UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA

COORDENAÇÃO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RAIFF DOS SANTOS SILVA

AUTOMAÇÃO DE CONTROLES DE SEGURANÇA EM AMBIENTES DE COMPUTAÇÃO NA NUVEM: ABORDAGEM PARA CONFORMIDADE CONTÍNUA NO GERENCIAMENTO DE VULNERABILIDADES

CAMPINA GRANDE
2024

RAIFF DOS SANTOS SILVA

AUTOMAÇÃO DE CONTROLES DE SEGURANÇA EM AMBIENTES DE COMPUTAÇÃO NA NUVEM: ABORDAGEM PARA CONFORMIDADE CONTÍNUA NO GERENCIAMENTO DE VULNERABILIDADES

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Andrey Brito

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Computação na nuvem

CAMPINA GRANDE, PARAÍBA

2024.

S586a Silva, Raiff dos Santos.
Automação de controles de segurança em ambientes de computação na nuvem: abordagem para conformidade contínua no gerenciamento de vulnerabilidades / Raiff dos Santos Silva. – Campina Grande, 2024.
175 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2024.
"Orientação: Prof. Dr Andrey Elísio Monteiro Brito".
Referências.

1. Segurança – Automação de Controles. 2. Computação na Nuvem. 3. Gerenciamento de Vulnerabilidade. 4. IaaS. I. Brito, Andrey Elísio Monteiro. II. Título.

CDU 004.056.52(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secpg@computacao.ufcg.edu.br

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

RAIFF DOS SANTOS SILVA

AUTOMATIZAÇÃO DE CONTROLES DE SEGURANÇA EM AMBIENTES DE COMPUTAÇÃO NA NUVEM: ABORDAGEM PARA CONFORMIDADE CONTÍNUA NO GERENCIAMENTO DE VULNERABILIDADES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 04/09/2024

Prof. Dr. ANDREY ELÍSIO MONTEIRO BRITO, UFCG, Orientador

Prof. Dr. REINALDO CÉZAR DE MORAIS GOMES, UFCG, Examinador Interno

Prof. Dr. PAULO DITARSO MACIEL JÚNIOR, IFPB, Examinador Externo

Prof. Dr. EDUARDO DE LUCENA FALCÃO, UFRN, Examinador Externo

Documento assinado eletronicamente por **ANDREY ELISIO MONTEIRO BRITO**, **PROFESSOR 3 GRAU**, em 04/09/2024, às 15:51, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Paulo Ditarso Maciel Júnior**, **Usuário Externo**, em 04/09/2024, às 16:46, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **REINALDO CEZAR DE MORAIS GOMES**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 05/09/2024, às 08:46, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Eduardo de Lucena Falcão**, **Usuário Externo**, em 12/09/2024, às 10:03, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **4757654** e o código CRC **CB9DC007**.

Referência: Processo nº 23096.059012/2024-18 SEI nº 4757654

Resumo

O atual cenário de segurança exige processos eficientes que correspondam ao avanço das técnicas empenhadas em ataques cibernéticos. Partindo desta perspectiva, a automação de processos é vista como alternativa para suprir a necessidade de segurança requerida nos sistemas de informação, beneficiando atividades manuais com otimizações e provendo escalabilidade. Observados os padrões de segurança ou muitas vezes legislações que exigem moldar uma postura de segurança para manutenção da privacidade de dados ou áreas específicas de atuação, a conformidade se vê presente como meio de comprovação das boas práticas implementadas.

A conformidade, muitas vezes é obtida por processos de auditoria de órgãos certificadores, também pode ser alcançada pelo uso de *frameworks* de segurança. Dessa forma, CIS *Controls* V8 e NIST *Cybersecurity Framework* possuem controles de segurança amplamente utilizados. Definimos neste estudo como utilizar estes *frameworks* de segurança como base de recomendações de segurança para prover conformidade para processos de gerenciamento de vulnerabilidades. Utilizamos CIS *Controls* V8 para extrair palavras-chave relevantes com uso de algoritmos de aprendizado de máquina que viabilizam a descoberta de ferramentas de segurança, assim obtendo suporte a nossas implementações técnicas de cibersegurança. Operacionalizamos a conformidade com uso de métodos para mensurar a conformidade usando métricas de verificação de vulnerabilidades e cobertura de recomendações de segurança.

Finalmente, agrupamos todo nosso aprendizado da operacionalização da conformidade para propor um modelo de conformidade contínua. Este modelo considera automação de processos para execução de ferramentas de verificação de vulnerabilidades para coleta de evidências de segurança sobre a arquitetura do sistema. Também são executadas as análises de vulnerabilidades nocivas à segurança com uso de nossa métrica de verificação que define a explorabilidade e o impacto relacionado aos ativos verificados em nossos processos.

Abstract

The current security landscape demands efficient processes that correspond to the advancement of techniques employed in cyber attacks. From this perspective, process automation is seen as an alternative to meet the security needs required in information systems, benefiting manual activities with optimizations and providing scalability. Observing security standards or often legislation that require shaping a security posture for maintaining data privacy or specific areas of operation, compliance is present as a means of proving the good practices implemented.

Compliance, often obtained through audit processes by certifying bodies, can also be achieved through the use of security frameworks. Thus, CIS Controls V8 and the NIST Cybersecurity Framework have widely used security controls. In this study, we define how to use these security frameworks as a basis for security recommendations to provide compliance for vulnerability management processes. We use CIS Controls V8 to extract relevant keywords with the use of machine learning algorithms that enable the discovery of security tools, thus supporting our technical cybersecurity implementations. We operationalize compliance by using methods to measure compliance using vulnerability verification metrics and security recommendation coverage.

Finally, we aggregate all our learning from the operationalization of compliance to propose a continuous compliance model. This model considers process automation for the execution of vulnerability verification tools to collect security evidence about the system architecture. It also includes the analysis of vulnerabilities harmful to security using our verification metric that defines the exploitability and impact related to the assets verified in our processes.

Agradecimentos

Agradeço a Deus por ter saúde e forças para me manter motivado no caminho da busca por conhecimento, sem o qual nada seria possível.

À minha esposa Helena, por acreditar em mim e me motivar a seguir em frente e ser sempre melhor a cada dia. Agradeço também pela paciência que ela teve quando eu precisava estudar e deixava os afazeres de casa em segundo plano.

À minha vó Sebastiana do Socorro que despertou em mim a vontade de ter a UFCG como parte da minha história.

Ao meu orientador e inspiração como pessoa e profissional, Andrey Brito, que me deu todo suporte, confiança e direcionamento necessários para desenvolver meu potencial frente aos diversos desafios que me proporcionaram grandes aprendizados.

Por ter participado da equipe WP4 da TED ANATEL/UFCG e pela experiência de trabalhar com pessoas inspiradoras. São eles: Alberi, Anderson, Eduardo, Jan, Raisson e Reinaldo. Um agradecimento especial a Jan Paulo pelas longas conversas realizadas quando estávamos prestes a travar em uma atividade complexa, mas sempre concluímos com soluções bem projetadas.

À FAPESQ pela oportunidade de exercer estudos na área de Big Data.

À participação no projeto de Estudos sobre Segurança Cibernética em Rede 5G - TED ANATEL/UFCG pela oportunidade de desenvolver pesquisa e competências em segurança da informação junto a tecnologias de grande relevância para as telecomunicações.

À participação no Projeto SSIP Supply Chain - Hewlett-Packard Brasil Ltda/UFCG pela oportunidade de realizar descobertas e experiências transformadoras como profissional.

Ao pessoal do LSD pelo companheirismo e diversas datas comemorativas acompanhadas das conversas LSD de grande importância para troca de conhecimento. Também ao pessoal do Suporte pela excelente ajuda perante as diversas dificuldades.

Conteúdo

1 Introdução **1**

1.1 Motivação 1

1.2 Objetivo 5

1.3 Organização do trabalho 6

2 Trabalhos relacionados **8**

2.1 Segurança definida por controles 8

2.2 Gestão das vulnerabilidades 13

2.3 Considerações 15

3 Segurança e computação na nuvem **17**

3.1 Conceitos fundamentais de computação na nuvem 17

3.1.1 Modelos de serviços 17

3.1.2 Virtualização 19

3.1.3 Instâncias no ambiente de computação na nuvem 20

3.1.4 Segurança em infraestruturas de computação na nuvem 20

3.2 Conformidade na segurança da informação 22

3.2.1 Conformidade contínua com controles e recomendações de segurança 22

3.2.2 Procedimentos do gerenciamento de vulnerabilidades 28

3.2.3 Segurança na cadeia de suprimentos 31

3.3 Especificações técnicas das ferramentas de segurança 32

3.3.1 Autoridades de relevância no fornecimento de ferramentas de segurança 32

3.3.2 Atributos das ferramentas de segurança 34

3.3.3 Considerações de usabilidade sem intervenção humana 38

4 Designação dos recursos base para conformidade 40

4.1 Concepção base de segurança da informação 40

4.2 Processo de identificação das ferramentas de segurança 44

4.3 Definições do escopo de interesse 49

5 Operacionalização da conformidade 54

5.1 Evidências de conformidade 54

5.2 Definição de métricas 56

5.2.1 Expressão quantitativa de recomendações de segurança 56

5.2.2 Métrica de Verificação do Nó 58

6 Provisionamento de um ambiente seguro 69

6.1 Implementações de segurança 69

6.1.1 Recomendações de segurança 69

6.1.2 Ferramentas de segurança 71

6.1.3 Escopo de implementação 77

6.2 Ambiente de testes 85

6.2.1 Execução em ambiente controlado 86

6.2.2 Execução em sistema em produção 98

6.3 Coleta de evidências e determinação de conformidade 103

6.4 Processo de automação 107

7 Conclusão 117

7.1 Considerações finais 117

7.2 Trabalhos futuros 119

A Termos de classificação de ferramentas de segurança 137

B Estudo de vulnerabilidades CVE do OpenStack 140

C Busca por ferramentas de segurança 144

D Strings de busca 151

E Tabelas de atribuição recomendação ferramenta **154**

F Recomendações do escopo de interesse **172**

G Processo de identificação das ferramentas de segurança **174**

Lista de Figuras

3.1 Modelos de serviço em computação na nuvem. 18

4.1 Tempo de inatividade Ncrack. 47
4.2 Tempo de inatividade Nmap. 47
4.3 Exemplo de grupo de controles de segurança de um *cluster* 48
4.4 Processo de detecção e relação de recomendações de segurança e ferramentas de segurança. 49
4.5 Escopo mediante o conceito de alto nível de requisitos de segurança IaaS. . 52

5.1 Mapeamento de ações e condições para definição de requisitos. 57

6.1 Relação entre componentes de gerenciamento de segurança e ferramentas de segurança. 76
6.2 Arquitetura do Kubescape CLI. 84
6.3 Verificação OpenSCAP para perfil de estação de trabalho nível 1. 87
6.4 Verificação OpenSCAP para perfil de servidor nível 2. 89
6.5 Verificação OpenSCAP OVAL Canonical. 90
6.6 Verificação atualizada OpenSCAP OVAL Canonical. 91
6.7 Verificação Lynis no Ubuntu 20.04 LTS. 92
6.8 Verificação de linha base de configuração. 98
6.9 Modelo de automação com implementação do *Compliance Monitor*. 109

B.1 Vulnerabilidades cadastradas no CVE por componente do OpenStack. . . . 140
B.2 Registros CVE para o OpenStack com avaliação CVSS separados por ano. . 141
B.3 Número de registros CVE classificados como alto impacto para diferentes componentes do OpenStack. 142

B.4 Número de registros CVE classificados como baixo impacto para diferentes componentes do OpenStack. 143

C.1 Recomendação de segurança CIS *Controls* V8. 145

C.2 *Clusters* do Kmeans. 147

Lista de Tabelas

4.1 Definição dos valores para critérios de exclusão em Novembro de 2023. . . 46

6.1 Ferramentas do projeto *Secure Code Box*. 72
6.2 Ferramentas resultantes do processo de busca. 73
6.3 Critérios de aceitação e descarte. 74
6.4 Ferramentas desconsideradas. 74
6.5 Recomendações alinhadas ao escopo de utilização. 77
6.6 Relação entre cenários e ferramentas de segurança. 82
6.7 Critérios de automação. 82
6.8 Mapeamento de evidências de conformidade. 104
6.9 Efeitos das métricas de explorabilidade e impacto. 107
6.10 Definição de conformidade para uso do SPIRE. 114
6.11 Seletores baseados em conformidade contínua para atestação de nó no SPIRE. 116

C.2 Desempenho das *strings* de busca. 149

E.1 Gerência de Identidade e Controle de Acesso. 156
E.2 Auditorias de Segurança. 160
E.3 Gerência de Vulnerabilidades. 166
E.4 Reposta de Emergência. 167
E.5 Recuperação de Desastres. 168
E.6 Backup. 169
E.7 Outros Relevantes. 171

F.1 Definição de recomendações para gerenciamento de vulnerabilidades e cadeia de suprimentos. 173

G.1 Ferramentas não consideradas. 175

Lista de Códigos Fonte

6.1 Arquivo config.ini. 111

Capítulo 1

Introdução

Nesta Seção apresentamos a introdução do trabalho por meio da motivação descrita na Seção 1.1 seguida dos objetivos detalhados na Seção 1.2 e por fim, a organização do trabalho detalhada na Seção 1.3.

1.1 Motivação

Sistemas de computação demandam segurança da informação, independente do seu tamanho e complexidade, seja um sistema simples para controle de estoque em um comércio local ou complexo para lidar com infraestruturas críticas como em *Industrial Control System* (ICS, em tradução, Sistema de Controle Industrial). Para os dois exemplos, a falta de segurança afeta o acesso por problemas de disponibilidade e integridade, prejudicando o uso do sistema, bem como problemas de confidencialidade para o segundo exemplo. Neste, há riscos de exposição de dados sensíveis e potenciais danos à reputação da organização. Os níveis de segurança devem contemplar os princípios da segurança da informação, como confidencialidade, integridade e disponibilidade [KMM+21].

A demanda por implementações de segurança impacta sistemas que lidam com dados sensíveis, tal como *Personally Identifiable Information* (PII, em tradução Informação Pessoalmente Identificável). Em setores como a saúde, segundo Mehrtak et al. [MSM+21], organizações têm hesitado em utilizar meios como a computação na nuvem por fatores de segurança que envolvem a confidencialidade das informações de pacientes. Questões de segurança também afetam Pequenas e Médias Empresas (PMEs), segundo Renaud et al.

[RW16], cibercriminosos têm as visado cada vez mais por terem uma defesa fraca. As PMEs desempenham fundamental importância para a economia global, porém são alvo de ataques, mesmo com a existência de medidas de proteção conhecidas [CZK22].

A segurança da informação é bastante discutida pelo setor acadêmico, industrial e governamental. É constante a busca por alternativas que mitiguem o impacto de agentes maliciosos na exploração de falhas de segurança, bem como a necessidade de criação e adequação a alternativas que fortaleçam medidas de segurança como as impostas pelas leis de proteção de dados como a Lei Geral de Proteção de Dados (LGPD[1]) e *General Data Protection Regulation* (GDPR[2], em tradução Regulamento Geral de Proteção de Dados). Leis como estas são fundamentais para fortalecer a proteção das PII, atuam estabelecendo regras para processamento e utilização de dados, tanto por setores públicos quanto privados. Isto mitiga problemas de insegurança jurídica que torna o país menos competitivo em cenários que demandam cada vez mais processamento de dados em diversos setores [dOGLN19].

Existem várias entidades e regulamentos que fornecem informações e guias sobre segurança da informação que podem se aplicar a sistemas de computação na nuvem além dos LGPD e GDPR, alguns são a *International Organization for Standardization* (ISO, em tradução Organização Internacional para Padronização), *National Institute of Standards and Technology* (NIST, em tradução Instituto Nacional de Padrões e Tecnologia), *Center for Internet Security* (CIS, em tradução Centro de Segurança na Internet), e *International Telecommunication Union* (ITU, em tradução União Internacional de Telecomunicações) [JAH23, Int20]. Também existem alternativas de aplicação mais restrita, como a instrução normativa nº 5[3], definida pelo governo federal brasileiro em 30 de agosto de 2021, que estabelece requisitos mínimos de segurança para soluções de computação na nuvem para entidades da administração pública federal.

Contemplar corretamente os importantes princípios de segurança e contornar dificuldades pertinentes à correta implementação são desafios enfrentados na área de segurança. Os desafios abrangem as organizações em diferentes aspectos. Segundo Fazenda [FF15], são

[1] https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm

[2] https://gdpr-info.eu/

[3] https://www.in.gov.br/en/web/dou/-/instrucao-normativa-n-5-de-30-de-agosto-de-2021-341649684

muitos os desafios relacionados à segurança, especificamente na manutenção de um Sistema de Gestão de Segurança de Informação (SGSI). O autor lista desafios como a falta de apoio da direção, falta de capacitação da área de segurança da informação, influência da cultura local, falhas na análise de risco e resistência a mudanças [FF15].

A utilização de meios para atuar como guia para boas práticas, auxílio a equipes e responsáveis pela segurança e medidas para evidenciar implementação de segurança se fazem necessários. Soluções que disponibilizam controles de segurança são uma forma adequada de considerar implementações de segurança. Os controles de segurança têm papel fundamental por meio de características que permitem o gerenciamento de risco, proteção da informação e considera métodos que abrangem estruturas organizacionais, administrativas, técnicas e legais [GFG17].

Uma grande quantidade de organizações disponibiliza controles de segurança com conjuntos de recomendações para implementação da segurança da informação. Porém, a grande oferta traz dificuldades para a escolha dos controles relevantes para implementação em um sistema. Essa oferta possibilitou a criação de *frameworks* que sumarizam controles de segurança recomendados por diferentes organizações. *Frameworks* como o CIS *Controls* v8 e o NIST *Cybersecurity Framework* categorizam diferentes aspectos de segurança que são relevantes para sistemas de computação.

Seguir e implementar medidas de segurança através da adoção de políticas e recomendações de segurança fornece garantias de conformidade. Segundo Bhaharin et al. [BASY19], a conformidade com as políticas de segurança é muito importante para a proteção contra ataques cibernéticos cada vez mais complexos. Implementar os meios necessários de proteção envolve esforços para garantir a conformidade seguindo diretrizes, padrões e estruturas de segurança frequentemente citados por recomendações e entidades relevantes da área de segurança da informação [BASY19]. No presente trabalho, a base de recomendações de segurança foi composta com os controles do CIS *Security controls* V8, NIST *Cybersecurity Framework* e categorias propostas pelo ITU direcionadas à segurança em computação na nuvem.

Assumir uma abordagem que considera implementações com base nestes documentos, traz benefícios para adequarmos à legislação de segurança que está cada vez mais madura.

Exemplificando, a Estratégia Nacional de Segurança Cibernética, E-ciber[4], recomenda o uso de alternativas do CIS e NIST quando pertinente. A E-ciber é uma estratégia definida pelo governo federal e um dos mecanismos da Política Nacional de Cibersegurança PNCiber[5]. A PNCiber é o decreto que traz um conjunto de diretrizes e objetivos estratégicos estabelecidos e supervisionados pela Comitê Nacional de Cibersegurança, CNCiber[6].

Garantir a conformidade com as recomendações de segurança demanda, em grande parte das vezes, por implementações técnicas de segurança. No entanto, recomendações são compostas em textos que descrevem o que deve ser feito, sendo assim limitadas à questões de implementações práticas como emprego de tecnologias adequadas. Isso expande a possibilidade de emprego de inúmeros procedimentos de como garantir a conformidade com recomendações de segurança, também abre espaço para vários níveis de implementação, não sendo apenas medido como verdadeiro para implementado ou falso quando não implementado. Essa limitação pode ser prejudicial no processo de implementação das recomendações de segurança. A título de exemplo, Potiguara et al. [CCCC20] descreve que observou a técnica de anonimização como sendo útil para proteção de dados segundo a LGPD e GDPR, no entanto não existem orientações de técnicas para acompanhar uma boa governança de dados. Em testes realizados no contexto de *Big Data*, a técnica de anonimização não garante a proteção de dados sem o emprego de outras técnicas [CCCC20].

