

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

**EMANUEL FELIPE GIROLDO MAZZER**

**IMPLEMENTAÇÃO DE UMA FERRAMENTA PARA  
AUXILIAR NA DETECÇÃO E VISUALIZAÇÃO DE  
VULNERABILIDADES EM REDES DE  
COMPUTADORES**

MONOGRAFIA

**CAMPO MOURÃO**

**2017**

**EMANUEL FELIPE GIROLDO MAZZER**

**IMPLEMENTAÇÃO DE UMA FERRAMENTA PARA  
AUXILIAR NA DETECÇÃO E VISUALIZAÇÃO DE  
VULNERABILIDADES EM REDES DE  
COMPUTADORES**

Proposta de Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso de Bacharelado em Ciência da Computação do Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Luiz Arthur Feitosa Santos

**CAMPO MOURÃO**

**2017**



# Resumo

---

Mazzer, Emanuel F. G. Implementação de uma ferramenta para auxiliar na detecção e visualização de vulnerabilidades em redes de computadores. 2017. 32. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2017.

A Internet vem se tornando cada vez mais popular e indispensável para a maioria da população. Hoje a Internet é uma das maiores fontes de informação, utilizada para a realização de diversas atividades do dia a dia, tais como: transações bancárias, compras *online*, entre outras. Porém, para que tudo funcione de maneira correta, é necessário garantir a segurança dos dados que trafegam tanto na Internet quanto em quaisquer outras redes de computadores. Para que a segurança dos dados ocorram, vulnerabilidades existentes em dispositivos conectados à rede devem ser encontradas e corrigidas, e novas vulnerabilidades devem ser evitadas, mas a detecção de vulnerabilidades nem sempre é uma tarefa simples, pois na maioria das vezes, vulnerabilidades estão contidas em *softwares* de terceiros, ou até mesmo nas configurações das redes, realizadas pelos próprios administradores e podem acabar passando despercebidas. Entretanto existem programas que são responsáveis por analisar os dispositivos conectados na rede em busca de vulnerabilidades, tais programas são chamados de *scanners*, e podem auxiliar os administradores de redes na detecção de vulnerabilidades. No entanto, quando as redes possuem centenas de dispositivos conectados, os *scanners* tendem a retornar grandes quantidades de informações, demandando um grande tempo para que os administradores analisem as informações, testem as vulnerabilidades encontradas e caso sejam confirmadas, corrijam tais vulnerabilidades. Dessa maneira, este trabalho tem como objetivo a implementação de uma ferramenta, que seja capaz de auxiliar na detecção e representação de vulnerabilidades em redes de computadores, utilizando *scanners*, os dispositivos conectados à rede serão analisados em busca de vulnerabilidades, vulnerabilidades encontradas serão armazenadas criando uma base de dados contendo, o *Internet Protocol* (IP) do dispositivo e as vulnerabilidades associadas a tal dispositivo. Essa base de dados será utilizada para montar um perfil para cada dispositivo analisado, tal perfil será comparado com perfis que já sofreram algum tipo de invasão, que serão obtidos através da análise de IPs cedidos pelo sistema Hórus, um sistema de alertas antecipados contra ataques cibernéticos. Caso haja alguma similaridade entre as vulnerabilidades comparadas, o administrador da rede será

notificado. Primeiramente, para testar qual *scanner* sera utilizado na implementação da ferramenta, foi utilizado um conjunto de teste, contendo 100 IPs desconhecidos, espalhados pela Internet. Tal conjunto foi analisado por dois *scanners* de vulnerabilidades diferentes, *Open Vulnerability Assessment System* (OpenVAS) e Nessus. Mais de 170 vulnerabilidades foram encontradas em 61 IPs distintos.

**Palavras-chaves:** Vulnerabilidades, Redes, Segurança, Cybersegurança.

# Lista de figuras

---

2.1	Arquitetura do <i>Open Vulnerability Assessment System</i> (OpenVAS) (OPENVAS, 2014). . . . .	15
4.1	Etapas necessárias para realização do projeto. . . . .	21
4.2	Arquivo <i>JavaScript Object Notation</i> (JSON) gerado a partir das informações .	22

# Lista de tabelas

---

2.1	Ferramentas utilizadas pelo OpenVAS . . . . .	16
4.1	Cronograma das Atividades . . . . .	24

# Siglas

---

CSRF: *Cross-Site Request Forgery*  
CVE: *Common Vulnerabilities and Exposures*  
CVSS: *Common Vulnerability Scoring System*  
GPL: *General Public Licence*  
GSA: *Greenbone Security Assistant*  
HTML: *HyperText Markup Language*  
IP: *Internet Protocol*  
JSON: *JavaScript Object Notation*  
NASL: *Nessus Attack Scripting Language*  
NVT: *Network Vulnerability Tests*  
OMP: *OpenVAS Manegement Protocol*  
OpenVAS: *Open Vulnerability Assessment System*  
OTP: *OpenVAS Transfer Protocol*  
pentest: *Penetration testing*  
PHP: *Personal Home Page*  
SID: *Session Identifier*  
SQL: *Structured Query Language*  
URL: *Uniform Resource Locator*  
XML: *eXtensible Markup Language*



# Sumário

---

<b>1</b>	<b>Introdução</b>	<b>8</b>
1.1	Considerações preliminares . . . . .	8
<b>2</b>	<b>Conceitos</b>	<b>10</b>
2.1	Ameaças . . . . .	10
2.2	Vulnerabilidades . . . . .	11
2.3	Ferramentas de análise de vulnerabilidades . . . . .	12
2.3.1	Nessus . . . . .	13
2.3.2	OpenVAS . . . . .	14
2.4	ElasticSearch . . . . .	16
2.5	Kibana . . . . .	16
<b>3</b>	<b>Trabalhos Relacionados</b>	<b>17</b>
<b>4</b>	<b>Proposta</b>	<b>20</b>
4.1	Metodologia . . . . .	20
4.2	Cronograma de Atividades . . . . .	23
<b>5</b>	<b>Experimentos e Resultados</b>	<b>25</b>
5.1	Experimentos . . . . .	25
5.2	Resultados . . . . .	26
5.3	Considerações Finais . . . . .	26
<b>6</b>	<b>Conclusões</b>	<b>27</b>
6.1	Considerações finais ou parciais . . . . .	27
6.2	Sugestões para Trabalhos Futuros . . . . .	27
	<b>Apêndices</b>	<b>29</b>
<b>A</b>	<b>Instalação de Ferramentas</b>	<b>30</b>
	<b>Referências</b>	<b>31</b>

---

# Introdução

---

## 1.1. Considerações preliminares

A Internet vem se tornando cada vez mais popular e indispensável para a maioria da população. Hoje a Internet é uma das maiores fontes de informação, utilizada para a realização de diversas atividades do dia a dia, tais como: transações bancárias, compras *online*, entre outras (FISCHER, 2014). Porém, para que tudo funcione de maneira correta, é necessário garantir a segurança dos dados que trafegam tanto na Internet quanto em quaisquer outras redes de computadores. Para que a segurança dos dados ocorram, vulnerabilidades existentes em dispositivos conectados à rede devem ser encontradas e corrigidas, e novas vulnerabilidades devem ser evitadas, mas a detecção de vulnerabilidades nem sempre é uma tarefa simples, pois na maioria das vezes, vulnerabilidades estão contidas em *softwares* de terceiros, ou até mesmo nas configurações das redes, realizadas pelos próprios administradores e podem acabar passando despercebidas.

