UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MATHEUS MARCHEZAN DE MARCO

AVALIAÇÃO DA SEGURANÇA DA REDE LAN DA UTFPR-MD MEDIANTE A UM TESTE DE PENETRAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA

2016

MATHEUS MARCHEZAN DE MARCO

AVALIAÇÃO DA SEGURANÇA DA REDE LAN DA UTFPR-MD MEDIANTE A UM TESTE DE PENETRAÇÃO

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Bacharel em Computação".

Orientador: Prof. Ms. Gloria Patricia Lopez Sepulveda.

MEDIANEIRA

2016

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso de Ciência da Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

AVALIAÇÃO DA SEGURANÇA DA REDE LAN DA UTFPR-MD MEDIANTE A UM TESTE DE PENETRAÇÃO

Por

MATHEUS MARCHEZAN DE MARCO

Este Trabalho de Conclusão de Curso foi apresentado às 10:00h do dia 09 de novembro de 2016 como requisito parcial para a obtenção do título de Bacharel no Curso de Ciência da Computação, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Neylor Michel
UTFPR - Câmpus Medianeira

Prof. Ricardo Sobjak
UTFPR - Câmpus Medianeira

Prof. Gloria Patricia Lopez Sepulveda
UTFPR - Câmpus Medianeira

Prof. Jorge Aikes Junior
UTFPR - Câmpus Medianeira

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

MARCHEZAN DE MARCO, Matheus. AVALIAÇÃO DA SEGURANÇA DA REDE LAN DA UTFPR-MD MEDIANTE A UM TESTE DE PENETRAÇÃO. 47 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2016.

Com a dependência que se tem hoje de sistemas informatizados e das redes de transmissão de dados para o funcionamento eficiente de instituições como bancos, empresas, universidades, entre outros, é fundamental que estes ofereçam níveis de segurança e confiabilidade satisfatórios. Considerando este cenário dentro da UTFPR-MD, propõe-se a realização de um teste de penetração na rede e sistemas empregados dentro da universidade. Apesar de não garantir que o sistema testado será totalmente seguro e confiável após a realização do mesmo, este tipo de teste tem demonstrado ser eficiente para avaliar o nível de segurança atual de um sistema e consequentemente aprimorá-lo. Esta trabalho propõe que o teste de penetração seja executado utilizando a metodologia *Penetration Testing Execition Standardard* (PTES), produzindo um relatório detalhado sobre a atual situação dos mecanismos de segurança presentes na rede de dados e sistemas da UTFPR-MD, além de apontar eventuais falhas e suas respectivas soluções. O proposto teste foi capaz de detectar várias potenciais vulnerabilidades em dispositivos encontrados ativos na rede LAN da universidade, mostrando que existem pontos onde a segurança da rede pode ser aperfeiçoada.

Palavras-chave: Segurança de redes, Confiabilidade, Pentest.

ABSTRACT

MARCHEZAN DE MARCO, Matheus. SECURITY EVALUATION OF UTFPR-MD'S LAN NETWORK THROUGH A PENETRATION TEST. 47 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2016.

Nowadays institutions such as banks, companies and universities are dependent on computerized systems and data transmission networks to be efficiently operational. For that reason, it is essential that those systems and networks provide a satisfying level of security and reliability. Considering that on the UTFPR-MD's cenario, the present paper proposes to conduct a penetration test on the network and systems employed within the university. Although it is not guaranteed that the tested system will be completely safe and secure after the completion of it, this type of test is the most effective way to assess the current security level of a system and consequently improve it. The test will be performed using the *Penetration Testing Execition Standardard* (PTES) methodology , producing a detailed report on the current situation of the security mechanisms running on the network and systems of UTFPR-MD's campus, as well as pointing any security faults and their solutions. The proposed test was able to detect several potential vulnerabilities in devices found active on the university's LAN network, showing that there are points where the network security can be improved.

Keywords: Network security, Reliability, Pentest.

LISTA DE FIGURAS

LISTA DE SIGLAS

ACK	Acknowledging
APT	Advanced Persistent Threat
COGETI	Coordenação de Tecnologia da Informação
COGETI	Coordenadoria de Gestão de Tecnologia de Informação
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection Systems
IP	Internet Protocol
ISOM	Institute for Security and Open Methodologies
MIT	Massachusetts Institute of Tecnology
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OSINT	Open Source Intelligence
OSSTMM	Open Source Security Testing Methodology Manual
PTES	Penetration Testing Execition Standardard
RDP	Remote Desktop Protocol
RST	Reset
SYN	Synchronise
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
US-CERT	United States Computer Emergency Readiness Team
UTFPR	Universidade Tecnológica Federal do Paraná
VLAN	Virtual Local Area Network
WPA2	Wi-Fi Protected Acess 2

SUMÁRIO

1 INTRODUÇÃO

Nos dias de hoje, em que grande parte das informações que são produzidas e acessadas estão armazenadas no formato digital, ou seja, estocadas em algum tipo de sistema computacional, certas informações tem um valor imensurável ou podem desencadear catástrofes se colocadas em mãos erradas, portanto é extremamente importante que estes sistemas computacionais sejam seguros e confiáveis. Segundo Schneier (1996), para alcançar estes requisitos um sistema deve garantir confidencialidade, integridade e disponibilidade. A confidencialidade assegura que as informações devem somente ser acessadas por usuários autorizados. Integridade se trata de garantir que informações não sejam modificadas por usuários não autorizados e por último, disponibilidade é a garantia de que o acesso ao sistema não será negado de forma maliciosa a usuários autorizados.

Quando um sistema não apresenta este tipo de segurança e confiabilidade, pessoas mal intencionadas podem aproveitar-se disto para acessar informações confidenciais buscando o autobeneficio ou até mesmo denegrir a imagem de alguém. Exemplo disto foi o que aconteceu em Julho de 2015, em que um grupo de hackers intitulados como *The Impact Team* invadiram os servidores do site de infidelidade *AshleyMadison.com* roubando informações pessoais de seus usuários, tais como, nomes, telefones, transações financeiras, endereços, históricos de busca, etc. Após o roubo, o grupo ameaçou divulgar publicamente as informações caso a empresa *Avid Life Media*, dona do web site, não o tirasse do ar. Como a organização não atendeu à solicitação dos hackers, as informações roubadas foram publicadas expondo os usuários do serviço, bem como denegrindo o nome da companhia.

Para evitar este tipo de situação existem os testes de penetrações, que apesar de não garantirem que um sistema seja completamente seguro e esteja a prova de ataques, eles aumentam o nível de segurança de um sistema computacional, detectando e removendo vulnerabilidades que são publicamente conhecidas, impedindo assim que hackers se utilizem destas para atacar o sistema em questão (PRANDINI; RAMILLI, 2010). Este trabalho tem como objetivo descrever a execução de um teste de penetração nos sistemas da Universidade Tecnológica Federal do Paraná (UTFPR), Campus Medianeira bem como possibilitar um aprimoramento da segurança presente utilizando os resultados obtidos.

1.1 PROBLEMÁTICA

Determinar qual é o nível de segurança oferecido pela rede de transmissão de dados presente na UTFPR-MD, bem como pelos sistemas que operam nesta localidade mediante a realização de um teste de penetração.

1.2 JUSTIFICATIVA

Considerando a importância da rede, dos sistemas e dos dados armazenados nos servidores da UTFPR-MD para a realização das atividades cotidianas dos alunos, professores e servidores, se torna essencial a presença de mecanismos de segurança capazes de assegurar a integridade, disponibilidade e a privacidade das informações contidas nos mesmos, bem como impedir o acesso de usuários não autorizados aos sistemas.

Para certificar que as políticas atualmente empregadas no campus são apropriadas para garantir um nível satisfatório de segurança, torna-se imprescindível a realização de um teste de penetração, apontando falhas e inconsistências, bem como soluções para estas, permitindo o aperfeiçoamento da segurança presente nos sistemas da universidade.

1.3 OBJETIVO GERAL

O objetivo geral é determinar, por meio de um teste de penetração, se os mecanismos de segurança atualmente implementados na rede de dados da UTFPR Campus Medianeira são capazes de garantir o funcionamento e a qualidade dos serviços de forma segura.

1.4 OBJETIVOS ESPECÍFICOS

A avaliação de segurança proposta neste trabalho se divide nas etapas apresentadas a seguir:

- Estudar os *exploits* presentes no *framework* Metasploit;
- Determinar se há potenciais falhas de segurança na rede da UTFPR Campus Medianeira que possam permitir que os serviços e sistemas presentes no campus sejam violados;
- Selecionar os *exploits* capazes de explorar as potenciais brechas encontradas;
- Explorar as potenciais falhas encontradas afim de penetrar o mais profundamente possível nestes sistemas;
- Determinar se os protocolos de segurança implementados na rede e nos sistemas são capazes de detectar e interromper ataques e intrusões;
- Apresentar soluções para as falhas exploradas, permitindo aumentar o nível de segurança da rede de dados da UTFPR Campus Medianeira.

1.5 HIPÓTESES

- Seria possível escanear a rede em busca de alvos em potencial, como servidores, por exemplo, que possam ser explorados com o objetivo de comprometer o funcionamento dos serviços ou o sigilo dos dados armazenados nos mesmos;
- Haveriam portas vulneráveis presentes nos alvos em potencial através de um Portscan;
- Seria possível obter acesso remoto não autorizado aos servidores do campus;
- Averiguar se o sistema de segurança presente é capaz de detectar e impedir tentativas de intrusão.

2 REFERENCIAL TEÓRICO

Nessa seção está descrito o estado da arte do tema escolhido. Inicialmente é apresentado um breve histórico da evolução dos tipos de ataque, em seguida, são descritos os tópicos referentes aos testes de penetração e sua execução.

2.1 TERMINOLOGIA

A seguir, são apresentados alguns termos que são utilizados no decorrer deste capitulo, bem como suas respectivas definições.

- **Backdoor** pode ser descrito como um meio secreto utilizado para contornar um sistema de autenticação comum, normalmente é utilizado para assegurar acesso remoto não autorizado a um computador ou sistema.
- **Exploit** é o meio pelo qual um atacante tira proveito de alguma falha presente em um sistema, aplicativo ou serviço. Em outras palavras, os atacantes utilizam *exploits* para atacar um sistema e produzir um resultado desejado que não foi previsto pelo desenvolvedor.
- **Hacker** é o termo utilizado para descrever pessoas que possuem um grande conhecimento técnico na área de segurança e utilizam-se deste conhecimento para obter acesso não autorizado a sistemas.
- **Malware** é uma abreviação de *malicious software*, ou software malicioso em português, que se refere a qualquer programa projetado para causar dano ou realizar ações indesejadas em sistemas computacionais.
- **Payload** pode ser definido como o código que um atacante deseja que seja executado pelo sistema alvo. Por exemplo, o *shell* reverso é um *payload* que cria uma conexão que parte do *host* alvo para computador do atacante. Uma simples sequência de comandos a serem executados pelo sistema operacional do *host* alvo pode ser considerado um *payload*.
- **Phishing** é a tentativa de conseguir dados confidenciais como logins, senhas e informações

sobre cartões de crédito, se passando por uma entidade confiável, fazendo com o que a vítima forneça estas informações.

- **Shellcode** é um conjunto de instruções usadas como *payload* quando a exploração acontece, normalmente escrito na linguagem *assembly*. Na maioria dos casos, após a execução destas instruções na maquina alvo um *shell* de comando deve ser fornecido.
- **Vulnerabilidade** pode ser considerada como um ponto fraco em um sistema que permite um atacante reduzir o nível de garantia da informação. Ou seja, é o meio que pelo qual o atacante pode ferir a confidencialidade, integridade ou disponibilidade de um sistema.