Para a resolução de problemas pertinentes a implementações técnicas, propõe-se a utilização de ferramentas de segurança, que provisionam técnicas e tecnologias essenciais para a garantia da conformidade. Cada recomendação de segurança, quando não classificada como organizacional, pode ser atribuída a uma ou mais ferramentas de segurança identificadas. É válido destacar que existem problemas que podem comprometer o processo de utilização das ferramentas de segurança, como por exemplo erros causados por operações humanas. Segundo Bhaharin et al. [BASY19] o erro humano é a principal ameaça à segurança da informação, resultante da negligência, ignorância e falha no cumprimento das políticas or-

[4] `https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2020/decreto/d10222.htm`

[5] `https://www.planalto.gov.br/ccivil_03/_ato2023-2026/2023/decreto/D11856.htm`

[6] `https://www.gov.br/gsi/pt-br/colegiados-do-gsi/comite-nacional-de-cibersegurança-cnciber`

ganizacionais de segurança da informação. Dessa forma, o processo automático de uso das ferramentas é uma alternativa para mitigar as chances de erros no processo de conformidade com as recomendações de segurança, tanto no aspecto de implementação, como avaliação. Dempsey et al. [DEM17] enfatiza que a automação das avaliações de controle de segurança é importante porque as ameaças à segurança estão se materializando em um ritmo acelerado.

Considerando todas as informações apresentadas até o momento, observamos que, para suporte ao processo de avaliação, bem como toda a proposta de implementação de segurança da informação com base em controles e recomendações de segurança, é necessário definir um contexto específico de segurança relacionado às recomendações. Desse modo, aplicá-lo a um caso de uso em sistemas específicos que compõem a arquitetura de computação da nuvem.

Dessa forma, consideramos os contextos de gerenciamento de vulnerabilidades, que tangenciam a proteção da cadeia de suprimentos, como dois aspectos de segurança muito relevantes na atualidade. A aplicação dos contextos de segurança sendo empregues em cenários como aplicação, plataforma e sistema operacional é composto por diversos componentes. Para aplicação temos servidores *web*, servidores *Proxy*, bancos de dados, caches distribuídos, entre outros [LPD+13]. O cenário de plataforma contém imagens, contêineres e Kubernetes com seus componentes. O cenário de sistema operacional contém componentes verificáveis nos contextos de segurança a serem observados, como configurações incorretas e softwares vulneráveis.

1.2 Objetivo

Temos como objetivo deste trabalho propor implementações de segurança para sistemas que executem em ambiente de nuvem em conformidade com uso de controles e recomendações de segurança. Propomos realizar adequações por meio da utilização de técnicas de segurança disponíveis em padrões recomendados e diretrizes de boas práticas. Dessa forma, as implementações de segurança são guiadas por documentos de referência previamente analisados, validados e selecionados para suprir a correta implementação e direcionamento dos esforços de trabalho nas implementações efetivas para segurança. Isso contribui para um processo de segurança consistente que inibe esforços desnecessários com implementações não relevan-

tes e direciona os trabalhos para implementações automatizadas que fortaleçam uma postura verificável e mensurável de suporte à conformidade contínua.

Os objetivos específicos deste trabalho são:

1. Identificar organizações, controles e recomendações de segurança por meio de análise, dessa forma criar bases de implementação de segurança que permitam garantias de conformidade.

2. Identificar ferramentas de segurança para suportar implementações técnicas de controles e recomendações, bem como realizar análise de funções e viabilidade de automação para exercer a utilização da ferramenta sem intervenção humana.

3. Introduzir conformidade através de controles, recomendações e ferramentas de segurança para provisionar implementações adequadas de segurança no gerenciamento de vulnerabilidades em ambientes de computação na nuvem.

4. Prover conformidade contínua através da automação de processos que executem ferramentas para fornecer evidências e produzir artefatos sobre o estado de segurança do sistema com métodos para auditar e quantificar a conformidade.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte forma. O Capítulo 2 apresenta trabalhos relacionados a conceitos relevantes da pesquisa. São discutidas questões chave como conformidade e análises de recomendações de segurança. No Capítulo 3 é disposto o referencial teórico. São abordados os conceitos sobre computação na nuvem e aspectos da implementação de segurança.

O Capítulo 4 aborda as metodologias definidas e utilizadas para estudo de documentações de segurança, análise de recomendações e mecanismos para descoberta de ferramentas. Neste capítulo também é definido o ambiente utilizado para implementações de segurança utilizando recomendações suportadas por implementações técnicas de ferramentas e caracterização de métricas para garantias de conformidade.

O Capítulo 5 estabelece uma implementação segura considerando o processo de gerenciamento de vulnerabilidades. São definidos cenários de teste que consistem em executar

testes e coletar resultados em ambientes simulados e controlados, assim como cenários em sistemas reais em produção. É definido um processo automático para garantir conformidade contínua e manutenção de meios para mensurar o nível de conformidade.

Por fim, o Capítulo 6 aborda os principais resultados obtidos e debate os trabalhos futuros.

Capítulo 2

Trabalhos relacionados

Para nosso estudo, partimos da necessidade de conhecer profundamente a estrutura dos processos que relacionam o cumprimento de recomendações de segurança e, então, expandir automação para o máximo de operações possíveis em um modelo proposto. Nesse processo, atividades são indispensavelmente compreendidas e processos realizados de forma manual são mapeados para atender à demanda de padronização de designs para viabilização da automação. Os estudos observados nesta Seção, tangenciam abordagens consideradas ao longo do nosso trabalho.

2.1 Segurança definida por controles

Considerar implementações de segurança por intermédio da conformidade com controles, leva a obrigatoriedade de basear-se em documentos pertinentes a este propósito. Frente à vasta quantidade de documentos de segurança, destacamos os *frameworks* de segurança como alternativa viável para compor bases de conformidade definidas por agrupamentos de recomendações. Diante da presente oferta de *frameworks* que, embora semelhante à dos documentos de segurança, é numericamente mais reduzida, estudos visam análises junto aos *frameworks* para inferir propriedades que exercem influência direta em projetos de segurança.

Em estudo realizado por Juma et al. [JAH23], foi feita uma revisão sistemática envolvendo *frameworks* de segurança. No estudo, foram observados fatores estruturais dos *frameworks* que atribuem recomendações a tópicos específicos para área de segurança, como

identificação, proteção, detecção, resposta e recuperação, para atuação direta nas questões de segurança. Na identificação, está envolvida a compreensão sobre os riscos de segurança. A proteção relaciona as implementações que necessariamente são demandadas. A detecção envolve as atividades de detecção de eventos que afetam a segurança. Na resposta, estão acomodados os procedimentos para limitar os efeitos de incidentes de segurança. Na recuperação, os preparativos adequados são formalmente definidos para garantir uma rápida restauração.

A compreensão estrutural dos *frameworks* de segurança praticadas no estudo de Juma et al. [JAH23] também auxiliam na compreensão das limitações da estrutura. Para os *frameworks* de nosso interesse, a compreensão sobre NIST *Cybersecurity Framework* define limitações sobre possível aplicabilidade em IoT (*Internet of Things*, em tradução Internet das Coisas), com aplicação deficitária em escalabilidade, verificações em tempo real e privacidade de dados. Além disso, foi avaliado que o NIST *Cybersecurity Framework* não possui mecanismos para comportar processos de mitigação de vulnerabilidade. No geral, o estudo traz CIS *Controls* V8 e NIST *Cybersercurity Framework* como alternativas comumente utilizadas e define a existência de limitações e desafios a serem mitigados com abordagens inovadoras para contextos IoT.

Construir uma compreensão formal da estrutura e definição de propósito estabelecidos a um *framework* de segurança é essencial. No entanto, podemos querer determinar similaridade e inferir resultados para melhor entendimento e propriedade investigatória sobre estas estruturas para utilização conjunta entre *frameworks*. Em trabalho realizado por Amiruddin et al. [AAN21], foi realizada a utilização do NIST *Cybersecurity Framework* como estrutura principal e CIS *Controls* V8 para definir um plano para gerenciamento de risco cibernético.

O estudo conduzido por Amiruddin et al. compreendeu o cumprimento de 6 etapas definidas pelo NIST que realizam a priorização de escopo que determina: um sistema específico observado; orientação sobre identificar ativos e detectar vulnerabilidade e ameaças; criar um perfil atual de estado da organização; condução da avaliação de riscos; definir um perfil de destino que consiste em concentrar avaliações em categorias específicas para conquista de estado desejado pela organização; e priorização das lacunas identificadas no processo para desenvolver planos para conquista do estado desejado.

As etapas definidas pelo estudo demonstram potencial para descoberta de vulnerabilida-

des, visto que a condução das etapas permitiu determinar os ativos de uma unidade de TI, que determinam o sistema observado e determinado no escopo. Amiruddin conduz a definição das fontes de ameaças e determina vulnerabilidades com base nos ativos e nas ameaças definidas. Ele também determina os controles para prevenção contra essas ameaças. Adiante o estudo determina o mapeamento entre CIS e NIST e define o perfil de estado de cibersegurança da organização considerando 4 níveis verificáveis determinados pelo NIST.

O autor determina que nem todos os controles do CIS são mapeados para oNIST. O cenário observado definiu que uma organização analisada foi estabelecida no nível 1 de gerenciamento de riscos, o que define que uma organização não tem uma prática formal de gerenciamento. O nível 3 é potencialmente atingido com o uso das ações de recomendações mapeadas entre o CIS e o NIST que um processo de gerenciamento de risco é formalmente aprovado e expresso por políticas e práticas de segurança.

Como termo relacionado a recomendações de segurança, a conformidade é tratada ao considerarmos a implementação das recomendações. Quando realizada de forma contínua, devem ser observados meios automatizados para suprir a necessidade de eficiência dos processos de gestão. Em estudo realizado por Cheng et al. [CVCLC18] é definido o mapeamento ontológico de requisitos para obtenção de conformidade através de gerenciamento automatizado. Cheng utiliza processamento de linguagem natural para obter o mapeamento de ações e condições necessárias que permitam o mapeamento de atividades de segurança para documentos de conformidade.

Partindo do princípio da existência de lacunas entre ferramentas de gestão de conformidade e ferramentas de gestão de segurança, definir um mapeamento conceitual de recomendações irá permitir a reutilização do modelo para vários padrões distintos que normalmente utilizam processos de análise caros e ineficientes. O estudo demonstra que o uso de processamento de linguagem natural obteve uma precisão de 69,38%, com ocorrência de potenciais erros de detecção de ações ocasionados por palavras com possibilidade de múltiplos significados.

Dificuldades em estabelecer o monitoramento de conformidade também são observadas em requisitos que não necessariamente dependem de verificações técnicas. Cheng utiliza *scripts* de verificação automática para determinar a conformidade executando tarefas mapeadas a recomendações. Portanto, em requisitos que demandam cumprimento de uma política

formal de segurança, como por exemplo, proibir o uso de softwares não autorizados, sai do escopo de verificação dos *scripts*, sendo uma política que demanda outros meios de verificação.

No estudo de Alromaih et al. [AIE22] trata da necessidade de conformidade para garantir uma postura robusta de segurança cibernética aplicada à transformação digital de cidades inteligentes. Em observação realizada no estudo, destaca-se que organizações que adotam transformação digital, em sua maioria eram certificadas em conformidade com segurança. Uma certificação define que determinada organização cumpre com o estabelecido em documentos técnicos ou padrões de segurança. No entanto, o estudo utiliza uma abordagem técnica de segurança que inclui detecção de vulnerabilidades e testes de penetração. É fato que, embora as organizações possuam certificação, elas ainda enfrentam ataques cibernéticos que causam danos extensos, pondo em evidência problemas de ineficiência ou desatualização da conformidade.

É proposto o uso de segurança técnica com implementação de gestão de vulnerabilidades e execução de testes de penetração para alcançar uma conformidade de forma contínua. A segurança técnica que dá suporte à gestão de vulnerabilidades é tida no estudo com o uso do software Nessus, uma ferramenta de verificação desenvolvida pela Tenable[1], enquanto os testes de penetração são realizados por ferramentas como Metasploit, Wireshark, W3AF e John the Ripper. O autor conclui que a conformidade de segurança certificada pode ser útil, porém, apenas quando a empresa continua com a segurança técnica, como a verificação de vulnerabilidades. É ressaltada a importância de ferramentas automatizadas para garantir conformidade contínua com os padrões e regulamentos de segurança cibernética. Este estudo expõe as evidentes necessidades de verificação contínua para manutenção da conformidade.

Em outro estudo realizado por Agarwal et al. [ABD+22], a conformidade relaciona-se com os termos de automação. Em específico, o trabalho trata da automação de processos, da perspectiva de conformidade regida por órgãos reguladores e padronizadores para definir o conjunto mínimo de controle de segurança cibernética. Neste ponto, uma organização deve necessariamente implementar controles de segurança para suporte da operações da empresa [ABD+22]. O estudo expressa a necessidade em manutenção da agilidade e demanda por modernização do gerenciamento de conformidade para abordagens automatizadas de ve-

[1] `https://pt-br.tenable.com/products/nessus`

rificação contínua. Ainda, direciona esforços para compreensão profunda de padrões que suportam métodos automatizados de verificação de conformidade.

Em específico, o estudo trata da conformidade como código e define geração e gerenciamento de artefatos como documentação necessária para a conformidade. É ressaltada a dificuldade com relação a verificações em diversos aspectos, como uso de ferramentas para gerenciar ativos, controles de acesso, verificação de vulnerabilidades, monitoramento de eventos de segurança, etc. Ferramentas usadas neste processo muitas vezes não foram projetadas para fins de conformidade e tendem a divergir nas interfaces e modelos de dados que dificultam processos de gestão de conformidade. Na proposta de conformidade como código, o estudo utiliza o *Open Security Controls Assessment Language* (OSCAL, em tradução, Linguagem Aberta para Avaliação de Controles de Segurança), um modelo em desenvolvimento pelo NIST em conjunto com a indústria, para tratar de estruturar a conformidade em um agrupamento correspondente de artefatos-chave impressos em formatos processáveis por máquina.

Estruturalmente a proposta do OSCAL abrange a especificação de requisitos, catálogos de recomendação e customização de perfis com linhas base para conformidade, além de captura de resultados e avaliação para suporte a auditorias. O estudo expressa fortemente a necessidade de automação, desenvolvimento de alternativas emergentes para auditorias de conformidade e a importância de expressar conformidade por meio de artefatos padronizados verificáveis. Com isso, a proposta dos autores, visa compreensão comum da conformidade em sistemas heterogêneos que lidam com inúmeras fontes de artefatos para seguir uma proposta de padronização em desenvolvimento pelo NIST.

Ainda partindo da necessidade de verificação contínua da conformidade, Hedjeres et al. [HBSH18] aborda uma alternativa baseada em conformidade orientada a eventos temporais. É proposta uma arquitetura que considera lógica temporal e paradigmas de ação e condição para composição de requisitos de conformidade usando orientação a eventos. É sugerido por Hedjeres que a conformidade de um sistema deve ser suficientemente alcançada quando é passível de verificação durante sua execução, visto que é quando emerge o comportamento real de um sistema e torna-se evidente a necessidade da verificação contínua de conformidade.

O estudo parte da ideia de mapear níveis de eventos que classificam mudanças por sig-

nificância, então é aplicado o uso de computação orientada a eventos baseada em regras de ECA (*Event-Condition-Action*) especificando qual ação deve ser executada em resposta a um evento específico. O modelo para verificação de conformidade é composto por uma estrutura de monitoramento que define os eventos analisados, gerenciadores de eventos para identificação e relatório, políticas de conformidade expressas como regras verificáveis e componentes que observam notificações reportadas para tomar decisões sobre análise e determinar a conformidade.

2.2 Gestão das vulnerabilidades

Como uma abordagem relevante para a segurança, atividades que compreendem o gerenciamento de vulnerabilidades auxiliam organizações nos cuidados sobre a superfície de ataque. Muitos *frameworks* de segurança dispõem de recomendações relacionadas à definição de processos de gerenciamento de vulnerabilidade. Esta atividade é verificada em diversos estudos que abordam a segurança da informação nos mais variados aspectos e tipos de abordagem.

Um estudo realizado por Kovačević et al. [KSLZ20] estabelece uma implementação de sistema de gerenciamento de vulnerabilidades. É essencialmente importante o entendimento sobre sistemas de gerenciamento de vulnerabilidades para compor o que é requerido em recomendações de segurança dos *frameworks*. No estudo é abordada a utilização do *National Vulnerability Database* (NVD, em tradução, Banco de Dados Nacional de Vulnerabilidades) como alternativa para descobrir vulnerabilidades conhecidas publicamente e descreve a proposta como uma alternativa automatizada de análise de vulnerabilidades de forma simplificada.

Para o processo de gerenciamento de vulnerabilidades, Kovačević propõe o uso de um Diagrama de Fluxo de Dados (DFD) responsável por representar especificações e estrutura do sistema que considera *Common Platform Enumeration* (CPE, em tradução, Enumeração Comum de Plataformas) como formato padrão. A ideia é partir de verificações utilizando um DFD submetido para detecção de vulnerabilidades com o uso de um componente de detecção que entra em contato com o NVD para solicitar as listas de vulnerabilidades. A abordagem tratada por Kovačević define conceitos utilizados por processos automatizados, como as con-

sultas realizadas a bases de dados de conhecimento sobre vulnerabilidades. Em conclusão do estudo, constatou-se a ineficiência do NVD devido à sua dependência do *Common Vulnerabilities and Exposures* (CVE, em tradução, Vulnerabilidades e Exposições Comuns).

A compreensão sobre o contexto de verificação de vulnerabilidades é necessária para determinar quais componentes são alvos de verificação. Um estudo realizado por Sasubilli et al. [SV21] torna evidente os problemas de segurança relacionados a computação na nuvem, em específico são observados os desafios enfrentados em computação na nuvem, as ameaças e vulnerabilidades. Considerando a camada de IaaS, o estudo aponta a relação em disponibilizar VMs aos usuários, isto relaciona o usuário de serviços de nuvem com atuação e responsabilidades sobre componentes como SO, tráfego de rede e aplicações. Sasubilli elenca 10 principais problemas das IaaS, como a falta de controles de segurança consistentes em ambientes locais e multi nuvem e incapacidade de monitorar sistemas de carga de trabalho em nuvem e aplicativos para vulnerabilidades.

No âmbito de avaliações de vulnerabilidades, o estudo desenvolvido por Santoso et al. [SIA23] baseia métodos de análise em verificação passiva e *Common Vulnerability Scoring System* (CVSS, em tradução Sistema de Pontuação de Vulnerabilidade Comum) para avaliação de aplicações *web*. A varredura passiva é realizada pela coleta de dados sobre o sistema para que a inferência sobre vulnerabilidades seja realizada. Isto é realizado sem interação direta e utilizando coleta informações sobre SO em uso, portas ativas e softwares instalados. As verificações propostas por Santoso tem objetivo de minimizar cargas de trabalho impostas ao alvo durante uma verificação e propor o CVSS para determinar classificação e priorização dos esforços de correção das vulnerabilidades.

Em análises realizadas no estudo, foram utilizados contêineres para avaliar a facilidade de implantação e gestão dos recursos do sistema durante os testes de segurança, bem como execução de ferramentas automatizadas de verificação de vulnerabilidades. As conclusões definem a viabilidade de verificação de vulnerabilidades utilizando contêineres para verificar sites de destino. Esta abordagem permite verificações simultâneas possibilitando a capacidade de ajustar a quantidade de processos de varredura de acordo com a disponibilidade de recursos, tornando o método de verificação flexível e mais eficiente.

2.3 Considerações

Desse modo, considera-se que o estudo tratado por Juma observa estruturalmente os *frameworks* de segurança e os aspectos de cobertura e limitações. Complementarmente, o trabalho de Amiruddin realiza comparação entre *frameworks* de segurança, sendo os *frameworks* do CIS e NIST mapeados entre sí e inferindo o potencial em determinar gerenciamento de riscos. Os dois estudos são relevantes para compreensão operacional dos *frameworks*, mas limitam-se na apresentação de soluções para implementações técnicas de segurança. Já no estudo de Cheng, as implementações começam a ser discutidas partindo do ponto de compreensão de requisitos contidos nas recomendações de segurança e mapeados com uso de processamento de linguagem natural. No entanto, o estudo destaca limitação em cumprir requisitos que exigem formalização de políticas de segurança.

Alternativamente, nossa abordagem realiza o mapeamento de frameworks de segurança, não entre si, mas com objetivo de aumentar o alcance da conformidade com uso complementar de recomendações para determinados contextos e expande implementações técnicas com propostas de implementação automática para abarcar políticas baseadas em recomendações organizacionais.