Entretanto existem programas que são responsáveis por analisar os dispositivos conectados na rede em busca de vulnerabilidades, tais programas são chamados de *scanners*, e podem auxiliar os administradores de redes na detecção de vulnerabilidades, os *scanners* utilizam bases de dados atualizadas e realizam vários testes para identificar possíveis vulnerabilidades (WEIDMAN, 2014). No entanto, quando as redes possuem centenas de dispositivos conectados, os *scanners* tendem a retornar grandes quantidades de informações, demandando um grande tempo para que os administradores analisem as informações, testem as vulnerabilidades encontradas e caso sejam confirmadas, corrijam tais vulnerabilidades.

Dessa maneira, este trabalho tem como objetivo a implementação de uma ferramenta, que seja capaz de auxiliar na detecção e representação de vulnerabilidades em redes de computadores, utilizando *scanners*, os dispositivos conectados à rede serão analisados em

busca de vulnerabilidades, vulnerabilidades encontradas serão armazenadas criando uma base de dados contendo, o *Internet Protocol* (IP) do dispositivo e as vulnerabilidades associadas a tal dispositivo. Essa base de dados será utilizada para montar um perfil para cada dispositivo analisado, tal perfil será comparado com perfis que já sofreram algum tipo de invasão, que serão obtidos através da análise de IPs cedidos pelo sistema Hórus, um sistema de alertas antecipados contra ataques cibernéticos. Caso haja alguma similaridade entre as vulnerabilidades comparadas, o administrador da rede será notificado.

O presente trabalho é dividido em capítulos, o capítulo 2 apresenta a fundamentação teórica, revisando conceitos importantes. O capítulo 3 comenta sobre os trabalhos relacionados. O capítulo 4 engloba a proposta do trabalho a ser realizado. No capítulo 4 encontra-se a metodologia e a proposta utilizada para a realização do trabalho. O capítulo 6 aborda os testes e resultados obtidos. Por fim, o capítulo 7 apresenta a conclusão parcial.

---

## Conceitos

---

A Internet se popularizou de tal maneira que tornou-se essencial para a nossa sociedade, com o grande crescimento da Internet, cresce também a preocupação com as informações que trafegam nas redes de computadores. De acordo com Nunes (2012), ataques com o objetivo de prejudicar o funcionamento de redes de computadores e sistemas de informação têm crescido tanto em número quanto em impacto. Especialistas têm estudado cada vez mais os meios para proteger as informações e sistemas contidos nas redes de computadores. A proteção de computadores e todos os seus componentes é chamada de cibersegurança. Porém, de acordo com Fischer (2014) é difícil encontrar uma definição exata para cibersegurança, ela geralmente se refere a um ou mais dos seguintes itens:

1. Um conjunto de atividades e medidas que visam proteger (de ataques, interrupções, ou outras ameaças) computadores, redes de computadores, ou qualquer software ou hardware relacionado;
2. A proteção resultante mediante ao uso de tais medidas;
3. Um amplo campo de atuação, focado na pesquisa e implementação das atividades e medidas de proteção citadas no item 1.

Portanto, para o melhor entendimento do conceito de cibersegurança, é necessário saber quais e o que são as vulnerabilidades e ameaças que podem por em risco a segurança dos dispositivos de redes. Deste modo, este capítulo apresentará brevemente as definições de vulnerabilidades e ameaças. Além disso, será discutido sobre as ferramentas utilizadas e tecnologias que embasam este trabalho.

### 2.1. Ameaças

Para Bishop (2005) ameaça é tudo o que pode potencialmente violar a segurança de sistemas, ou seja, tudo aquilo que de maneira intencional, ou não, pode disparar vulnerabilidades e

causar impactos no sistema afetado. Segundo Stoneburner et al. (2002) as ameaças podem ser classificadas como:

- Ameaças Naturais: Causados por fenômenos naturais, ocorrem sem a intervenção humana, como por exemplo, enchentes, terremotos, tempestades, incêndios naturais, entre outras;
- Ameaças Humanas: Causados por seres humanos, podemos dividir estas ameaças em dois tipos, voluntárias e involuntárias. Ameaças voluntárias ocorrem quando existe a intenção de executar uma ação maliciosa, como por exemplo, *hackers* que procuram roubar informações. Ameaças involuntárias ocorrem quando, não há intenção alguma de causar alguma ação maliciosa;
- Ameaças Ambientais: Riscos que ocorrem devido ao ambiente em que os sistemas se encontram, como por exemplo, picos de energia, poluição, vazamento de líquidos, entre outros.

As ameaças não apresentam riscos caso não existam vulnerabilidades que possam ser disparadas (ameaças naturais, ameaças ambientais ou ameaças humanas involuntárias) ou exploradas (ameaças humanas voluntárias). Por exemplo, a ameaça ambiental de queda de energia não apresenta nenhum risco para um computador que esteja ligado em um *nobreak*. Portanto, para diminuir os riscos que as ameaças representam deve-se diminuir as vulnerabilidades existentes em redes ou sistemas de computadores. Deste modo é necessário entender o que são vulnerabilidades.

## 2.2. Vulnerabilidades

De acordo com Stoneburner et al. (2002) vulnerabilidades são fraquezas ou falhas de segurança no projeto, implementação, operação ou gerenciamento de sistemas, podem ser acidentalmente desencadeadas (ameaças naturais, ameaças humanas, ameaças ambientais) ou intencionalmente exploradas (ameaças humanas), resultando em brechas ou violações na segurança de sistemas. Algumas vulnerabilidades, quando exploradas, permitem que usuários não autorizados obtenham o controle de sistemas, podendo assim realizar várias ações maliciosas, como por exemplo, realizar novos ataques, além de obter acesso a informações confidenciais que podem gerar grandes prejuízos. (NAKAMURA; GEUS, 2007).

