CVSS: 7,5
- Classificação OpenVAS: Severidade Alta

Versões da biblioteca OpenSSL anteriores a 0.9.8o/1.0.0a são vulneráveis a corrupção de memória, sendo possível que atacantes ao utilizar certas estruturas em suas requisições consigam comprometer aplicações dependentes da biblioteca e assim permitir a execução remota de códigos arbitrários. Caso o ataque não obtenha sucesso é possível que o mesmo tenha como efeito colateral a negação de serviço. A solução para essa vulnerabilidade é a atualização da biblioteca OpenSSL para uma das versões superiores a as afetadas por essa vulnerabilidade.

CVE-2009-3245

- Pontuação CVSS: 10
- Classificação OpenVAS: Severidade Alta

De acordo com a página oficial da biblioteca OpenSSL, todas as versões da biblioteca da série 0.9.8x inferiores a versão 0.9.8m são vulneráveis a uma falha de alocação de memória. Essa falha é ocasionada pelo fato de que no código da aplicação o resultado da função bn_wexpand() nem sempre é

checado, podendo fazendo com que seja possível o crash da aplicação até mesmo a execução remota de código arbitrário. Novamente a correção do problema passa pela atualização da biblioteca OpenSSL.

Códigos CVE com 22 Ocorrências

- Pontuação CVSS: Variada
- Classificação OpenVAS: Severidade Alta

Os identificadores CVE com 22 ocorrências apontam para diversas vulnerabilidades associadas ao fato dos servidores estarem usando uma versão da linguagem php desatualizada. Muitas delas podem comprometer o conteúdo de um sistema feito na linguagem e até mesmo permitir a execução de código remoto na máquina atacada. A solução para as vulnerabilidades associadas a esses identificadores CVE é a atualização da linguagem para uma versão mais recente.

4.2.2 Resultados da Varredura Externa

A segunda varredura de vulnerabilidades foi feita externa a rede alvo e a rede da UFSC. Era esperado que ela levasse mais tempo para ser executada do que a primeira varredura devido a latência da rede, entretanto com um número de serviços reduzidos para serem testados ela levou aproximadamente 5 horas.

Como resultado, a varredura de vulnerabilidades externa encontrou o total de 253 vulnerabilidades de alto risco, 202 de médio risco, 78 de baixo risco e 1098 *logs*. As vulnerabilidades encontradas estavam associadas a 226 identificadores CVE distintos que se repetiram num total de 1745 vezes. A Figura 34, contém uma visão geral dos resultados por *host*.

Host	Most Severe Result(s)	High	Medium	Low	Log	False Posit
150.162.60.2 (apolo.inf.ufsc.br)	Severity: Log	0	0	0	7	0
150.162.60.3 (bdine.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.5 (ceres.inf.ufsc.br)	Severity: High	49	17	4	58	0
[redacted]						
150.162.60.12 (sol.inf.ufsc.br)	Severity: High	10	10	1	20	0
[redacted]						
150.162.60.14 (triton.inf.ufsc.br)	Severity: High	18	6	0	17	0
150.162.60.16 (classificados.inf.ufsc.br)	Severity: High	14	7	0	26	0
150.162.60.18 (webmail.inf.ufsc.br)	Severity: High	6	8	6	23	0
150.162.60.19 (wmail.inf.ufsc.br)	Severity: High	7	7	3	24	0
150.162.60.20 (java.inf.ufsc.br)	Severity: High	12	17	3	26	0
150.162.60.21 (www.inf.ufsc.br)	Severity: High	21	5	1	20	0
150.162.60.23 (euryale.inf.ufsc.br)	Severity: High	10	9	1	18	0
150.162.60.24 (svn.inf.ufsc.br)	Severity: High	8	8	4	23	0
150.162.60.25 (wwwexe.inf.ufsc.br)	Severity: High	13	17	2	27	0
150.162.60.28 (admrede2.inf.ufsc.br)	Severity: High	7	5	3	32	0
150.162.60.29 (mail2.inf.ufsc.br)	Severity: High	2	1	1	15	0
150.162.60.30 (ldapn.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.31 (rcmail.inf.ufsc.br)	Severity: Medium	0	1	2	12	0
150.162.60.34 (mv2.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.36 (admredeu.inf.ufsc.br)	Severity: Log	0	0	0	8	0
[redacted]						
150.162.60.40 (ceresd.inf.ufsc.br)	Severity: Medium	0	1	2	44	0
150.162.60.41 (bdd.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.42 (admd.inf.ufsc.br)	Severity: High	3	1	1	18	0
150.162.60.43 (wwwd.inf.ufsc.br)	Severity: Medium	0	3	3	24	0
[redacted]						
150.162.60.65 (venus.inf.ufsc.br)	Severity: Log	0	0	0	10	0
150.162.60.66 (marte.inf.ufsc.br)	Severity: Log	0	0	0	8	0
[redacted]						
150.162.60.129 (inca.inf.ufsc.br)	Severity: Log	0	0	0	9	0
150.162.60.133 (tcc.inf.ufsc.br)	Severity: High	1	3	6	21	0
150.162.60.134 (plab.inf.ufsc.br)	Severity: Medium	0	2	1	16	0
150.162.60.140 (pegasus.inf.ufsc.br)	Severity: Log	0	0	0	13	0
150.162.60.141 (gufsc.inf.ufsc.br)	Severity: High	8	3	1	27	0
150.162.60.144 (bixin.inf.ufsc.br)	Severity: Medium	0	1	1	11	0
150.162.60.151 (jason.inf.ufsc.br)	Severity: Log	0	0	0	9	0
150.162.60.152 (liate.inf.ufsc.br)	Severity: Medium	0	5	1	27	0
150.162.60.154 (biblio.inf.ufsc.br)	Severity: High	2	5	3	35	0
150.162.60.155 (concerto.inf.ufsc.br)	Severity: High	1	9	1	27	0
150.162.60.157 (uzi.inf.ufsc.br)	Severity: Medium	0	1	1	17	0
150.162.60.158 (apilab.inf.ufsc.br)	Severity: Log	0	0	0	7	0
150.162.60.161 (lisaserver.inf.ufsc.br)	Severity: High	3	11	6	41	0
150.162.60.164 (minicursos.inf.ufsc.br)	Severity: High	1	1	2	21	0
150.162.60.165 (servidor.iate.ufsc.br)	Severity: Medium	0	1	1	16	0
150.162.60.166 (saas2.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.168 (saad.inf.ufsc.br)	Severity: Medium	0	2	5	23	0
150.162.60.178 (servidort.iate.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.180 (bksetic.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.181 (saas.inf.ufsc.br)	Severity: Medium	0	1	0	20	0
150.162.60.182 (sestatnet.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.183 (agil.inf.ufsc.br)	Severity: Log	0	0	0	8	0
150.162.60.184 (ambiente.etec.ufsc.br)	Severity: Medium	0	2	2	15	0
150.162.60.185 (redmine.saas.etec.ufsc.br)	Severity: Medium	0	1	0	12	0
150.162.60.186 (projeto.etec.ufsc.br)	Severity: High	1	1	2	13	0
150.162.60.187 (tri.unasus.ufsc.br)	Severity: High	2	5	2	23	0
[redacted]						
Total: 65		253	202	78	1098	0