No campo da conformidade, Alromaih expõem a necessidade das verificações contínuas em detrimento de certificações, que são limitadas pela capacidade de verificação da conformidade. Indo além, foram abordadas ferramentas para verificação técnica contínua com execução de verificação de vulnerabilidades e testes de penetração. Portanto, o estudo não aponta para alternativas de código aberto nem foca especificamente em definir agrupamento de recomendações para exercer o uso das ferramentas com base nas recomendações direcionadas.

Como observação complementar, o estudo de Agarwal expressa a necessidade de automação da conformidade e destaca a limitação das ferramentas de segurança em não serem projetadas para suporte a verificações de conformidade com indicativo para adoção de uma padronização OSCAL. Em nosso trabalho, observamos o benefício de uma padronização com uso de especificações SCAP em artefatos para definição de processos automatizados de verificação.

A abordagem de conformidade orientada a eventos tratada por Hedjeres considera mu-

danças para mover ações de verificação de conformidade, porém é exigida uma interpretação de eventos em níveis para determinar quais verificações serão executadas. Em nosso estudo, as verificações são realizadas conforme definido na frequência de verificação, exercendo integralmente as atividades necessárias para definição de conformidade em períodos de tempo determinados.

Capítulo 3

Segurança e computação na nuvem

3.1 Conceitos fundamentais de computação na nuvem

A computação na nuvem envolve uma série de recursos que são disponibilizados através de uma infraestrutura. Dentre os recursos estão o armazenamento, bancos de dados, redes, servidores e softwares. Uma das definições utilizadas sobre computação na nuvem é a estabelecida pelo NIST[1], do qual indica tratar-se de um modelo que permite acesso sob demanda a recursos computacionais dispostos em rede, que podem ser configuráveis permitindo suporte para disponibilização de redes, aplicativos e serviços. A computação na nuvem tem um rápido provisionamento de recursos, demandando pouco esforço em seu gerenciamento e pode garantir elasticidade e disponibilidade definindo também modelos de serviços [MG11].

A computação na nuvem envolve uma série de definições bem particulares e atraentes, principalmente nos custos de utilização. Segundo Bello et al. [BOA+21], o princípio fundamental, que guia a computação em nuvem, é partir da reutilização da estrutura fazendo com que os custos de computação possam ser reduzidos.

3.1.1 Modelos de serviços

Para cobertura adequada de componentes considerando princípios fundamentais de segurança, considera-se a nuvem em seus aspectos arquiteturais. Dessa forma, a computação na nuvem se divide em modelos de serviços e modelos de implantação, além de utilizar a

[1] `https://csrc.nist.gov/glossary/term/cloud_computing`

virtualização permitindo o gerenciamento de *Virtual Machine* (VMs, em tradução, Máquina Virtual) e suporte a alocação de recursos. Os modelos de serviço como visto na Figura 3.1 são *Infrastructure as a Service* (IaaS, em tradução, Infraestrutura como Serviço), *Platform as a Service* (PaaS, em tradução, Plataforma como Serviço), e SaaS *Software as a Service* (SaaS, em tradução, Software como Serviço) [MZ21].

Figura 3.1: Modelos de serviço em computação na nuvem.

Fonte: Adaptado de Redhat. [Hat]

IaaS é o modelo que integra os recursos primários no qual os provedores de nuvem fornecem computação, infraestrutura virtualizada, servidores virtuais, armazenamento e rede. Os clientes têm controle total sobre a configuração e implantação de aplicativos e sistemas operacionais dentro da infraestrutura virtualizada. Eles são responsáveis pela gestão e manutenção do software e das aplicações, enquanto o provedor de nuvem cuida da infraestrutura subjacente e provê serviços de computação e recursos de armazenamento e rede quando houver demanda.

PaaS oferece os meios necessários para desenvolvedores criarem e distribuírem programas sem preocupações com infraestrutura, fornecendo uma plataforma para o gerenciamento

de aplicativos. O provedor de nuvem gerencia tanto a infraestrutura quanto o ambiente de execução. Os clientes podem se concentrar no desenvolvimento e configuração de aplicativos, enquanto o provedor de nuvem cuida da configuração e escalabilidade da plataforma.

SaaS é o modelo no qual os usuários têm acesso a serviços completos hospedados na nuvem por meio de interfaces *web* ou de aplicativos disponíveis. Os aplicativos são entregues como serviços prontos para uso e os usuários podem acessá-los por meio de variados dispositivos e configurações. Este é um modelo que tem foco na interface com o usuário, onde este utiliza e gerencia o software que está disponível na nuvem. Nesse modelo, os usuários não precisam se preocupar com infraestrutura, plataforma ou manutenção do software, pois todas essas responsabilidades são assumidas pelo provedor de nuvem.

A compreensão dos diferentes modelos de serviço praticados em computação na nuvem deixa claro a alternância da responsabilidade sobre os componentes entre os provedores de nuvem e os clientes, além de delimitar quais são os recursos primários da infraestrutura. Independente da parte responsável, componentes da infraestrutura virtualizada demandam por requisitos de segurança por tratar-se de uma superfície de ataque com potencial risco de comprometimento.

3.1.2 Virtualização

A virtualização é fortemente utilizada em computação na nuvem por fornecer uma abstração lógica para aplicativos e sistemas operacionais permitindo a independência do sistema lógico da estrutura física. A virtualização se estende para componentes como processadores, rede, armazenamento, memórias e permite isolamento, aumento da segurança, flexibilidade e melhora do gerenciamento de recursos, além de possibilitar o balanceamento de carga e respostas à recuperação de desastres [Cha14].

O gerenciamento e a manutenção dos recursos de Tecnologia da Informação TI é simplificado pela virtualização, pois as VMs podem ser facilmente implantadas, migradas e escaladas conforme necessário. O correto gerenciamento de VMs envolve o processo de mapear VMs em *hosts* físicos e considerações sobre capacidades de memória, armazenamento, largura de banda e *Central Processing Unit* (CPUs, em tradução, Unidade Central de Processamento). Decisões sobre migração das VMs definidas por políticas de mapeamento determinam o processo de migração de VMs permitindo que o gerenciamento das VMs seja

feito sem prejuízos para serviços que estão em execução [BOA+21].

3.1.3 Instâncias no ambiente de computação na nuvem

A disponibilidade de recursos em computação na nuvem abrange componentes fundamentais da infraestrutura, pelos quais a segurança deve ser resguardada com uso de controles e boas práticas. Tomando o trabalho de Sandhu [San21] como exemplo, a solução de computação na nuvem oferece economia e escalabilidade para a área de big data. Em seu estudo, ele define ser conveniente o acesso a recursos de computação configuráveis, dentre os quais estão servidores, redes e soluções de armazenamento baseados em pagamento para uso [San21]. No entanto, componentes comumente utilizados neste contexto, como Sistemas Operacionais (SOs), instâncias, volumes e redes, devem ser corretamente gerenciados para manutenção da segurança.

Como fator crucial na arquitetura de computação na nuvem, as instâncias de máquinas virtuais estão relacionadas a diversos aspectos de segurança. O processo de virtualização se relaciona aos componentes de rede, sistemas operacionais, softwares, entre outros. Modi et al. [MPB+13] destaca que as redes são um componente fundamental na arquitetura de computação na nuvem e evidencia em seu estudo problemas em que instâncias constituem uma nova base para invasores. Exemplificando, o comprometimento de uma instância pode permitir o uso de um *backchannel* por parte dos invasores. Desse modo, um ataque a camada física seria viabilizado pela leitura da memória de acesso aleatório de *hosts* virtualizados [MPB+13].

3.1.4 Segurança em infraestruturas de computação na nuvem

Considerando os modelos de serviço praticados em computação na nuvem, a evidente flexibilidade permite que organizações utilizem o melhor modelo para atender a seus objetivos. Embora a mudança da responsabilidade pelos componentes em computação na nuvem varie entre provedor de nuvem e cliente, nos modelos definidos na Seção 3.1.1, o modelo IaaS delimita a infraestrutura primária, gerenciada por um provedor de nuvem ou uma organização quando se adota o modelo de nuvem privada, e a infraestrutura virtualizada.

Por meio da estrutura primária, uma série de outros recursos e consequentemente mo-

delos de serviços são definidos. São variados os recursos que compõem a arquitetura de computação na nuvem, sendo assim, torna-se difícil a manutenção bem ajustada sobre o gerenciamento de segurança. No estudo de Sasubilli [SV21], são definidos problemas de segurança nos diferentes modelos de serviço, de modo que o aumento da complexidade da estrutura demanda maior tempo e esforço de implementação e manutenção. Entre os problemas identificados estão a incapacidade de manter a conformidade regulatória, a dificuldade em monitorar carga de trabalho em nuvem e aplicativos para identificar vulnerabilidades, e a falta de controles de segurança consistentes em ambientes local e de nuvem. Estes são apenas alguns dos problemas destacados [SV21].

Algumas alternativas de segurança são amplamente discutidas e utilizadas. Estas, quando implementadas, resolvem problemas específicos de segurança. Alguns exemplo disso são o surgimento do DevSecOps, computação confidencial e o *Secure Production Identity Framework For Everyone* (SPIFFE, em tradução Estrutura de Identidade de Produção Segura para Todos) e SPIFFE *Runtime Environment* (SPIRE, em tradução Ambiente de Execução).

O DevSecOps [MCP17] teve início com a necessidade de agregar mais confiabilidade aos produtos desenvolvidos beneficiando com diminuição de explorações de vulnerabilidades, aumento da qualidade dos códigos e velocidade de implementação. A computação confidencial surge da necessidade de segurança à medida que cada vez mais dados sensíveis estão presentes em computação na nuvem. Esta proporciona o benefício de minimizar a necessidade de confiar nas infraestruturas. É estabelecida a prevenção contra o acesso não autorizado aos dados, estando estes em processamento, em trânsito ou em repouso [BSS+20]. O SPIFFE tem como objetivo a identificação segura de sistemas de software em ambientes heterogêneos. Ele também realiza atestação de nós e cargas de trabalho, com isso emite SPIFFE *Verifiable Identity Document* (SVIDs, em tradução Documento de Identidade Verificável) [2] de forma segura baseando-se em propriedades que serão coletadas durante a execução.

Apesar de soluções como estas, temos como escopo o interesse sobre a infraestrutura virtualizada e os componentes da superfície de ataque potencialmente exposta a exploração de vulnerabilidades com uso de bibliotecas vulneráveis, componentes ou softwares de terceiros. Exemplificando, um SO em execução na VM pode conter vulnerabilidades, portas expostas e softwares desatualizados que comprometam diretamente a segurança.

[2] `https://spiffe.io/docs/latest/deploying/svids/`

3.2 Conformidade na segurança da informação

3.2.1 Conformidade contínua com controles e recomendações de segurança

A segurança interessa a governos, empresas e indivíduos. O desenvolvimento tecnológico é acompanhado pelo crescimento do cibercrime, que ganhou evidência e constante presença desde a década de 80 [JAH23]. A ampla adesão de tecnologias por setores de alto risco, como o setor bancário [KB19], exige que sejam adotados meios de proteção eficientes.

Problemas de segurança que culminaram em grandes comprometimentos de sistemas, como exemplo, o ocorrido com a SolarWinds[3][JAH23] afetando milhares de organizações em todo o mundo por meio da vulnerabilidade CVE-2020-10148[4], enfatizam a inegável necessidade de estabelecer controles de segurança cibernética que devem ser implementados pelas organizações, bem como a necessidade de adequar-se a leis como LGPD e GDPR. Estes e outros fatores tornam evidente o esforço de órgãos reguladores e padronizadores, como a ISO, *Federal Risk and Authorization Management Program* FedRAMP e também guias, normas, manuais e documentos de segurança desenvolvidos por órgãos como CIS, NIST, CSA, NSA, ENISA e ITU [ABD+22].

Órgãos como o ITU[5], voltado para telecomunicações, propõem padrões para questões específicas de segurança através de documentos especializados. O ITU é uma agência das Nações Unidas que considera a conectividade em escala internacional. Este órgão define padrões técnicos para garantir interconexão contínua de redes e tecnologias com esforços para melhoria das *Information and Communication Technologies* (ICTs, em tradução, Tecnologias da Informação e Comunicação). A entidade também trata de segurança para IaaS em computação na nuvem através de documentos especializados como o ITU-T X.1605. Este documento define segurança em IaaS distribuídas em 6 componentes de gerenciamento de segurança, são eles:

- **Gestão de identidades e controle de acessos:** Componente que consistem na administração criteriosa e na proteção das identidades dos usuários, juntamente com

[3] `https://www.cisecurity.org/solarwinds`
[4] `https://nvd.nist.gov/vuln/detail/CVE-2020-10148`
[5] `https://www.itu.int/en/about/Pages/default.aspx`

a regulação do acesso aos recursos dos sistemas. Esta prática é crucial e incorpora metodologias avançadas de autenticação e autorização, assegurando que somente indivíduos ou serviços devidamente autorizados possam interagir com os recursos apropriados, essencial para a salvaguarda de informações sensíveis. Neste componente, o ITU tem propostas robustas, entre as quais se destacam o apoio à autenticação multifatores *Multi-Factor Authentication* (MFA, em tradução, Autenticação Multifator), a aplicação do princípio de minimalismo de privilégios para administradores, e a implementação rigorosa de sistemas de registro de *logs*. Estes registros são cruciais para o monitoramento de atividades relacionadas à autenticação, autorização e outras operações inerentes à gerência de identidades e ao gerenciamento de acessos *Identity and Access Management* (IAM, em tradução, Gestão de Identidade e Acesso) [Int20].

- **Auditorias de segurança:** Representa procedimentos sistemáticos e precisos, essenciais para avaliar a efetividade e conformidade das estratégias de segurança em uma organização. Estas auditorias abrangem uma análise criteriosa de políticas, processos, controles e registros pertinentes, com o intuito de identificar falhas de segurança, riscos potenciais e eventuais transgressões às normas de proteção. O propósito primordial dessas auditorias é assegurar adequadamente a defesa de ativos, informações e sistemas organizacionais contra ameaças, tanto internas quanto externas [Int20].

  Auditorias de segurança são vitais para sugerir medidas corretivas e preventivas. Elas desempenham um papel crucial na manutenção da integridade, confidencialidade e disponibilidade dos recursos corporativos, além de garantir a observância às regulamentações e padrões vigentes de segurança. Auditoria de segurança é um componente que propõe alternativas como a utilização de registros para monitoramento de atividades de autenticação, autorização e gestão, a implementação de mecanismos de segurança contra manipulações indevidas, a sincronização precisa dos relógios de rede e a inclusão de dados minuciosos nos registros de auditoria [Int20].

- **Gestão de vulnerabilidades:** Também detalhada na Seção 3.2.2, constitui um conjunto de procedimentos estratégicos para a identificação, avaliação e correção de falhas de segurança. Este processo continuamente executado é fundamental para assegurar a integridade de ambientes computacionais, reduzindo lacunas de segurança e defen-

dendo contra ameaças emergentes. No contexto deste componente de gerenciamento de segurança, o ITU propõe orientações específicas, incluindo a implementação de sistemas para inventário detalhado de ativos e suas respectivas versões, para isto podem ser utilizados *Software Bill of Materials* (SBOMs, em tradução, Lista de Materiais de Software) como discutido na Seção 3.2.3. Além disso, também estão incluídas avaliações periódicas de vulnerabilidades, a aplicação de correções e atualizações de segurança, o estabelecimento de configurações de segurança padrão e a auditoria de alterações nas configurações. A adoção de uma abordagem abrangente na gerência de vulnerabilidades é imprescindível para reforçar a segurança e assegurar a proteção eficaz dos ativos contra incursões maliciosas [Int20].

- **Resposta a emergências:** Constitui um elemento fundamental para assegurar a operacionalidade eficaz dos sistemas de computação em nuvem, especialmente após incidentes de segurança. Um plano de resposta de emergência, estruturado com eficiência, estabelece um conjunto de procedimentos e medidas técnicas essenciais. Este plano tem como objetivo mitigar o impacto de incidentes de segurança em plataformas e serviços de nuvem, fornecendo diretrizes claras para a equipe de segurança. Equilibrando detalhamento com flexibilidade, o plano é vital para a gestão de riscos. O desenvolvimento e a gestão de um plano de resposta de emergência seguem um ciclo de melhoria contínua, dividido em três fases: desenvolvimento, teste e implementação, e manutenção [Int16].

  Durante a fase de desenvolvimento, análises de risco e impacto são realizadas para definir os requisitos de recuperação e delinear o escopo da resposta. A fase de teste e implementação compreende exercícios, treinamentos e avaliações para assegurar a eficácia do plano. Por último, na fase de manutenção, ocorre a revisão regular do plano, ajustando-o às mudanças no ambiente e mantendo sua efetividade contínua [Int16].

- **Recuperação de desastres:** Engloba um conjunto de protocolos e procedimentos essenciais, destinados a restaurar a operacionalidade de sistemas ou ambientes após eventos adversos. Este processo envolve ações estratégicas para reduzir impactos e danos, enfatizando a restauração de dados, infraestrutura e serviços críticos. O propósito central da recuperação de desastres é assegurar a disponibilidade e a integridade

de recursos vitais, bem como a rápida retomada das operações [Int20].

Este processo abarca a formulação e execução de planos de contingência, monitoramento contínuo e atualizações periódicas, visando atenuar as consequências de desastres. Segundo o ITU, práticas como a definição de objetivos de recuperação específicos, *backups* regulares de sistemas e dados, validação frequente do plano de recuperação, e avaliação constante dos riscos são fundamentais. Estas ações incluem a análise de impacto nos negócios, armazenamento de dados em múltiplas localidades e realização de simulações de recuperação.

Estas orientações são parte do gerenciamento de segurança, estão alinhadas ao controle de Gestão de Respostas a Incidentes (CIS *Control* 17), focando em procedimentos pós-incidente para restabelecer a normalidade operacional, diferentemente das ações imediatas de resposta a emergências.

- ***Backup*:** No âmbito da computação em nuvem, a execução de *backups* é um procedimento crítico, essencial tanto para clientes quanto para provedores de serviços. Os processos de *backup* visam assegurar a disponibilidade e proteção de dados e sistemas vitais em eventos de perda ou corrupção de informações. A eficiência e confiabilidade na recuperação de dados são fundamentais para restabelecer o ambiente a um estado seguro, anterior a incidentes adversos [Int16].

  As políticas de *backup* devem ser meticulosamente formuladas, levando em conta as especificidades de cada cliente ou serviço em nuvem. Aspectos cruciais incluem a metodologia de armazenamento de dados, considerando opções como criptografia, a localização física dos armazenamentos e a implementação de procedimentos para a verificação da integridade dos *backups*.

  De acordo com orientações do ITU, é imprescindível definir uma estratégia de *backup* apropriada, contemplando a frequência das cópias de segurança e determinando prazos para a retenção de dados. Ressalta-se também a importância de procedimentos para validar a eficácia dos *backups*, incluindo verificações de integridade e testes de recuperação. Adicionalmente, é sugerida cautela no uso de *snapshots* de volumes de máquinas virtuais, evitando exageros que possam comprometer tanto o desempenho quanto o espaço de armazenamento.

Tomar medidas de segurança é essencial às organizações, fundamentar suas implementações com base em controles e recomendações de segurança de órgãos relevantes, permite que organizações possam derivar recomendações aplicáveis a seu contexto de uso [ABD+22]. A ampla oferta de soluções dispostas em documentos de segurança torna difícil a tarefa de estabelecer meios para identificar a melhor solução dentre as disponíveis. Como potencial solução, o surgimento de *Frameworks* de segurança facilita o processo de identificação de soluções. Estes sumarizam conjuntos de controles e recomendações de segurança e motivam uma forte resposta ao surgimento de novos ataques cibernéticos. Em estudo realizado, Juma et al. [JAH23] identificou as principais estruturas de segurança utilizadas globalmente para se proteger de qualquer tipo de ameaça. São listados CIS *Controls*[6] e NIST *Security Framework*[7], além de ISO/IEC 27001[8], *Control Objectives for Information and related Technology* (COBIT[9], em tradução, Objetivos de Controle para Informação e Tecnologia Relacionada) e *SysAdmin, Audit, Network, and Security* (SANS[10] *Critical Security Controls*, em tradução, Administrador de Sistemas, Auditoria, Rede e Segurança).