2.2 IMPORTÂNCIA DA SEGURANÇA

É difícil dizer exatamente como o termo *hacker* surgiu, mas seguindo HOLLINGER e LANZA-KADUCE (1988), o mesmo começou a ser utilizado nos laboratórios de inteligência artificial do Massachusetts Institute of Tecnology (MIT) como apelido que se dava a indivíduos que conseguiam descobrir uma nova maneira para resolver problemas matemáticos, isso por volta dos anos 60. Utilizado posteriormente para se referir a programadores talentosos ou aficionados por computadores, hoje o termo tem uma certa carga pejorativa, pois a mídia sempre utiliza o termo *hacker* relacionado a *cybercrimes* .

Estes crimes cibernéticos acontecem desde os primórdios da computação, porém alguns crimes e criminosos entraram para a história devido as proporções ou prejuízo causado, como é o caso de Jonathan James. Com 15 anos de idade, em 1999, Jhonathan invadiu várias redes de companhias americanas, incluindo a Bell South, o departamento de defesa dos Estados Unidos e também a *National Aeronautics and Space Administration* (NASA). Durante estas invasões o criminoso conseguiu obter o código fonte do programa que controlava a Estação Espacial Internacional, avaliado em $1.7 milhões de dólares. Além disso, quando a intrusão foi detectada pela NASA, seus computadores foram desligados por 3 semanas para investigações, causando um prejuízo enorme para a agência (STOUT, 2000).

Outro criminoso que conseguiu gravar seu nome na história foi Gary McKinnon, que realizou o maior ataque a computadores militares de todos os tempos. Gary invadiu 97 computadores das forças armadas dos Estados Unidos e da NASA entre fevereiro de 2001 e março de 2002. As autoridades americanas afirmam que ele apagou arquivos críticos de sistemas operacionais, fazendo com que a rede das forças armadas do Estados Unidos em

Washington, contendo mais de 2.000 computadores, fosse desligada por 24 horas. Gary também deletou logs de armamento da estação de armas navais de Earl, fazendo com que a rede da estação, contendo 300 computadores, ficasse inoperável e paralisando assim as entregas de munição para a frota do Atlântico da marinha americana. As autoridades americanas declararam que o custo para rastrear e corrigir os problemas causados pelo criminoso passaram de $700.000 dólares (BURNST, 2009).

Já a *cybergang* intitulada como *Carbanak* conseguiu mudar a forma de roubar instituições financeiras em 2015. A gangue executou o que pode ser considerado o maior e mais sofisticado roubo a bancos da história, roubando dinheiro de mais de 100 bancos e instituições financeiras em mais de 30 países incluindo o Brasil. Estes criminosos foram capazes de invadir os sistemas dos maiores bancos do mundo e roubar cerca de $1 bilhão de dólares durante aproximadamente 2 anos de ação (SANGER; PERLROTH, 2015).

Ainda em 2015, segundo a Symantec (2016), foram encontrados mais de 430 milhões de tipos exclusivos de *malwares*. Meio bilhão de registros pessoais foram roubados de corporações ou deletados e também foram interceptados cerca de 100 milhões de golpes de falso suporte técnico, onde criminosos cibernéticos fazem chamadas e convencem as vítimas a entregar seu dinheiro, se passando por prestadores de suporte. Além disso, as campanhas de *phishing* contra funcionários de grandes corporações aumentou em 55% no último ano.

Existem também alguns grupos famosos que se intitulam como *hackers* ativistas, *Anonymous* e *WikiLeaks* por exemplo, que fazem o uso de seus conhecimentos na área de segurança da informação para atacar sistemas como forma de protesto. Estes protestos podem se dar por meio da invasão de web sites, em que os atacantes desfiguram as páginas inciais com mensagens de revolta. Outras formas são os ataques de negação de serviço, fazendo com que serviços importantes fiquem fora do ar, e a distribuição de informações roubadas com o fim de causar danos o expor planos ou fragilidades sobre corporações ou governos (PENDERGRASS, 2013).

Acontecimentos como esses deixam claro o tamanho da importância da segurança em redes de computadores e sistemas de informação. Pequenas falhas quando exploradas por pessoas mal intencionadas e com grande conhecimento técnico podem se tornar catastróficas.

2.3 HACKING

Quando se fala sobre ciber ataques, *Hacking* é um termo muito comum utilizado para descrevê-los, porém pode-se distinguir o significado desses dois termos. O termo ataque normalmente refere-se a qualquer atividade não autorizada que busca causar dano ou violar algum tipo de tecnologia ou sistema. *Hacking* é um tipo de ataque mais sofisticado que visa estudar e aprender sobre todos os aspectos tecnológicos e de infra-estrutura, e assim, usar este conhecimento para explorar vulnerabilidades. Sendo assim, para praticar *hacking* é necessário um amplo entendimento sobre sistemas operacionais, redes de computadores, protocolos de comunicação, medidas de segurança bem como falhas em aplicações (ALSUNBUL et al., 2013).

Os *hackers* existem desde os primórdios da computação e vem atacando sistemas computacionais de maneiras cada vez mais sofisticadas. Segundo Cisco (2014) é possível mapear as grandes mudanças ocorridas nas formas de ataque utilizadas ao decorrer dos anos, ilustrado na Figura 1.

Primeiramente surgiram os vírus, que em sua essência, quando são executados tem a capacidade de se reproduzir ou infectar outros programas tendo como alvo os sistemas operacionais. Em seguida houve o surgimento dos *worms*, que são capazes de se auto-replicar espalhando-se para outros computadores, normalmente através da rede de dados, mas diferentemente dos vírus, os *worms* não precisam estar vinculados a um programa para serem executados. Depois disto, com a popularização dos aplicativos e dispositivos online, apareceram os *spywares*, programas rodam em *background* coletando informações pessoais dos usuários, e os *rootkits*, programas criados por *hackers* que os possibilitam obter acesso administrativo ao computador da vítima. Atualmente existem ataques mais avançados e direcionados como é o caso das ameaças persistentes avançadas, *Advanced Persistent Threat* (APT) , que consistem em obter acesso a uma rede e permanecer conectado sem ser detectado com o objetivo de roubar informações valiosas.

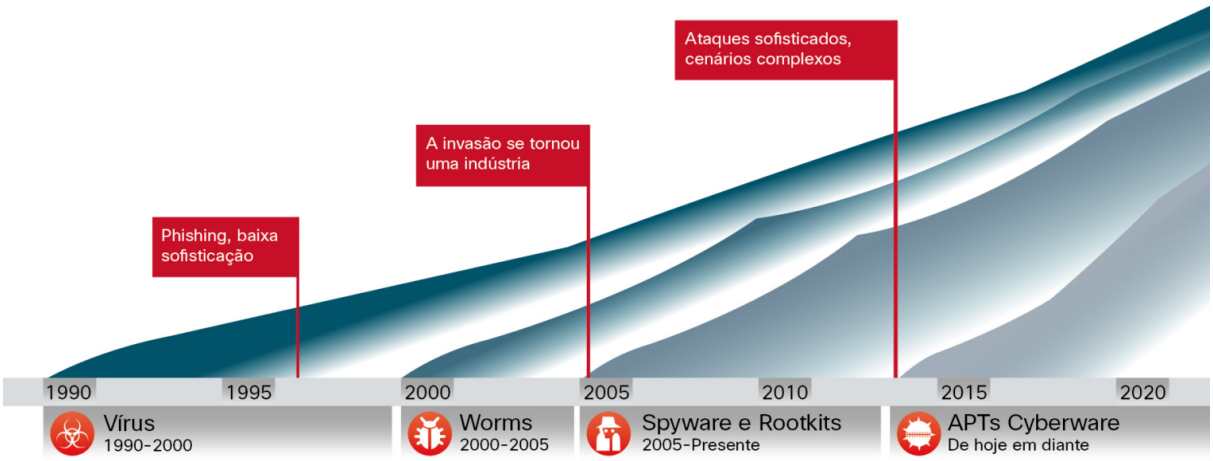


Figura 1 – A evolução das formas de ataque.

Fonte: (CISCO, 2014).

Apesar da evolução e sofisticação dos meios de ataque, os objetivos finais sempre foram obter algum tipo de ganho financeiro ou causar algum tipo de dano à vítima. Segundo Ahmadian et al. (2015), entre as mais recentes e sofisticadas formas de ataque estão os *Ransomwares*, *malwares* que restringem o acesso ao sistema ou arquivos infectados exigindo o pagamento de um resgate ao desenvolvedor do *malware* para que a vítima volte a ter acesso aos documentos ou sistemas comprometidos. Normalmente esta restrição se dá por meio da criptografia de arquivos dentro de um sistema, obrigando assim a vítima a pagar pela chave capaz de descriptografar o que foi infectado, porém há variações onde o sistema simplesmente é bloqueado e são exibidas mensagens tentando forçar a vítima a realizar o pagamento, como ilustrado na Figura 2.

Figura 2 – Mensagem exibida em uma máquina infectada pelo *PRISM CryptoLocker ransomeware*.
Fonte: Adaptado de http://hyphenet.com/ransomware-screenshots

2.4 ÉTICA

No contexto de segurança, são utilizados vários termos distintos para fazer referência aos tipos diferentes de *hackers*, como *white hat* ou *pentester*, que referenciam os *hackers* éticos, e *black hat*, ou *crackers* que correspondem aos *hackers* maliciosos. Como afirma Weidman (2014), é importante notar que os *white hats* quase sempre executam as mesmas atividades que um *hacker* malicioso, inclusive utilizando as mesmas ferramentas na execução destas tarefas. Para realizar um teste de penetração que seja realmente útil e relevante para a organização onde o mesmo será executado, o *pentester* precisa esforçar-se para agir e pensar como um *black hat*, possibilitando assim a simulação ataques reais. Apesar destes dois tipos de *hackers* desempenharem funções muito parecidas, eles se diferenciam por três pontos chaves: autorização, motivação e intenção.

A primeira e mais simples maneira de diferenciar *white hats* e *black hats* é a autorização, que nada mais é que o processo pelo qual obtêm-se aprovação da organização para a realização de qualquer tipo de ataque ou teste. Neste processo um escopo é definido, ou seja, quais recursos, sistemas e alvos farão parte do teste e o que pode ou não ser executado. Os *white hats* sempre respeitarão a autorização findada e jamais irão além do escopo definido para o teste enquanto os *black hats* não possuem a autorização da organização e nem restrições de escopo (WEIDMAN, 2014).

Outra maneira de diferenciá-los é examinar a motivação do para a realização dos ataques. Caso este seja motivado por ajudar a organização a aprimorar a segurança de seus sistemas, possuindo uma autorização prévia, este pode ser considerado um *hacker* ético. Entretanto, se a motivação for algum ganho pessoal, como obter renda por meio de extorsão ou outros métodos deturpados para extrair dinheiro da vítima, fama ou vingança, este deve ser considerado um *black hat* (ENGEBRETSON, 2011).

Ainda segundo Engebretson (2011), se a intenção do for fornecer uma simulação de ataque realista possibilitando que a organização melhore sua segurança detectando precocemente e reduzindo as vulnerabilidades, e acima disso, não compartilhando as informações confidenciais obtidas durante o teste com ninguém além da própria organização, pode-se considera-lo um *hacker* ético. Contudo, se a intenção for utilizar-se destas informações para obter qualquer tipo de ganho pessoal, considera-se o *hacker* como malicioso.

Existe ainda uma terceira categoria, definida como *grey hat*, que é um meio termo entre *white hat* e *black hat*. Estes *hackers* normalmente atacam sistemas de informação sem a autorização do proprietário, entretanto não visam benefício próprio, ou seja, comumente a motivação é demostrar aos proprietários as brechas e os pontos fracos presentes em seus sistemas.