FIGURA 34: VISÃO GERAL DOS RESULTADOS DA VARREDURA DE VULNERABILIDADES EXTERNA

Os 25 identificadores que mais apareceram durante a varredura de vulnerabilidades externa estão exibidos no gráfico da Figura 35: Histograma dos 25 Identificadores CVE Mais Encontrados (Varredura Externa).

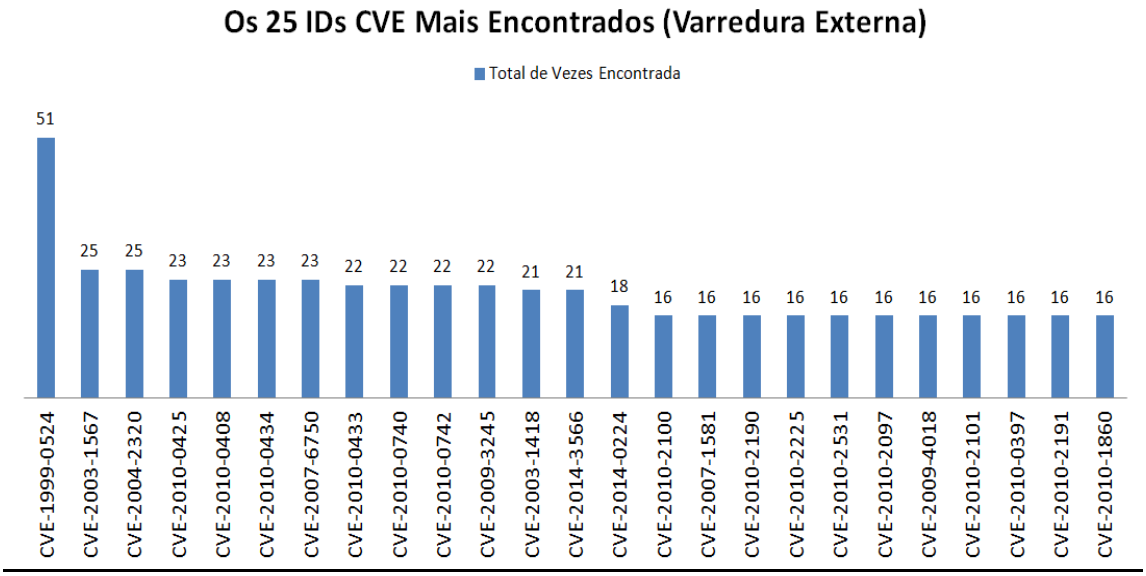


FIGURA 35: HISTOGRAMA DOS 25 IDENTIFICADORES CVE MAIS ENCONTRADOS (VARREDURA EXTERNA)

Desses identificadores apenas 4 não estão presentes entre os 25 mais encontrados na varredura interna. São esses os identificadores: CVE-2010-0408, CVE-2010-0425, CVE-2010-0434 e CVE-2007-6750.

CVE-2010-0408, CVE-2010-0425, CVE-2010-0434 e CVE-2007-6750

- Pontuação CVSS: 10
- Classificação OpenVAS: Severidade Alta

Esse 4 identificadores estão ligados a uma vulnerabilidade em versões do servidor Apache inferiores a 2.2.15. A vulnerabilidade encontrada é uma de corrupção de memória. É possível que um atacante utilize dessa vulnerabilidade para realizar negação de serviço ou até mesmo que ele execute código remoto com privilégios de sistema . A solução para esse problema é a atualização do servidor apache para uma versão mais recente (BUGTRAQ, 2014).

4.2.3 Resultados da Varredura Feita a Partir do Laboratório de Redes e Gerência (LRG)

A terceira e última varredura de vulnerabilidades foi feita a partir da rede do Laboratório de Redes e Gerência (LRG), como dito anteriormente a rede do LRG fica fora da rede alvo porém dentro da rede da UFSC. A última varredura tomou um pouco mais do que 8 horas para ser realizada, e foi obtido como resultado o total de 124 vulnerabilidades de alto risco, 164 de médio risco, 46 de baixo risco e 1321 *logs*. As vulnerabilidades encontradas estavam associadas a 124 identificadores CVE distintos que se repetiram num total de 794 vezes. A Figura 36, contém uma visão geral dos resultados por *host*.