Embora todas tratem de questões de segurança, cada uma tem sua finalidade específica. O COBIT, embora trate questões de segurança, exerce influência na governança e gerenciamento de TI, com ênfase em tarefas administrativas como a governança corporativa [JAH23]. A ISO/IEC 27001 exige investimento em certificação, sua relevância internacional torna a certificação uma ótima alternativa para organizações que necessitam validar sua conformidade, além de estender a confiabilidade da ISO para seus produtos e serviços. O SANS *Critical Security Controls*[11] anteriormente composto por 20 controles de segurança criados pela SANS, passou por uma atualização que resultou no CIS *Controls* V8 com 18 controles atualmente mantido pela CIS.

O CIS *Controls* é um padrão desenvolvido pelo CIS, contém controles para auxílio a organizações e indivíduos com objetivo de direcionar esforços e concentração a aspectos importantes para a defesa contra ataques cibernéticos [AAN21]. Para desenvolver os con-

[6] https://www.cisecurity.org/controls/cis-controls-navigator

[7] https://www.nist.gov/cyberframework

[8] https://www.iso.org/standard/27001

[9] https://engage.isaca.org/brasiliachapter/certificacao/outras-certificacoes/cobit5

[10] https://www.sans.org/

[11] https://www.cisecurity.org/controls/cis-controls-list

troles, uma ampla comunidade de profissionais ligados a governo e indústria participaram do processo. O CIS *Controls* V8 é fundamentalmente acessível a organizações de grande e pequeno porte, seu desenvolvimento considera um projeto focado na simplicidade quando comparado com outras estruturas de controle [BS22]. NIST *Cybersecurity Framework* é um padrão desenvolvido pelo NIST, tem como objetivo agrupar diretrizes de segurança aplicáveis a todos os setores de infraestrutura crítica. Sua abordagem fornece flexibilidade, isso permite estabelecer um projeto iterativo com atividades priorizadas focado em desempenho sustentável [BS22].

Em conjunto com atividades atribuídas à implementação de controles de segurança, organizações devem direcionar esforços correspondentes para obter conformidade. A conformidade é o esforço empregado para seguir orientações de maneira adequada para mitigar qualquer risco de segurança potencialmente nocivo à organização [AIE22]. Dessa maneira, organizações empenhadas no uso de controles e recomendações de segurança devem ter ciência dos meios para lidar com questões de conformidade. Segundo Zelmati et al. [ZOE23], em situações que envolvem certificação a normas como a ISO/IEC 27001, o rigoroso processo de conformidade é repleto de desafios. Auditores lidam com constantes dificuldades devido a atualizações de requisitos regulatórios, acompanhamento em tempo real e gestão de documentos e relatórios [ZOE23].

A importância da conformidade em TI estende-se a casos em que uma certificação é exigida por lei, contratos ou padrão industrial, como os seguinte exemplos: *Payment Card Industry* (PCI, em tradução, Indústria de Cartões de Pagamento) *Data Security Standard* (DSS, em tradução, Padrão de Segurança de Dados), em que a *Visa International Service Association* (VISA, em tradução, Associação de Serviços Internacionais Visa) exige o uso de software compatíveis com PCI DSS por empresas que processam transações de cartão de crédito ou o governo dos EUA que exigem certificação com a FedRAMP a seus fornecedores de nuvem [KSTE20].

Atualmente, o interesse em eficiência é estimado em todas as áreas ligadas a TI. Dessa maneira, a automação é considerada em diversos processos para torná-los mais eficientes, isso é naturalmente estendido para a conformidade que está sendo constantemente estudada e desenvolvida [FAH+18].

Existem diferentes alternativas que tratam de automação no contexto de conformidade.

Uma delas é a conformidade contínua, tida como um processo automatizado que objetiva manter ativos de TI em conformidade com políticas corporativas, estruturas regulatórias ou outras práticas que lidam com recomendações de segurança [FAH+18]. Além da conformidade contínua, termos como conformidade em tempo real, conformidade orientada a eventos e conformidade como código são facilmente encontrados nesse contexto. Como fator que estimula a conformidade contínua, a postura que trata de auditorias manuais está relacionada a problemas com custos, julgamento e amostragem. Os custos se relacionam à mão de obra para execução de tarefas, o julgamento considera eventual ocorrência de erro por motivação humana no processo, enquanto a amostragem, esta é relacionada à execução de verificações em amostras aleatórias evitando análise integral em procedimentos manuais e custosos [KSTE20].

Critérios técnicos para prestar suporte ao gerencimento de conformidade podem ser obtidos através do uso de ferramentas. Para Filepp et al. [FAH+18], a conformidade é vista com frequência como a outra face do gerenciamento de segurança, do qual o gerenciamento de segurança envolve *design* de sistema para fortalecimento da segurança, enquanto o gerenciamento de conformidade envolve verificação e reparo de configurações de segurança. Neste cenário, as ferramentas de segurança se aplicam a verificações em vulnerabilidades e outros aspectos do sistema, implementando condições necessárias para a conformidade com recomendações de segurança e contribuindo para sistemas cada vez mais robustos e imunes à vulnerabilidade [FAH+18].

3.2.2 Procedimentos do gerenciamento de vulnerabilidades

As vulnerabilidades são um grande fator de risco para a segurança. Embora sua exploração ocasione sérios problemas de comprometimento dependendo de fatores como impacto, severidade e risco, vulnerabilidade é o elemento mais controlável. Isto é tido pelo fato de que é possível a identificação e resposta quando esforços são aplicados no desenvolvimento de processos eficazes de gestão de vulnerabilidades [BK19]. O problema com vulnerabilidades afeta inúmeras áreas da TI. Em computação na nuvem, Modi et al. [MPB+13] esclarece que as vulnerabilidades são lacunas na arquitetura de segurança que podem ser exploradas por agentes maliciosos através de técnicas sofisticadas para acesso à rede e recursos de infraestrutura. Ações desse tipo afetam a disponibilidade e até mesmo os benefícios econômicos da

computação em nuvem [MPB+13].

Inúmeros fatores podem levar a problemas com vulnerabilidades. Em estudo realizado por Hamadi [Ham19], vulnerabilidades podem estar vinculadas a fraquezas de políticas de segurança, fraqueza de tecnologia e fraqueza de configuração. Recomendações de segurança de órgãos como CIS e NIST, evidenciam a relevância em compor um processo para mapear vulnerabilidades, a existência de ferramentas para esse propósito, auxilia especialistas na descoberta. As verificações executadas por ferramentas de segurança incluem descobertas em vulnerabilidades de design, de implementação e operacionais. A título de exemplo, as vulnerabilidades de *designs* são relacionadas a fraquezas nas especificações de software, vulnerabilidades de implementação consideram problemas decorrentes de *bug* no software e as vulnerabilidades operacionais são resultantes de configuração inadequada de softwares [Ham19].

O processo de gerenciamento de vulnerabilidades é estabelecido pelas definições dos controles de segurança de órgãos reguladores e padronizadores. Desse modo, é possível cumprir ações necessárias para compor corretamente o processo. Alguns exemplos são encontrados no ITU-T X.1605 [Int20] reportando a necessidade de esforços para estabelecer mecanismos de avaliação de vulnerabilidades, formulação de linha base de configuração de segurança, execução regular de verificação e geração de relatórios de avaliação. Em modo geral, controles de segurança como estes definem em alto nível como é constituído o processo de gerenciamento de vulnerabilidades [Int20].

Para atender a necessidade de cada organização, a flexibilidade na frequência de verificação, amplitude e profundidade é definida por cada organização. Desse modo, estabelecer esses parâmetros depende de fatores como a criticidade dos sistema. O NIST trata de questões da cobertura de vulnerabilidades nos documentos NIST *Special Publication* SP 800-53 *Revision* 5 [Nat20] e NIST *Special Publication* 800-53A *Revision* 4 [Nat14], que definem amplitude em relação a componentes dentro do sistema, tipos de sistema, criticidade dos sistemas ou número de vulnerabilidades verificadas. A profundidade da cobertura de verificação está relacionada ao nível de *design* do sistema a ser monitorado por uma determinada organização. Itens como componentes, módulos, subsistemas, bibliotecas, código, entre outros, são fatores determinantes para o nível de profundidade da verificação [Nat20].

As vulnerabilidades são um problema para a segurança que relacionam-se com outras

dimensões de comprometimento dos princípios de segurança de um sistema, como ameaças e ataques. As ameaças se referem a qualquer elemento com potencial de causar danos e comprometer a segurança, os ataques são as ações intencionais que objetivam causar danos à segurança [Ham19]. Entender a dificuldade relacionada a uma vulnerabilidade é fator decisivo no processo de priorização de soluções, bem como na compreensão dos níveis de risco [SIA23]. Bases como o CVE são amplamente utilizadas na comunidade de segurança. O objetivo é padronizar a comunicação sobre vulnerabilidades, além de conter informações sobre impactos potenciais, medidas de mitigação e correção disponíveis.

Frequentemente, os registros CVE recebem avaliação de severidade de acordo com o padrão CVSS. Este é um sistema que prevê uma representação numérica entre 0 e 10 para a criticidade de uma vulnerabilidade. Desse modo, as métricas são grupos que definem pontuação básica, temporal e ambiental, produzindo números que se convertem em classificações qualitativas, como baixa, média, alta ou crítica para auxiliar priorização e esforços necessários de remediação.

Exemplificando a importância e uso do CVE e padrão CVSS, um estudo realizado no projeto de Estudos sobre Segurança Cibernética em Rede 5G - TED ANATEL/UFCG evidenciou a necessidade de conhecimento sobre vulnerabilidades de segurança no OpenStack[12], uma plataforma de código aberto que faz uso de recursos virtualizados para gerenciar nuvens públicas e privadas. Isso resultou em um processo de análise das informações da base de dados do CVE[13] sobre o OpenStack. Os detalhes da análise são encontrados no Apêndice B.

Ferramentas de segurança que realizam scanner de vulnerabilidades utilizam as vulnerabilidades conhecidas em bases de inteligência sobre vulnerabilidades como o CVE. Dessa forma, identificam intercorrências tanto relacionadas a vulnerabilidades como configurações de segurança inadequadas. Essa tarefa é realizada automaticamente, eliminando a necessidade das organizações acessarem e baixarem manualmente bases de dados de vulnerabilidades.

[12] `https://www.redhat.com/pt-br/topics/openstack`

[13] `https://www.cve.org/`

3.2.3 Segurança na cadeia de suprimentos

Para o alcance da segurança em softwares, duas direções podem ser tomadas. A primeira delas é o desenvolvimento seguro, em seguida a proteção contra vulnerabilidades de software [DD21]. O desenvolvimento seguro, fortemente apoiado pela implementação de DevSecOps, como descrito na Seção 3.1.4, é seguido por preocupações com a cadeia de suprimentos de software, termo que ganhou popularidade após a divulgação de departamentos de cibersegurança dos Estados Unidos, sobre software como um produto de várias etapas semelhante a cadeia de suprimentos de produtos físicos [The21].

O rápido desenvolvimento demandado pelo mercado exige que softwares sejam desenvolvidos continuamente no menor tempo possível para enfrentar os concorrentes. Por consequência, os desenvolvedores criam vulnerabilidades de segurança na programação ou usam módulos ou componentes de terceiros que são vulneráveis [THH+20]. Nos últimos anos, os ataques à cadeia de suprimentos cresceram ao redor do mundo chegando ao acumulado de 742% [Son21]. Ataques à cadeia de suprimentos objetivam injetar vulnerabilidades ou *malware* em produtos que consequentemente podem ser explorados quando forem distribuídos ou executados. Exemplificando, o uso de softwares de código aberto que carregam vulnerabilidades, quando usados em projetos de código fechado, acabam injetando vulnerabilidades nesses projetos [Son21].

Estratégias começaram a ser desenvolvidas para garantir a proteção da cadeia de suprimentos, como o *Framework Supply-chain Levels for Software Artifacts* (SLSA[14], em tradução, *Framework* de Níveis de Cadeia de Suprimentos para Artefatos de Software) que propõem métricas guia para integridade e proveniência dos artefatos de software e o SBOM que contém um inventário de todas as dependências diretas quando usadas pelo código do produto e transitivas quando usadas pelas dependências do produto. Um SBOM trata-se de uma lista de todos os componentes de software que identifica informações sobre os componentes e relacionamentos da cadeia de suprimentos entre eles [SCW+23].

No contexto de ferramentas de segurança, algumas ferramentas podem adotar esse padrão para gerar SBOM como saída de sua execução, ou consumir SBOM para realizar scanner de vulnerabilidades da lista de materiais em busca de identificação de vulnerabilidades conhecidas.

[14] `https://slsa.dev/`

3.3 Especificações técnicas das ferramentas de segurança

3.3.1 Autoridades de relevância no fornecimento de ferramentas de segurança

Na área de segurança da informação, além de organizações internacionais que definem regras ou padrões de segurança dispostos em documentos que provisionam controles de segurança, existem organizações que dispõem da manutenção, desenvolvimento e provimento da área de segurança com fomento de ferramentas. O emprego de ferramentas de segurança quando feito apropriadamente, garante que importantes questões de segurança sejam tratadas utilizando a tecnologia adequada para solução de problemas. As ferramentas de segurança estão relacionadas à segurança da informação de diferentes maneiras, além de proporcionar meios técnicos e tecnológicos para implementações de segurança. Cheng et al. [CVCLC18] definem ferramentas relacionadas à segurança em aspectos como gestão de conformidade e gestão de segurança presentes em sistemas de governança e gerenciamento de risco.

Ao considerar o emprego de ferramentas nas soluções de segurança, organizações como *Open Worldwide Application Security Project* (OWASP, em tradução, Projeto Aberto de Segurança de Aplicações Mundial), Kali Linux e *Cloud Native Computing Foundation* (CNCF, em tradução, Fundação para Computação Nativa em Nuvem) sumarizam informações sobre ferramentas, tal como utilização para fortalecer a postura de segurança em sistemas de informação. As três organizações citadas possuem reconhecimento internacional e dispõem de alternativas de segurança através da utilização de ferramentas. Cada uma possui propósitos específicos em aplicações distintas, porém segurança é um atributo em comum.

A OWASP foi fundada em 1° de dezembro de 2001, objetiva fortalecer a segurança dos softwares desenvolvidos, além de ser uma organização sem fins lucrativos, dispõe de diversos projetos de software de código aberto, milhares de membros e conferências educacionais. A comunidade aberta foca na compreensão, desenvolvimento e obtenção de sistemas confiáveis. Todas as ferramentas, documentos e fóruns OWASP são gratuitos e disponíveis para interessados em aprender sobre segurança [ASMK20]. Dentre os projetos da OWASP, pode-se destacar o *Secure Code Box*[15] que reúne ferramentas utilizadas para testes de segurança e o

[15] `https://owasp.org/www-project-securecodebox/`

Ten Web Application Security Risks [16] com consenso sobre riscos de segurança em aplicações *web*.

O projeto *Secure Code Box*, inclui diversas ferramentas de segurança conhecidas e amplamente utilizadas por profissionais de segurança, o projeto permite que vulnerabilidades em rede e nos aplicativos sejam descobertas. Em Modi et al. [MPB+13], é abordado o problema de segurança em nível de rede, que é a espinha dorsal de computação na nuvem. Por serem ferramentas presentes no projeto desenvolvido por organizações de relevância internacional como a OWASP, o conjunto de ferramentas presente serve de base para pesquisa por ferramentas de segurança amplamente utilizadas e relevantes. O projeto *Ten Web Application Security* auxilia o direcionamento de esforços de segurança em conjunto com a utilização de alternativas como o *Secure Code Box*. [HFF+20].

O Kali Linux é uma plataforma amplamente utilizada na área de segurança da informação. Esta alternativa disponível para aplicação nas questões de segurança possibilita sua utilização em diversas tarefas da segurança da informação. É utilizada por *hackers* e profissionais de segurança como distribuição Linux que exige conhecimento profundo sobre como o SO funciona e como utilizar as ferramentas disponíveis [Ham19]. Dentre as diversas alternativas de ferramentas disponíveis no Kali Linux, como *kali-tools-reverse-engineering, kali-tools-passwords, kali-tools-fuzzing,* entre outras, destaca-se o *kali-tools-vulnerability*. Cada um desses exemplos citados são dependências que reúnem ferramentas em determinados contextos. Nosso foco é direcionado para o pacote de ferramentas de vulnerabilidades, mais detalhes sobre vulnerabilidades estão descritos na Seção 3.2.2.

O Kali Linux, por meio do pacote *kali-tools-vulnerability*, disponibiliza uma lista relevante de ferramentas de segurança para esta finalidade. Para Hamadi [Ham19], o gerenciamento de vulnerabilidades se divide em vulnerabilidades de design, de implementação e operacionais. Ele acrescenta que a diversidade de ferramentas utilizadas no Kali Linux auxilia na descoberta das vulnerabilidades mapeadas nos diferentes aspectos mencionados. Considerar o *kali-tools-vulnerability* como fonte que compõe uma lista relevante de ferramentas de segurança é benéfico para interessados em utilizar ferramentas para compor um processo de gerenciamento de vulnerabilidades.

Com grande influência para o desenvolvimento da computação na nuvem, CNCF é uma

[16] `https://owasp.org/www-project-top-ten/`

importante organização com reconhecimento internacional, seu propósito é fomentar o crescimento e evolução do ecossistema de computação na nuvem. Sendo uma organização sem fins lucrativos, a CNCF mantém diversos projetos de código aberto. Seus esforços estão relacionados com importantes projetos, como Kubernetes, Prometheus e Envoy. O grande envolvimento da organização com desenvolvimento e apoio em projetos relacionados a soluções para computação na nuvem, levou a CNCF ao desenvolvimento do Landscape[17].

Pensado para entregar uma experiência visual, o Landscape auxilia na descoberta de soluções relacionadas à computação na nuvem. Isso é dado pela facilidade de uso e grande número de informações disponíveis, tornando-o o Landscape eficiente para descoberta de soluções. A CNCF consegue fornecer um agrupamento de ferramentas e serviços recomendados e atualizados. O popular interesse por computação na nuvem torna necessária a proposta do Landscape no auxílio a profissionais como desenvolvedores e engenheiros, no desenvolvimento que envolve computação na nuvem. A CNCF, bem como a OWASP e Kali Linux, servem de importante base para a descoberta de ferramentas para compor soluções de segurança em sistemas de informação [RDP22].

3.3.2 Atributos das ferramentas de segurança

As ferramentas de segurança são programas de computador, que executam papel fundamental na área de segurança da informação. A realização de operações por meio de ferramentas de segurança se relaciona a muitos conceitos que precisam ser compreendidos por quem as utiliza, sejam organizações ou profissionais de segurança. Para entendimento inicial da ampla aplicabilidade das ferramentas de segurança, alguns termos ajudam no agrupamento das ferramentas em contextos, que auxilia a compreensão e implementação. Alguns termos são classificados por Dimov et al. [DD21] em duas categorias, ferramentas de verificação de segurança e ferramentas de proteção de tempo de execução. Sendo assim, as ferramentas de verificação de segurança incluem termos como, *Static Application Security Testing* (SAST, em tradução, Teste de Segurança de Aplicações Estáticas), *Dynamic Application Security Testing* (DAST, em tradução, Teste de Segurança de Aplicações Dinâmicas), *Interactive Application Security Testing* (IAST, em tradução, Teste de Segurança de Aplicações Interativo) e *Software Composition Analysis* (SCA, em tradução, Análise de Composição de Software).

[17] `https://landscape.cncf.io/`

Enquanto a proteção em tempo de execução temos *Web Application Firewalls* (WAF, em tradução, Firewalls de Aplicações Web) como exemplo. Além do que é apresentado por Dimov et al. alguns termos relacionados a ferramentas de segurança são apresentados no Apêndice A.

As ferramentas SAST, DAST e IAST são aplicadas em testes de segurança de aplicações, mas variam em sua implementação. Estes testes consideram o rápido desenvolvimento e competitividade entre organizações, isto faz com que desenvolvedores gerem vulnerabilidades ou utilizem componentes de terceiros contendo vulnerabilidades. Segundo Tudela et al. [THH+20], as ferramentas SAST realizam a análise de segurança de caixa branca. O código fonte e os executáveis são analisados, em seguida considerações sobre a segurança do programa podem ser determinadas. A finalidade é considerar todo o programa, bem como mapear toda a superfície de ataque. As ferramentas também devem proporcionar suporte para verificação de várias linguagens de programação, bem como prover a detecção de falsos positivos [THH+20].