Este trabalho tem foco nas vulnerabilidades de software, como por exemplo, programas desatualizados, senhas de autenticação fracas, entre outras. Tais vulnerabilidades podem ser descobertas com a ajuda de algumas ferramentas, chamadas de *scanners*, ferramentas que podem ajudar tanto o atacante quanto o defensor, os atacantes utilizam os *scanners* para localizar e então explorar as vulnerabilidades, enquanto os defensores utilizam os *scanners* para localizar e corrigir as vulnerabilidades encontradas (FISCHER, 2014).

## 2.3. Ferramentas de análise de vulnerabilidades

Segundo Weidman (2014), a maioria das empresas com orçamentos consideráveis aplicados para a segurança de sistemas, têm suas informações confidenciais roubadas por ataques que exploram vulnerabilidades já conhecidas, ou seja, os atacantes não utilizam vulnerabilidades recentemente descobertas (*zero day*), mas sim vulnerabilidades das quais a existência já eram dadas há algum tempo, sendo que para maioria, as correções já estavam disponíveis. Testes de intrusão (*pentest*) são utilizados para encontrar tais vulnerabilidades. O *Penetration testing* (*pentest*) é um processo que tem como objetivo identificar as vulnerabilidades existentes nas redes, dispositivos e aplicativos.

Para Epling et al. (2015) um *pentest*, é quando uma empresa contrata profissionais (programadores, *hackers*, ou qualquer outra pessoa com um conhecimento elevado na área de segurança da informação) na área de segurança da computação para avaliar e explorar sua própria rede, servidores e serviços, antes que pessoas com intenções maliciosas façam o mesmo. Ataques reais são realizados para respectivamente, descobrir, explorar vulnerabilidades encontradas. O *pentest* é essencial para qualquer ambiente corporativo. Porém, os preços dos testes de intrusão podem aumentar de maneira significativa dependendo da complexidade e tamanho da infraestrutura de rede a ser avaliada.

De acordo com Allen et al. (2014) existem dois principais tipos de *pentest*, são eles, testes de caixa preta (*black box testing*) e os testes de caixa branca (*white box testing*). No teste de caixa preta, o profissional de segurança não tem nenhum conhecimento a respeito do ambiente alvo, usando apenas suas habilidades, conhecimentos e ferramentas para encontrar e explorar as vulnerabilidades existentes. Já no teste de caixa branca, o profissional de segurança tem conhecimento de todas as tecnologias utilizadas no ambiente alvo, desde topologia da rede, equipamentos utilizados, sistemas operacionais, entre outras.

Segundo Weidman (2014) os pentests possuem as seguintes etapas:

- Pré-compromisso: Antes que os testes comecem, os profissionais de segurança e os clientes que contrataram o *pentest* se reúnem e conversam sobre o que deve ser testado, como deve ser testado, entre outras especificações;
- Levantamento de informações: Os profissionais responsáveis pelo *pentest* devem reunir a maior quantidade de informações possíveis sobre os clientes, por exemplo, sistemas utilizados, softwares que estão executando, entre outras;
- Modelagem de ameaças: Baseando-se na informações obtidas no levantamento de informações, os profissionais tentam enxergar quais as possíveis vulnerabilidades existentes nos sistemas utilizados pelos clientes, além de estratégias para explorá-las;
- Análise de vulnerabilidades: Vulnerabilidades começam a ser investigadas, *scanners* são utilizados para auxiliar os profissionais a descobrirem a maior quantidade possível de vulnerabilidade nos sistemas analisados;

- Exploração: As vulnerabilidades descobertas começam a ser exploradas, os profissionais de segurança tentam de todo modo invadir os sistemas dos clientes;
- Pós-exploração: Nesta etapa os profissionais de segurança identificam quais informações podem ser obtidas dos sistemas analisados a partir da exploração das vulnerabilidades;
- Relatórios: Relatório contendo tudo o que foi descoberto pelos profissionais são entregues aos clientes. Um relatório completo é elaborado, comentando sobre todas as vulnerabilidades encontradas e possíveis prejuízos resultantes da exploração de tais vulnerabilidades.

Para realizar a etapa de análise de vulnerabilidades, geralmente, *scanners* de vulnerabilidades são utilizados. Segundo Ulbrich e Valle (2003), *scanners* são programas utilizados para varrer dispositivos em redes de computadores à procura de vulnerabilidades, de acordo com Weidman (2014), os *scanners* usam bases de dados atualizadas e a partir de vários testes identificam as vulnerabilidades existentes. Além disso, os *scanners* também podem classificar as vulnerabilidades de acordo com os riscos que elas representam ao sistema (*low, medium, high*).

Em seguida são explicadas as ferramentas estudadas para a realização desse trabalho: Nessus e OpenVAS. E também é explicado o motivo da escolha da ferramenta OpenVAS como ferramenta final a ser utilizada.

### 2.3.1. Nessus

Nessus é um dos melhores e mais bem mantidos *scanner* de vulnerabilidades existentes hoje. Originalmente era uma ferramenta de código aberto, lançado sobre a licença *General Public Licence* (GPL), porém a partir da versão 3.0 a Tenable Network Security decidiu fechar seu código para uso comercial. De acordo com Im et al. (2016), Nessus contém todas as funções básicas das ferramentas de varredura de rede. Além de oferecer contramedidas de proteção adequadas para descobrir e analisar potenciais vulnerabilidades. O Nessus funciona em uma arquitetura cliente-servidor, onde o servidor é responsável pelo processamento das varreduras e detecção das vulnerabilidades e o cliente é responsável pela interface web, onde os usuários podem analisar tais vulnerabilidades.

O Nessus realiza uma varredura de portas, e é capaz de identificar falhas e vulnerabilidades em serviços mesmo que não estejam executando em suas portas padrões. Além disso, o Nessus funciona de maneira inteligente, testes para um programa específico só serão executados caso esse programa seja encontrado nos sistemas alvo, por exemplo, testes para aplicações web só serão executados caso uma aplicação web seja encontrada (ANDERSON, 2003).

Para encontrar e categorizar vulnerabilidades o Nessus executa vários programas menores, esses programas são chamados de *plugins*, os *plugins* são escritos em uma linguagem própria, chamada de *Nessus Attack Scripting Language* (NASL), os *plugins* possuem todas

as informações sobre determinadas vulnerabilidade, um conjunto genérico de correção e um algoritmo para testar a presença da vulnerabilidade na rede (TENABLE, 2017).

A Tenable Network Security disponibiliza três códigos de ativação para o Nessus, sendo um deles sem custo, porém com algumas limitações, são eles:

- *Nessus Home*: É a versão sem custo da ferramenta de análise de vulnerabilidades, permite realizar a varredura de até 16 IPs consecutivos, com a mesma velocidade e profundidade de um assinante Nessus. Destinados a usuários domésticos.
- *Nessus Professional*: Capaz de realizar varreduras em ilimitados IPs, além de possuir diferentes tipos de varreduras de rede.
- *Nessus Manager*: Combina todas as funcionalidades do Nessus com a possibilidade de compartilhar os recursos de varredura com outros membros do sistema, de maneira colaborativa.