Host	Most Severe Result(s)	High	Medium	Low	Log	False Positiv
150.162.60.2	Severity: Log	0	0	0	4	0
150.162.60.3	Severity: Log	0	0	0	5	0
150.162.60.5	Severity: High	10	3	3	45	0
[redacted]						
150.162.60.12	Severity: High	8	7	0	22	0
[redacted]						
150.162.60.14	Severity: High	15	3	1	19	0
150.162.60.16	Severity: High	9	4	2	24	0
150.162.60.18	Severity: High	4	4	3	16	0
150.162.60.19 (wmail.inf.ufsc.br)	Severity: High	8	6	2	25	0
150.162.60.20 (java.inf.ufsc.br)	Severity: High	11	15	1	27	0
150.162.60.21	Severity: High	7	1	0	19	0
150.162.60.23	Severity: High	4	6	0	17	0
150.162.60.24	Severity: High	4	4	1	18	0
150.162.60.25	Severity: High	5	9	0	18	0
150.162.60.26	Severity: Log	0	0	0	10	0
150.162.60.28	Severity: Medium	0	3	1	28	0
150.162.60.29	Severity: Low	0	0	1	19	0
150.162.60.30 (ldapn.inf.ufsc.br)	Severity: Medium	0	1	0	12	0
150.162.60.31 (rcmail.inf.ufsc.br)	Severity: High	6	5	0	23	0
150.162.60.34 (mv2.inf.ufsc.br)	Severity: Log	0	0	0	10	0
150.162.60.36 (admredeu.inf.ufsc.br)	Severity: Medium	0	2	0	17	0
[redacted]						
150.162.60.40 (ceresd.inf.ufsc.br)	Severity: Medium	0	1	3	43	0
150.162.60.41 (bdd.inf.ufsc.br)	Severity: Log	0	0	0	10	0
150.162.60.42 (admd.inf.ufsc.br)	Severity: High	3	2	3	28	0
150.162.60.43 (wwwd.inf.ufsc.br)	Severity: Medium	0	3	1	23	0
[redacted]						
150.162.60.65 (venus.inf.ufsc.br)	Severity: Medium	0	1	0	13	0
150.162.60.66 (marte.inf.ufsc.br)	Severity: Medium	0	1	0	10	0
[redacted]						
150.162.60.129	Severity: Medium	0	1	0	17	0
150.162.60.133 (tcc.inf.ufsc.br)	Severity: High	1	3	3	26	0
150.162.60.134	Severity: Medium	0	2	0	18	0
150.162.60.140 (pegasus.inf.ufsc.br)	Severity: Medium	0	1	0	15	0
150.162.60.141 (gufsc.inf.ufsc.br)	Severity: High	8	1	0	26	0
150.162.60.144 (bixin.inf.ufsc.br)	Severity: Medium	0	1	1	16	0
150.162.60.151 (jason.inf.ufsc.br)	Severity: Medium	0	1	0	15	0
150.162.60.152 (liate.inf.ufsc.br)	Severity: Medium	0	5	2	25	0
150.162.60.154	Severity: High	1	8	1	42	0
150.162.60.155 (concerto.inf.ufsc.br)	Severity: High	3	13	0	31	0
150.162.60.157 (uzi.inf.ufsc.br)	Severity: Medium	0	1	1	19	0
150.162.60.158 (apilab.inf.ufsc.br)	Severity: High	2	3	0	24	0
150.162.60.161 (lisaserver.inf.ufsc.br)	Severity: High	3	12	1	43	0
150.162.60.164 (minicursos.inf.ufsc.br)	Severity: High	1	1	1	24	0
150.162.60.165 (servidor.iate.ufsc.br)	Severity: Medium	0	2	0	27	0
150.162.60.166 (saas2.inf.ufsc.br)	Severity: Medium	0	1	0	20	0
150.162.60.168	Severity: Medium	0	3	2	29	0
150.162.60.178 (servidort.iate.ufsc.br)	Severity: Medium	0	1	1	19	0
150.162.60.180 (bksetic.inf.ufsc.br)	Severity: Medium	0	1	0	14	0
150.162.60.181 (saas.inf.ufsc.br)	Severity: Medium	0	1	0	27	0
150.162.60.182 (sestatnet.inf.ufsc.br)	Severity: High	1	2	0	25	0
150.162.60.183 (agil.inf.ufsc.br)	Severity: Medium	0	2	0	20	0
150.162.60.184 (ambiente.etec.ufsc.br)	Severity: Medium	0	2	1	22	0
150.162.60.185 (redmine.saas.etec.ufsc.br)	Severity: High	1	1	0	20	0
150.162.60.186 (projeto.etec.ufsc.br)	Severity: High	1	1	0	19	0
150.162.60.187 (tri.unasus.ufsc.br)	Severity: High	2	5	1	35	0
[redacted]						
Total: 66		124	164	46	1321	0

FIGURA 36: VISÃO GERAL DOS RESULTADOS DA VARREDURA DE VULNERABILIDADES FEITA DO LRG

Os 25 identificadores que mais apareceram durante a varredura de vulnerabilidades feita a partir do LRG estão exibidos no gráfico da

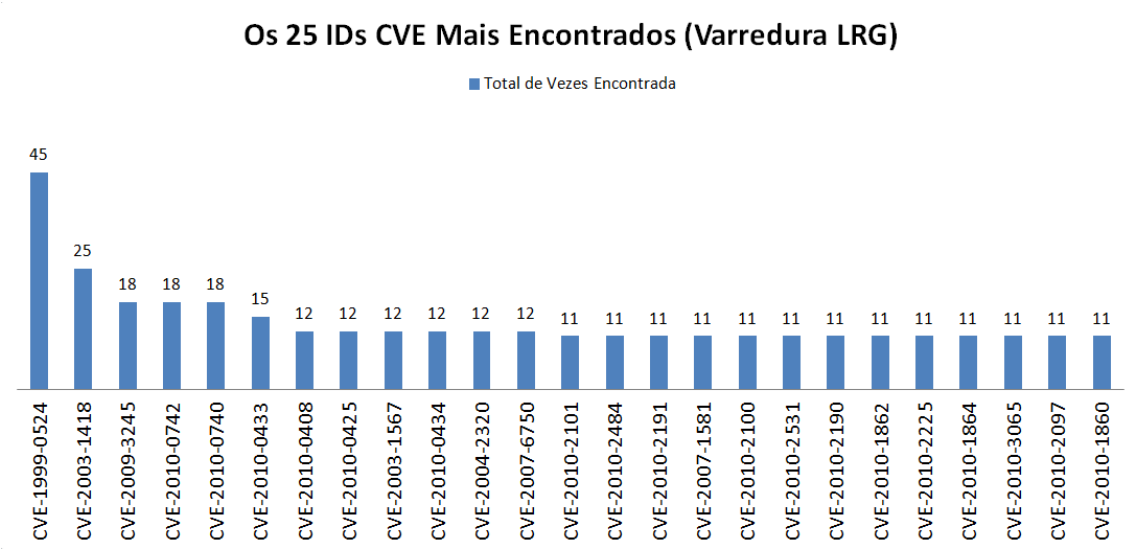


FIGURA 37: HISTOGRAMA DOS 25 IDENTIFICADORES CVE MAIS ENCONTRADOS (VARREDURA LRG)

Desses 25 identificadores, 5 não aparecem na listagem dos 25 mais encontrados na varredura de vulnerabilidades interna, porém se comparado a varredura de vulnerabilidades externa a diferença entre os 25 identificadores mais encontrados cai para 1. Os identificadores diferentes são os seguintes: CVE-2010-0408, CVE-2010-0425, CVE-2010-0434, CVE-2007-6750 (presentes nos 25 mais encontrados da varredura externa porém não nos da interna) e CVE-2014-3566 (exclusivo entre os 25 mais encontrados da varredura feita a partir do LRG).

CVE-2014-3566

- Pontuação CVSS: 4,3
- Classificação OpenVAS: Severidade Média

Essa vulnerabilidade está associada ao fato de que um servidor possibilita que conexões SSL sejam estabelecidas utilizando a versão SSLv3 do protocolo com modo de cifra do tipo CBC. É conhecido que esse tipo de conexão está sujeita a uma vulnerabilidade de vazamento de informações.