2.5 TESTE DE PENETRAÇÃO

Um teste de penetração, ou *penetration test*, ou *pentest*, é um processo de avaliação de segurança que simula ataques reais de atacantes maliciosos a um sistema, visando encontrar e remover potenciais vulnerabilidades (FARAH, 2015). Existe também a avaliação de vulnerabilidades, que pode ser confundida com o teste de penetração por também ter como objetivo reportar falhas de segurança. Porém segundo Engebretson (2011), a avaliação de

vulnerabilidades consiste em avaliar os serviços e sistemas em busca de potenciais falhas de segurança, enquanto o teste de penetração busca explorar possíveis falhas provando que estas são reais. Em outras palavras, o teste de penetração vai além da avaliação, simulando o comportamento real de um atacante. Sendo assim, a avaliação de vulnerabilidades pode ser considerada uma fase do teste de penetração.

Os *pentests* podem ser divididos em duas categorias distintas, *black box*, ou caixa preta e *white box*, caixa branca. Antunes e Vieira (2014) descrevem que em um teste de penetração que se baseia na técnica de caixa preta, o testador não tem acesso ao comportamento interno do sistema, ou seja, ele tem acesso apenas ao ponto de vista de um usuário externo. Já quando utiliza-se da técnica de caixa branca, analisa-se o sistema internamente buscando vulnerabilidades por meio da revisão do código fonte do sistema.

2.6 METODOLOGIAS PARA EXECUÇÃO DE TESTES DE PENETRAÇÃO

Para obter sucesso na execução de um *pentest* é preciso escolher um entre os inúmeros padrões para execução de testes de penetração disponíveis no mercado. O *Open Source Security Testing Methodology Manual* (OSSTMM) por exemplo, é uma metodologia para a aplicação de testes de penetração que foi introduzida no ano de 2000 e é mantida, na versão atual 3.0, pelo *Institute for Security and Open Methodologies* (ISOM). Esta metodologia prega que os elementos de infraestrutura dentro de um sistema interagem, sendo assim, para aumentar o nível de segurança de determinado sistema é preciso controlar e restringir essa interação (HOLIK et al., 2015).

O OSSTMM divide o teste de penetração em 4 fases principais: Indução, Interação, Averiguação e Intervenção. O resultado de cada fase servirá de entrada para a fase seguinte. A primeira delas, a fase de indução, esclarece qual será o período de tempo para a realização do teste de penetração, bem como os tipos de testes que poderão ser utilizados. Na fase de interação, determina-se quais são os alvos que farão parte do escopo do teste. Durante a averiguação deve-se obter o máximo de informação possível sobre os alvos dentro do escopo. Na última fase, a intervenção, verifica-se as funcionalidade dos mecanismos de segurança e alerta (HOLIK et al., 2015).

Outro exemplo de metodologia é a definida pelo *National Institute of Standards and Technology* (NIST), que também divide o teste em 4 fases principais, porém com uma

nomenclatura diferente. As fases são: Planejamento, Descoberta, Ataque e Relatório. Sendo que estas se dividem em sub-fases durante a execução, como ilustrado na Figura 3 (SCARFONE et al., 2008).

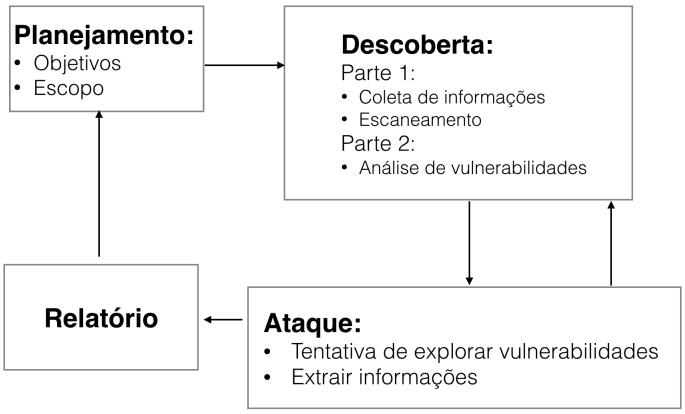


Figura 3 – Fases da metodologia NIST.
Fonte: Adaptado de (HOLIK et al., 2015).

Segundo Farah (2015), apesar das metodologias apresentarem nomes ou uma ordem diferente para execução das etapas do teste, as tarefas a serem executadas são fundamentalmente as mesmas. Por ser uma metodologia recomendada por vários autores e possuir uma carga teórica menor que as citadas anteriormente, a metodologia adotada neste trabalho será a do Padrão para Execução de Teste de Penetração, do inglês *Penetration Testing Execition Standardard* (PTES), que divide o *pentest* em 7 etapas descrevendo as tarefas que devem ser cumpridas em cada uma destas. Partindo da comunicação inicial com o cliente e motivação do teste, passando pela coleta de informações, modelagem de ameaças, análise de vulnerabilidades, exploração, pós-exploração e por fim a criação do relatório técnico. Cada etapa estabelecida pelo PTES está descrita detalhadamente nas subseções seguintes.

2.6.1 Acordos prévios

Segundo Weidman (2014), antes de iniciar qualquer teste de penetração é importante esclarecer qual é a motivação para o teste, quais razões levaram o cliente a buscar a realização do teste, quais os objetivos do mesmo e que tipo de falha preocupa o cliente. A falta de esclarecimento neste ponto pode levar a situações embaraçosas, como quando o cliente deseja

apenas uma avaliação de vulnerabilidades e não deixa isto explícito para o testador, uma vez que o teste de penetração é muito mais intrusivo.

Além disto, é necessário definir o escopo do teste, determinando quais *hosts* e endereços de IP fazem parte de escopo e quais não fazem. Clarificar se o testador terá total liberdade para explorar possíveis falhas considerando que certos *exploits* podem interromper o funcionamento do sistema ou se ele deve limitar-se a encontrar potenciais falhas e reportá-las. Determinar se o cliente deseja que o teste seja realizado em dias e horários específicos. Também é crucial a obtenção de uma autorização por escrito em forma de contrato para a realização do teste diz Weidman (2014).

2.6.2 Coleta de informação

Nesta fase o objetivo é conseguir a maior quantidade de informação possível sobre o alvo a ser testado, uma vez que, quanto mais informação se tem sobre o alvo, maior será a gama de vetores de ataque que poderão ser empregados na fase de modelagem das ameaças. Para isso, a OSINT, que pode ser definida como uma maneira de gerir a coleta de informações publicamente disponíveis analisando-as visando encontrar algo útil para criação dos planos de ataque, apresenta três formas de coleta: passiva, semi-passiva e ativa (pentest-standard.org).

- **Passiva:** na coleta passiva o testador não interage com o alvo enviando pacotes para o mesmo. Deve utilizar-se apenas de informações armazenadas em repositórios públicos. Sendo assim, esta forma aplica-se a casos em que estas atividades de coleta não podem, de maneira alguma, ser detectadas pelo alvo. Porém não há garantia de que os dados obtidos são atualizados ou corretos.
- **Semi-passiva:** a coleta semi-passiva permite a interação entre testador e alvo, no entanto o tráfego gerado nesta interação deve ser semelhante ao produzido por um usuário comum, sem despertar a atenção do alvo. Para isso, analisa-se apenas os metadados presente em arquivos e documentos disponíveis, sem buscar por serviços ou diretórios ocultos.
- **Ativa:** nesta forma de coleta o testador busca mapear a infra-estrutura de rede do alvo, procurando por serviços, servidores, arquivos ou diretórios ocultos, bem como por vulnerabilidades nos serviços publicados. Serão empregados nesta fase *port scans* e outras ferramentas que produzirão um tráfego anormal na rede, possivelmente sendo detectadas e até mesmo bloqueadas pelo alvo. Por este motivo é importante que os endereços de IP utilizados nesta fase sejam completamente diferentes do que serão

empregados na execução dos ataques.

2.6.3 Modelagem de ameaças

Neste ponto, de posse de uma grande quantidade de informações que dizem respeito ao funcionamento do sistema, determina-se qual é o objetivo do testador perante ao alvo e quais os métodos de ataque mais eficientes para alcançar este objetivo (KENNEDY et al., 2011). Como Weidman (2014) exemplifica, se a organização alvo possuir um software proprietário, o atacante poderia causar um dano catastrófico à organização conseguindo acesso aos repositórios onde o código fonte do software é desenvolvido e testado e vendendo seus segredos comerciais para organizações concorrentes.

2.6.4 Análise de vulnerabilidades

Após a modelagem das ameaças, inicia-se a análise de vulnerabilidades na qual o testador busca efetivamente por potenciais falhas que irão determinar quão efetivas serão suas estratégias de ataque definidas na fase anterior (WEIDMAN, 2014). O PTES descreve duas maneiras de busca por vulnerabilidades que podem ser aplicadas nesta fase: passiva e ativa (pentest-standard.org).

- **Passiva:** baseia-se na leitura e interpretação da *metadata*, informações que descrevem os arquivos, presente nos documentos. Um arquivo criado usando o Microsoft Office por exemplo, pode conter em sua *metadata* informações como nome do autor, data de criação, data da última modificação, entre outros. Muitos arquivos permitem criação de *metadata* customizada, ou seja, estes arquivos podem conter caminhos para servidores ocultos ou endereços de IP internos, dentre outras informações que podem ajudar o testador a encontrar vulnerabilidades no sistema em teste. Outra forma de adquirir informações passivamente é o monitoramento de tráfego, que consiste em conectar-se a uma rede de dados e capturar pacotes para analisá-los posteriormente (pentest-standard.org).
- **Ativa:** diferente da passiva, a análise ativa de vulnerabilidades envolve interação direta com o sistema em teste, sendo que esta interação pode ser automatizada ou manual. A

> automação permite a execução de tarefas repetitivas, como é o caso do *port scanner* por exemplo, que verifica individualmente cada uma das 65.535 portas de um *host* buscando por alguma que esteja aberta para conexões. (pentest-standard.org).

Existem vários tipos de *scanners* que podem ser executados nesta etapa, os baseados em portas, ou *port scanners*, por exemplo, são normalmente os primeiros *scanners* executados pelo testador em um teste de penetração, pois estes dão uma visão geral sobre quais portas estão disponíveis no alvo. Este tipo de *scanner* utiliza protocolos baseados no IP como TCP, UDP, ICMP, por exemplo, para checar quais portas estão disponíveis para receber conexões (pentest-standard.org).

Já os *Scanners* baseados em serviços, fazem o uso de protocolos específicos para tentar se comunicar com as portas abertas no *host* alvo sendo capazes de determinar que tipo de serviço está sendo executado em uma porta específica baseando-se no protocolo pelo qual a porta é capaz de responder corretamente (pentest-standard.org).

Durante o processo de conexão a uma porta normalmente o serviço que está rodando na mesma retorna informações como o nome da aplicação ou serviço e qual a versão que está em execução. O processo de capturar e analisar estas informações para identificar o que esta rodando em uma porta especifica é chamado de *banner grabbing* (pentest-standard.org).

Da mesma forma, os *scanners* de vulnerabilidades, como o *NeXpose*, também são capazes de identificar as versões de sistemas e serviços ativos em um determinado *host*, no entanto, estes *scanners* cruzam os resultados obtidos com bases de dados de vulnerabilidades, como o *National Vulnerability Database* (NVD), para encontrar falhas de software ou de configuração relacionadas aos sistemas e serviços identificados. Segundo The MITRE Corporation (2016), o NVD baseia-se e utiliza a lista de falhas mantida pelo *Common Vulnerabilities and Exposures* (CVE), que é um dicionário internacional de vulnerabilidades e exposições publicamente conhecidas na área de segurança da informação, que até o presente momento conta com mais 78.700 enumeradas.