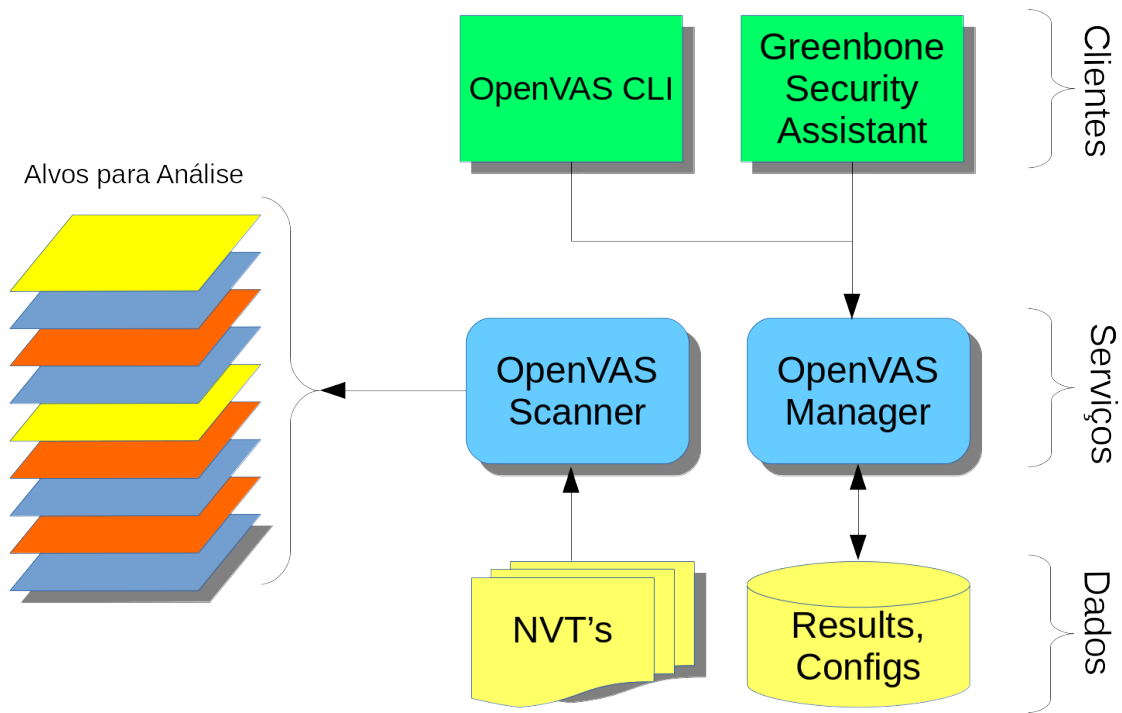
O Nessus é capaz de gerar relatórios com todas as informações sobre as vulnerabilidades encontradas e os passos necessários para corrigir essa vulnerabilidade, os relatórios estão disponíveis em vários formatos, como *HyperText Markup Language* (HTML), *eXtensible Markup Language* (XML), LaTeX ou texto simples. Além disso, esses relatórios podem ser organizados de diversas maneiras, como por exemplo, vulnerabilidades agrupadas por IP (lista os IPs encontrados durante a varredura e as vulnerabilidades associadas a esse IP), vulnerabilidades agrupadas por *plugin* (lista todas as vulnerabilidades encontradas durante a varredura), entre outras.

### 2.3.2. OpenVAS

OpenVAS é uma ferramenta de código aberto para a análise de vulnerabilidades, mais precisamente é um arcabouço de vários serviços e ferramentas que oferecem uma poderosa solução para *scanning* e gerenciamento de vulnerabilidades (OPENVAS, 2014). O OpenVAS foi criado a partir de variações dos códigos do Nessus, depois que a empresa Tenable Network Security fechou o código para uso comercial.

O OpenVAS possui uma arquitetura baseada em cliente-servidor, o servidor é encarregado do processamento e armazenamento das varreduras e configurações realizadas. Enquanto o cliente fornece uma interface web, na qual o administrador de rede é capaz de configurar varreduras e visualizar os relatórios gerados. A Figura 2.1 mostra a arquitetura do OpenVAS.





**Figura 2.1.** Arquitetura do OpenVAS (OPENVAS, 2014).

De acordo com Kim et al. (2016), a seguir serão descritos os componentes que formam a arquitetura do OpenVAS:

- *Greenbone Security Assistant* (GSA): Fornece aos usuários uma interface web, na qual os mesmos podem gerenciar as configurações, criar varreduras e visualizar os relatórios de varreduras já executadas.
- *OpenVAS CLI*: Cliente OpenVAS, responsável por auxiliar o usuário através da interface de linha de comando, permitindo que usuários realizem as mesmas funções providas pelo GSA, sem a necessidade de acessar uma interface gráfica.
- *OpenVAS Manager*: Serviço principal do OpenVAS, controla o *scanner* por um protocolo chamado de *OpenVAS Transfer Protocol* (OTP), além de ser responsável por armazenar a configuração e os resultados das varreduras. Também oferece funções adicionais, como por exemplo, agendamento de varreduras, geração de relatórios entre outras, com uma ferramenta baseada em XML, chamada de *OpenVAS Management Protocol* (OMP).
- *OpenVAS Scanner*: Núcleo da arquitetura do OpenVAS, executa vários testes chamados de *Network Vulnerability Tests* (NVT), esses testes verificam a presença de vulnerabilidades em sistemas. Os NVTs são desenvolvidos utilizando *scripts* da linguagem NASL, e assim como o Nessus o OpenVAS também possibilita a criação de seus próprios *plugins* (NVT) para a verificação de vulnerabilidades.

Como já dito, o OpenVAS é um *framework* composto de vários serviços e ferramentas, segundo Allen et al. (2014), as ferramentas que compõem o OpenVAS são mostradas na tabela 2.1:

**Tabela 2.1.** Ferramentas utilizadas pelo OpenVAS

Ferramenta	Descrição
Amap	Ferramenta para detecção de protocolo de aplicações
Ike-scan	<i>Scanner</i> para detecção e testes de sistemas IPSec e VPN
Ldapsearch	Extraí informações dos dicionários LDAP
Nikto	Realiza análise de vulnerabilidades em servidores web
Nmap	Realiza uma varredura das portas de um sistema
Ovaldi	Realiza análise de vulnerabilidades em um sistema
pncan	Realiza uma varredura das portas de um sistema
Portbunny	Realiza uma varredura das portas de um sistema
Seccubus	Automatiza as varreduras realizadas pelo OpenVAS
SLAD	Várias ferramentas de segurança (John-the-Ripper, Chkrootkit, ClamAV, Snort, Logwa
Snmpwalk	Extraí data dos protocolos SNMP
Strobe	Realiza uma varredura das portas de um sistema
w3af	Realiza ataques em aplicações web

O OpenVAS é uma ferramenta completa, podendo ser utilizada para analisar qualquer tipo de rede. Capaz de realizar desde simples varreduras de portas, utilizando ferramentas como Nmap, até quebras de senhas fracas, utilizando john-the-Ripper. Unindo tais características com o fato de ser uma ferramenta de código aberto e possuir uma comunidade forte e crescente, o OpenVAS se destaca cada vez mais quando comparados com outros *scanners* do gênero.