Para resolver o problema o servidor deve ser configurado para não possibilitar conexões SSL como as descritas acima.

4.2.4 Conclusões das Varreduras de Vulnerabilidades

O número de vulnerabilidades encontrado nas máquinas varridas nesse experimento é bastante alto, além disso boa parte dos *host*s possui alguma vulnerabilidade considerada grave pela ferramenta OpenVAS. É interessante ressaltar porém que para boa parte dessas vulnerabilidades não há expoloits prontos para uso na web nem em ferramentas de exploração como o Metasploit. No entanto esse não é um motivo para que nenhuma atitude seja tomada pelos responsáveis por cada um dos *host*s. Como demonstrado na listagem dos 25 identificadores CVE mais encontrados, para grande parte das vulnerabilidades a atualização das ferramentas envolvidas é o bastante para sanar o problema, sendo essa uma atitude que deveria ser encorajada.

4.3 Exploração de Vulnerabilidades

A penúltima etapa do teste de penetração é a exploração de vulnerabilidades. Para realização dessa etapa foi escolhida a ferramenta Metasploit Framework com a interface interativa MSFconsole.

Dentro do MSFconsole é possível realizar buscas por exploits relativos aos identificadores CVE e do Bugtraq que foram encontrados durante a varredura de vulnerabilidades. Para realizar uma busca por cve basta utilizar o comando “search” dentro do MSFconsole com o operador “cve:<identificador-cve>”, já para buscar pelo identificador Bugtraq se realiza um pesquisa com o comando “search” associado ao operador “bid:<identificador-bugtraq>”. Além da possibilidade de se pesquisar pelos identificadores a ferramenta também permite uma busca livre por texto, nesse caso basta usar o comando “search <texto_livre>”. Um exemplo de busca livre pode ser visto na Figura 38.

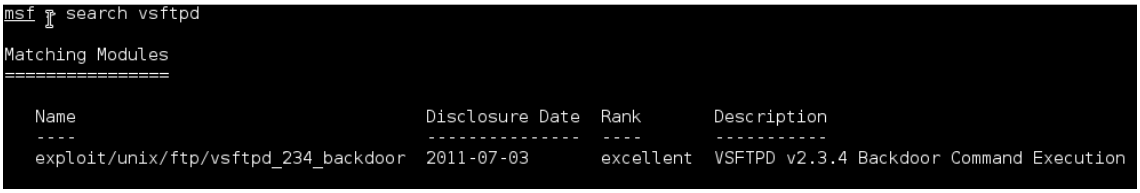


FIGURA 38: EXEMPLO DE BUSCA COM TEXTO LIVRE

Uma vez que é encontrado um *exploit* para a vulnerabilidade do sistema alvo o próximo passo é utilizá-la. Para utilizar um *exploit*, é usado o comando “use” seguido do nome do *exploit* (nesse caso o caminho completo até ele, como pode ser visto na figura 26). Uma vez que o comando use foi digitado, a ferramenta entra no contexto do módulo que está sendo utilizado. Alguns comandos são sensíveis a contexto, como por exemplo o comando “info”. Quando o comando “info” é usado dentro do contexto de um módulo ele irá exibir todas as informações relativas ao módulo do contexto atual. Entre as informações exibidas pelo comando info estão os parâmetros para a execução do *exploit*, que são listados abaixo do título “*Basic options*” como na Figura 39.

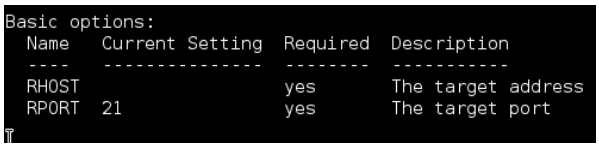


FIGURA 39: LISTAGEM DOS PARÂMETROS NECESSÁRIOS PARA EXECUÇÃO DE UM EXPLOIT

Outro comando sensível a contexto é o comando “show payloads”, quando esse comando é executado dentro do contexto de um módulo ele vai exibir apenas os *payloads* compatíveis com o módulo atual. Por exemplo, observe na Figura 40 a saída do comando “show payloads” dentro do contexto do *exploit* “vsftpd_234_backdoor”, que vem sido utilizado até agora como exemplo e a saída do mesmo comando dentro do contexto padrão do MSFconsole.

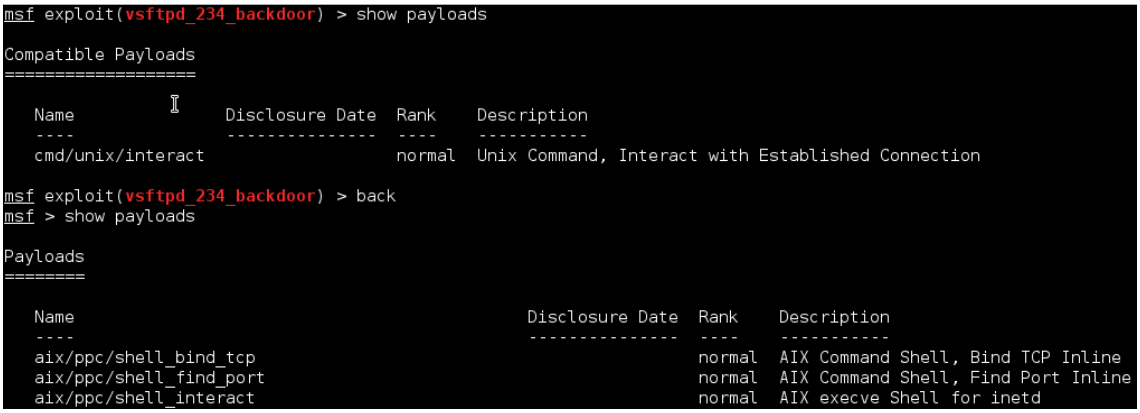


FIGURA 40: COMANDO "SHOW PAYLOADS" EM CONTEXTOS DIFERENTES

Para definir o valor dos parâmetros que serão utilizados na exploração é utilizado o comando “set”. O comando “set” é utilizado da seguinte maneira “set <nome do parâmetro> <valor>”, um exemplo do uso desse comando é mostrado na Figura 41.