Patrocinado pelo *United States Computer Emergency Readiness Team* (US-CERT), que é uma organização dentro do departamento de segurança interna dos Estados Unidos, o CVE tem como objetivo facilitar o compartilhamento de informações sobre vulnerabilidades entre ferramentas, repositórios e serviços através da enumeração das mesmas. Em outras palavras, o CVE dá um nome (CVE ID) para cada vulnerabilidade e mantém informações como sistemas em que a mesma pode estar presente, o impacto causado sobre o sistema pela sua exploração, a forma como ela pode ser explorada e o que é preciso para a remediação (The MITRE Corporation).

2.6.5 Exploração

Na fase de exploração o testador executa os *exploits* tentando comprometer o sistema alvo, porém deve-se tomar cuidado para executá-los com precisão ao invés de usar força bruta. Um *exploit* deve ser executado apenas quando se está seguro de que o mesmo será bem sucedido, por esta razão antes de executar qualquer tipo de ataque é importante considerar todos os possíveis mecanismos de segurança ativos no sistema alvo. Medidas de segurança imprevistas como *Intrusion Detection Systems* (IDS), ou sistemas de detecção de intrusão em português, podem ser encontradas, no entanto executar *exploits* em massa esperando que algum obtenha sucesso não é produtivo muito menos recomendado, já que alguns destes podem disparar mecanismos de segurança impedindo assim que o testador prossiga (KENNEDY et al., 2011).

2.6.6 Pós-exploração

Após concluída a fase de exploração, espera-se que o testador tenha obtido acesso ao sistema alvo ou a parte dele. Na etapa de pós-exploração o testador produz informações muito relevantes sobre o teste de penetração. Nesta fase, o objetivo é atacar sistemas específicos, identificar infra-estruturas críticas, e buscar por informações que deveriam ser protegidas pela organização. Quando o testador está executando esta parte do teste, ele visa demonstrar quais os ataques que teriam maior impacto sobre a organização. O testador deve também determinar quais são os vários sistemas exploráveis e quais são os níveis de permissão dos usuários dentro destes sistemas (KENNEDY et al., 2011).

No caso de uma invasão onde o testador consegue acessar o sistema com os privilégios de administrador do mesmo, seria possível comprometer o sistema financeiro utilizado para realizar o pagamento de funcionários da organização transferindo todo o dinheiro para uma conta externa por exemplo? Conseguiria o testador comprometer as propriedades intelectuais da organização, copiando o código fonte de algum sistema proprietário e distribuindo publicamente? Considerando hipoteticamente que a organização em teste venda aplicações customizadas para fábricas, seria possível alterar o código fonte entregue aos clientes inserindo um *backdoor* comprometendo assim todos os clientes da organização? O que poderia ser feito para prejudicar a credibilidade da organização alvo? Estas são as hipóteses que o testador deve levantar nesta etapa, pensando como um atacante malicioso, sendo criativo, se adaptando

rapidamente a indo além das ferramentas automatizadas, completa Kennedy et al. (2011) sobre a fase de pós-exploração.

2.6.7 Relatório

Saber como reportar as vulnerabilidades encontradas durante a realização do teste de penetração é tão importante quanto encontrá-las e explorá-las. Neste relatório o testador deve descrever quais vulnerabilidades foram descobertas, como foram exploradas e além disto, como a organização pode corrigir as falhas de segurança encontradas. As informações produzidas durante a realização do teste são vitais para que os responsáveis pela segurança na organização consigam impedir novos ataques, já que o testador trabalha simulando o ponto de vista de um atacante malicioso, que dificilmente a organização visualiza (KENNEDY et al., 2011).

Segundo Weidman (2014), o relatório deve ser dividido em um sumário executivo e um relatório técnico. O sumário deve descrever de maneira geral os objetivos do teste, uma visão do que foi encontrado, bem como recomendações e estratégias para remediar os problemas descobertos, já que este destina-se aos executivos da organização, que possivelmente não possuem conhecimento técnico abrangente sobre segurança. Por outro lado, o relatório técnico deve conter os detalhes específicos de cada fase do teste realizado.

2.7 REDE DE DADOS DA UTFPR CAMPUS MEDIANEIRA

A rede de dados pública da UTFPR Campus Medianeira permite conexões sem fio e via cabo, sendo redes diferentes e com políticas de segurança distintas. Ambos os tipos de conexão fazem o uso de servidores *Dynamic Host Configuration Protocol* (DHCP), que são responsáveis por atribuir automaticamente endereços de IP a qualquer dispositivo que se conecte a rede, porém utiliza-se faixas de IP diferentes para as redes. Na rede cabeada os o servidor DHCP atribui endereços IP classe C (192.168.0.0), com a mascara de sub-rede contendo 24 bits. Além disso estão presentes dois servidores *Domain Name System* (DNS), que são responsáveis pela tradução das URLs (www.exemplo.com) em endereços de IP. O servidor primário tem o endereço 192.168.2.240, com o sistema operacional Microsoft Windows Server 2012 Standad Edition e o secundário no endereço 200.134.18.52, com o sistema operacional Linux com *kernel*

na versão 2.6.9. Para navegar na internet utilizando a rede cabeada do campus, é necessária a configuração do *proxy* no dispositivo em que se deseja navegar, o mesmo está no endereço 200.134.81.53 na porta 3128, não sendo requerido nenhum tipo de autenticação do usuário.

Já a rede sem fio utiliza IPs classe A (10.10.0.0) com a mascara de sub-rede de 16 bits, permitindo até 65534 endereços diferentes na mesma rede. Os servidores de DNS para a rede sem fio estão nos endereços 200.134.80.11 e 10.10.0.1. Para se conectar a esta rede é necessária uma autenticação por parte do usuário, onde o mesmo deve acessar utilizando dados de acesso previamente criados, sendo o login seu número de registro acadêmico precedido pela letra "a", e a senha definida pelo mesmo anteriormente. Esta rede não foi avaliada durante os testes pois a maior parte dos dispositivos conectados a ela são de terceiros (alunos), para possibilitar a avaliação dos mesmos seria necessário obter a autorização dos proprietários, o que se tornou inviável.

Existe ainda a rede administrativa da universidade, nomeada como UTFPR-ADM, a qual não foi possível obter informações sobre pois a mesma possui um sistema de autenticação *Wi-Fi Protected Acess 2* (WPA2) *Enterprise*, exigindo usuário e senha de funcionários para acesso. Esta rede também não foi avaliada no teste.

2.8 METASPLOIT

Metasploit é um *framework*, ou seja, uma suite de algoritmos que oferece a infra-estrutura necessária para automatizar tarefas mundanas, rotineiras e até mesmo complexas da área de segurança. Metasploit foi originalmente concebido por H. D. Moore, quando se deu conta de que gastava a maior parte de seu tempo verificando e validando códigos de *exploits* públicos, enquanto trabalhava em um empresa de segurança. Ele decidiu então criar uma estrutura flexível e sustentável para o desenvolvimentos de *exploits*, publicando assim a primeira versão do Metasploit, baseada na linguagem *Perl* e contendo um total de 11 *exploits*, em 2003 (KENNEDY et al., 2011).

Já em 2004, a versão 2.0 do *framework* foi publicada, tendo seu código completamente reescrito com a ajuda de Spoonm e contendo desta vez 19 *exploits* além de mais de 27 *payloads*. Logo após isto o projeto começou a receber um forte suporte da comunidade de segurança, rapidamente se tornando uma ferramenta necessária e muito útil na realização de testes de penetração e exploração. Em 2007 o Metasploit 3.0 foi publicado pelo time de desenvolvimento,

tendo sido completamente reescrito, porém desta vez na linguagem *Ruby*, sendo adotado em larga escala pela comunidade de segurança. Por fim, em 2009 o projeto Metasploit foi adquirido pela Rapid7, empresa líder na área de escaneamento de vulnerabilidades, que continua seu desenvolvimento e aperfeiçoamento atualmente (KENNEDY et al., 2011).

2.9 KALI LINUX

Kali é uma distribuição baseada no Debian Linux voltada para testes de penetração avançados e auditoria de segurança. Kali oferece centenas de ferramentas que auxiliam em várias tarefas relacionadas a segurança, por exemplo, testes de penetração, análise forense e engenharia reversa, sendo também uma plataforma suportada pelo Metasploit *Framework* como ilustra a Figura 4 (Kali Linux). Além disso, Kali é um projeto de código aberto que foi criado e é mantido pela *Offensive security*, uma empresa que oferece treinamentos e serviços nas áreas de segurança e testes de penetração, porém o time responsável pelo desenvolvimento da ferramenta é constituído por um grupo bem pequeno de desenvolvedores confiáveis e que apenas esses tem permissão para realizar modificações nos projeto. É possível instalá-lo no disco rígido da máquina, executa-lo diretamente de um live CD ou em uma máquina virtual, como *VirtualBox* (BROAD; BINDNER, 2014).

Figura 4 – Kali Linux executando o MSFConsole do Metasploit *framework*.

Fonte: Autoria própria.

3 MATERIAL E MÉTODOS

Neste capítulo são apresentadas as ferramentas e tecnologias empregadas na realização deste trabalho bem como a metodologia utilizada para a execução do teste de penetração na rede de dados da UTFPR Campus Medianeira.

3.1 MATERIAL

Os recursos citados nesta seção incluem todos os softwares necessários para a realização deste trabalho:

- VirtualBox versão 5.0.2;
- Kali Linux versão 2016.1;
- Nmap versão 7.01;
- Netcraft;
- Whois;
- NeXpose;
- Metasploit Framework versão 4.11.7;

***VirtualBox*:** é uma ferramenta de virtualização para multiplataformas, ou seja, uma vez instalada em ambiantes como Windows, Mac OS, Linux ou Solaris, permite estender a capacidade do computador em que foi instalado para executar vários sistemas operacionais ao mesmo tempo ()VirtualBox.org). Neste trabalho utilizou-se desta ferramenta com o intuito de permitir a execução do Kali Linux sem a necessidade de instalação no disco rígido da máquina utilizada para realizar os testes.

***Kali* Linux:** apresentado na Seção 2.9, Kali foi o sistema operacional utilizado para a realização do teste, uma vez que este já possui uma ampla gama de ferramentas que serão utilizadas durante a execução do teste já instaladas e pré configuradas como *Metasploit framework* e *Nmap*, bem como os *exploits* e *payloads* necessários.

***Nmap*:** do inglês *Network Mapper*, é uma ferramenta gratuita e de código aberto para descoberta de rede a auditoria de segurança. Ela utiliza pacotes IP de maneiras diferentes para determinar quais *hosts* estão ativos em uma rede, que serviços estes *hosts* estão disponibilizando, qual é o sistema operacional e a versão que eles estão executando, que tipo de filtro de pacotes e *firewalls* estão em uso e outras dezenas de características. NMAP foi projetado para escanear rapidamente grandes redes, porém funciona muito bem para apenas um único *host*, além de poder ser executado em todos os principais sistemas operacionais (nmap.org).

Esta ferramenta possui dois tipos básicos de escaneamento: *TCP connect* e o *Synchronise (SYN) scan* ou *Stealth Scan*, que em português seria escâner discreto. Ambos os escâneres são baseados no protocolo TCP, entretanto o primeiro, *TCP connect*, tem esse nome pois faz o uso da chamada de sistema *connect()* presente em sistemas UNIX para iniciar uma conexão com o dispositivo remoto. Se a chamada obtiver sucesso a conexão é criada e então o dispositivo é considerado ativo, caso contrario ele é considerado inativo (nmap.org).

Por outro lado o *Stealth scan* utiliza o processo de *handshake* triplo do protocolo TCP, que envolve a troca de 3 pacotes entre os *hosts* para sincronizá-los e estabelecer a conexão. O dispositivo que inicia a conexão envia uma pacote SYN para o dispositivo a ser conectado, que por sua vez responde com um SYN e um *Acknowledging* (ACK), que indica o recebimento do SYN anterior, combinados em um único pacote. Por fim o dispositivo que iniciou o *handshake* responde com um pacote ACK que indica o recebimento do pacote (SYN/ACK), então a conexão é estabelecida e a transferência de dados pode ser iniciada. O *Stealth Scan* não deixa que a conexão seja estabelecida completamente, ele apenas envia o pacote SYN ao dispositivo alvo e aguarda a resposta. Se o dispositivo responder com o pacote SYN/ACK significa que a porta esta aberta no dispositivo alvo e este está tentando estabelecer uma conexão TCP, neste caso o escâner envia um pacote *Reset* (RST) que derruba a conexão antes que a mesma possa ser completamente estabelecida (nmap.org).