## 2.4. ElasticSearch

## 2.5. Kibana

---

## Trabalhos Relacionados

---

Em seu trabalho Altaf et al. (2015) foca na identificação e remoção de vulnerabilidades web, o artigo fala principalmente de avaliação de vulnerabilidades (*vulnerability assessment*), que é o processo de identificação, quantificação e classificação de vulnerabilidades. Primeiramente é explicado o processo de uma avaliação de vulnerabilidades, que é basicamente encontrar o sistema alvo e extrair informações. Nesse procedimento são realizados vários tipos de testes de penetração, como por exemplo, teste de caixa preta (*black box testing*) e teste de caixa branca (*white box testing*).

Durante esses testes, profissionais de segurança ou *hackers*, tentam encontrar qualquer vulnerabilidade para depois explorá-la, e então ganhar acesso ao sistema. Altaf et al. (2015) cita os principais motivos para a realização de um teste de penetração, são eles:

1. Para identificar os meios que um atacante pode obter acesso ao sistema.
2. Para saber qual é a maior ameaça do sistema, e corrigi-la assim que possível.
3. Para identificar as vulnerabilidades que sistemas automatizados não conseguem identificar.
4. Identificar os riscos comerciais existentes. Uma empresa perderá a fé de seus clientes caso seu site seja hackeado, ou algo do tipo.
5. Verificar se o sistema responde bem aos ataques comuns.
6. Mostrar que qualquer tipo de ataque pode ser realizado em sistemas vulneráveis, e assim convencer as organizações a investir mais em segurança.

Também é comentado sobre dois tipos de testes de penetração, os testes de penetração automáticos, onde softwares procuram por vulnerabilidades em aplicações web e em todas as paginas relacionadas a essa aplicação, no final do teste é gerado um relatório contendo as vulnerabilidades e os métodos utilizados para resolver tais vulnerabilidades. E os testes de penetração manuais, aonde um profissional de segurança utiliza técnicas do mundo real, as mesmas utilizadas por *hackers*, para explorar e ganhar acesso ao sistema, o profissional em

segurança utiliza seu conhecimento para encontrar, explorar e concertar as vulnerabilidades encontradas durante o processo de teste.

Também é discutido os principais tipos de vulnerabilidades baseadas em injeção de SQL (*SQL injection*), um tipo de vulnerabilidade onde o atacante consegue "injetar" *Structured Query Language* (SQL), em uma base de dados e conseguir informações restritas. Em seu trabalho Altaf et al. (2015) introduz uma metodologia capaz de identificar declarações em aplicações *Personal Home Page* (PHP), que podem estar vulneráveis para a injeção de SQL.

Em seu artigo Lukanta et al. (2014) comenta sobre as vulnerabilidades de gerenciamento de sessão (*session management vulnerabilities*) existentes em aplicações web. De modo resumido, vulnerabilidades de gerenciamento de sessão são muitas vezes vulnerabilidades que ainda estão sendo descobertas. O método de gerenciamento de sessão mais comum utiliza um identificador de sessão (*session identifier*), esse *Session Identifier* (SID) é um par "nome=valor". O valor é um numero correspondente a uma sessão na web, o SID deve ser enviado em cada requisição feita. Em aplicações web, o SID geralmente é enviado em um campo oculto ou em um *cookie* HTTP. Na maioria das vezes o gerenciamento de sessão é implementado incorretamente, o que acaba gerando as seguintes vulnerabilidades:

- Correção de sessão (*Session Fixation*): É uma vulnerabilidade que ocorre quando o atacante visita uma pagina web e recebe um SID, após isso, o invasor manda uma *Uniform Resource Locator* (URL) contendo o SID para a vitima, quando a vitima visita a URL e se autentica o SID será o mesmo que o do atacante.
- *Cross-Site Request Forgery* (CSRF): Um invasor manda uma URL manipulada para a vitima, quando visitado o URL faz uma requisição para o servidor web, sem nenhum reconhecimento da vitima.
- Insuficiente atributos *Cookies*: O *cookie* geralmente serve como um container para o SID. Portanto o desenvolvedor deve tomar um cuidado especial quando estiver lidando com esses atributos, caso contrario, o atacante será capaz de roubar o *cookie* contendo o SID.

Para lidar para essas vulnerabilidades, Lukanta et al. (2014) propõe uma solução de duas partes, a primeira é uma extensão para navegadores web, e a segunda parte é um plugin desenvolvido para a ferramenta de análise de vulnerabilidades web Nikto. A primeira parte (extensão para navegadores web) serve como um identificador de vulnerabilidades único. E a segunda parte (Nikto) é usada para suportar o teste continuo, repetindo todo o processo feito na primeira parte automaticamente.

Gawron et al. (2015) comenta sobre os problemas para a detecção e representação de vulnerabilidades em sistemas informáticos. Em seu trabalho ele comenta sobre o grande aumento da complexidade das redes, sistemas únicos e componentes de hardware e software. Tal complexidade é tão alta que se torna praticamente impossível de gerenciar os riscos de segurança manualmente. Gawron comenta também sobre outro desafio, que é a falta de

um banco de dados com análises suficientes para automatizar o processo de detecção de vulnerabilidades, isso faz com que a investigação manual de uma vulnerabilidade específica necessite de várias pesquisas em diferentes fontes. Sendo assim Gawron et al. (2015) propõe a criação de uma ferramenta capaz de detectar vulnerabilidades automaticamente. Além de descrever duas abordagens que ajudam o usuário a proteger seu sistema.