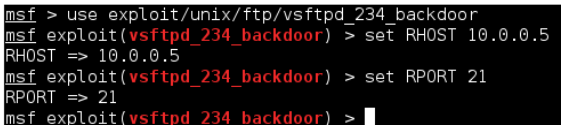


FIGURA 41: DEFINIÇÃO DOS VALORES DOS PARÂMETROS DE UM EXPLOIT

Com os valores dos parâmetros definidos o que resta a fazer é executar o *exploit* com um comando que leva esse mesmo nome “exploit”. Quando acionado o comando “exploit”, realiza automaticamente a exploração do sistema alvo, caso a exploração tenha sucesso o *payload* é executado logo em seguida. Na figura 30, o *exploit* “vsftpd_234_backdoor” é executado com sucesso e como conseqüência o *payload* associado a ele é executado. Nesse caso o *payload* é o primeiro da figura 28, “cmd/unix/Interact”, esse *payload* inicia um *shell* Unix interativo, que é como qualquer *shell* normal de um sistema do tipo *NIX. Devido ao *exploit* que foi utilizado o usuário associado ao *shell* aberto é o usuário root, fazendo com que o processo de exploração nessa máquina esteja completo (Figura 42).

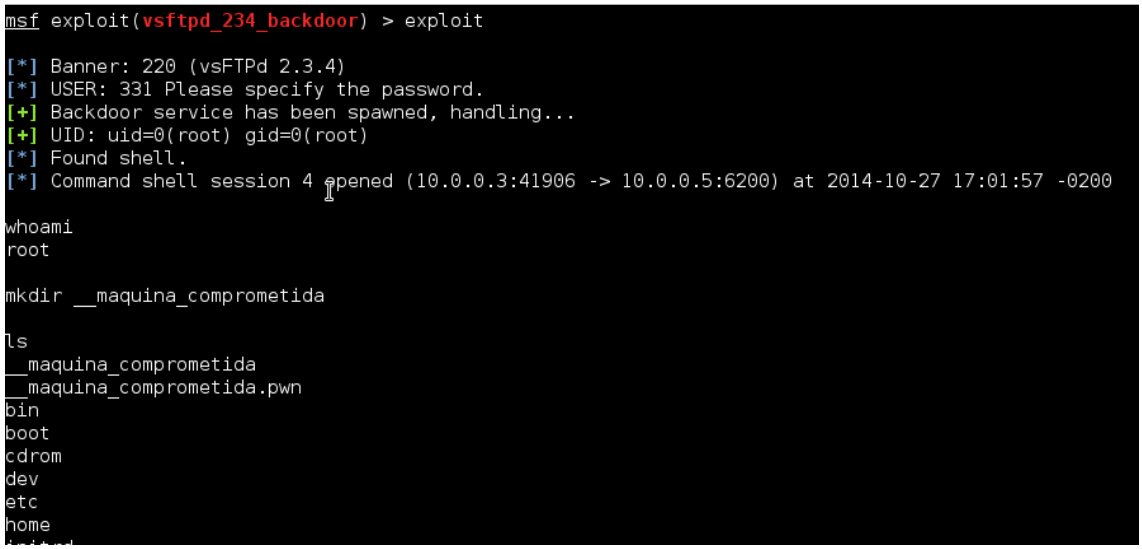


FIGURA 42: SHELL RESULTANTE DS EXPLORAÇÃO DO SISTEMA ALVO

4.3.1 Resultado da Fase de Exploração de Vulnerabilidades na Rede do INE

Quando o teste foi proposto ao gerente da rede do departamento de informática algumas condições foram estabelecidas, entre elas estavam as condição de que os testes não deveriam ser obstrutivos e nem deveria gerar instabilidade na rede. Em razão disso as vulnerabilidades encontradas que possuíam como falha a possibilidade de negação de serviço não foram exploradas. Além das vulnerabilidades de negação de serviço, as vulnerabilidades que possibilitavam ataques do tipo *man-in-the-middle* também não foram exploradas, pois não era conhecido o caráter das mensagem trocadas pelas máquinas testadas. Por fim pelo fato do foco do trabalho ser o teste de penetração através da exploração de serviços de rede, vulnerabilidades que possibilitam XSS e injeção de SQL também não foram exploradas.

O fato de grande parte das vulnerabilidades encontradas fazerem parte das categorias apresentadas no parágrafo anterior, fez com que o número de vulnerabilidades interessantes para esse teste fosse bastante reduzido. Sendo que não foram encontrados *exploits* para pronto uso relacionados a tais vulnerabilidades.

Apesar disso a rede possui diversas vulnerabilidades que deveriam ser remediadas o quanto antes, pois atacantes reais não respeitarem limites estabelecidos em contrato.

4.4 Documentação

4.4.1 Sumário Executivo do Teste de Penetração Realizado na Rede 150.162.60.0

O teste de penetração realizado na rede 150.162.60.0, teve como alvo 76 equipamentos como computadores e roteadores conectados a rede, e nele todos esses equipamentos tiveram a sua segurança testada.

O teste de penetração tem como objetivo descobrir que tipo de software roda em cada um dos computadores testados, e se esses softwares possuem falhas de segurança que podem ser exploradas por um atacante.

O teste que foi realizado nessa pequena parte da rede do departamento e obteve resultados bastante preocupantes. Nele foi constatado que a rede testada tem mais de 700 vulnerabilidades na sua segurança.

Apesar do grande número de vulnerabilidades de segurança encontradas nos equipamentos testados, o teste não conseguiu ganhar acesso a nenhum deles. Entretanto esse teste não englobou diversas modalidades de ataque que poderiam causar a negação do serviço oferecido pelas máquinas, a interceptação de comunicações, nem o roubo de credenciais de acesso aos sistemas hospedados neles.

O resultado do teste aponta que é aconselhável a correção dos problemas encontrados com certa urgência, dado o grande número de vulnerabilidades de alto risco encontradas (que na sua maioria são fáceis de serem sanadas).

Maiores detalhes sobre os problemas encontrados podem ser encontrados no relatório técnico detalhado e nos dados extraídos das ferramentas utilizadas.