Se a porta em questão se encontrar fechada, o dispositivo alvo enviará um pacote RST ao invés de SYN/ACK como resposta ao pacote SYN enviado pelo escâner. Dessa forma este tipo de escâner impede que a tentativa de conexão apareça em logs das aplicações no dispositivo alvo, entretanto pode ser facilmente detectado por mecanismos de detecção de intrusão (nmap.org). Por esta razão este o *SYN scanner* foi utilizado.

***Netcraft*:** é uma ferramenta web capaz de obter informações como versão do servidor web, sistema operacional e o endereço IP do servidor de um determinado site através de sua *Uniform Resource Locator* (URL) (Figura 5).

***Whois*:** esta ferramenta consegue buscar informações sobre o proprietário do site dada uma

URL, uma vez que todos os registradores de domínio mantém registros sobre os domínios que eles hospedam, incluindo telefone, endereço e email (Figura 5).

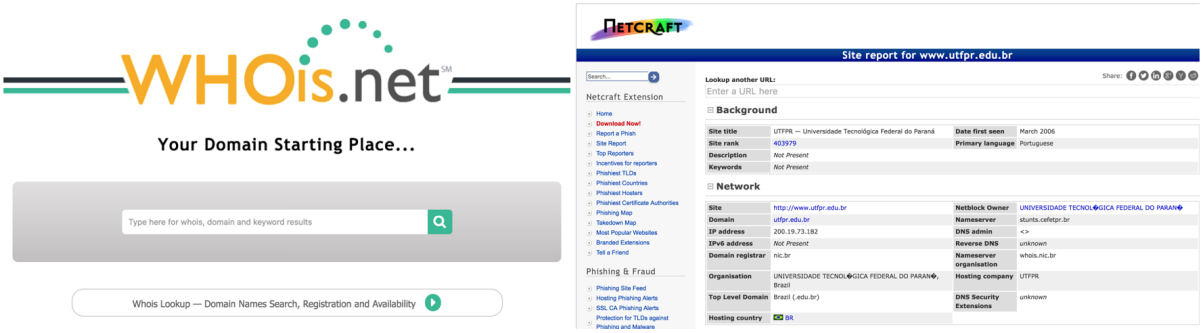


Figura 5 – Ferramentas Whois e Netcraft.

Fonte: Adaptado de https://www.whois.net/ e http://www.netcraft.com/.

***NeXpose*:** é um *scanner* de vulnerabilidades desenvolvido pela empresa *Rapid7*, que permite a avaliação e gerenciamento das mesmas. Além de buscar por vulnerabilidades no sistema alvo, esta ferramente analisa as descobertas e às divide em categorias de acordo com o risco de exploração que estas oferecem. Como nem todas as vulnerabilidades podem ser exploradas, o *NeXpose* se integra perfeitamente com o *Metasploit framework* para avaliar a explorabilidade de cada vulnerabilidade encontrada, relacionando o modulo ou *exploit* do *framework* capaz de explorá-la. Ele também apresenta correções efetivas para as falhas em questão e gera vários tipos de relatórios contendo informação sobre tudo que foi encontrado (Rapid7).

O *NeXpose* oferece uma gama de *templates*, ou modelos, que definem como o escâner será executado e como o mesmo irá interagir com os alvos. Para os testes aqui realizados, o *template* escolhido foi o *Full audit without Web Spider*. Este realiza uma auditoria completa em todos os sistemas utilizando apenas verificações seguras, incluindo vulnerabilidades de rede, verificação de instalação de *patches*/hotfix e auditoria completa na camada de aplicação. Somente as portas padrão são escaneadas, fazendo com que sua execução seja mais rápida, caso contrário, o escaneamento poderia durar dias se tratando de um grande número de dispositivos. Este *template* também não faz o uso de *Web Spiders* ou *Web Crawlers*, que são utilizados para indexação de páginas web e coleta de informações sobre as mesmas.

***Metasploit framework*:** como descrito na Seção 2.8, este *framework* oferece os *exploits* e *payloads* que foram utilizados durante o teste para atacar os sistemas alvo a tentar explorar as vulnerabilidades encontradas.

3.2 MÉTODOS

O teste foi constituído pelas etapas descritas na Seção 2.6, que são as seguintes:

1. Definir o escopo do teste juntamente com a administração da universidade;
2. Coletar informações sobre a infra-estrutura e sistemas;
3. Modelar as ameaças;
4. Buscar vulnerabilidades;
5. Explorar as vulnerabilidades;
6. Explorar os sistemas comprometidos;
7. Produzir um relatório técnico sobre o teste.

Etapa 1: nesta etapa discutiu-se com a Coordenação de Tecnologia da Informação (COGETI) da universidade os aspectos referentes ao escopo do teste. Foram definidos os horários para a execução do teste, os sistemas e *hosts* que fariam parte do teste e também quais ataques poderiam ser realizados, tendo em vista que certos ataques seriam capazes de interromper o funcionamento dos sistemas alvo. Ao fim desta etapa, um documento contendo os detalhes do escopo e uma autorização formal para a realização do teste foi criado e assinado pelo responsável pela COGETI da UTFPR Campus Medianeira. O modelo do documento que foi criado e assinado encontra-se no Apêndice A.

Etapa 2: após finalizada a etapa anterior, utilizou-se das coletas ativa e passiva de informações descritas na Seção 2.6. Para obter informações passivamente sobre os serviços online foram utilizadas as ferramentas *Netcraft* e *Whois*. Já para encontrar portas abertas e descobrir que serviços estão rodando nestas portas, ou seja, coletar informações ativamente, utilizou-se das funcionalidades do *Nmap scanner*. Com a conclusão desta etapa, foi criado um arquivo contendo informações como endereços IP, versão do sistema operacional, e quais serviços estão sendo executados em cada *host* pertencente ao escopo do teste.

Etapa 3: com o conhecimento obtido sobre o sistema nas etapas anteriores, foi possível ter uma visão geral de como os sistemas em teste operavam, o que possibilitou a determinação dos meios pelos quais o testador seria capaz de causar dano aos usuários e/ou administradores do sistema. Por exemplo, obter acesso a base de dados dos alunos da universidade e roubar informações sigilosas ou até mesmo alterar informações dentro da base de dados. Finalizada esta etapa, estavam definidos os objetivos a serem alcançados durante o teste.

Etapa 4: com as informações obtidas na etapa 2, foi executado o *scanner* de vulnerabilidade *NeXpose*, citado na Seção 2.6. Este foi utilizado para analisar os sistemas e serviços que estavam sendo executados em cada *host*, apontando quais são as potenciais vulnerabilidades em cada um

deles, podendo ser exploradas a fim de alcançar os objetivos definidos na etapa 3.

Etapa 5: na etapa da exploração de vulnerabilidades o objetivo foi obter acesso não autorizado aos sistemas alvo, explorando as vulnerabilidades encontradas na etapa anterior. Para tanto, foram utilizados *exploits* e *payloads* presentes no *Metasploit framework* e Kali Linux, porém estes são capazes apenas de explorar falhas específicas, ou seja, não existem *exploits* ou *payloads* genéricos, portanto eles foram escolhidos de acordo com as potenciais vulnerabilidades encontradas na etapa anterior. Uma vez obtido o acesso ao sistema alvo, esta etapa se da por encerrada.

Etapa 6: nesta etapa já deve ser possível acessar o sistema alvo de forma não autorizada, mas o que se pode fazer com isso para causar o maior dano possível para a universidade? Ou então, é possível comprometer algo a mais a partir do sistema comprometido? Estas são as questões que deveriam ser respondidas nesta fase.

Etapa 7: na última fase, o objetivo foi criar um relatório técnico descrevendo todas as vulnerabilidades encontradas, como foram exploradas, o que foi comprometido a partir disto e acima de tudo, indicar medidas que possam ser tomadas para eliminar as vulnerabilidades encontradas. Por fim, este relatório deve ser apresentado para o responsável pala COGETI da universidade, possibilitando assim o aperfeiçoamento dos mecanismos de segurança em uso.

4 RESULTADOS E DISCUSSÃO

Antes de dar início ao teste, foi criado um termo de autorização que descreve como deveria ser executado o teste, qual seria o escopo, quais os horários para execução e como seriam expostos os resultados. Este termo foi apresentado a Coordenadoria de Gestão de Tecnologia de Informação (COGETI) e assinado em duas vias pelo professor responsável e pelo aluno que se propôs a realizar o teste. O modelo do documento assinado encontra-se no Apêndice A.

4.1 COLETA DE INFORMAÇÕES

Após a obtenção da autorização, deu-se início a fase da coleta de informações, na qual utilizou-se das ferramentas *Whois*, *Netcraft* e *Nmap scanner* para obter informações sobre a rede passivamente e ativamente. Como o escopo do teste foi limitado a rede e serviços ativos no Campus Medianeira da UTFPR, as duas primeiras ferramentas não produziram resultados úteis, já que as mesmas tem como principal função obter informações sobre os domínios registrados para instituições ou companhias. Como o web site da universidade fica hospedado em servidores na reitoria, em Curitiba, o web site está fora do escopo deste teste.

Já o *Nmap scanner* que foi utilizado para obter informações sobre dispositivos e serviços ativos na rede. Para tanto, este *scanner* foi executado a partir de uma máquina conectada à rede cabeada da universidade utilizando o seguinte comando:

```
1 nmap -sS -O -oN WiredStealthScan.txt 192.168.*.*
```

A *flag* -sS indica que o tipo de escâner executado será o *Stealth Scan* descrito no capítulo anterior. Já a *flag* -O faz com que durante o escaneamento a ferramenta tente detectar qual o sistema operacional que está um uso no dispositivo alvo. O conjunto -oN WiredStealhScan.txt indica que a ferramenta deve criar um arquivo do tipo texto chamado 'WiredStealthScan' contendo os resultados obtidos. Por fim, a última parte do comando indica quais são os endereços a serem escaneados pela ferramenta, que neste caso são todos os

endereços IP entre 192.168.0.1 e 192.168.255.254. O escâner foi executado para os endereços das sub-redes 200.134.18.48/28 e 200.134.81.48/28 também.

Ao todo foram 65534 endereços de IP avaliados pela ferramenta dentro da faixa 192.168.*.*, que encontrou 193 dispositivos ativos, bem como seus respectivos sistemas operacionais e portas abertas. Já para a rede 200.134.*.* foram avaliados 28 endereços pertencentes as sub-redes 200.134.18.48/28 e 200.134.81.48/28, dos quais 6 foram encontrados ativos. Ao fim da execução os arquivos de saída foram gerados contendo os detalhes de todos os dispositivos encontrados ativos. A Figura 6 ilustra o resultado gerado pelo *Nmap* para um dos dispositivos encontrados, exibindo que o *host* encontra-se no endereço 192.168.30.218. Revela que as portas 22, 80, 139, 445, 3306, 8000, 8009 e 9418 estão aceitando conexões, bem como os respectivos serviços que estão atendendo em cada uma delas. Por fim, mostra que o sistema operacional em execução no dispositivo é Linux com o *kernel* na versão entre 3.11 e 3.14 e o número de saltos necessários para alcançar-lo, ou seja, por quantos dispositivos de rede os pacotes passam para saírem da máquina que está executando o escâner e chegarem ao dispositivo escaneado, que neste caso é 1.

```
Nmap scan report for 192.168.30.218
Host is up (0.0011s latency).
Not shown: 992 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3306/tcp open  mysql
8000/tcp open  http-alt
8009/tcp open  ajp13
9418/tcp open  git
MAC Address: 46:B9:EC:66:AC:96 (Unknown)
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.11 - 3.14
Network Distance: 1 hop
```

Figura 6 – Saída obtida a partir da ferramenta Nmap.