---

# Proposta

---

A sequencia deste capitulo apresenta a metodologia utilizada para implementar a ferramenta,

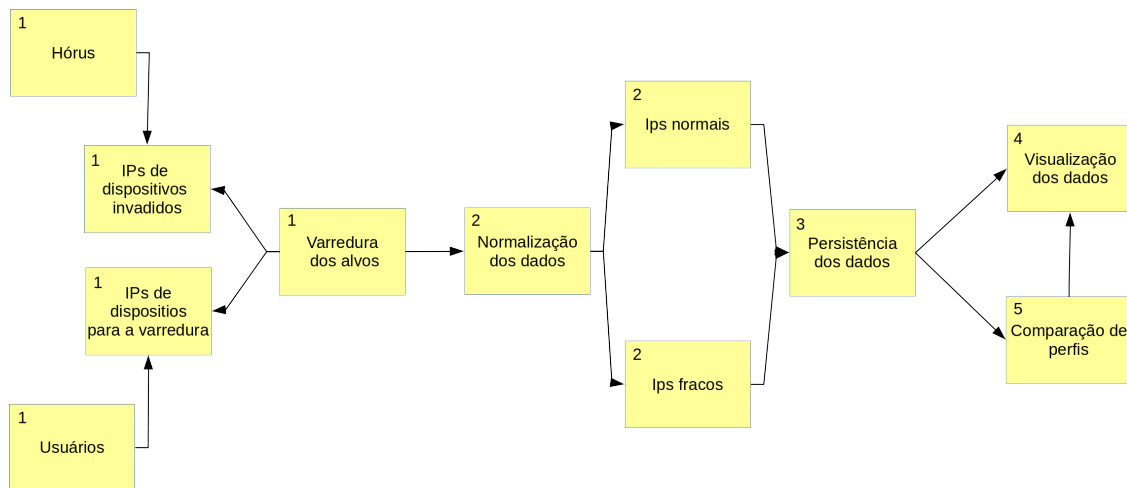
## 4.1. Metodologia

A complexidade das redes de computadores aumentaram consideravelmente nos últimos anos. De acordo com Gawron et al. (2015) tal complexidade é tão grande que se tornou praticamente impossível gerenciar os riscos de segurança manualmente. Deste modo, ferramentas que auxiliam os administradores de redes na realização de tal gerenciamento tornaram-se necessárias, como por exemplo os *scanners* de vulnerabilidade. Porém, os *scanners* tendem a gerar quantidades significativas de informações, demandando um grande tempo (Ainda nao encontrei referencia!!) do administrador de rede para analisar tais informações.

Deste modo, a ferramenta proposta deve auxiliar tanto na detecção das vulnerabilidades quanto na visualização dos dados retornados. A figura 4.1 mostra as etapas necessárias para implementar a ferramenta:

1. Utilizando *scanners* de vulnerabilidades os alvos serão analisados;
2. Normalização dos dados retornados, selecionando apenas as principais informações;
3. Persistência dos dados normalizados em um banco de dados;
4. Visualização simplificada das informações;
5. Comparação de perfis, utilizando as informações persistidas.

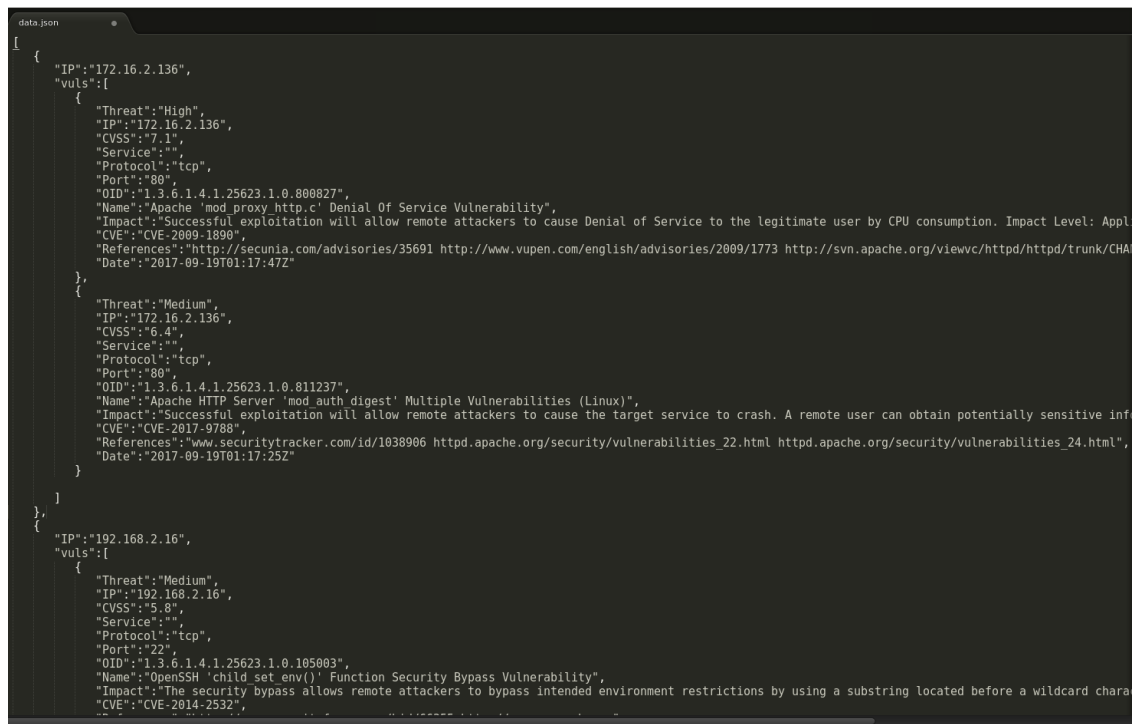
Tais etapas serão explicadas detalhadamente a seguir.



**Figura 4.1.** Etapas necessárias para realização do projeto.

Na etapa de varredura, *scanners* de vulnerabilidades serão utilizados para analisar os alvos. Os *scanners* devolverão um arquivo XML, contendo todas as vulnerabilidades encontradas durante a varredura, além de informações sobre os dispositivos analisados. Nesta etapa serão analisados tanto os IPs vindo do usuário quanto os IPs cedidos pelo sistema Hórus, porém, o usuário não terá conhecimento que outra varredura está sendo realizada. Para testar qual *scanner* será utilizado na implementação da ferramenta, foi utilizado um conjunto de teste, contendo 100 IPs desconhecidos, espalhados pela Internet. Tal conjunto foi analisado por dois *scanners* de vulnerabilidades diferentes, *Open Vulnerability Assessment System* (OpenVAS) e Nessus. Os detalhes dos testes e resultados se encontram no Capítulo XX.