4.4.2 Relatório Técnico do Teste de Penetração Realizado na Rede 150.162.60.0

4.4.2.1 Introdução

O teste de penetração realizado na rede 150.162.60.0, teve como alvo 76 *host*s encontrados ativos, os endereços associados aos *host*s testados são:

- 150.162.60.2
- 150.162.60.3
- 150.162.60.4
- 150.162.60.5
- 150.162.60.6
- 150.162.60.12
- 150.162.60.13
- 150.162.60.14
- 150.162.60.16
- 150.162.60.18
- 150.162.60.19
- 150.162.60.20
- 150.162.60.21
- 150.162.60.23
- 150.162.60.24
- 150.162.60.25
- 150.162.60.26
- 150.162.60.27
- 150.162.60.28
- 150.162.60.29
- 150.162.60.30
- 150.162.60.31
- 150.162.60.34
- 150.162.60.36
- 150.162.60.38
- 150.162.60.40
- 150.162.60.41
- 150.162.60.42
- 150.162.60.43
- 150.162.60.60
- 150.162.60.62
- 150.162.60.65
- 150.162.60.66
- 150.162.60.67
- 150.162.60.94
- 150.162.60.103
- 150.162.60.106
- 150.162.60.118
- 150.162.60.119
- 150.162.60.120
- 150.162.60.121
- 150.162.60.122
- 150.162.60.123
- 150.162.60.126
- 150.162.60.129
- 150.162.60.133
- 150.162.60.134
- 150.162.60.140
- 150.162.60.141
- 150.162.60.144
- 150.162.60.147
- 150.162.60.148
- 150.162.60.151
- 150.162.60.152
- 150.162.60.154
- 150.162.60.155
- 150.162.60.157
- 150.162.60.158
- 150.162.60.161
- 150.162.60.164
- 150.162.60.165
- 150.162.60.166
- 150.162.60.168
- 150.162.60.172
- 150.162.60.173
- 150.162.60.178
- 150.162.60.179
- 150.162.60.180
- 150.162.60.181
- 150.162.60.182
- 150.162.60.183
- 150.162.60.184
- 150.162.60.185
- 150.162.60.186

- 150.162.60.187
- 150.162.60.254

O objetivo do teste realizado era o de encontrar possíveis vulnerabilidades de segurança associadas aos serviços de rede (SMPT, SSH, HTTP, etc.) e tentar explorá-las para ganhar acesso aos *host*s. A exploração de vulnerabilidades associadas a sistemas web, injeção de SQL, XSS, ataques *man-in-the-middle* e negação de serviço não fez parte do escopo do presente teste, entretanto essas vulnerabilidades são reportadas aqui e no conjunto "cru" de dados gerados pelas ferramentas utilizadas.

4.4.2.2 Vulnerabilidades Encontradas

Foram encontradas mais de 700 vulnerabilidades nos *host*s testados. A varredura em busca de vulnerabilidades foi realizada com a ferramenta OpenVAS. As vulnerabilidades encontradas por essa ferramenta de serem classificadas pelo risco (alto, médio, baixo e apenas informativo) também apresentam:

- Texto de descrição das vulnerabilidades.
- Nível de correspondência com qualquer identificador CVE, os níveis são: parcial, total ou ambos. CVE é sigla para *Common Vulnerabilities and Exposures* (vulnerabilidades e exposições comuns). O CVE, de acordo com a sua página oficial, é um dicionário internacional, gratuito e de uso público que contém vulnerabilidades de conhecimento público. Um identificador CVE é um código que identifica unicamente uma vulnerabilidade.
- Valor do CVSS associado a uma vulnerabilidade. CVSS ou *Common Vulnerability Scoring System* (sistema comum para pontuação de vulnerabilidades), é um sistema de pontuação desenvolvido com o objetivo de disponibilizar um método aberto e padronizado para atribuir uma pontuação para vulnerabilidades de acordo com a sua severidade, a pontuação CVSS vai de 0 até 10. O CVSS é responsabilidade do NIST, e todas as vulnerabilidades catalogadas com CVE possuem uma pontuação CVSS.

- Identificador Bugtraq (BID): O Bugtraq é uma famosa lista de emails onde profissionais da área de segurança podem falar sobre vulnerabilidades, como explorá-las e se proteger delas.

4.4.2.3 As Vulnerabilidades de Alto Risco mais Vezes Encontradas

CVE-2014-0224

- Pontuação CVSS: 6,8
- Total de Vezes Encontrada: 35

Servidores cuja versão da biblioteca OpenSSL é anterior a 1.0.1 e a 1.0.2-beta1 são vulneráveis a ataques do tipo *man-in-the-middle*. A vulnerabilidade encontrada nessa biblioteca pode ser explorada por um atacante através de uma solicitação de *handshake* desenvolvida com o objetivo de forçar que a chave simétrica gerada para a comunicação seja fraca e fácil de ser quebrada, o que possibilita o ataque *man-in-the-middle*. A solução para essa vulnerabilidade é atualização da biblioteca OpenSSL.

CVE-2004-2320 e CVE-2003-1567

- Pontuação CVSS: 5,8
- Total de Vezes Encontrada: 34

De acordo com o relatório gerado pelo OpenVAS, servidores web que permitem o uso dos métodos TRACE ou TRACK estão sujeitos a ataques a ataques de XSS ou *cross-site-scripting*. Os métodos TRACE e TRACK são utilizados para testar conexões com servidores web, entretanto associados a vulnerabilidades encontradas em browsers é possível que um atacante realize um ataque XSS nesse caso chamado de XST (*cross-site-tracing*). Para solucionar essa vulnerabilidade é necessário desabilitar os métodos TRACE e TRACK nos servidores web.

CVE-2010-0742

- Pontuação CVSS: 7,5
- Total de Vezes Encontrada: 24

Versões da biblioteca OpenSSL anteriores a 0.9.8o/1.0.0a são vulneráveis a corrupção de memória, sendo possível que atacantes ao utilizar certas estruturas em suas requisições consigam comprometer aplicações dependentes da biblioteca e assim permitir a execução remota de códigos arbitrários. Caso o ataque não obtenha sucesso é possível que o mesmo tenha como efeito colateral a negação de serviço. A solução para essa vulnerabilidade é a atualização da biblioteca OpenSSL para uma das versões superiores a as afetadas por essa vulnerabilidade.

CVE-2009-3245

- Pontuação CVSS: 10
- Total de Vezes Encontrada: 24

De acordo com a página oficial da biblioteca OpenSSL, todas as versões da biblioteca da série 0.9.8x inferiores a versão 0.9.8m são vulneráveis a uma falha de alocação de memória. Essa falha é ocasionada pelo fato de que no código da aplicação o resultado da função bn_wexpand() nem sempre é checado, podendo fazendo com que seja possível o crash da aplicação até mesmo a execução remota de código arbitrário. Novamente a correção do problema passa pela atualização da biblioteca OpenSSL.

CVE-2010-2097,CVE-2010-2191,CVE-2010-2101,CVE-2007-1581,CVE-2010-2484,CVE-2009-4018,CVE-2010-2100,CVE-2010-0397,CVE-2010-2190,CVE-2010-1860,CVE-2010-2225,CVE-2010-1862,CVE-2010-2531,CVE-2010-3065,CVE-2010-1864

- Pontuação CVSS: Variada
- Total de Vezes Encontrada: 22 vezes em média

Esses identificadores apontam para diversas vulnerabilidades associadas ao fato dos servidores estarem usando uma versão da linguagem desatualizada. Muitas delas podem comprometer o conteúdo de um sistema feito na linguagem e até mesmo permitir a execução de código remoto na máquina atacada. A solução para as vulnerabilidades associadas a esses identificadores CVE é a atualização da linguagem para uma versão mais recente.