Fonte: Autoria própria.

Analisando os resultados obtidos com o *Nmap* foi possível se ter o conhecimento de como é a infraestrutura da rede cabeada, já que é possível identificar os roteadores e dispositivos de rede através de seu sistema operacional. Sabendo também quantos saltos são necessários para

alcançar cada dispositivo, se teve uma visão geral sobre estrutura física da rede. Além disso, foram identificados os dispositivos com uma maior possibilidade de conter vulnerabilidades analisando o número de portas abertas e quais serviços estavam sendo executados nas mesmas.

4.2 ANÁLISE DE VULNERABILIDADES

Com as informações sobre os dispositivos, sistemas e serviços ativos, iniciou-se a fase de análise de vulnerabilidades. Os dispositivos foram agrupados de acordo com o seu endereço de rede para a avaliação feita pelo *NeXpose*. Para o primeiro grupo, que continha os 193 dispositivos ativos com os endereços de IP iniciando em 192.168.*.*, foram reportadas no total 2.435 potenciais vulnerabilidades. Já para o segundo, contendo 6 dispositivos cujo os endereços pertenciam às sub-redes 200.134.18.48/28 e 200.134.81.48/28, o número total de potenciais vulnerabilidades encontradas foi 34.

Para classificar as vulnerabilidades pelo seu grau de severidade, o *NeXpose* utiliza o *Common Vulnerability Scoring System* (CVSS), que consiste em um modelo quantitativo capaz de calcular pontuações para vulnerabilidades. Este modelo é mantido pelo NIST, com o objetivo de criar um padrão para classificação de vulnerabilidades. Segundo Weidman (2014), este sistema atribui a pontuação com base na complexidade de acesso, na autenticação requerida e no impacto causado ao sistema caso a vulnerabilidade seja explorada, permitindo assim a criação de um *ranking* de vulnerabilidades, no qual quanto maior a pontuação, mais crítica a vulnerabilidade é, e maior a prioridade de remediação. Na Figura 7 é ilustrado como foram classificadas as vulnerabilidades encontradas de acordo com as suas respectivas pontuações CVSS.

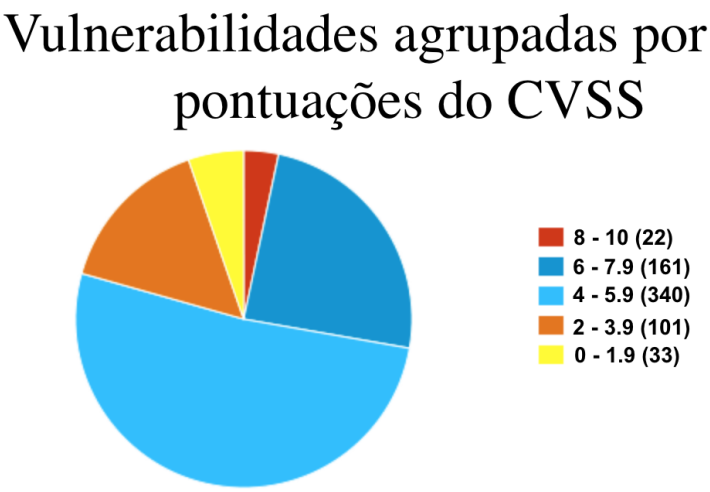
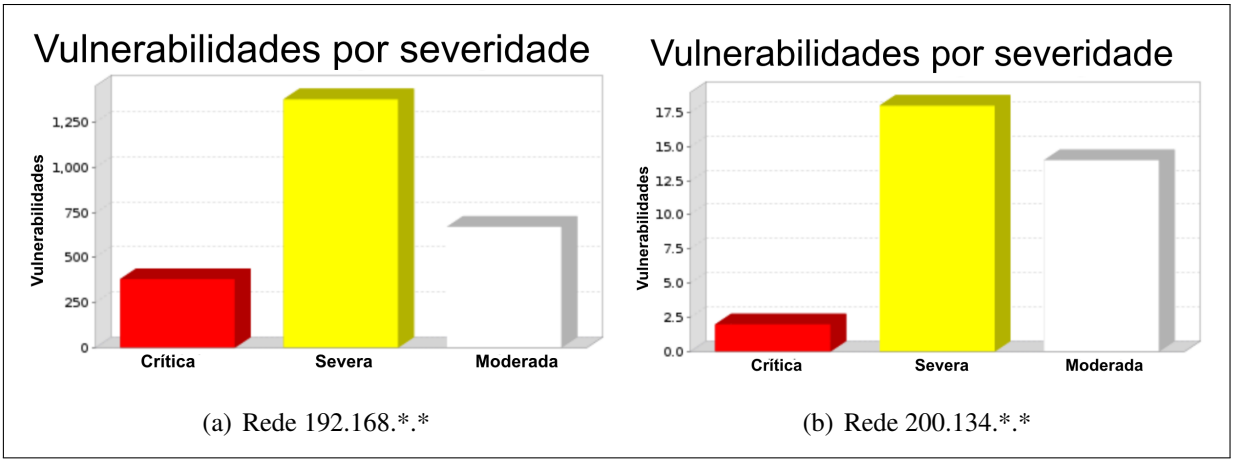


Figura 7 – Vulnerabilidades classificadas de acordo com sua pontuação do CVSS.
Fonte: Relatório criado pela ferramenta *NeXpose*.

As vulnerabilidades também podem ser categorizadas entre críticas, severas e moderadas. Sendo que as críticas requerem remediação imediata, pois estas são relativamente simples de ser exploradas por atacantes e podem fornecer acesso total ao sistema afetado. As vulnerabilidades classificadas como severas normalmente são de mais difícil exploração e em geral não fornecem o mesmo acesso aos sistemas afetados. Já as categorizadas como moderadas também devem ser removidas, porém não demandam da mesma urgência que as anteriores. Normalmente estas apenas fornecem informações que podem ajudar o atacante a arquitetar ataques subsequentes. Na Figura 8(a) é ilustrado como foram categorizadas, de acordo com este padrão, as vulnerabilidades encontradas na rede 192.168.*.*. Na Figura 8(b) está ilustrado o mesmo porém para as sub-redes 200.134.*.*.



(a) Rede 192.168.*.* (b) Rede 200.134.*.*

Figura 8 – Vulnerabilidades de acordo com sua severidade.

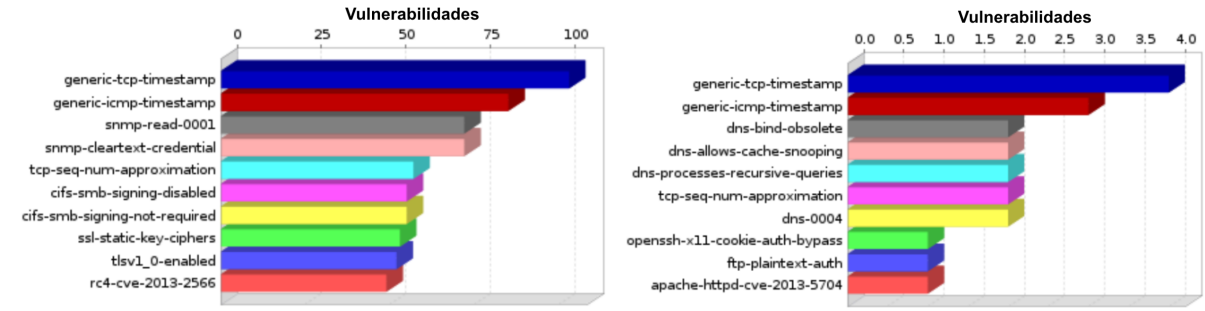
Fonte: Relatório criado pela ferramenta *NeXpose*.

Outra forma de classificação de vulnerabilidades disponível na ferramenta é através do nível de habilidade necessário para explorar tal vulnerabilidade. A Figura 9 ilustra como foram classificadas as potenciais vulnerabilidades encontradas, onde novato é a classe que contém as vulnerabilidades mais fáceis de se explorar, e consequentemente as que precisam de maior atenção do administrador. A classe Especialista contém as que são consideradas mais difíceis ou que exigem maior conhecimento para a execução dos *exploits* capazes de explorá-las. Por fim, a classe Exploração desconhecida contém as vulnerabilidades já publicadas porém não possuem *exploits* publicamente disponíveis relacionados.

Figura 9 – Vulnerabilidades classificadas de acordo com o nível de habilidade necessário para explorá-las.

Fonte: Relatório criado pela ferramenta *NeXpose*.

Nas Figuras 10(a) e 10(b) são ilustradas quais foram as vulnerabilidades mais comuns encontradas em cada uma das redes. Para ambas, as vulnerabilidades mais comuns foram *generic-tcp-timestamp* e *generic-icmp-timestamp*. As duas são categorizadas com o nível de severidade moderado, já que quando exploradas apenas proveem informações sobre o sistema. A primeira faz com que o dispositivo responda com um *TCP timestamp*, permitindo ao atacante determinar o seu horário de funcionamento. Já a segunda faz com que o dispositivo responda a um *ICMP timestamp request*, nesta resposta estão contidas a data e hora do sistema, que em teoria, podem ser utilizadas em alguns sistemas para explorar geradores de números aleatórios baseados em tempo fracos presentes em outros serviços.

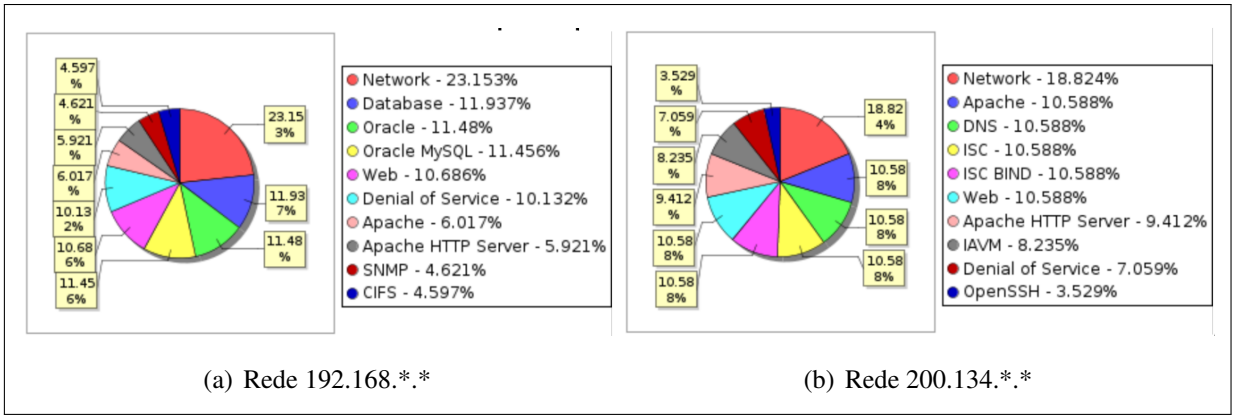


(a) Rede 192.168.*.* (b) Rede 200.134.*.*

Figura 10 – Vulnerabilidades mais comuns encontradas em cada rede.

Fonte: Relatório criado pela ferramenta *NeXpose*.

Outra forma de agrupar as vulnerabilidades encontradas disponível na ferramenta *NeXpose*, é pelo seu tipo, como ilustrado nas Figuras 11(a) e 11(b). As categorias aqui podem ser agrupadas em 5 classes principais: Vendedor, *Oracle* por exemplo; Vendedor e produto, como *Oracle MySQL*; Classe do produto, como *Database*; Tipo da vulnerabilidade, *Denial of Service* ou *Remote Execution* por exemplo; e protocolo, como HTTP.