As ferramentas DAST classificam-se nos testes de caixa preta permitindo que verificações sejam realizadas ao programa em uso. Após a realização do mapeamento da superfície de ataque, as ferramentas DAST procedem analisando o comportamento do sistema. A título de exemplo, considerando sistemas *web*, são realizadas entradas maliciosas de dados nas entradas de fontes externas. Análises de vulnerabilidade relacionadas às portas utilizadas pela aplicação também podem ser executadas. A diferença com relação às ferramentas SAST é que nesse caso o código do sistema não é conhecido. Para as ferramentas IAST, são considerados os elementos tanto para verificação estática, quanto dinâmica. As ferramentas irão considerar a verificação de código-fonte na descoberta de vulnerabilidades, assim como testes realizados durante a execução da aplicação [THH+20].

O SCA, relaciona-se à segurança da cadeia de suprimentos, como descrito na Seção 3.3.3. Tem como finalidade a análise e gerenciamento de implementações de código aberto. O inventário de componentes de software é fundamental para mapear todos os componentes, versões de uso, licenças e vulnerabilidades conhecidas. As ferramentas SCA desempenham papel fundamental na capacidade de detecção de vulnerabilidades em componentes de código aberto utilizados nos softwares. Esta prática fortalece a segurança dos sistemas que utilizam diversas bibliotecas e dependências de terceiros em seus projetos [ITW21].

Nas ferramentas WAF o foco é na proteção de tempo de execução. Usado diretamente em aplicações *web*, fornece um escudo de segurança em aplicações que utilizam o protocolo HTTP. Diferente da utilização das ferramentas citadas anteriormente, o WAF irá atuar diretamente como um componente de segurança para a aplicação. Disposto na rede, WAF define uma barreira entre a rede interna e externa. Além de prevenir ameaças e invasores, essas ferramentas irão filtrar dados de entrada, saída e interromper o tráfego quando o considerar potencialmente perigoso. Esta consideração irá de acordo com regras previamente definidas para compor as políticas de execução [MBHR20].

Pensado para fortalecer os procedimentos de automação de segurança, o NIST mantém o *Security Content Automation Protocol* (SCAP[18], em tradução, Protocolo de Automação de Conteúdo de Segurança), uma síntese de especificações em constante desenvolvimento pela comunidade. Seguindo especificações relevantes da agenda de automação de segurança do NIST, SCAP possibilita padronizar formato e nomenclatura comuns trazendo benefícios à segurança pelo uso de informações que podem ser legíveis tanto a máquinas, quanto humanos [LAT]. O CIS[19] destaca o poder do SCAP em contribuir para que organizações cumpram mais facilmente com a conformidade.

As especificações contribuem para conformidade com políticas de segurança que envolvem o gerenciamento de vulnerabilidades, além de permitir que configurações de sistemas sejam comparadas com diretrizes de segurança populares em todo o mundo. Alguns dos benefícios do SCAP incluem lidar com problemas de gerenciamento de vulnerabilidades permitindo a manutenção da segurança por meio de atualizações ou reconfiguração de sistemas e a grande quantidade de ferramentas de segurança que utilizam conteúdo e terminologia proprietários adicionando incompatibilidade entre si [LAT].

As especificações SCAP contemplam diversos aspectos da segurança, e podem ser classificadas em 5 categorias, são elas:

- Línguas que definem um padrão para expressar políticas, como *Extensible Configuration Checklist Description Format* (XCCDF, em tradução, Formato Extensível de Descrição de Lista de Verificação de Configuração), *Open Vulnerability and Assessment*

[18] `https://csrc.nist.gov/projects/security-content-automation-protocol`

[19] `https://www.cisecurity.org/insights/blog/secure-configurations-and-the-power-of-scap`

Language (OVAL, em tradução, Linguagem Aberta de Vulnerabilidade e Avaliação) e *Open Check List Interactive Language* (OCIL, em tradução, Linguagem Interativa de Lista de Verificação Aberta);

- Formatos de relatórios para reportar informações, como *Asset Reporting Format* (ARF, em tradução, Formato de Relatório de Ativos), desse modo uma saída padrão pode ser estabelecida após a execução de uma ferramenta;

- Esquemas de identificação para conceituar produtos de software, vulnerabilidades e configurações, como o *Common Platform Enumeration* (CPE, em tradução, Enumeração Comum de Plataformas), *Software Identification* (SWID, em tradução, Identificação de Software), *Common Configuration Enumeration* (CCE, em tradução, Enumeração Comum de Configuração) e CVE;

- Sistemas de medição como o CVSS, dessa forma é possível definir a severidade de uma vulnerabilidade usando um padrão definido;

- Integridade para contribuir com a manutenção de um sistema íntegro, como *Trust Model for Security Automation Data* (TMSAD, em tradução, Modelo de Confiança para Dados de Automação de Segurança).

Em situações específicas, a execução de ferramentas requer o uso de artefatos para que um processo seja iniciado. Isso envolve fatores como a realização de consultas a informações externas, arquivos de dados, consumo de regras estabelecidas por políticas, scripts para execução de tarefas, entre outros. A maneira como os artefatos são consumidos varia de acordo com a ferramenta utilizada, porém existem alguns padrões comuns para uma ou mais ferramentas. Além de especificações como OVAL e XCCDF, temos padrões usados por ferramentas como Nmap, OpenSCAP e Lynis, são eles:

- **NSE:** O *Nmap Scripting Engine* (NSE)[20], são códigos desenvolvidos para execução junto a ferramenta Nmap. Escritos em Lua, os scripts são continuamente atualizados e permitem verificação de uma série de vulnerabilidades conhecidas, além de automatizar uma ampla variedade de tarefas de rede [LZZ+20][LCFX21];

[20] `https://nmap.org/book/man-nse.html`

- **SSG:** O SCAP *Security Guide* (SSG)[21], reúne uma série de conteúdos SCAP usados para verificações de segurança em diversos SOs. O projeto iniciou com colaborações de agências dos Estados Unidos, atualmente reúne esforços da comunidade e diversos fornecedores de sistemas operacionais, o projeto compõe fluxos de dados SCAP.

- **Rego:** o Rego[22] é uma linguagem de consulta desenvolvida para fácil leitura e escrita. Permite extensões que oferecem suporte a modelos de documentos estruturados, como JSON (*JavaScript Object Notation*.) Usado em conjunto com o *Open Policy Agent*, um mecanismo de política, Rego é uma linguagem declarativa que permite especificação de políticas como código.

A tendência de meios de automação e interoperabilidade entre ferramentas, leva naturalmente a adoção de especificações padrão para que a comunicação entre diferentes ferramentas de segurança seja possível. Atualmente existem ferramentas que implementam especificações SCAP, embora o processo ainda demande por tempo para que cada vez mais ferramentas de segurança implementem essa padronização.

3.3.3 Considerações de usabilidade sem intervenção humana

A busca por eficiência em TI está tornando a automação cada vez mais comum em diversos processos [FAH+18]. Esse fator, aliado a possíveis erros causados por intervenções humanas, leva ao desenvolvimento e discussão de métodos de implementação de automação de forma padrão para alguns processos comuns e amplamente utilizados. Grandes exemplos de esforços envolvendo automação são o SOAR e SCAP descritos na Seção 3.3.2. O SOAR propõe a definição do limite da intervenção humana passando a tratar questões de automação para realização de tarefas e execução de processos. Enquanto o SCAP é parte da agenda de automação do NIST para tratar questões de segurança utilizando especificações padronizadas potencialmente usadas para definir comunicação comum entre diferentes ferramentas de segurança.

Para tratar de questões de segurança, o fator humano na realização de tarefas deve ser considerado. Embora as capacidades humanas possibilitem inovação e resolução de proble-

[21] `https://github.com/ComplianceAsCode/content`
[22] `https://www.openpolicyagent.org/docs/latest/policy-language/`

mas, o nível de acesso humano ao sistema permite que funcionários ou clientes com acesso administrador possam ser fator crítico para beneficiar ou danificar a segurança de um sistema [TKR20]. Outros elementos ligados à interação humana nos processos envolve o erro humano como a principal ameaça para a segurança devido a negligência, ignorância e falha no cumprimento de políticas de segurança [BASY19].

A automação implementada em processos que envolvem o uso de ferramentas de segurança auxilia no cumprimento de controles de segurança e são fundamentais para a resposta a ameaças de segurança que se materializam em um ritmo acelerado [DEM17]. A alta velocidade em que a quantidade de ameaças de segurança está disposta, somadas a capacidades insuficientes de profissionais em relação à segurança, ou mesmo com profissionais com alta experiência em segurança, não conseguem acompanhar a demanda por soluções. Em cenários como este, tarefas comuns passam a ser automatizadas para dar suporte ao trabalho humano [SV21]. Desse modo, soluções automatizadas usando ferramentas de segurança, processos que demandam eficiência e diminuição de intervenção humana para reduzir o acometimento de erros, se estendem a tarefas de conformidade e dão suporte a métodos como a conformidade contínua que somam esforços mediante automação para manter ativos de TI em conformidade com recomendações de segurança.

Capítulo 4

Designação dos recursos base para conformidade

A implementação de medidas de segurança demanda a definição de uma série de atributos com fundamentação e suporte suficientemente alinhados à realidade de uma organização. Dessa forma, a escolha dos controles de segurança é influenciada por diversos fatores que compreendem desde a qualidade das soluções empregadas, até a conformidade com a legislação aplicada no país. Neste capítulo, tratamos dos recursos para conformidade, compostos por métodos para identificar recomendações de segurança, viabilizar a implementação dessas recomendações com o uso de ferramentas de código aberto e determinar como conduzir as soluções para um escopo de interesse na área de segurança. Como resultado, formalizamos os meios necessários para que as organizações possam realizar implementações de segurança seguindo recomendações.

4.1 Concepção base de segurança da informação

A base da segurança da informação deve ser provisionada por recomendações relevantes da área de segurança. Um bom nível de relevância pode ser atingido por meio de controles e recomendações de segurança de órgãos reconhecidos internacionalmente. Neste sentido, a segurança da informação é abordada em diversos documentos como, guias, normas, padrões, manuais, etc. Estes materiais formam a base para políticas de segurança adotadas por empresas ou entidades, isto conduz a implementação de métodos para fortalecimento da postura de

segurança perante os diversos desafios existentes, bem como constitui uma conduta segura com incentivo à conformidade.

Mediante a grande oferta de soluções providas para o ecossistema de segurança, torna-se difícil e complexo estabelecer um processo para avaliar qual dentre as possíveis alternativas é mais benéfica para uma determinada organização, quando considerada a implementação de segurança em conformidade com recomendações relevantes. Fatores como custos, contexto e abrangência são apenas algumas das variáveis relacionadas ao processo de apuração.

Para estabelecer a base de segurança da informação utilizada neste estudo, algumas considerações foram inicialmente definidas com o intuito de identificar propósitos comuns entre as alternativas. Em síntese, a detecção de alternativas direcionadas à segurança da informação, especificamente em computação na nuvem, foi observada nas propostas da ISO nos documentos ISO/*International Electrotechnical Commission* (IEC, em tradução, Comissão Eletrotécnica Internacional) 27017, ISO/IEC 27018 e CSA.

Embora todas essas alternativas sejam voltadas para computação na nuvem, a ISO/IEC 27017 e a ISO/IEC 27018 exigem normatização e implementação de medidas de segurança específicas, enquanto a CSA provisiona capacitação e treinamentos para ajuste às condições adequadas de segurança. Tanto a ISO quanto a CSA são exemplos de organizações de relevância internacional que dispõem de aplicação em contexto de computação na nuvem. Além desses órgãos, foram identificados *frameworks* de outras entidades, como o CIS, NIST e também o *framework* COBIT. Eles sumarizam conjuntos de recomendações de segurança de diversos órgãos em um conjunto conciso de controles e recomendações de segurança.

Os *frameworks* de segurança são formados por controles e recomendações de segurança da informação. Dessa forma, mapeiam inúmeros documentos de segurança que estão relacionados a suas recomendações propostas. Sendo assim, ao cumprir uma recomendação de segurança disponível em um *framework*, a conformidade se estende para vários outros documentos mapeados. Considerando o potencial de abrangência, o uso de um *framework* é escalável se comparado a alternativas como ISO e CSA. Em pesquisa por órgãos de segurança realizada na Internet, foram identificados, além dos já citados, órgãos que tratam de segurança da informação e segurança em computação na nuvem. O processo de compreensão de cada alternativa é custoso e recorrente em cada um dos órgãos, do contrário, os *frameworks* de segurança são um ponto de partida que unificam e integram as recomendações

de vários órgãos de segurança.

Tomando como exemplo, considerando os mapeamentos realizados pelo CIS *Controls* V8, pode-se constatar a abrangência dos documentos considerados pelos *frameworks* de segurança. Alguns dos documentos mapeados e relacionados aos órgãos identificados em pesquisa na internet foram: CISA *Cybersecurity Performance Goals*, CSA *Cloud Controls Matrix* v4, ISO/IEC 27001:2002, NIST *Cybersecurity framework*, NIST SP 800-53 *Revision 5 Moderate Baseline*, MITRE *Enterprise* ATT&CK v8.2, NIST SP 800-171, CISA *Cross-sector Cybersecurity Performance Goals*, ISACA COBIT 19 e NIST SP 800-53 *Revision 5 Low Baseline*.

Com base nas considerações apresentadas sobre órgãos de segurança, a alternativa acatada no presente estudo utiliza *frameworks* de segurança para compor a base de segurança da informação. Estes fornecem recomendações de segurança abrangentes e alinhadas com diversos órgãos relevantes de segurança da informação. O decreto E-ciber do governo federal[1] enfatiza que devem ser adotadas normas de órgãos internacionais, como a ISO, COBIT, NIST e CIS. Os *frameworks* considerados foram CIS Controls na versão 8, NIST *Security Framework* e COBIT 5, o que exclui a ISO como uma das alternativas para a utilização por não ser um *framework* como os demais.

Em análise inicial sobre o propósito de cada um dos *frameworks* e órgãos identificados e detalhados na Seção 3.3.1, CIS e NIST se demonstraram alternativas prioritariamente voltadas para segurança da informação, enquanto COBIT 5 é um *framework* utilizado em governança e gestão de TI, não sendo aplicado especificamente para segurança da informação. O *framework* COBIT 5 foi então descartado para utilização na base de segurança. As alternativas selecionadas para compor a base de segurança da informação foram CIS *Contols* V8 e NIST *Cybersecurity Framework*. A alternativa do CIS foi desenvolvida para ser simples e abranger organizações de grande e pequeno porte, enquanto o *framework* do NIST é direcionado para segurança de infraestruturas críticas.

As recomendações de segurança do CIS *Controls* V8 e NIST *Security Framework* direcionam a um escopo geral de segurança da informação. Desta forma, estão contidas nas documentações, recomendações que são utilizadas e também explicadas em ambientes de

[1] `https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2020/decreto/d10222.htm`

computação na nuvem. Para contextualizar as recomendações ajustadas à computação na nuvem, realizou-se a classificação das recomendações de segurança, utilizando-se os componentes de gerenciamento de segurança para IaaS do ITU-T X.1605. Os componentes de gerenciamento de segurança como discutidos na Seção 3.2.1 são: Gerência de identidade e controle de acesso; Gerência de vulnerabilidades; Auditoria de segurança; Resposta de emergência, Recuperação de desastres e *Backup*.

O motivo pelo qual um procedimento de classificação é relevante envolve a necessidade de categorizar os controles do CIS *Controls* V8 e do NIST *Cybersecurity Framework* relevantes para computação na nuvem. Para a classificação de recomendações de segurança, foram utilizados os componentes de gerenciamento de segurança disponíveis no documento ITU-T X.1605, que abrangem todas as instalações de segurança necessárias para suportar os serviços em nuvem. Alguns passos de trabalho seguem ordenados para obtenção da classificação, como:

- Mapear os controles do CIS *Controls* V8 nos componentes de gerenciamento de segurança do ITU-T X.1605;
- Incluir componentes de segurança do ITU atribuídos às recomendações de segurança do CIS *Controls* V8 juntamente com as recomendações do NIST *Security Framework*;
- Análise de possíveis recomendações de segurança não classificadas;
- Considerar recomendações não classificadas, porém relevantes para segurança e compor uma categoria específica para esse propósito.

Obtém-se como produto da etapa de classificação, uma lista de componentes de gerenciamento de segurança orientados à computação na nuvem. Temos cada componente de gerenciamento proposto pelo ITU sendo populado com recomendações de segurança do CIS *Controls* V8 e cada recomendação do NIST *Security Framework* sendo implementada de forma complementar quando pertinente. Assim, consideramos a diferença entre os grupos de recomendação e desconsideramos a intersecção entre recomendações. Este é um significativo passo na direção de estabelecer um escopo de recomendações de segurança fortemente orientado por órgãos de segurança comprovadamente relevantes em nível mundial. Além

disso, aborda segurança na infraestrutura necessária para organizações com interesse em serviços na nuvem e direciona os esforços necessários para conformidade requeridos por leis como a LGPD.

4.2 Processo de identificação das ferramentas de segurança

A identificação de ferramentas de segurança tem como objetivo propor o uso de ferramentas de código aberto bem consolidadas para atender a implementação técnica de recomendações de segurança. Portanto, foram necessários meios para compor um processo de identificação. Foram elaborados métodos para atender duas etapas importantes para proposta de implementação técnica. Na primeira etapa detalhada no Apêndice C, são realizadas buscas para identificação das ferramentas de segurança. A atribuição das ferramentas identificadas às recomendações precisa ser feita adequadamente este trabalho é realizado na segunda etapa. O processo deve contemplar a compreensão de recomendações de segurança e alinhar ferramentas que atuem para o cumprimento efetivo provendo conformidade com as recomendações.

Os procedimentos descritos para a segunda etapa da metodologia visam alcançar alguns objetivos importantes para viabilizar a efetivação de conformidade. Tem-se como objetivos, utilizar as ferramentas de segurança identificadas com base no documento CIS *Controls* V8, considerar as recomendações de segurança dos documentos CIS *Controls* V8 e NIST *Cybersecurity Framework* e realizar a atribuição das ferramentas às recomendações de segurança.

Ferramentas de segurança identificadas no processo de busca por ferramentas de segurança do Apêndice C, devem ser vinculadas às recomendações de segurança do CIS *Controls* V8. O processo de identificação das ferramentas considerou todas as recomendações do documento de segurança do CIS, a lista de ferramentas resultante não tem vínculo com nenhuma recomendação de segurança de forma explícita. Sendo assim, estabelecemos um processo para vinculação de cada ferramenta identificada a uma ou mais recomendações de segurança. O processo de atribuição requer a execução de tarefas de análise executadas por etapas, além de relacionar um processo manual de leitura e compreensão. As etapas contemplam as tarefas (T1) listadas abaixo:

- **T1:** Análise do propósito da ferramenta de segurança;

- **T2:** Identificação de princípios funcionais da ferramenta de segurança;
- **T3:** Análise de desígnio da recomendação de segurança.

O levantamento de informações pertinentes de ferramentas de segurança requer fontes confiáveis de informação, a análise deve considerar apenas informações presentes em páginas oficiais e documentação oficial da ferramenta de segurança. Abaixo seguem detalhes dos critérios de aceitação CA e descarte CD.

Critérios de aceitação:

- **CA1:** Ferramenta de segurança;
- **CA2:** Ferramenta de código aberto;
- **CA3:** Suporte ativo pela comunidade de desenvolvimento;
- **CA4:** Documentação disponível.

Critérios de descarte:

- **CD1:** Não é de código aberto. Ferramenta disponível somente em versões *premium*;
- **CD2:** Não é uma ferramenta de segurança que se estabeleça para o escopo desejado. A ferramenta não se classifica nas categorizações apresentadas na Seção 3.4.2 ou está fora do escopo dos componentes de gerenciamento de segurança do ITU-T X.1605;
- **CD3:** A ferramenta encontra-se sem suporte ativo por parte da comunidade de desenvolvedores;
- **CD4:** Ausência de documentação disponível para consulta;
- **CD5:** Ferramenta obsoleta e sem atualizações recentes;
- **CD6:** Baixa popularidade dentro da comunidade de usuários. O valor para definição de baixa popularidade considera a soma de estrelas e *forks* do repositório menor que 1.068 como definido na Tabela 4.1;
- **CD7:** Data do último *fork* indicando inatividade no repositório. O valor para definição da data limite considera o período de 3.6 anos;

- **CD8:** Conflitos de compatibilidade com as recomendações de segurança;
- **CD9:** Impossibilidade de localizar o repositório da ferramenta.