Na normalização, o arquivo XML é filtrado, visando diminuir a quantidade e a complexidade dos dados retornados pelos *scanners*, tal filtragem é realizada utilizando um *script*, implementado em python, que percorre as *tags* XML e coleta as informações contidas nas mesmas. Apenas as principais informações serão mantidas, as que forem consideradas desnecessárias serão excluídas. O *script* retorna as informações filtradas no formato JSON, tal formato foi escolhido devido a sua simplicidade e a facilidade de integração com o banco de dados não-relacional Elasticsearch, que armazena os dados no formato JSON. A Figura 4.2 apresenta o arquivo obtido após a filtragem das informações.



```

{
  "IP": "172.16.2.136",
  "vuls": [
    {
      "Threat": "High",
      "IP": "172.16.2.136",
      "CVSS": "7.1",
      "Service": "",
      "Protocol": "tcp",
      "Port": "80",
      "OID": "1.3.6.1.4.1.25623.1.0.800827",
      "Name": "Apache 'mod proxy http.c' Denial Of Service Vulnerability",
      "Impact": "Successful exploitation will allow remote attackers to cause Denial of Service to the legitimate user by CPU consumption. Impact Level: Appl",
      "CVE": "CVE-2009-1890",
      "References": "http://secunia.com/advisories/35691 http://www.vupen.com/english/advisories/2009/1773 http://svn.apache.org/viewvc/httpd/httpd/trunk/CHA",
      "Date": "2017-09-19T01:17:47Z"
    },
    {
      "Threat": "Medium",
      "IP": "172.16.2.136",
      "CVSS": "6.4",
      "Service": "",
      "Protocol": "tcp",
      "Port": "80",
      "OID": "1.3.6.1.4.1.25623.1.0.811237",
      "Name": "Apache HTTP Server 'mod auth digest' Multiple Vulnerabilities (Linux)",
      "Impact": "Successful exploitation will allow remote attackers to cause the target service to crash. A remote user can obtain potentially sensitive inf",
      "CVE": "CVE-2017-9788",
      "References": "www.securitytracker.com/id/1038906 httpd.apache.org/security/vulnerabilities_22.html httpd.apache.org/security/vulnerabilities_24.html",
      "Date": "2017-09-19T01:17:25Z"
    }
  ]
},
{
  "IP": "192.168.2.16",
  "vuls": [
    {
      "Threat": "Medium",
      "IP": "192.168.2.16",
      "CVSS": "5.8",
      "Service": "",
      "Protocol": "tcp",
      "Port": "22",
      "OID": "1.3.6.1.4.1.25623.1.0.105003",
      "Name": "OpenSSH 'child_set_env()' Function Security Bypass Vulnerability",
      "Impact": "The security bypass allows remote attackers to bypass intended environment restrictions by using a substring located before a wildcard chara",
      "CVE": "CVE-2014-2532",
      "References": ""
    }
  ]
}

```

**Figura 4.2.** Arquivo JSON gerado a partir das informações .

O JSON que será persistido no banco de dados contém um vetor de objetos com os campos “IP” e “vuls”. O campo “IP” contém o IP do dispositivo que foi analisado. O campo “Vuls” é um vetor com todas as vulnerabilidades encontradas para o determinado IP, contendo os seguintes campos:

- *Threat* (ameaça): O nível de ameaça que tal vulnerabilidade representa para o sistema em qual foi detectada, os *scanners* testados no presente trabalho possuem 4 níveis de ameaças, *high* (alto), *medium* (médio), *low* (baixo) e *logs* (informações), esses níveis são determinados utilizando a pontuação obtida durante o cálculo do *Common Vulnerability Scoring System* (CVSS). Os *logs* não são considerados ameaças, são informações que o *scanner* conseguiu adquirir do sistema alvo (Sistema operacional, versões de aplicativos instalados, entre outras);
- CVSS: Uma métrica padrão para calcular o nível de gravidade das vulnerabilidades. O cálculo é feito levando em consideração a facilidade para explorar a vulnerabilidade e o impacto da exploração. Após o cálculo, uma pontuação é atribuída a tal vulnerabilidade, podendo ir de 0 até 10, onde 10 é considerado gravidade crítica.(reF);
- *Service* (serviço): Não sei ainda
- *Protocol* (protocolo): Protocolo de comunicação de rede que estava sendo utilizado quando a vulnerabilidade foi detectada;
- *Port* (porta): Porta em qual a vulnerabilidade foi detectada;
- *OID*: Identificador único de vulnerabilidades dentro da base de dados do OpenVAS;
- *Name* (nome): Nome da vulnerabilidade;



- *Impact* (impacto): Impacto para o sistema caso ocorra a exploração da vulnerabilidade;
- *Common Vulnerabilities and Exposures* (CVE): Identificador único para a vulnerabilidade;
- *References* (referências): Informações a respeito das vulnerabilidades;
- *Date* (data): A data exata contendo dia, mês, ano, horas, minutos e segundos na qual a vulnerabilidade foi detectada.

Como dito anteriormente, um perfil será criado para cada dispositivo analisado, contendo o IP do dispositivo e as vulnerabilidades identificadas durante sua análise. Formatando os dados da maneira mostrada na Figura 4.2, os perfis acabam sendo criados implicitamente. Portanto, os dados que serão persistidos no banco são exatamente os perfis dos dispositivos.

O banco de dados não-relacional Elasticsearch foi escolhido para a persistência dos arquivos JSON, principalmente por ser um software de código aberto, e também por apresentar integração com o *plugin* Kibana, que será utilizado para visualização dos dados. Serão persistidos dois grupos de dados diferentes. O primeiro representa os dados obtidos na varredura dos IPs que foram analisados pelo cliente, estes serão chamados de “IPs normais”. O segundo grupo de dados representa os dados gerados a partir da varredura dos IPs cedidos pelo sistema Hórus, este grupo será chamado de “IPs fracos”.

Como dito anteriormente, Kibana é um *plugin* para Elasticsearch, o *plugin* utiliza os dados existentes no banco para gerar gráficos dos mais variados estilos. No presente trabalho, o Kibana será utilizado para facilitar a visualização das vulnerabilidades encontradas. Através de uma interface web, o usuário poderá visualizar os gráficos gerados, facilitando na administração da rede que está sendo analisada.

Para comparar os perfis, será utilizado uma consulta do próprio banco de dados. Será selecionado todos os IPs do grupo “IPs normais” que possuir, uma ou mais, vulnerabilidades em comum com qualquer IP do grupo “IPs fracos”. Caso a consulta retorne algum IP, o mesmo será mostrado de maneira chamativa no Kibana, juntamente com sua vulnerabilidade.

## 4.2. Cronograma de Atividades

Nesta seção são apresentadas as atividades a serem desenvolvidas para a execução da proposta. O cronograma de realização das tarefas é apresentado na Tabela 4.1.