(a) Rede 192.168.*.* (b) Rede 200.134.*.*

Figura 11 – Tipos de vulnerabilidades encontrados em cada rede.

Fonte: Relatório criado pela ferramenta *NeXpose*.

4.3 EXPLORAÇÃO

Devido ao grande número de dispositivos encontrados e consequentemente o de potenciais vulnerabilidades, tornou-se inviável a validação da explorabilidade de todas. Entretanto, entre as vulnerabilidades mais críticas, escolheu-se uma de forma aleatória para a exploração, a *MS12_020*, que consiste em uma falha no protocolo *Remote Desktop Protocol* (RDP) da *Microsoft*, que por padrão roda na porta 3389. Este protocolo foi construído para oferecer o serviço de acesso remoto, com uma interface gráfica capaz se conectar a outro computador através de uma conexão de rede. A falha em questão afeta todas as versões do *Windows* liberadas até de março de 2012 incluindo o *Windows XP*, *Windows Server 2003*, *Windows Vista*, *Windows Server 2008* e *Windows 7*. A mesma foi descoberta pelo *NeXpose* na etapa anterior, estando presente em 3 dispositivos na rede, entretanto apenas 1 foi explorado.

Com o fim de validar a presença da falha nos dispositivos citados utilizou-se o módulo *auxiliary/scanner/rdp/ms12_020_check*, indicado pela ferramenta *NeXpose* e presente no *Metasploit Framework*. Esta é capaz de checar se a vulnerabilidade está presente em um dispositivo sem explorá-la. Para tanto, foram utilizados os seguintes comandos:

```
1 msfconsole
2 use  auxiliary/scanner/rdp/ms12_020_check
3 set RHOST "IP do alvo"
4 run
```

Onde 1 inicia o *MSFConsole* do *Metasploit*, o comando 2 seleciona o módulo a ser utilizado, 3 indica ao módulo qual é o endereço do dispositivo a ser avaliado, e por fim, o comando 4 executa o módulo. A Figura 12 ilustra os resultados obtidos para os *hosts* nos endereços 192.168.30.110, 192.168.30.102 e 192.168.6.174.

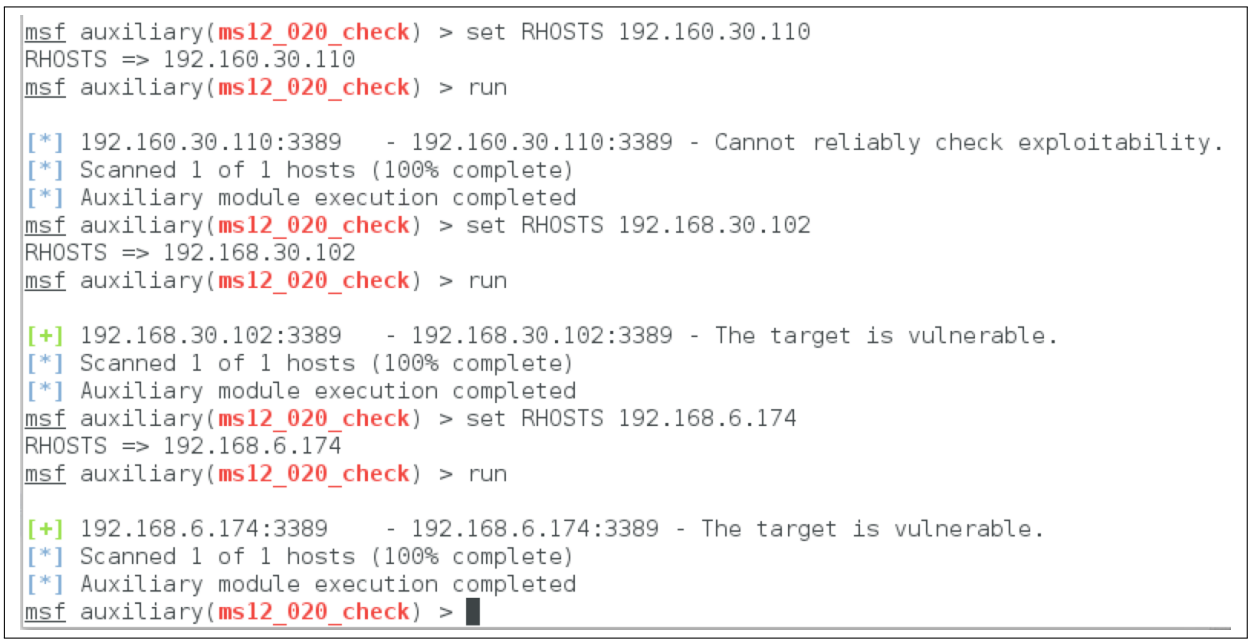


Figura 12 – Utilização do módulo ms12_020_check para validar a presença da vulnerabilidade *MS12_020*.

Fonte: Autoria própria.

A exploração desta falha pode acarretar no travamento do sistema operacional da máquina e no aparecimento da tela azul do *Windows* contendo os códigos de erros. Para validar a explorabilidade desta falha, escolheu-se um dos *hosts* mencionados acima aleatoriamente e então o *exploit MS10_020_maxchannelids*, também sugerido pelo *NeXpose*, foi utilizado. A fim de realizar o ataque os comandos a seguir foram executados no *MSFConsole*:

```
1 use  auxiliary/dos/windows/rdp/ms12_020_maxchannelids
2 set RHOST "IP do alvo"
3 show options
4 exploit
```

O comando 1 seleciona o *exploit* a ser utilizado, em seguida, o comando 2 indica qual será o alvo, 192.168.6.147 neste caso. Por fim, o comando 4 executa o ataque. Para visualizar a atual configuração do *exploit* ou módulo em uso, pode-se utilizar o comando 3, ilustrado na Figura 13, bem como a execução do ataque, que foi bem sucedida. Ao término da execução

do *exploit* não foi mais possível se comunicar com o *host* atacado, indicando o travamento completo do sistema.

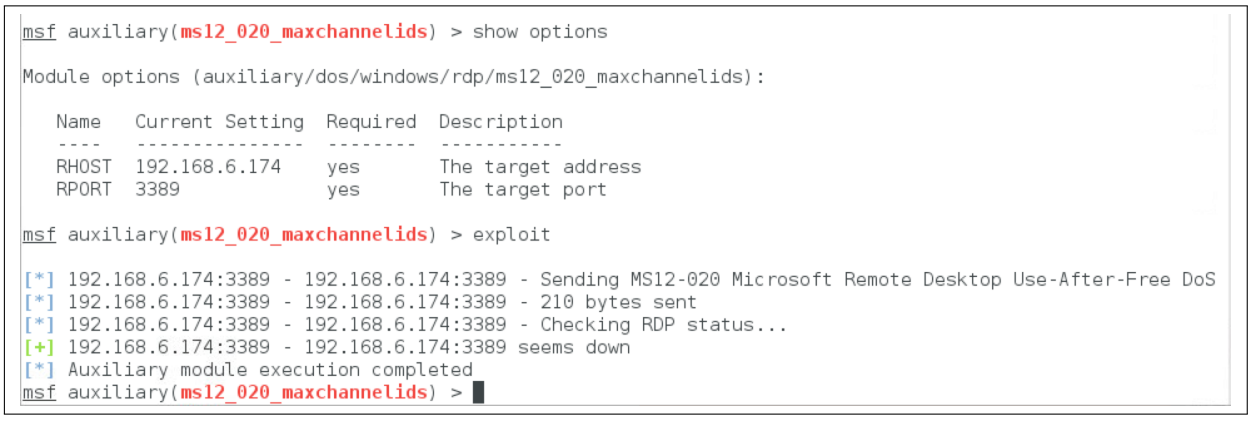


Figura 13 – Utilização do *exploit* ms12_020_maxchannelids para explorar a vulnerabilidade *MS12_020*.

Fonte: Autoria própria.

Na fase de pós-exploração o objetivo seria avaliar o que mais poderia ser explorado a partir dos sistemas comprometidos na fase de exploração, ou seja, seria necessária a obtenção de acesso em algum sistema ou a desativação de algum mecanismo de segurança na fase anterior. Como na fase de exploração apenas obteve-se o travamento de um dispositivo, que não era essencial para o funcionamento da rede e nem rodava algum tipo sistema de defesa da rede a fase de pós-exploração não obteve resultados.

Para remediar esta vulnerabilidade nos dispositivos afetados, basta acessá-los, fazer o *download* e instalar a correção oferecida pela *Microsoft*, disponível no seguinte link:

```
1 http://go.microsoft.com/fwlink/?LinkId=232664
```

4.4 RELATÓRIO

Para apresentar todas as potenciais falhas encontradas durante este teste de penetração à COGETI da universidade foram criados 2 sumários executivos, um para cada rede escaneada, bem como 2 relatórios técnicos. Os sumários executivos dão uma visão geral sobre qual é a atual situação de cada uma das redes, ilustrando o número de vulnerabilidades encontradas,

classificando-as de acordo com a sua severidade, apresentando quais foram as vulnerabilidades mais comuns encontradas, listando quais são as que oferecem maior risco, quais os sistemas operacionais presentes na rede, categorizando as vulnerabilidades de acordo com os serviços que as mesmas afetam entre outras. Todas estas informações estão expostas de forma intuitiva, por meio de gráficos, permitindo que pessoas desprovidas de conhecimento técnico na área de computação tenham um panorama da situação da segurança oferecida rede.

Por outro lado, a fim de possibilitar aos responsáveis pela manutenção da rede corrigir as potenciais falhas encontradas, os relatórios técnicos trazem todas as informações relativas a cada falha. Listam todos os dispositivos encontrados, seus endereços de IP, seus sistemas operacionais e seus pseudônimos. Além disso, descrevem cada vulnerabilidade apresentando uma breve descrição da mesma, qual é o possível resultado de sua exploração, quais foram os dispositivos onde esta falha foi encontrada, referências onde o responsável pode buscar mais informações, e por fim, como ela pode ser remediada.

4.5 CONSIDERAÇÕES

O teste foi realizado a partir de uma máquina virtual criada e hospedada em um dos servidores presentes na rede interna da universidade, com o IP 192.168.30.164. Por esta razão, alguns mecanismos de segurança podem ter sido de certa forma evitados, uma vez que a *Virtual Local Area Network*(VLAN) na qual os servidores estão contidos possui privilégio que não são oferecidos em outras VLANs.

Contudo, durante todos os escaneamentos e ataques realizados neste teste não se teve a preocupação em se manter imperceptível, pois um dos objetivos era avaliar se existem mecanismos de segurança presentes na rede capazes de detectar atividades suspeitas, alertando os responsáveis para que os mesmos pudessem tomar alguma medida ou até mesmo interrompendo as atividades em questão de forma automatizada. Para o escaneamento feito com o NMAP utilizou-se o modo *Stealth Scan*, que apesar de não aparecer em *logs* de aplicações, cria um tráfego incomum na rede durante sua execução, podendo ser facilmente detectado por sistemas de detecção/prevenção de intrusão.

O escâner de vulnerabilidades utilizado também gera um tráfego anormal durante sua execução, deixando clara a inexistência de sistemas como estes na rede, já que nenhuma das duas ferramentas foi bloqueada durante sua execução. Sistemas de detecção/prevenção de

intrusão são uma ótima maneira de dificultar o trabalho de atacantes, pois estes emitem alertas ou bloqueios quando detectam a execução de atividades maliciosas, tornando assim muito mais difícil a obtenção de informações sobre a infraestrutura da rede por parte do atacante.