Para os critérios de descarte, baixa popularidade e data do último *fork* são valores de referência que precisam ser definidos. Sendo assim, os valores foram extraídos de amostras da lista de ferramentas da OWASP. Para análise e obtenção dos valores, utilizou-se as ferramentas Ncrack, Nmap e ThreatDragon como amostra. A seleção das ferramentas segue os critérios de tempo de desenvolvimento e popularidade. A popularidade é definida pela soma do número de estrelas e número de *forks* no repositório do GitHub. Vale lembrar que esse valor é atualizado no decorrer das interações dos usuários com o repositório.

A popularidade das ferramentas no momento da análise tem os seguintes valores como detalhado na Tabela 4.1. Ncrack tem popularidade 1.233, Nmap tem popularidade 10.934 e ThreatDragon tem popularidade 902. No tempo de desenvolvimento, o Ncrack possui *commits* feitos há mais de 14 anos, o Nmap possui *commits* feitos há mais de 16 anos e o ThreatDragon possui *commits* feitos há mais de 8 anos. Dessa forma a amostra de ferramentas contém variação de popularidade e tempo de desenvolvimento.

Critérios de descarte	**Nmap**	**Ncrack**	**ThreatDragon**
Popularidade da ferramenta	10.934	1.233	902
Tempo de desenvolvimento	16 anos	14 anos	9 anos
Tempo de inatividade no repositório	2 meses	84 meses	14 meses
Baixa popularidade (Média das duas menores popularidades)	-	1.067,5	-
Data do último *fork* (média do tempo de inatividade dos maiores tempos de desenvolvimento)	43 meses	-	-

Tabela 4.1: Definição dos valores para critérios de exclusão em Novembro de 2023.

O valor de baixa popularidade é a média das duas menores pontuações de popularidade da amostra. O valor estabelecido foi 1.068, considerando as ferramentas Ncrack [2] e ThreatDragon [3]. Para a data do último *fork* foi considerado o tempo máximo de inatividade do repositório. Considerando Ncrack [4] e Nmap [5] pois são duas ferramentas com longo tempo de

[2] `https://github.com/nmap/ncrack`
[3] `https://github.com/OWASP/threat-dragon`
[4] `https://github.com/nmap/ncrack/graphs/contributors`
[5] `https://github.com/nmap/nmap/graphs/contributors`

desenvolvimento e variação na pontuação de popularidade. As Figuras 4.1 e 4.2 mostram informações de inatividade das ferramentas, o período máximo sem identificação de *commits* nos repositórios do Ncrack e Nmap respectivamente. Para a ferramenta Ncrack, a análise identificou um período de 84 meses de inatividade no repositório. Para a ferramenta Nmap, foram 2 meses de inatividade.

Dessa forma, o valor definido para o critério de exclusão pela data do último *fork* foi a média de inatividade entre Ncrack e Nmap. O valor estabelecido para o critério de exclusão data do último *fork* é 3.6 anos.

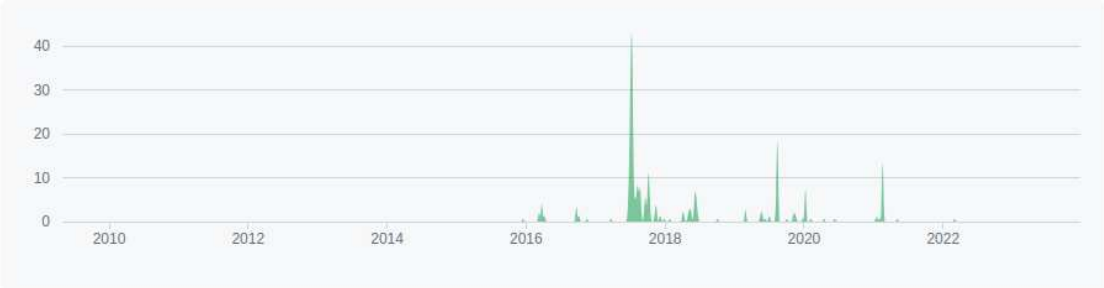


Figura 4.1: Tempo de inatividade Ncrack.

Fonte: Repositório do Ncrack no Github, 2023.

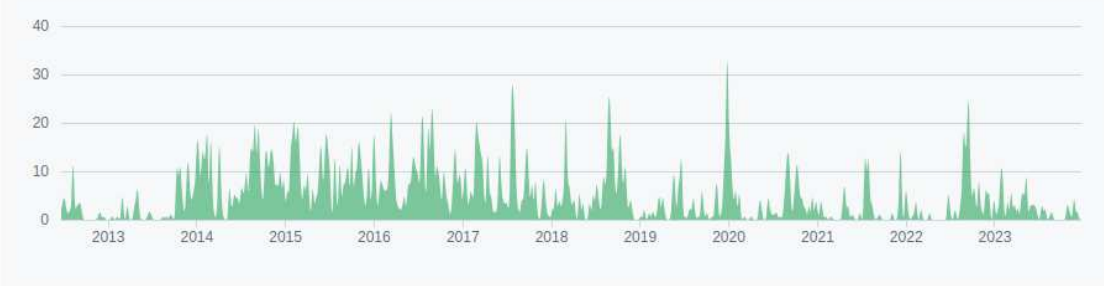


Figura 4.2: Tempo de inatividade Nmap.

Fonte: Repositório do Nmap no Github, 2023.

Para o processo de atribuição, o algoritmo Kmeans permitiu direcionar as ferramentas descobertas a grupos de recomendações. Cada *cluster* do Kmeans contém palavras-chave agrupadas, que permite o mapeamento das recomendações de segurança ao qual as palavras-chave foram extraídas. Após realizado o mapeamento das relações entre palavras-chave e recomendações de segurança, obtém-se um grupo de recomendações de segurança para cada

um dos *clusters*. A contribuição do algoritmo Kmeans para a vinculação é detalhada na Figura 4.4. Ao utilizar a lista de palavras relevantes geradas pelo TF-IDF na criação das strings de busca, a aleatoriedade das palavras com a falta de agrupamento exige considerar a análise integral das recomendações de segurança pertencentes ao documento CIS *Controls* V8.

Ao utilizar os *clusters* do Kmeans, embora seu uso não apresente melhoria dos resultados no processo de busca, como detalhado no Apêndice C, o agrupamento de recomendações de segurança mapeados para os *clusters* direciona a análise para setores específicos do documento CIS *Controls* V8. É demonstado na Figura 4.3 um *cluster* com controles mapeados e a quantidade de recomendações em cada um. Isso reduz consideravelmente o número de recomendações de segurança que precisam ser analisadas para realizar a atribuição.

```
GRUPO de Controles 0:
Controle                              ReferenciaControle
Network Monitoring and Defense        CIS Control 13        11
Access Control Management             CIS Control 6          8
Network Infrastructure Management     CIS Control 12         8
Account Management                    CIS Control 5          2
Email and Web Browser Protections     CIS Control 9          2
Service Provider Management           CIS Control 15         2
Data Protection                       CIS Control 3          1
Security Awareness and Skills Training CIS Control 14        1
```

Figura 4.3: Exemplo de grupo de controles de segurança de um *cluster*

.

Fonte: De autoria própria.

Após a definição de parâmetros para o processo de atribuição de ferramentas de segurança às recomendações de segurança, as atividades deste processo podem ser executadas. Isso dá suporte ao processo de análise e atribuição adequadamente. Esse é um processo significativo, no qual a capacidade da implementação das recomendações de segurança dependem diretamente das ferramentas à ela atribuídas.

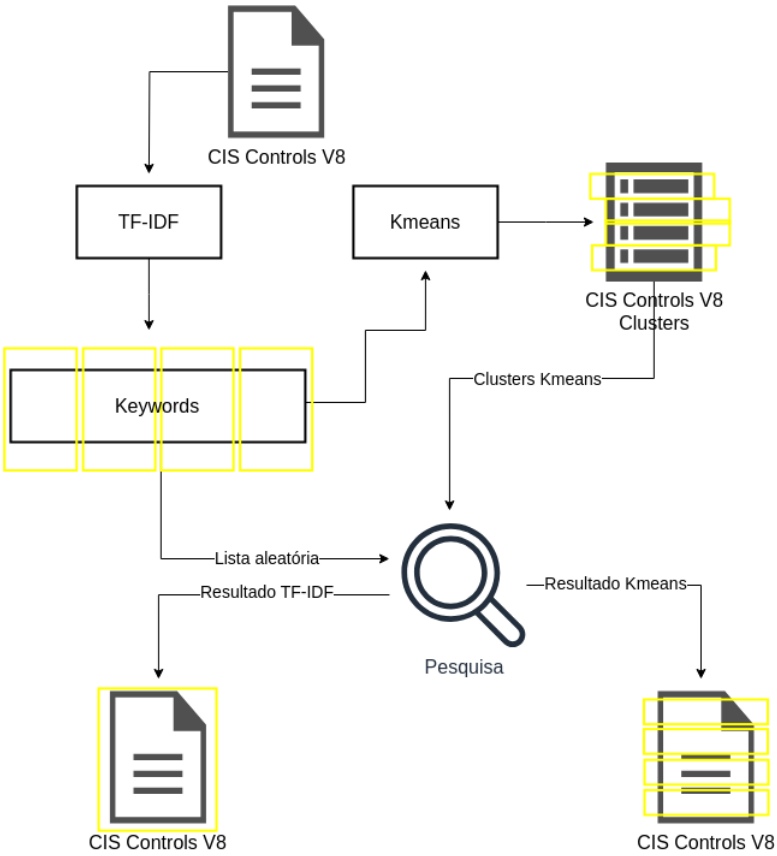


Figura 4.4: Processo de detecção e relação de recomendações de segurança e ferramentas de segurança.

Fonte: De autoria própria.

4.3 Definições do escopo de interesse

Até o momento, na Seção 4.1 e 4.2 foram consideradas recomendações e ferramentas de segurança em escopo geral. Nesse sentido, a abrangência se estende para diferentes componentes e contextos de utilização. Mediante isso, propomos um contexto que motiva o escopo para implementação e testes, consequentemente viabilizando as medições detalhadas na Seção 5.

Para determinar o escopo de utilização, necessitou-se definir dois fatores. O primeiro está relacionado às recomendações de segurança, o segundo se refere ao ambiente de implementação no cenário de computação na nuvem. Desse modo, as recomendações de segurança são submetidas à definições específicas, que conduz a aplicação de recomendações

para tratar questões de segurança contidas em um dado objetivo. A definição considerada para as recomendações de segurança foi o gerenciamento de vulnerabilidades e a segurança da cadeia de suprimentos de software, estes descritos nas Sessões 3.2.2 e 3.2.3. Direcionar recomendações de segurança para questões como estas torna a implementação bastante relevante. Áreas de grande importância para a segurança da informação são contempladas com conjuntos de recomendações, ferramentas para fortalecimento da segurança e métricas para avaliação da conformidade.

Para consolidação de uma lista de recomendações de segurança, critérios para mapeamento de recomendações pertinentes ao gerenciamento de vulnerabilidades, bem como segurança da cadeia de suprimentos precisam ser estabelecidos. Desse modo, além de palavras-chave padrão como *vulnerability* ou *supply chain*, a análise da descrição dos componentes de gerenciamento de segurança do ITU possibilitou a extração de palavras-chave que estão relacionadas com as determinadas recomendações de segurança. A lista de palavras-chave definidas foi *vulnerability, threat, asset, configuration, patch, assessment, provider, third-party* e *supply chain*.

Indo além da identificação por palavras-chave, definiu-se níveis de alinhamento ao escopo, para identificar recomendações de segurança que são classificadas nos níveis determinados. Os níveis definidos de alinhamento são descritos abaixo em três categorias referenciadas como AL1, AL2 e AL3:

- Para AL1, focamos em processos que podem ser medidos tanto no processo de construção quanto em tempo de execução de forma automatizada;
- Para AL2, consideramos recomendações parcialmente satisfeitas pela implementação proposta;
- Para AL3, abordamos implementações que não satisfazem literalmente a recomendação mas mitigam seus impactos.

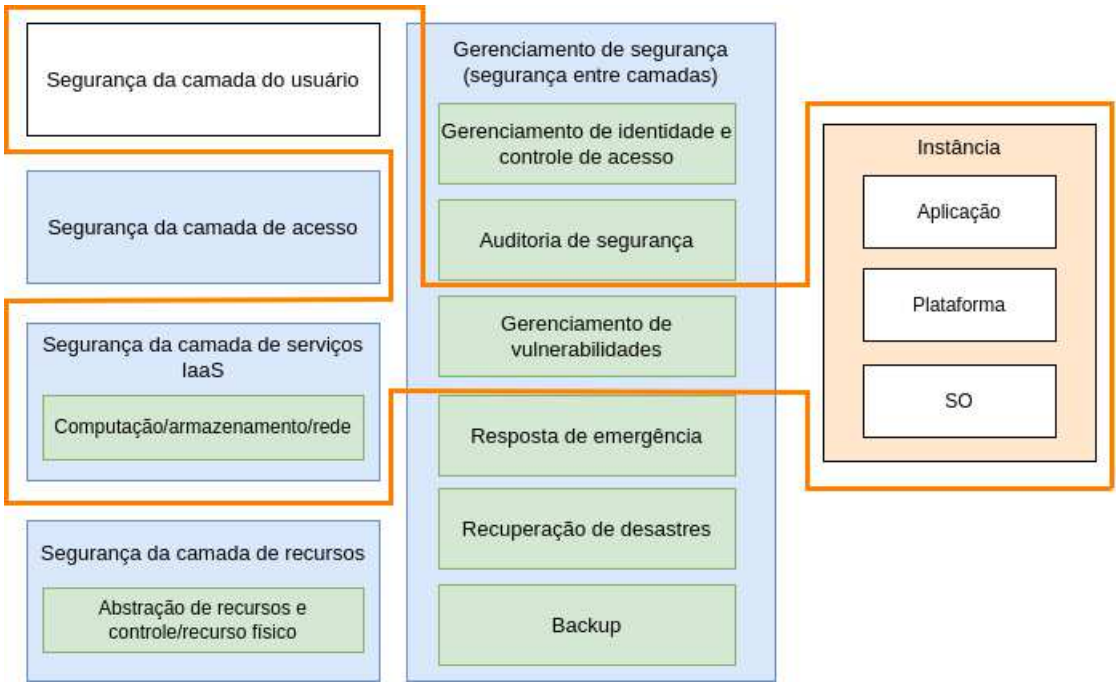
Para implementações no cenário de computação na nuvem é considerada a viabilidade de implementação técnica, montagem de ambientes de teste e instalação das ferramentas de segurança. Tomou-se por base o modelo de serviços de computação na nuvem IaaS. Os recursos primários são fornecidos pelos provedores de nuvem e o total controle de implementações e configurações fica a cargo dos clientes. Como o gerenciamento dos recursos

virtualizados. A existência de soluções de segurança aplicadas no cenário comercial atual, como DevSecOps e computação confidencial, fortalecem a postura de segurança de sistemas de computação. Soluções como estas oportunizam a condução de nossa implementação para componentes cruciais em IaaS, atuando de modo complementar com soluções de segurança acatadas opcionalmente de maneira flexível e adaptável aos interesses de segurança das organizações. Desse modo, como visto na Seção 3.1.3, as instâncias são um exemplo e parte fundamental do ambiente de computação na nuvem. O termo instância está relacionado a VMs, redes, sistemas operacionais, etc.

Embora algumas partes da infraestrutura tenham boas soluções de segurança, como a computação confidencial que minimiza a necessidade de confiar na infraestrutura, quando considerada para trabalhar com dados PII, também há a utilização de DevSecOps. Assim, entregando mais confiabilidade às aplicações. Nosso foco está no ambiente computacional gerenciado pelos usuários em IaaS, levando em conta elementos com potencial demanda a procedimentos para identificação de vulnerabilidades, bem como a possível implementação de componentes, bibliotecas ou softwares de terceiros como discutido na Seção 3.1.4. A Figura 4.3, que representa o conceito de alto nível de requisitos de segurança em IaaS definidos pelo ITU [Int20].

Tomando por base a representação de alto nível do ITU, a segurança da camada de acesso e camada de recursos possuem responsabilidades atribuídas aos provedores de serviços de nuvem. A segurança da camada de serviços IaaS, em parte, exige responsabilidades de segurança ligadas aos provedores com controle da implementação dos serviços como computação, armazenamento e redes ofertados a clientes de serviços de nuvem. A segurança da camada do usuário e camada de serviços IaaS pode ser gerenciadas por usuários.

Sendo assim, a camada de usuário representa componentes funcionais que incluem operações dos usuários consumindo recursos de serviços de nuvem. Usuários empregando ferramentas próprias na camada do usuário são encarregados pela manutenção da segurança. Na camada de serviços, os recursos disponibilizados por provedores, como serviços de computação e armazenamento, são postos à disposição dos usuários, que passam a se responsabilizar pela segurança.



[Int20].

Figura 4.5: Escopo mediante o conceito de alto nível de requisitos de segurança IaaS.

Fonte: Adaptado de ITU.

Os cenários de implementação tem como propósito sinalizar componentes que demandam por implementações de segurança necessitando de adequação à recomendações de segurança e conformidade, e também definição do ambiente para execução de testes. Foram então constituídos os cenários de aplicação, plataforma e SO. Como demonstrado na Figura 3.1, além de aplicação e SO, estão compreendidos dados, tempo de execução e *middleware*. Destes componentes, dados e *middleware* estão fora do escopo definido. O componente tempo de execução está contido dentro do cenário classificado como plataforma, pelo qual integra itens como imagens de contêiner e orquestração.

No cenário de aplicação, os componentes relacionados a essa camada são servidores *web*/aplicativos, servidores *Proxy*, bancos de dados NoSQL, caches distribuídos, bancos de dados relacionais, servidores de aplicativos e outros [LPD+13]. O cenário de teste atribuído em aplicação deve suportar a utilização de ferramentas que executam *scanner* de servidores para descoberta de problemas nos serviços dispostos. Desta forma, são cobertos diversos serviços que demandam execução neste cenário. Os aplicativos *web* constituem uma fra-

ção importante neste cenário. A OWASP[6] possui uma lista de ferramentas para *scanner* de vulnerabilidades com foco em aplicativos *web*. As ferramentas para essa finalidade são frequentemente denominadas DAST.

No cenário plataforma foi considerada a utilização do Kubernetes. Com a crescente utilização de contêineres, quase metade das organizações que utilizam contêiner executam o Kubernetes [Clo22] no gerenciamento. Assim, testes das ferramentas de segurança para esta camada devem considerar o uso do Kubernetes. Desse modo, o *scanner* de vulnerabilidades e verificação de configuração incorretas dos componentes utilizados se faz necessário. No ambiente de teste, os *scanners* de vulnerabilidades atuam na descoberta de vulnerabilidades conhecidas no ambiente em que os componentes alvo estão sendo utilizados. Isto contempla por exemplo imagens, *clusters*, segredos, repositórios, lista de materiais de software, etc. O *scanner* de configurações incorretas verifica possíveis problemas de segurança decorrentes da má configuração dos componentes ou IaC, dessa forma fugindo dos padrões de configuração recomendados para manter os níveis de segurança.

O cenário SO está presente em instâncias de VMs, estas podem ser utilizadas em servidores, nós que compõem *clusters* Kubernetes e podem relacionar-se a problemas como portas expostas, vulnerabilidades em componentes e problemas de privilégios de usuários, além de configurações incorretas. Estabelecer testes de ferramentas no SO considera a análise em versões específicas de um dado SO. Neste, os resultados esperados devem fortalecer a postura de segurança do sistema operacional, seja implementando técnicas de *hardening* [Fer21] por meio de ferramentas ou a descoberta de vulnerabilidades conhecidas e problemas de configuração incorreta.

[6] `https://owasp.org/www-community/Vulnerability_Scanning_Tools`

Capítulo 5

Operacionalização da conformidade

Neste capítulo, tratamos dos recursos necessários para a operacionalização da conformidade. Elaboramos a definição de evidências para a implementação das recomendações de segurança, a decomposição de recomendações em requisitos compostos por ações e condições e a criação de métricas para mensurar a conformidade e estimar vulnerabilidades com base em análises de explorabilidade e impacto. Além disso, visamos permitir que essas medições sejam realizadas automaticamente, fornecendo suporte a processos automatizados de verificação.

5.1 Evidências de conformidade

Desenvolver métodos que concedem provas do cumprimento de recomendações de segurança é tão importante quanto o cumprimento da própria recomendação. Mediante esta consideração, a coleta de evidências de conformidade permite, por meio de artefatos gerados por ferramentas de segurança, a criação de provas necessárias para o cumprimento das recomendações e consequentemente a garantia de conformidade. No contexto de implementação da ferramenta de segurança, uma evidência foi definida como o artefato resultante, ação executada ou atividade definida para uma ferramenta que esteja alinhada com os princípios de uma recomendação de segurança.