CVE-2011-3192

- Pontuação CVSS: 7.8
- Total de Vezes Encontrada: 15

Servidores Apaches com versões inferiores a 2.2.20 estão sujeitos a uma vulnerabilidade de negação de serviço. Neles uma falha no modo em que o servidor lida com algumas requisições faz com que o mesmo aloque muita memória e utilize muitos ciclos de processamento fazendo com que o ele fique instável ou até mesmo pare de funcionar. A solução para essa vulnerabilidade é a atualização do servidor para uma versão que não sofra dessa vulnerabilidade, de preferência a versão mais atual o possível.

CVE-2010-0408, CVE-2010-0425, CVE-2010-0434 e CVE-2007-6750

- Pontuação CVSS: 10
- Classificação OpenVAS: Severidade Alta

Esse 4 identificadores estão ligados a uma vulnerabilidade em versões do servidor Apache inferiores a 2.2.15. A vulnerabilidade encontrada é uma de corrupção de memória. É possível que um atacante utilize dessa vulnerabilidade para realizar negação de serviço ou até mesmo que ele execute código remoto com privilégios de sistema . A solução para esse problema é a atualização do servidor apache para uma versão mais recente (BUGTRAQ, 2014).

CVE-2014-3566

- Pontuação CVSS: 4,3
- Classificação OpenVAS: Severidade Média

Essa vulnerabilidade está associada ao fato de que um servidor possibilita que conexões SSL sejam estabelecidas utilizando a versão SSLv3 do protocolo com modo de cifra do tipo CBC. É conhecido que esse tipo de conexão está sujeita a uma vulnerabilidade de vazamento de informações. Para resolver o problema o servidor deve ser configurado para não possibilitar conexões SSL como as descritas acima.

4.4.2.4 Exploits Encontrados

O objetivo do teste realizado na rede do INE era o de ganhar acesso as máquinas testadas, entretanto para esse objetivo não foram encontrados *exploits* para pronto uso, entretanto, para outras vulnerabilidades do sistema foram encontrados dois *exploits* para pronto uso a partir dos metasploit, são eles:

- Apache Tomcat Transfer-Encoding Information Disclosure and DoS: *exploit* baseado na vulnerabilidade de com indentificador cve CVE-2010-2227, com esse *exploit* é possível realizar negação de serviço.
- phpLDAPadmin query_engine Remote PHP Code Injection: *exploit* baseado na vulnerabilidade CVE-2011-4075, permite que um atacante injete código PHP posteriormente executado pela função “create_function()”. Através desse *exploit* é possível por exemplo fazer o donwload de qualquer arquivo permitido ao servidor web hospedeiro da aplicação.

4.4.2.5 Recomendações Técnicas

É recomendável que o responsável por cada um dos *host*s tenha acesso ao relatório gerado pelo OpenVAS, pelo menos a parte que fale sobre o seu próprio host, para que o mesmo se conscientize da situação de sua máquina. As vulnerabilidades com alto grau de risco são majoritariamente consertadas através de uma simples atualização do sistema, para todas as outras existem medidas possíveis no próprio relatório e *links* para referências externas que falam sobre medidas a serem adotadas.

5 Conclusão e Trabalhos Futuros

O trabalho conseguiu alcançar o objetivo de desenvolver as principais etapas de um teste de penetração. Foram explicados vários conceitos básicos de segurança, sobre os testes propriamente ditos, além de diversas ferramentas e técnicas terem sido explicadas e demonstradas.

Além disso, foi efetuado um teste de penetração real na rede do Departamento de Informática e Estatística da UFSC. Fato que ajudou a ilustrar muito melhor os conceitos apresentados nos primeiros capítulos do trabalho. Além disso, é importante mencionar que os resultados do teste de penetração realizado, estão sendo utilizados pelo responsável da rede testada como um guia para consertar as falhas de segurança encontradas e para ver constatar se as regras de firewall implantadas estão funcionando ou não.

Por todos os fatos citados, esse trabalho conseguiu alcançar seu objetivo de se tornar um guia introdutório sobre testes de penetração para outros alunos da área de informática que tenham interesse sobre o assunto.

Como trabalho futuro, seria interessante expandir esse guia para incluir mais detalhes sobre a fase de exploração de software, tanto serviços de rede como aplicações web. Da fase de exploração, muitos aspectos referentes ao desenvolvimento de *exploits* poderiam ser agregados a esse guia, tornando-o assim um material muito mais completo.

Por último é interessante comentar que os dados completos produzidos pelas ferramentas durante a execução da parte prática desse trabalho são de caráter sigiloso devido ao risco associado a sua exposição, sendo esse o motivo pelo qual não são reproduzidos em sua totalidade nesse trabalho.

6 Referências

ABOUT OpenVAS Software. Disponível em: <http://www.openvas.org/software.html>. Acesso em: 7 jun. 2014.

ALLEN, Lee; HERIYANTO, Tedi; ALI, Shakeel. **Kali Linux:** Assuring Security by Penetration Testing. Birmingham: Packt Publishing, 2014. 454 p.

ANSARI, Sabeel; S.G., Rajeev; H.S., Chandrashekar. Packet Sniffing: A Brief Introduction. Ieee Potentials, v. 21, n. 5, p.17-19, dez. 2002. Bimestral.

BECHTSOUDIS, Anestis; SKLAVOS, Nicolas. Aiming at Higher Network Security Through Extensive Penetration Tests. Ieee Latin America Transactions, v. 10, n. 3, p.1752-1756, abr. 2012. Bimestral.

BUGTRAQ. **Apache 'mod_isapi' Memory Corruption Vulnerability.** Disponível em: <http://www.securityfocus.com/bid/38494/discuss>. Acesso em: 4 dez. 2014.

BUGTRAQ. **BugTraq.** 2006. Disponível em: <http://www.securityfocus.com/archive/1/description>. Acesso em: 2 out. 2014.

CLAUSEN, Lars R.. Concerning Etags and Datestamps. In: INTERNATIONAL WEB ARCHIVING WORKSHOP, 4., 2004, Bath.

COMPARE cURL Features with Other Download Tools. Disponível em: <http://curl.haxx.se/docs/comparison-table.html>. Acesso em: 10 maio 2014.

CROSS-SITE Scripting (XSS). Disponível em: <https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)>. Acesso em: 18 abr. 2014.