1. Estudo da ferramenta Elasticsearch.
2. Persistência dos dados no banco.
3. Implementação da Ferramenta Sem nome.
4. Estudo da ferramenta Kibana.
5. Realização dos experimentos utilizando a ferramenta sem nome.
6. Teste e análise dos resultados obtidos.
7. Escrita do TCC2

8. Entrega do TCC 2.

9. Apresentação do TCC 2.

**Tabela 4.1.** Cronograma das Atividades

Atividade	2018					
	Jan	Fev	Mar	Abr	Mai	Jun
<b>1</b>	X					
<b>2</b>	X					
<b>3</b>	X	X	X	X	X	
<b>4</b>		X				
<b>5</b>			X	X		
<b>6</b>				X	X	
<b>7</b>	X	X	X	X	X	X
<b>8</b>						X
<b>9</b>						X

---

## Experimentos e Resultados

---

Texto de ligação/introdução do capítulo...

(ATENÇÃO - veja com o seu orientador se você vai ter este capítulo e se este vai ter nome!)

### 5.1. Experimentos

Descreva os experimentos realizados...

(ATENÇÃO - Essa seção é uma sugestão, veja com o seu orientador se você vai ter essa e se vai ter esse nome!)

TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO

## 5.2. Resultados

Aqui você pode descrever os resultados obtidos nos experimentos e/ou analisar/discutir tais resultados.

(ATENÇÃO - Essa seção é uma sugestão, veja com o seu orientador se você vai ter essa e se vai ter esse nome!)

TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
 TEXTO TEXTO TEXTO

## 5.3. Considerações Finais

Esta é uma sugestão de seção para dar um fechamento em cada uma dos capítulos.

(ATENÇÃO - veja com o seu orientador se é uma seção necessária (pois trate-se de estilo de escrita))

---

## Conclusões

---

Texto de ligação/introdução da conclusão...

### 6.1. Considerações finais ou parciais

Descreva as conclusões parciais (TCC1) e finais (TCC2) do seu trabalho.

(ATENÇÃO - Essa seção é uma sugestão, veja com o seu orientador se você vai ter essa e se vai ter esse nome!)

TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO  
TEXTO TEXTO TEXTO

### 6.2. Sugestões para Trabalhos Futuros

Descreva como é possível continuar esse trabalho, ou suas ideias depois desse trabalho.

[illegible]

# Apêndices

---

## Instalação de Ferramentas

---

Os apêndices são usados para disponibilizar materiais extras que por questões de espaço ou estilo de escrita não foram colocados diretamente no texto. Por exemplo, *scripts*, instruções de instalação das ferramentas utilizadas pelo trabalho, partes de código fonte e questionários que tenham sido aplicados, tabelas com resultados...

(ATENÇÃO - veja com o seu orientador se é necessário disponibilizar algum material extra sobre algum capítulo em anexo!)



# Referências

---

- ALLEN, L.; HERIYANTO, T.; ALI, S. *Kali Linux – Assuring Security by Penetration Testing*. Packt Publishing, 2014. (Community experience distilled). ISBN 9781849519496. Disponível em: <<https://books.google.com.br/books?id=QcBGAAwAAQBAJ>>.
- ALTAF, I.; RASHID, F. u.; DAR, J. A.; RAFIQ, M. Vulnerability assessment and patching management. In: *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*. [S.l.: s.n.], 2015. p. 16–21.
- ANDERSON, Harry. *Introduction to nessus*. 2003.
- BISHOP, M. *Introduction to Computer Security*. Addison-Wesley, 2005. ISBN 9780321247445. Disponível em: <<https://books.google.com.br/books?id=Z-lQAAAAMAAJ>>.
- EPLING, Lee; HINKEL, Brandon; HU, Yi. Penetration testing in a box. In: *Proceedings of the 2015 Information Security Curriculum Development Conference*. New York, NY, USA: ACM, 2015. (InfoSec '15), p. 6:1–6:4. ISBN 978-1-4503-4049-6. Disponível em: <<http://doi.acm.org/10.1145/2885990.2885996>>.
- FISCHER, Eric A. *Cybersecurity Issues and challenges: in brief*. [S.l.]: Congressional Research Service, 2014.
- GAWRON, Marian; AMIRKHANYAN, Aragats; CHENG, Feng; MEINEL, Christoph. Automatic vulnerability detection for weakness visualization and advisory creation. In: *Proceedings of the 8th International Conference on Security of Information and Networks*. New York, NY, USA: ACM, 2015. (SIN '15), p. 229–236. ISBN 978-1-4503-3453-2. Disponível em: <<http://doi.acm.org/10.1145/2799979.2799986>>.
- IM, Sun young; SHIN, S. H.; RYU, Ki Yeol; ROH, Byeong hee. Performance evaluation of network scanning tools with operation of firewall. In: *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. [S.l.: s.n.], 2016. p. 876–881.
- KIM, S. S.; LEE, D. E.; HONG, C. S. Vulnerability detection mechanism based on open api for multi-user's convenience. In: *2016 International Conference on Information Networking (ICOIN)*. [S.l.: s.n.], 2016. p. 458–462.
- LUKANTA, R.; ASNAR, Y.; KISTIJANTORO, A. I. A vulnerability scanning tool for session management vulnerabilities. In: *2014 International Conference on Data and Software Engineering (ICODSE)*. [S.l.: s.n.], 2014. p. 1–6.
- NAKAMURA, Emilio Tissato; GEUS, Paulo Lício de. *Segurança de redes em ambientes cooperativos*. [S.l.]: Novatec Editora, 2007.

NUNES, Paulo Viegas. A definição de uma estratégia nacional de cibersegurança. *Nação e Defesa-Revista Quadrimestral* n. <sup>o</sup>, v. 133, p. 113–127, 2012.

OPENVAS. *OpenVAS Open Vulnerability Assessment System*. 2014. <<http://www.openvas.org/about.html>>. Acessado em 29/10/2017.

STONEBURNER, Gary.; GOGUEN, Alice.; FERINGA, Alexis.; STANDARDS, National Institute of; (U.S.), Technology. Book, Online. *Risk management guide for information technology systems [electronic resource] : recommendations of the National Institute of Standards and Technology / Gary Stoneburner, Alice Goguen, and Alexis Feringa*. [S.l.]: U.S. Dept. of Commerce, National Institute of Standards and Technology Gaithersburg, Md, 2002.

TENABLE. *Tenable Network Security*. 2017. <<https://www.tenable.com/products/nessus-vulnerability-scanner>>. Acessado em 29/10/2017.

ULBRICH, HC; VALLE, J Della. *Universidade Hacker-Desvende todos os segredos dos submundos dos hackers*. 2<sup>a</sup>. [S.l.]: Digerati, São Paulo, Brazil, 2003.

WEIDMAN, G. *Penetration Testing: A Hands-On Introduction to Hacking*. No Starch Press, 2014. ISBN 9781593275648. Disponível em: <[https://books.google.com.br/books?id=T\\\_\\\_LlAwAAQBAJ](https://books.google.com.br/books?id=T\_\_LlAwAAQBAJ)>.