A lista de ferramentas que compõem a base da implementação técnica das recomendações de segurança possui o atributo que foi analisado como saída da execução, detalhado na Seção 6.1.3. Desse modo, é possível que uma determinada ferramenta salve os resultados de

uma execução em um arquivo de saída de formato previamente definido. Existem vários formatos de saída disponíveis, um dos formatos mais comuns entre ferramentas de segurança é o JSON. A saída produzida pela execução da ferramenta de segurança torna-se uma evidência quando fornece indício de que a atividade requerida em uma recomendação de segurança foi executada. Nem todas as saídas geradas pela execução de uma ferramenta de segurança podem ser consideradas uma evidência sem uma análise e correlação prévia.

Para definir as evidências de conformidade, é preciso a análise de critérios, bem como estabelecer previamente quais são as evidências de conformidade. Este procedimento considera etapas de análise das ferramentas utilizadas, a análise de desígnio da recomendação de segurança e quais saídas do processamento são evidências de seu cumprimento. Os critérios analisados para definição das evidências podem contemplar documentos, registros ou informações que relacionam-se ao cumprimento de condições estabelecidas nas recomendações de segurança. Desse modo, o processo que envolve análise para definir a produção de evidências por ferramentas de segurança envolve os seguinte fatores:

- Ferramenta utilizada;
- Artefato produzido;
- Cenário de implementação;
- Alinhamento da recomendação, definido na Seção 4.3;
- Condição requerida na recomendação, definida na Seção 5.2;
- Título identificador da evidência de conformidade.

A consideração destes critérios leva ao estabelecimento de evidências de conformidade geradas na execução de ferramentas de segurança. Seguindo além do que é resultado de uma execução, esse procedimento metodológico propõe estabelecer evidências que de fato estejam alinhadas a recomendações de segurança, servindo de prova de sua implementação e apoiando o uso de métodos para quantificação por meio de métricas. No processo que considera tarefas automatizadas, as evidências produzidas e analisadas refletem o estado de segurança do sistema e definem os meios que irão garantir conformidade de forma contínua.

5.2 Definição de métricas

Tarefas de verificação estão presentes em empresas interessadas em cumprir com padrões, leis, normas ou regulamentos. A conformidade pode ser verificada por meio de processos, muitas vezes realizados por procedimentos manuais guiados por auditores de segurança, como por exemplo nos processos de certificações da ISO. A elaboração de métricas pretende atender procedimentos de verificação da conformidade, seja de forma manual ou parte de processos de automatização e verificação contínua. A partir dessa perspectiva, propor os meios para mensurar a conformidade, além de conceder suporte para automação, irá permitir a definição de processos mais eficientes de verificação da conformidade de forma contínua, contribuindo também com indicadores de adequação a leis como LGPD.

Para expressar a importância da verificação de conformidade e, consequentemente o papel fundamental das métricas, evidenciar os mecanismos de segurança adotados, bem como comprovar seus efeitos para a segurança são itens requeridos pela LGPD. Tendo como exemplo o capítulo VII da LGPD, o Art. 48 expõe as necessidades de reportar medidas praticadas em caso de incidentes de segurança, como a indicação das medidas técnicas e de segurança utilizadas para a proteção dos dados. As sanções administrativas impostas pela lei encontradas no capítulo VIII Art. 52, também possibilitam a oportunidade de ampla defesa.

Considerando entre outros parâmetros e critérios, a cooperação do infrator, a adoção reiterada e demonstrada de mecanismos e procedimentos internos para minimizar danos e a adoção de políticas de boas práticas e governança. Neste sentido, as métricas possibilitam a validação da aderência às recomendações de segurança através da análise quantitativa de sua efetiva implementação, beneficiando organizações com a mensuração da conformidade e adequando-se aos critérios estabelecidos por lei, comprovando fatos em casos oportunos.

5.2.1 Expressão quantitativa de recomendações de segurança

Os documentos de referência para conformidade, como os CIS *Controls* e NIST *Cybersecurity Framework*, são fontes de dados não estruturados. Para estimar a conformidade com recomendações de segurança devem ser aplicados métodos que possibilitem a quantificação. Considerando para análise cada recomendação de segurança de forma individual dos documentos de referência, identificamos a estrutura da recomendação de segurança como uma

série de declarações imperativas. A identificação das declarações podem ser mapeadas para ações e condições que formam uma declaração imperativa da recomendação de segurança. Estas declarações podem ser entendidas como um requisito a ser satisfeito em uma recomendação de segurança. Em Cheng et al. [CVCLC18] é definido que uma relação semântica entre dois conceitos é expressa por um verbo em textos de linguagem natural, sendo necessária a identificação do verbo na declaração. Posteriormente sua relação com sintagmas nominais para cognição do sentido.

Tomando como exemplo uma das recomendações de segurança do CIS *Controls* V8, no texto descritivo da recomendação 7.1 é realizado o mapeamento das ações e condições que formam a recomendação. Na Figura 5.1, duas ações podem ser identificadas e claramente indicam o que deve ser realizado, sendo então definidas como requisitos da recomendação. Para cada ação, sua condição é interpretada e classificada como satisfeita ou não satisfeita em um processo de verificação de conformidade, seja em uma tarefa manual por intervenção humana ou um processo automatizado de verificação.

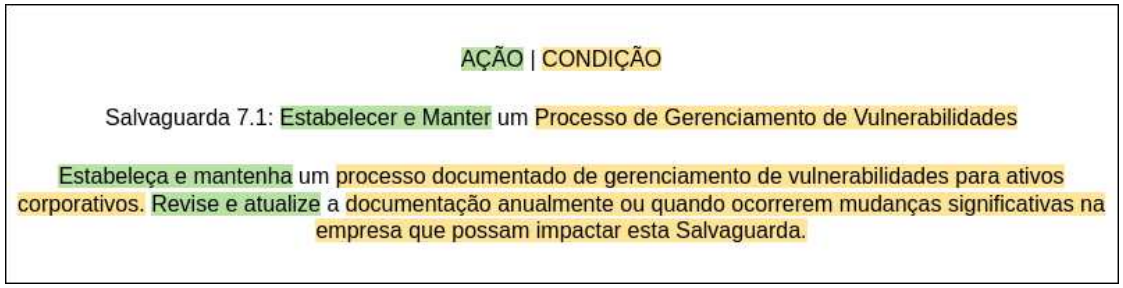


Figura 5.1: Mapeamento de ações e condições para definição de requisitos.

Fonte: De autoria própria.

O suporte de implementação técnica das recomendações por meio da atribuição de ferramentas de segurança, bem como a identificação de evidências produzidas pela execução das ferramentas, permite definir quais requisitos das recomendações estão sendo cumpridos de cada uma das recomendações em particular. Após mapeada a quantidade de ações e condições de cada uma das recomendações de segurança, podemos então estabelecer um processo que indica quais requisitos estão sendo satisfeitos e realizar um processo de verificação para estimar em expressão quantitativa a conformidade com as recomendações de segurança.

5.2.2 Métrica de Verificação do Nó

Para tratar das métricas, consideramos a instância definida na Seção 3.1.3 como um nó, ou seja, um componente dentro da infraestrutura de computação. Assim sendo, o processo de gerenciamento de vulnerabilidades definido por recomendações de segurança para um nó, é um processo em alto nível, comum entre as organizações, possuindo fases bem definidas para escaneamento, identificação, priorização e resolução.

Questões que definem as verificações de vulnerabilidade tratadas na Seção 3.2.2 como a amplitude sendo a cobertura da verificação de vulnerabilidades, bem como a profundidade que define o nível de *design* do sistema em que uma organização pretende monitorar, são particularidades de cada organização [NIST 800-53] e sofrem variação com a criticidade dos sistemas e informações utilizadas. Portanto, o processo de desenvolvimento das Métricas de Verificação do Nó (MVN) deve conceber para a métrica a capacidade de adaptação à necessidade da organização.

Para definir a capacidade de adaptação da métrica às necessidades da organização, assim como dar suporte a um processo automático e contínuo de verificação, foram definidos dois conjuntos com parâmetros para determinar o comportamento da verificação durante a execução das ferramentas, bem como avaliar o nó mediante os resultados de *scanner* de vulnerabilidades. O comportamento da verificação durante a execução da ferramenta é determinado por parâmetros informados pela organização divididos em 3 grupos de atributos: interface de rede, definindo atributos como portas utilizadas por serviços, endereços IP e tipos de serviços executados no nó verificado, como servidor *web*, LDAP, banco de dados, etc; verificação, definindo a abrangência das ferramentas utilizadas no processo de verificação, como uso de ferramenta de padronização SCAP ou verificação geral com resultados não padronizados; e alerta que define o envio de notificações a endereços de E-mail quando necessário.

Para avaliação do nó mediante os resultados de *scanner* de vulnerabilidades, precisou-se definir atributos relacionados à probabilidade de exploração por meio de vulnerabilidades e o impacto nos casos de exploração bem sucedida. Utilizamos as métricas do CVSS V3.1, onde a Figura B.2 da Seção 3.2.2 que trata sobre o gerenciamento de vulnerabilidades, demonstra o atual crescimento da utilização do CVSS para definir a severidade dos CVEs. A partir do uso do CVSS é possível definir a explorabilidade do nó por meio de métricas *Attack Vector*

(AV), *Attack Complexity* (AC), *Privileges Required* (PR), *User Interaction* (UI), *Scope* (S) e o impacto definido pelas métricas *Confidentiality* (C), *Integrity* (I) e *Availability* (A).

Para definir os valores de cada métrica que compõem a explorabilidade e o impacto, foi projetado um formulário de perguntas direcionado às organizações que gerenciam o nó. O formulário pretende mapear a condição de segurança do nó gerando uma MVN com finalidade de conduzir indicativos de vulnerabilidades potencialmente nocivas ao ativo alvo da verificação de vulnerabilidades. Cada métrica do CVSS foi analisada no contexto de verificação de nó, posteriormente foi definido um esquema de valores baseado no valor da constante associada que é usada nas fórmulas para cálculo do escore CVSS[1], isso permite calcular o valor de uma métrica a partir de várias perguntas elaboradas no formulário. A descrição da análise do nó considerando cada métrica, bem como justificativa e direcionamento auxiliar das perguntas são apresentadas a seguir:

- **Métrica *Attack Vector*:** Mede a exposição necessária para a exploração de possíveis vulnerabilidades. Quanto maior a exposição, seja em uma rede ou na internet, maior a quantidade de potenciais atacantes e consequentemente, maior a pontuação de severidade atribuída. Embora este seja o fator determinante para o peso mediante a exposição, o meio de acesso ao nó também é considerado quando o acesso *Local* ou *Physical* a um nó permite a exploração de todas as vulnerabilidades em contexto de maior exposição, como *Network*.

  Justificativa: As perguntas para definição da métrica AV consideram o nível de exposição requerido pelo nó e os meios de acesso para definir quais vulnerabilidades podem comprometer sua segurança caso exploradas.

  - **AV1:** Em ordem decrescente de exposição de *Network* a *Physical*, o nível de exposição para atuação do nó requer:
    - ***Network*:** Conexão com a internet;
    - ***Adjacent*:** Vínculo em uma LAN;
    - ***Local*:** Acesso via login local ou remoto;
    - ***Physical*:** Manipulação física do *hardware*.

[1] `https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf`

Além de identificar a exposição requerida pelo nó, o meio de acesso é fator determinante para sua operação, seja para fins de atualização, manutenção ou operação de uso. Visto que o acesso *Physical* é menor em número de potenciais atacantes, é o maior privilégio possível, então todas as categorias de vulnerabilidade podem ser exploradas. As perguntas a seguir definem o meio de acesso ao nó assumindo respostas como verdadeiro ou falso.

  - **AV2.1:** O acesso físico ao *hardware* do nó é de responsabilidade da organização;
  - **AV2.2:** Se sim em AV2.1, a organização declara que o controle e acesso físico ao *hardware* é feito por pessoal autorizado e garante a segurança do *hardware*;
  - **AV3:** É possível o acesso local ao nó por meio de *login* ou remotamente, como por exemplo através de conexão SSH. Acessos que permitem operações de leitura, escrita e execução;
  - **AV4:** O acesso ao nó é feito através de uma LAN;
  - **AV5:** O acesso ao nó é feito através da Internet.

- **Métrica *Attack Complexity*:** Para cumprir condições necessárias de exploração de vulnerabilidades, o nível de AC determina o esforço necessário por parte de agentes maliciosos. Dessa forma, a métrica estabelece o nível de complexidade para comprometimento do nó, isso considera mecanismos de segurança adotados para dificultar as condições de exploração de vulnerabilidades. Portanto, mecanismos de segurança como o controle de acesso e alertas para detecção de incidentes aumentam a AC para auxílio à proteção do nó.

  Justificativa: As perguntas visam detectar a existência de mecanismos e processos que atuam diretamente na postura de segurança do nó.

  - **AC1:** A detecção de ações, como, por exemplo, meios de assegurar a detecção de ações executadas de forma maliciosa, identificação de acessos não autorizados ou tentativas de *login*, modificação indevida em arquivos ou configurações e monitoramento de *logs*;

  - **AC2:** Processos para assegurar que apenas as portas necessárias para operações legítimas e serviços específicos estejam abertas e acessíveis, dessa forma, minimizando a superfície de ataque disponível para agentes maliciosos;
  - **AC3:** Mecanismos para garantir que apenas usuários autorizados tenham acesso a recursos e quando pertinente, usar criptografia nos protocolos de comunicação para impedir que dados sejam interceptados por agentes maliciosos.

- **Métrica *Privileges Required*:** Essa métrica refere-se ao nível de privilégio necessário para operação diária, bem como a existência de políticas de gerenciamento de privilégios praticados por parte da organização. O PR definido no SO do nó se relaciona ao PR necessário para exploração de vulnerabilidades.

  Justificativa: Identificar o nível de gerenciamento de privilégios no nó para usuários e processos. Quanto maior o privilégio dos usuários e processos sem políticas de gerenciamento, maior as chances de comprometimento de configurações, arquivos e componentes significativos em caso de exploração de vulnerabilidades. A lógica do privilégio dos usuários de SO é inversa ao PR da vulnerabilidade. A falta de gerenciamento de privilégios oportuniza problemas com escalonamento de privilégios ou exploração de vulnerabilidades com nenhum privilégio necessário. Para o privilégio do usuário, nenhum < baixo < alto, enquanto PR da vulnerabilidade nenhum > baixo > alto.

  - **PR1:** Os possíveis valores indicam que usuários locais tenham nenhum, baixo ou alto privilégio. Dessa forma, o privilégio de usuários locais pode fornecer controle significativo sobre componentes vulneráveis, ou executar operações críticas com baixo privilégio requerido.
  - **PR2:** Mesmo sentido dos privilégios dos usuários mas aplicável aos processos. O privilégio dos processos pode fornecer controle significativo sobre componentes vulneráveis, ou executar operações críticas com baixo privilégio requerido.
  - **PR3:** A organização declara que existe um plano para gerenciamento de privilégios de usuários e processos para o nó que contempla operações de leitura, escrita e execução. Por exemplo, apenas administradores tem maiores níveis de permissão, enquanto usuários regulares tem restrições de privilégio.

- **Métrica *User Interaction*:** Esta métrica define a necessidade de interação humana além do invasor. A medição determina se a vulnerabilidade pode ser explorada apenas pelo invasor ou requer ação de um usuário separado ou um processo iniciado por usuário que deva participar com alguma interação. As perguntas AV2-2, AV3 e todas as perguntas de PR definem o peso para determinar o valor da métrica UI.

  Justificativa: Detectar os meios de acesso ao nó, bem como os níveis de privilégio e gerenciamento de privilégios direcionados a este, para estimar a atuação de usuários na exploração de vulnerabilidades.

- **Métrica *Scope*:** O Escopo determina o impacto da exploração de uma vulnerabilidade estendendo para além da autoridade de segurança (nó com serviços, aplicações e SO).

  Justificativa: Identificar se o comprometimento do nó por meio da exploração de vulnerabilidades é capaz de comprometer componentes além da sua autoridade de segurança.

  - **S1:** As operações realizadas pelo nó não exigem interação com outros sistemas através de uma rede.
  - **S2:** Presença de mecanismos para abordagem proativa de segurança com intuito de barrar possíveis impactos sofridos ou gerados por exploração de vulnerabilidades.
  - **S3:** Uso de virtualização ou contêineres para isolamento e contenção de problemas de segurança é praticado.

Tomando como base as considerações e direcionamentos realizados sobre as métricas do CVSS, foram elaboradas as perguntas para compor as MVNs aplicando-se o formulário para coleta de dados, definição dos pesos de cada resposta e regras de relação entre perguntas para concepção da métrica para o nó.

As métricas AV são definidas por um conjunto de perguntas para detectar nível de exposição e acesso ao nó. A combinação das perguntas foi elaborada para dar contexto e melhoria da precisão na coleta das respostas para estimar a métrica AV determinando os valores AV em vulnerabilidades que são nocívas para o nó. Sendo assim, a pergunta $AV1$ pode assumir os valores *Network*, *Adjacent*, *Local* ou *Physical*. Em $AV2.1$, $AV2.2$, $AV3$, $AV4$ e $AV5$, é

definido que cada pergunta pode assumir os valores *True* ou *False*. Para as perguntas $AV2.1$ e $AV2.2$, definimos se $AV2.2$ for *True*, então $AV2.1$ deve ser *true* ($AV2.2 \rightarrow AV2.1$). Para as perguntas $AV4$ e $AV5$, definimos $(AV4 \vee AV5) =$ *False* se $(AV1 \neq N) \wedge (AV1 \neq A)$. Então definimos AV como:

$$AV = \begin{cases} [N, A, L, P] & \text{se } (AV2.1) \vee (AV2.1 \wedge \neg AV2.2) \\ [N, A, L] & \text{se } (\neg AV2.1 \wedge AV3) \\ [N, A] & \text{se } (\neg AV2.1 \wedge \neg AV3) \end{cases}$$

A métrica AV recebe um vetor que representa o nível de exposição das vulnerabilidades que são potencialmente nocivas para o nó. As perguntas que compõem a métrica AV seguem abaixo:

- $AV1$**:** Qual é o nível de exposição necessário para o funcionamento do nó?
- $AV2.1$**:** A organização é responsável pelo acesso físico aos equipamentos de *hardware*?
- $AV2.2$**:** Se sim em $AV2.1$. Existe uma política de segurança e controle de acesso físico ao *hardware*?
- $AV3$**:** É possível acessar o nó através de *login* local ou remoto?
- $AV4$**:** O acesso ao nó é feito através de uma LAN?
- $AV5$**:** O acesso ao nó é feito através da internet?

A métrica AC possui 3 perguntas para estimar seu valor, sendo assim, definimos que cada pergunta pode assumir os seguinte valores:

$$\textit{True} = 1$$

$$\textit{False} = 0$$

Então, para $AC1$, $AC2$, $AC3$ podendo ser *True* ou *False*, definimos AC como:

$$AC = \begin{cases} \textit{Low} & \text{se } \textit{count}(AC1, AC2, AC3) \leq 1 \\ \textit{High} & \text{se } \textit{count}(AC1, AC2, AC3) \geq 2 \end{cases}$$

onde *count*$(AC1, AC2, AC3)$ representa a quantidade de afirmações que são verdadeiras. As perguntas que compõem a métrica AC seguem abaixo:

- $AC1$**:** Existe monitoramento de atividades suspeitas executadas dentro do nó?
- $AC2$**:** Existe controle e gerenciamento de portas utilizadas por serviços do nó?
- $AC3$**:** Os serviços em execução possuem autenticação e/ou criptografia para melhoria da segurança?

A métrica PR possui 3 perguntas, para $PR1$ e $PR2$: As respostas podem ser *None*, *Low*, ou *High*, com pesos atribuídos de 0.85, 0.62 e 0.27, respectivamente. Em $PR3$, as respostas podem assumir *True* ou *False*, com *True* recebendo um peso de 0.27 e *False* recebendo peso de 0.85. Então, para PR podendo ser *None*, *Low* ou *High*, definimos PR como:

$$PR = \begin{cases} \textit{None} & \text{para valores} > 0.62, \\ \textit{Low} & \text{para valores} > 0.27 \text{ e } \leq 0.62, \\ \textit{High} & \text{para valores} \leq 0.27. \end{cases}$$

As perguntas que compõem a métrica PR seguem abaixo. A média dos valores de $PR1$, $PR2$ e $PR3$ definem o valor para PR:

- $PR1$**:** Qual o nível de gerenciamento de privilégio dos usuários locais, visando a segurança do sistema?
- $PR2$**:** Qual o nível de gerenciamento de permissão para a execução de processos no nó, visando a segurança do sistema?
- $PR3$**:** Existe uma política para determinar e aplicar os níveis de permissão necessários para usuários e softwares?