CVE. **Common Vulnerabilities and Exposures:** The Standard for Information Security Vulnerability Name. 2014. Disponível em: <https://cve.mitre.org/>. Acesso em: 15 set. 2014.

DAIGLE L. **WHOIS Protocol Specification:** Request for Comments 3912. Disponível em: <http://www.rfc-editor.org/rfc/rfc3912.txt>. Acesso em: 7 jun. 2014.

ENGEBRETSON, Patrick. The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy. Waltham: Syngress, 2011.

ESPENSCHIED, Jon. Five free pen-testing tools. Disponível em: <http://www.computerworld.com/s/article/9087439/Five_free_pen_testing_tools >. Acesso em: 4 maio 2014.

FRIEDL, Steve. SQL Injection Attacks by Example. Disponível em: <http://www.unixwiz.net/techtips/sql-injection.html>. Acesso em: 18 abr. 2014.

GUTTMAN, Barbara; ROBACK, Edward. An Introduction to Computer Security: The NIST Handbook. Washington: U.S. Government Printing Office, 1995.

HARPER, Allen et al. Gray Hat Hacking: The Ethical Hacker's Handbook. 3. ed. New York: Mcgraw-hill, 2011.

HTTRACK Website Copier - Free Software Offline Browser (GNU GPL). Disponível em: <http://www.httrack.com/page/9/en/index.html>. Acesso em: 10 maio 2014.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION / INTERNATIONAL ELECTROTECHNICAL COMMISSION. 27005: Information technology - Security techniques - Information security risk management. 1 ed. Suíça: Iso, 2008.

INTERNET Archive: Wayback Machine. Disponível em: <https://archive.org/web/>. Acesso em: 14 maio 2014.

INTERNET CRIME COMPLAINT CENTER (IC3). (Org.). 2013 Internet Crime Report. 2013. Disponível em: <http://www.ic3.gov/media/annualreport/2013_IC3Report.pdf>. Acesso em: 14 jun. 2014.

JASWAL, Nipun. **Mastering Metasploit:** Write and implement sophisticated attack vectors in Metasploit using a completely hands-on approach. Birmingham: Packt Publishing, 2014. 378 p.

KATHYAT, Vivek. **How to use TheHarvester on Backtrack 5 [Tutorial[.** 2013. Disponível em: <http://www.hackyshacky.com/2013/03/how-to-use-theharvester-on-backtrack-5.html>. Acesso em: 15 jul. 2014.

KENNEDY, David et al. **METASPLOIT:** The Penetration Tester's Guide. San Francisco: no Starch Press, 2011. 332 p.

LYON, Gordon. Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Sunnyvale: Insecure.com Llc, 2009.

MESSER, James. Secrets of Network Cartography: A Comprehensive Guide to nmap. Networkuptime.com, 2005.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Common Vulnerability Scoring System.** Disponível em: <http://nvd.nist.gov/cvss.cfm>. Acesso em: 15 set. 2014.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Minimum Security Requirements for Federal Information and Information Systems. Washington, 2006.

NORTHCUTT, Stephen et al. Penetration Testing: Assessing Your Overall Security Before Attackers Do. Sans. Disponível em: <https://www.sans.org/reading-room/analysts-program/PenetrationTesting-June06>. Acesso em: 14 jun. 2014.

NOVELL. **Potential Apache Security Vulnerability discloses iNode information with the FileETag method.** 2004. Disponível em: <https://www.novell.com/support/kb/doc.php?id=10090670>. Acesso em: 2 out. 2014.

OPENSSL. **OpenSSL Security Advisory:** SSL/TLS MITM vulnerability (CVE-2014-0224). 2014. Disponível em: <http://www.openssl.org/news/secadv_20140605.txt>. Acesso em: 2 out. 2014.

OPENSSL. **OpenSSL Security Advisory:** Record of death" vulnerability in OpenSSL 0.9.8f through 0.9.8m. 2010. Disponível em: <http://www.openssl.org/news/secadv_20100324.txt>. Acesso em: 2 out. 2014.

OREBAUGH, Angela; PINKARD, Becky. **Nmap in the Enterprise Your Guide to Network Scanning.** Burlington: Syngress, 2008. `

POSTEL, J. Transmission Control Protocol: Request for Comments 793. 1981. Disponível em: <http://www.ietf.org/rfc/rfc793.txt>. Acesso em: 7 jun. 2014.

ROUSE, Margaret. **Metasploit Project:** Metasploit Framework. 2011. Disponível em: <http://searchsecurity.techtarget.in/definition/Metasploit-Project-Metasploit-Framework>. Acesso em: 2 out. 2014.

SALGADO, Roberto. The SQL Injection Knowledge Base. Disponível em: <http://websec.ca/kb/sql_injection>. Acesso em: 18 abr. 2014.

SHIREY R. Internet Security Glossary: Request for Comments 2828. Disponível em: <http://www.ietf.org/rfc/rfc2828.txt>. Acesso em: 5 abr. 2014.

SHODAN. **The search engine for the Web:** Shodan is the world's first search engine for Internet-connected devices.. 2014. Disponível em: <https://www.shodan.io/>. Acesso em: 14 jun. 2014.

STALLINGS, William. Criptografia e Segurança de Redes: Princípios e Práticas. 4. ed. São Paulo: Prentice Hall, 2007.

STEWART, Ed. R.. **Stream Control Transmission Protocol:** Request for Comments: 4960. 2007. Disponível em: <http://www.rfc-editor.org/rfc/rfc4960.txt>. Acesso em: 14 jun. 2014.

SYMANTEC. CYBERCRIME REPORT 2012. 2012. Disponível em: <http://now-static.norton.com/now/en/pu/images/Promotions/2012/cybercrimeReport/NCR-Country_Fact_Sheet-Brazil.pdf>. Acesso em: 14 jun. 2014.

TANENBAUM, Andrew S.; WETHERALL, David J. . COMPUTER NETWORKS. 5. ed. Boston: Pearson Prentice Hall, 2011.

THE Harvester: The information gathering suite. Disponível em: <http://www.edge-security.com/theharvester.php>. Acesso em: 17 maio 2014.

THE Wget Wgiki. Disponível em: <http://wget.addictivecode.org/Wget>. Acesso em: 10 maio 2014.

WAGNER, Jan-oliver et al. OpenVAS Compendium. 2009. Disponível em: <http://wald.intevation.org/frs/download.php/558/openvas-compendium-1.0.1.pdf>. Acesso em: 7 jun. 2014.

WGET. Disponível em: <https://en.wikipedia.org/wiki/Wget>. Acesso em: 10 maio 2014.