Por esta razão, sugere-se a implantação de um sistema de detecção de intrusão de rede como o *Snort*. Segundo Snort.org (2016), esta é uma ferramenta de código aberto capaz de analisar o tráfego de uma rede em tempo real para detectar escaneamentos e ataques, emitindo alertas para a administração possibilitando que contra-medidas sejam adotadas. O *Snort* possui 3 modos de operação, o modo *sniffer* faz com que a ferramenta leia os pacotes de rede a apresente-os no console, o modo *packet logger* que lê e armazena os pacotes em disco, possibilitando análises posteriores e o modo *intrusion detection*, onde a ferramenta analisa o tráfego de acordo com uma série de regras definidas pelo usuário, realizando ações específicas baseadas no que foi identificado na análise.

Sugere-se também o uso de ferramentas como o *NeXpose* pela administração da rede da universidade. A ferramenta pode ser utilizada para realizar auditorias periódicas e o gerenciamento das vulnerabilidades, bem como seus riscos, permitindo assim que as mesmas sejam encontradas e corrigidas antes que possam ser exploradas por atacantes.

5 CONCLUSÕES

Um dos objetivos deste trabalho foi determinar se exitem falhas de segurança ou vulnerabilidades que possam permitir que os sistemas e serviços presentes no campus sejam violados. Com os resultados obtidos utilizando o *NeXpose*, foram identificadas mais de 2.000 potenciais vulnerabilidades na rede escaneada, o que foi um número muito acima do esperado, tornando inviável validação de todas elas.

Grande parte das vulnerabilidades encontradas não possuem *exploits* publicamente conhecidos capazes de explorá-las, ou seja, a exploração das mesmas demanda de muito conhecimento técnico por parte do testador. Entretanto para as que possuem, a tarefa de encontrá-los foi simples, uma vez que o próprio escâner de vulnerabilidades indica quais são quando eles existem.

Já a fase de exploração das vulnerabilidades trouxe resultados abaixo do esperado, já que foi possível explorar com sucesso apenas uma das vulnerabilidades encontradas, a qual ocasionou o travamento da máquina, não sendo possível a obtenção de acesso remoto a nenhum dispositivo.

Sobre determinar se os protocolos de segurança implementados atualmente são capazes de detectar e interromper ataques, as resposta foi não. Durante todo o escaneamento da rede, que foi feito a partir da mesma máquina com um endereço de IP fixo, não foi encontrado nenhum tipo de barreira ou impedimento, o que indica a inexistência de mecanismos de detecção de intrusão e sugerindo sua implantação.

A solução para cada potencial falha encontra-se no relatório técnico entregue para a administração da rede da universidade, possibilitando assim que as mesmas sejam corrigidas antes que atacantes possam explorá-las.

O objetivo geral do trabalho foi determinar se os mecanismos de segurança presentes na rede da universidade são capazes de garantir o funcionamento e a qualidade dos serviços em execução nesta localidade. Analisando o número de vulnerabilidades encontradas e explorando pelo menos umas delas, comprovou-se que as falhas existem e são exploráveis, ilustrando assim que o cenário atual da rede não oferece um nível de segurança satisfatório.

5.1 TRABALHOS FUTUROS

Para trabalhos futuros, existem alguns tópicos que devem ser estudados, como a validação da explorabilidade de outras vulnerabilidades encontradas, já que grande parte delas não foi explorada, e qual o seu impacto dentro da rede, bem como validar se as correções sugeridas neste trabalho foram aplicadas e se elas são efetivas.

A rede administrativa da universidade (UTFPR-ADM) não foi avaliada durante a realização deste trabalho pois a mesma é destinada ao uso exclusivo dos funcionários, possuindo uma autenticação WPA2 *Enterprise*, exigindo usuário e senha de um funcionário para acesso . A segurança desta rede também deve ser avaliada, seja por meio de um teste de penetração ou de uma análise de vulnerabilidades.

Outro ponto a ser estudado é o impacto da implantação do sistema de detecção de intrusão, tentando executar os mesmos escâneres executados aqui após a implantação do mesmo, testando assim sua efetividade.

Existem muitas outras possibilidades de estudo relacionadas a esta área, estes tópicos são apenas algumas delas.

APÊNDICE A – TERMO DE AUTORIZAÇÃO

Acordo para realização do teste de penetração

Este acordo é firmado entre Matheus Marchezan De Marco, residente na rua Paraná, 1888, centro, aluno do curso de Ciência da computação na Universidade Tecnológica Federal do Paraná no câmpus Medianeira (UTFPR-MD), portador do RA 131523, orientado pela professora Glória Patricia Lopez Sepulveda. E a UTFPR-MD localizada na avenida Brasil, 4232, parque independência.

A UTFPR-MD e o aluno Matheus Marchezan De Marco entraram em acordo em relação aos seguintes termos sobre a execução do teste de segurança a ser realizado:

1. O aluno realizará um teste de penetração parcialmente automatizado que tentará encontrar falhas de segurança/vulnerabilidades ou erros de configuração na rede de dados presente no câmpus Medianeira da universidade.

2. Detalhes sobre as restrições do escopo do teste estão contidos no Anexo 1.

3. O aluno tem a autorização da universidade para a execução dos testes durante o período estabelecido no Anexo 2.

4. A qualquer momento durante a execução do teste a universidade pode requerer que o mesmo seja pausado.

5. O aluno realizará o teste de maneira responsável e profissional de acordo com as melhores práticas do setor de segurança. Não poderá modificar ou alterar intencionalmente quaisquer aplicações, dados, programas ou componentes presentes na rede de dados ou nos sistemas computacionais da universidade.

6. O aluno poderá divulgar apenas as falhas encontradas que forem remediadas e com o aval da universidade. Quaisquer outras informações relacionadas aos testes são confidenciais e serão tratadas como tal.

7. Informações confidenciais obtidas pelo aluno que não serão necessárias para a criação do relatório técnico serão destruídas imediatamente após a realização dos testes. O restante das informações serão destruídas após a entrega do relatório.

8. Este acordo e anexos constitui a totalidade do acordo entre as partes relacionadas ao teste. Qualquer mudança, alteração ou modificação só será válida, caso seja feita por escrito, datada e assinada por ambas as partes.

Anexo 1 - A universidade autoriza a execução do teste de penetração pelo aluno afim de avaliar as políticas de segurança implementadas atualmente incluindo todos os dispositivos e recursos presentes neste câmpus exceto (Ip's, URL's, servidores, VLAN's, ou qualquer outro recurso ou dispositivo):

_____

_____

_____

_____

_____

_____

Anexo 2 - O aluno está autorizado pela UTFPR-MD para a realização dos testes apenas durante o período e horários descritos aqui:

_____

_____

_____

_____

_____

_____

Nome: Matheus Marchezan De Marco	Nome: _____
Título: Testador	Título: _____
Data: __/__/__	Data: __/__/__
Assinatura:_____	Assinatura: _____

Firmado em Medianeira, Paraná, no dia _____ em duas vias originais assinadas, cada parte recebendo uma.

REFERÊNCIAS

AHMADIAN, M. M.; SHAHRIARI, H. R.; GHAFFARIAN, S. M. Connection-monitor & connection-breaker: A novel approach for prevention and detecting of high suvivable ransomwares. **12th ISCISC International Iranian Society of Cryptology Conference on Information Security and Cryptology, 2015, Proceedings**, p. 79–84, 2015.

ALSUNBUL, S.; LE, P.; TAN, J. A defense security approach for infrastructures against hacking. **12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom, 2013, Proceedings**, p. 1600–1606, 2013.

ANTUNES, N.; VIEIRA, M. Penetration testing for web services. **Computer (Volume:47, Issue: 2)**, IEEE Computer Society, p. 30–36, 2014. ISSN 0018-9162.

BROAD, J.; BINDNER, A. **Hacking with Kali**: Practical penetration testing techniques. [S.l.: s.n.], 2014. 1–227 p. ISBN 9780124077492.

BURNST, J. F. **Hacker's Extradition to U.S. More Likely**. 2009. Acessado em 11/2016. Disponível em: <http://www.nytimes.com/2009/08/01/world/europe/01britain.html>.

CISCO. Como lidar com a sequência de ataque completa : antes , durante e depois de um ataque É hora de ter um novo modelo de segurança. p. 1–8, 2014. Acessado em 05/2016. Disponível em: <http://www.cisco.com/c/dam/r/pt/br/internet-of-everything-ioe/assets/pdfs/sec_bda_wp_cte_pte_etmg_pt-br_39724.pdf>.

ENGEBRETSON, P. **The Basics of Hacking and Penetration Testing**: Ethical Hacking and Penetration Testing Made Easy. [S.l.: s.n.], 2011. 178 p. ISSN 13534858. ISBN 9781597496551.

FARAH, T. SQLi Penetration Testing of Financial Web Applications : Investigation of Bangladesh Region. **(WorldCIS) World Congress on Internet Security, 2015**, IEEE, p. 146–151, 2015.

HOLIK, F. et al. Effective penetration testing with Metasploit framework and methodologies. **CINTI - 15th IEEE International Symposium on Computational Intelligence and Informatics, 2014, Proceedings**, p. 237–242, 2015.

HOLLINGER, R. C.; LANZA-KADUCE, L. The process of criminalization: The case of computer crime laws*. **Criminology**, Blackwell Publishing Ltd, v. 26, p. 101–126, 1988. ISSN 1745-9125.

Kali Linux. **What is Kali?** 2016. Acessado em 05/2016. Disponível em: <https://www.kali.org/>.

KENNEDY, D. et al. **Metasploit**. [S.l.: s.n.], 2011. 1–299 p. ISBN 9781593272883.

nmap.org. **Nmap security scanner**. 2016. Acessado em 05/2016. Disponível em: <https://nmap.org/>.

PENDERGRASS, W. S. What is Anonymous?: A case study of an information systems hacker activist collective movement. **ProQuest Dissertations and Theses**, n. May, p. 405, 2013.

pentest-standard.org. **Penetration Testing Execition Standard**. 2016. Acessado em 04/2016. Disponível em: <http://www.pentest-standard.org/index.php>.

PRANDINI, M.; RAMILLI, M. Towards a practical and effective security testing methodology. In: . [S.l.]: IEEE Computer Society, 2010. (ISCC '10), p. 320–325. ISBN 978-1-4244-7754-8.

Rapid7. **Nexpose: Reduce your risk of a breach**. 2016. Acessado em 05/2016. Disponível em: <https://www.rapid7.com/products/nexpose/>.

SANGER, D.; PERLROTH, N. **Bank Hackers Steal Millions via Malware**. 2015. Acessado em 10/2016. Disponível em: <http://www.nytimes.com/2015/02/15/world/bank-hackers-steal-millions-via-malware.html?ref=business&_r=1>.

SCARFONE, K. A. et al. **SP 800-115. Technical Guide to Information Security Testing and Assessment**. Gaithersburg, MD, United States, 2008.

SCHNEIER, B. **Applied Cryptogarphy: Protocols, Algorithms and source code in C**. [S.l.]: John Wiley & Sons, Inc., 1996. ISBN 0471128457.

SNORT.ORG. **Snort**. 2016. Acessado em 10/2016. Disponível em: <https://www.snort.org>.

STOUT, D. **Youth Sentenced in Government Hacking Case**. 2000. Acessado em 11/2016. Disponível em: <http://www.nytimes.com/2000/09/23/us/youth-sentenced-in-government-hacking-case.html>.

SYMANTEC. Internet Security Threat Report. p. 1–81, 2016.

The MITRE Corporation. **Common Vulnerabilities and Exposures**. 2016. Acessado em 10/2016. Disponível em: <https://cve.mitre.org/>.

VirtualBox.org. **VirtualBox Manual**. 2016. Acessado em 05/2016. Disponível em: <https://www.virtualbox.org/manual/ch01.html>.

WEIDMAN, G. **Penetration Testing**: A hands-on introduction to hacking. [S.l.: s.n.], 2014. 1–531 p. ISBN 9781593275648.