A métrica UI possui relação com duas perguntas da métrica AV. Para $AV2.2$ e $AV3$: As respostas podem ser *True* ou *False*. Então, para UI podendo ser *Required* ou *None*, definimos UI como:

$$UI = \begin{cases} \textit{Required} & \text{se } (AV2.2 \vee AV3) \text{ é verdadeiro.} \\ \textit{None} & \text{se } (AV2.2 \vee AV3) \text{ é falso.} \end{cases}$$

A métrica S possui 3 perguntas para estimar seu valor, sendo assim, definimos que cada pergunta pode assumir os seguinte valores:

$$\textit{True} = 1$$

$$\textit{False} = 0$$

Então, para $S1$, $S2$, $S3$ podendo ser *True* ou *False*, definimos S como:

$$S = \begin{cases} \textit{Changed} & \text{se } \textit{count}(S1, S2, S3) \leq 1 \\ \textit{Unchanged} & \text{se } \textit{count}(S1, S2, S3) \geq 2 \end{cases}$$

onde $\textit{count}(S1, S2, S3)$ representa a quantidade de afirmações que são verdadeiras. As perguntas que compõem a métrica S seguem abaixo:

- $S1$**:** O nó opera isoladamente? Isto é, não interage com outros sistemas em uma rede.
- $S2$**:** Existe algum mecanismo de segurança no nó com objetivo de minimizar impactos sofridos ou gerados para outros componentes na mesma rede?
- $S3$**:** O nó usa virtualização ou contêineres para isolamento?

Para definir as métricas relacionadas a impacto, os possíveis valores de resposta e os pesos atribuídos referem-se às métricas de *Confidentiality*, *Integrity* e *Availability*, cada uma destas possui 4 perguntas. Sendo assim, para as métricas C, I ou A cujos valores são representados por x, as respostas podem ser *None*, *Low* ou *High* com base nas respostas e pesos atribuídos a $x1$, $x2$, $x3$ e $x4$ representando cada uma das perguntas. Para $x1$, definimos que deve assumir *True* recebendo um peso de 0.56 ou *False* recebendo peso de 0.22.

Para $x2$, $x3$ e $x4$, as respostas podem ser *None*, *Low* ou *High*, com pesos de 0.0, 0.22 e 0.56, respectivamente, conforme indicado pela documentação do CVSS V3.1[2]. Então, para x podendo ser *None*, *Low* ou *High*, definimos x como:

[2] `https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf`

$$X = \begin{cases} \textit{None} & \text{para pesos} = 0.0, \\ \textit{Low} & \text{para pesos} > 0.0 \text{ e } \leq 0.22, \\ \textit{High} & \text{para pesos} \geq 0.23. \end{cases}$$

a média dos valores de $x1$, $x2$, $x3$ e $x4$ definem o valor para x. O direcionamento auxiliar das perguntas que compõem as métricas C, I e A seguem abaixo:

- **Métrica *Confidentiality*:** Esta métrica se refere ao nível confidencialidade das informações para a organização, para um valor muito alto de confidencialidade sugere que a organização tem baixa tolerância a vulnerabilidades com impacto na confidencialidade, estas são potencialmente nocívas.

  Justificativa: As perguntas direcionam o valor para a métrica C, considerada a sensibilidade das informações, criticidade das informações para a organização e o controle de acesso ao nó.

  - $C1$**:** O nó é utilizado para armazenamento temporário ou permanente de informações sensíveis ou utilizado para processar dados sensíveis de usuários, da organização, de funcionários, entre outros;
  - $C2$**:** O nó em seu estado de atuação realiza manipulação de informações que não requer confidencialidade, ou são informações que demandam baixa ou alta confidencialidade.
  - $C3$**:** O nó em seu estado de atuação provê serviços que não são críticos para a organização ou são de baixa criticidade ou alta criticidade. A criticidade é definida como a importância do ativo para a organização.
  - $C4$**:** Caso não exista o controle, o valor é nenhum. Caso exista um processo de controle para acesso ao nó, seja acesso de forma física ou virtual, o nível de restrição pode ser considerado baixo ou alto.

- **Métrica *Integrity*:** Esta métrica se refere ao nível de confiança que a organização demanda das informações e estado do nó. Isto sugere que em altos valores para integridade, a organização tem alta demanda por confiabilidade do nó e veracidade das informações.

Justificativa: As perguntas direcionam para identificar um valor para a métrica I mediante a existência de mecanismos para acompanhar modificações e confiabilidade do nó.

- $I1$**:** Define a existência de mecanismos para gerenciamento de *logs*, bem como controle de acesso e mecanismos de alerta sobre modificações inesperadas.
- $I2$**:** Caso não exista controle para gerenciamento de configurações, o valor é nenhum. Caso exista um processo de controle para gerenciamento de configurações no nó, o nível de restrição pode ser considerado baixo ou alto.
- $I3$**:** Define a existência de um processo ou mecanismo para garantir o uso de softwares confiáveis executados no nó.
- $I4$**:** Define a existência de um processo ou mecanismo para gerenciar vulnerabilidades conhecidas de componentes de software executados no nó.

- **Métrica *Availability*:** Esta métrica representa o nível de disponibilidade requerido da organização para o nó em operação. O impacto da disponibilidade do nó para a organização é alto quando a tolerância para disponibilidade é baixa.

  Justificativa: As perguntas definem o valor da métrica A mediante as capacidades de recuperação de incidentes, relevância do nó em operação, bem como as informações nele processadas e o nível de dano de qualquer natureza causado por questões de disponibilidade do nó.

  - $A1$**:** Em caso de comprometimento, um plano de recuperação visa fornecer uma rápida resposta a incidentes que comprometam a disponibilidade do nó;
  - $A2$**:** Define a relevância do nó considerando sua importância para a infraestrutura da organização. Por exemplo, para manutenção de serviços essenciais, o nó pode não ser necessário, bem como pode ter um nível baixo ou alto de relevância na infraestrutura;
  - $A3$**:** Define a importância das informações disponíveis no nó para a manutenção do pleno funcionamento da organização. As informações disponíveis podem não ter importância, bem como podem ter um baixo nível de importância ou alto nível de importância;

  - $A4$**:** Define o dano causado para a organização, caso ocorra o comprometimento da disponibilidade do nó. Desse modo, pode não ocorrer dano ou o dano seja em um nível baixo ou alto para a organização.

Tomando como base as considerações de direcionamento e justificativas, as perguntas elaboradas para compor o formulário que definem as métricas de impacto seguem abaixo:

- $C1$**:** Armazena e/ou processa dados sensíveis?
- $C2$**:** Qual o nível de criticidade dos dados do nó para a organização?
- $C3$**:** Qual o nível de criticidade dos serviços do nó para a organização?
- $C4$**:** Qual o nível de restrição de acesso dos usuários ao nó?
- $I1$**:** Existe um plano para identificar e auditar as modificações realizadas no sistema operacional?
- $I2$**:** Qual o nível de restrição no acesso ao gerenciamento de configurações?
- $I3$**:** Qual o nível de controle de uso de softwares de fontes confiáveis?
- $I4$**:** Qual o nível de checagem de vulnerabilidades conhecidas no inventário de softwares?
- $A1$**:** Existe um plano de recuperação e resposta a incidentes para o nó?
- $A2$**:** Qual o nível de dependência do nó para a infraestrutura da organização?
- $A3$**:** Qual o nível de dano para a organização com a perda de acesso às informações do nó?
- $A4$**:** Qual o nível de dano para a organização caso ocorra a queda do nó?

A submissão do formulário de perguntas que define a MVN possibilita direcionar a atenção a vulnerabilidades com potencial risco de danos ao nó.

Capítulo 6

Provisionamento de um ambiente seguro

O capítulo anterior tratou de operacionalizar a conformidade e neste capítulo, serão apresentadas as metodologias definidas nesta operacionalização. São executados todos os passos necessários para provisionar ambientes seguros acordados com nossa proposta de conformidade. Definimos concordância entre recomendações aplicadas por ferramentas de segurança identificadas e direcionadas a nosso escopo de gerenciamento de vulnerabilidades. Executamos testes de validação e aderência entre recomendações, ferramentas, atividades de coleta de evidências mensuráveis de conformidade e processos de automação. Para automação, propomos um modelo composto por operações que utilizam ferramentas de segurança para produção recorrente de evidências verificáveis de conformidade.

6.1 Implementações de segurança

Como definido na Seção 4.2, procedemos nesta Seção com atuações para estabelecer a base de recomendações de segurança e detectar ferramentas de código aberto em autoridades reconhecidas definidas na Seção 3.3.1 e uso de *strings* com palavras-chave para realização das buscas.

6.1.1 Recomendações de segurança

Propomos basear implementações de segurança com a utilização adequada de recomendações. Partindo deste princípio, recomendações quando não expressam uma política ou ati-

vidades de gestão de segurança da informação, classificadas como recomendações organizacionais, demandam implementações de cibersegurança. Os procedimentos necessários da recomendação são descritos pela composição de requisitos que expressam ações mensuráveis a serem tomadas, como visto na Seção 5.2 e detalhadas estruturalmente na Seção 3.2.1. Para cumprir o máximo de recomendações alinhadas a um determinado propósito, a finalidade identificada de uma recomendação de segurança a torna passível de agrupamento. Isso facilita sua implementação para propósitos específicos nos sistemas de informação que demandem conformidade.

O processo para definir a base de segurança com uso de recomendações é descrito na Seção 4.1. Desse modo, quatro passos essenciais foram estabelecidos no processo. Inicialmente foram classificados os controles do CIS *Controls* V8, conjunto C de recomendações dispostas nos componentes de gerenciamento de segurança definidos pelo ITU-T X.1605. Posteriormente, os componentes de gerenciamento de segurança são complementarmente dispostos por controles de segurança do NIST *Cybersecurity Framework*, conjunto N de recomendações. A redundância de recomendações deve ser evitada, considerando primeiramente as recomendações do CIS, e posteriormente as recomendações do NIST que não estão sendo abarcadas $C \cup (N - C)$. Os dois outros passos do processo são referentes às tarefas: analisar as recomendações não classificadas para nenhum dos componentes de gerenciamento de segurança existentes; listar recomendações não classificadas, porém relevantes para segurança e compor uma categoria específica para esse propósito.

Realizar a atribuição entre controles e componentes exige que seja feita a análise de cada componente e cada controle de segurança. Adicionalmente, o propósito de um controles de segurança é reforçado por todas as recomendações de segurança que o compõem, sendo assim, a análise individual de cada recomendação é fundamental. É preciso garantir que o controle atribuído ao componente de gerenciamento de segurança foi corretamente conduzido e disposto. A atribuição deve partir de uma etapa mais geral, no nível de controle de segurança, posteriormente um nível mais detalhado do qual cada recomendação é disposta em um dos componentes do ITU.

Partindo das considerações realizadas, nas discussões precedentes exploramos os controles de segurança correlacionados diretamente com os componentes de gerenciamento de segurança delineados pelo ITU. Através de uma análise comparativa, identificamos contro-

les sugeridos por CIS *Controls* V8 e NIST *Cybersecurity Framework* que não se encaixam perfeitamente nos moldes estabelecidos pelo ITU. Dentro deste escopo, categorizamos os controles remanescentes em um grupo, desse modo o componente "outros relevantes"agrupa os controles e recomendações que agrega a segurança igualmente aos demais componentes do ITU. No Apêndice E encontra-se a Tabela E.7 que representa os controles classificados no componente outros relevantes.

Após realização da análise e detalhamento dos componentes do ITU-T X.1605, foram mapeados os controles de segurança do CIS e NIST para cada componente conforme procedimento descrito na Seção 4.1. O resultado compõe parte do esforço para gerar os desfechos dispostos no Apêndice E com a exposição dos controles na primeira coluna das tabelas que resultaram do mapeamento de controles e referenciam cada um dos componentes de gerenciamento de segurança.

Em segundo momento, a análise visa mapear recomendações, feito individual e reiteradamente em todas as recomendações de segurança. Seguindo um maior nível de detalhamento, são consideradas as descrições de cada recomendação de segurança para que estas sejam adequadamente inseridas e alinhadas aos componentes de gerenciamento de segurança do ITU. O resultado desse processo também é parte do esforço para gerar os desfechos dispostos no Apêndide E. Sendo assim, as tabelas são compostas para representar os componentes de gerenciamento de segurança integrados com controles de segurança mapeados em um primeiro momento e recomendações de segurança mapeadas em um segundo momento de análise. As tabelas também representam a relação entre recomendações e ferramentas de segurança, que serão detalhadas na próxima Seção.

6.1.2 Ferramentas de segurança

Após a execução de buscas por ferramentas para compor a implementação técnica de recomendações de segurança, consideramos as autoridades de relevância no fornecimento de ferramentas de segurança descrito na Seção 3.3.1, bem como o processo de identificação das ferramentas de segurança disposto no Apêndice C. Levando em conta o contexto geral das recomendações de segurança, o *Secure Code Box*, projeto da OWASP, foi analisado e disponibilizou o conjunto de ferramentas apresentado na Tabela 6.1.

Amass	Git RepoScanner	Nmap
CMSeek	Kube Hunter	Screenshooter
ffuf	Kubeaudit	Semgrep
Ncrack	Nikto	SSH-audit
SSH	SSLyze	Typo3Scan
Trivy	Whatweb	Elastichsearch
WPScan	ZAP	Dependency-track

Tabela 6.1: Ferramentas do projeto *Secure Code Box*.

O processo de busca por ferramentas de segurança fez uso dos parâmetros e *string* de busca descrita na alternativa 2 da Tabela C.2 de *strings* de busca, para realização das buscas seguindo o que foi estabelecido no Apêndice C. Desse modo, o resultado das buscas concebeu a lista de ferramentas de segurança definidas para o escopo geral de utilização, considerando palavras-chave referentes a todas as recomendações de segurança. Após identificação de ferramentas de segurança nos resultados obtidos através do uso das *strings* de busca, foram coletados nomes de potenciais ferramentas de segurança. Posteriormente, as ferramentas identificadas foram submetidas aos critérios de aceitação definidos na Tabela 6.3. A atividade realizada nas buscas e análise aos critérios de aceitação teve como resultado as ferramentas dispostas na Tabela 6.2.

OpenIAM	OpenSCAP	GRR Rapid Response
Keycloak	Lynis	ELK Stack
FreeIPA	OSSEC	cloudEndure
OpenLDAP	Security Onion	Bacula
Shibboleth	Snort	Rclone
OpenIDM	OpenVAS	TestDisk
CASB	Wazuh	Duplicity
Suricata	Nikto	SystemRescueCD
SSLyze	Trivy	cloudBerry Backup
Typo3Scan	Whatweb	UrBackup
Zap Advanced	Kube-bench	Amanda
Metaspoit	Wapiti	Restic
Zeek	MISP	DLP Data Loss Prevention
TheHive	Dependency-Track	VSAQ
Sigstore	dm-crypt	SLSA
git	IPTables	kube-Score
Microsoft TMT	OpenSearch	Puppet
PRSense	SPIRE	Terraform
Threat Dragon	USBGuard	VeraCrypt

Tabela 6.2: Ferramentas resultantes do processo de busca.

Para obtenção de uma lista concisa de ferramentas de segurança, os critérios de descarte foram empregados sobre as ferramentas listadas anteriormente. Ao realizar o processo de análise aos critérios de descarte foram identificadas as ferramentas compatíveis com um ou mais dos itens estabelecidos. A execução desta tarefa considerou ferramentas do OWASP e do processo de busca com uso de *strings*. Nas informações dispostas na Tabela 6.4, o nome da ferramenta está atribuído ao código do critério de descarte. Para maiores informações, a Seção 4.2 descreve a finalidade de cada um dos critérios de aceitação e descarte.

Uma série de ferramentas de segurança aprovadas nos critérios de aceitação e não elegíveis aos critérios de descarte foi estabelecida. Para realizar a atribuição de ferramentas à recomendações de segurança, é preciso o entendimento adequado das ferramentas de segurança, desse modo são consideradas as tarefas T1, T2 e T3 estabelecidas na Seção 4.2. Com as tarefas essenciais obrigatoriamente cumpridas, como a análise de disposição da ferramenta de segurança e identificação de princípios funcionais é possível estabelecer a compreensão

Critério de aceitação	**Número**	**Critério de descarte**	**Número**
Ferramenta de segurança.	CA1	Não é uma ferramenta de segurança de código aberto.	CD1
Ferramenta de código aberto.	CA2	Não é uma ferramenta de segurança.	CD2
Suporte ativo pela comunidade de desenvolvimento.	CA3	Suporte inativo da comunidade.	CD3
Documentação disponível.	CA4	Documentação indisponível.	CD4
		Ferramenta depreciada.	CD5
		Popularidade mínima abaixo de 1.068.	CD6
		Último fork superior a 3.6 anos.	CD7
		Incompatibilidade com recomendações de segurança.	CD8
		Repositório não encontrado	CD9

Tabela 6.3: Critérios de aceitação e descarte.

Ferramenta	**Referência**	**Ferramenta**	**Referência**
Git RepoScanner	CD4	Metasploit	CD8
Screenshooter	CD6, CD7	Wapiti	CD6
SSH-Audit	CD7	ELK Stack	CD9
SSH_scan	CD5, CD6	cloudEndure	CD9
Typo3Scan	CD6	TestDisk	CD8
Whatweb	CD6	Duplicity	CD6
OpenIAM	CD4	SystemRescueCD	CD4, CD9
FreeIPA - 6	CD6	cloudBerry Backup	CD4, CD9
Shibboleth	CD4	UrBackup	CD6
OpenIDM	CD6	Amanda	CD4, CD9
CASB	CD4	DLP Data Loss Prevention	CD8

Tabela 6.4: Ferramentas desconsideradas.

sobre os problemas que a ferramenta de segurança tende a resolver.

Mediante análise documental e site oficial das ferramentas, os princípios funcionais foram mapeados. Tomando por exemplo informações levantadas referentes à ferramenta Trivy, é possível destacar que suas características indicam que seu uso permite o *scanner* de vulnerabilidades, a verificação de problemas de configuração, a geração de relatórios de conformidade definidos em arquivos YAML para diretrizes e políticas específicas da organização, e a criação de artefatos como SBOM, assim como a verificação de vulnerabilidades usando estes artefatos como alvo.

Após feitas todas as considerações de análise, foi realizado o processo de atribuição entre ferramentas e recomendações de segurança. Para suporte a esta tarefa, as informações

geradas nesta Seção são somadas aos processos anteriores, descritos na Seção 6.1.1. Na realização das atribuições, devem ser considerados os itens descritos a seguir: A *string* de busca usada para identificar a ferramenta de segurança, isso restringe significativamente a quantidade de recomendações de segurança analisadas como descrito na Seção 4.2 e Figura 4.4; Informações sobre a recomendação de segurança, nome e descrição devem ser analisadas; Informações sobre a disposição e princípios funcionais das ferramentas; Informações sobre os componentes de gerenciamento de segurança do ITU.

Uma vez realizado o processo de atribuição das ferramentas às recomendações de segurança, as tabelas dispostas no Apêndice E contém as informações sobre todas as atribuições realizadas. Observa-se que o relacionamento entre ferramentas e recomendações é de muitos para muitos, uma recomendação pode conter inúmeras ferramentas e cada ferramenta pode estar atribuída a inúmeras recomendações. A tabela também deixa claro que uma recomendação de segurança além de poder ter várias ferramentas de segurança atribuídas, ela também pode não ter ferramentas atribuídas.

Como resultado do processo de atribuição, temos a Figura 6.1 resumindo a relação entre os componentes de gerenciamento de segurança e as ferramentas de segurança de código aberto. O número associado ao componente de gerenciamento de segurança do ITU-T X.1605 no lado esquerdo da figura, especifica quantas ferramentas escolhidas foram associadas a cada um dos componentes. Observa-se que determinadas ferramentas de segurança podem ser aplicadas a mais de um componente (ilustrado pelo número associado ao nome das ferramentas no lado direito da figura).

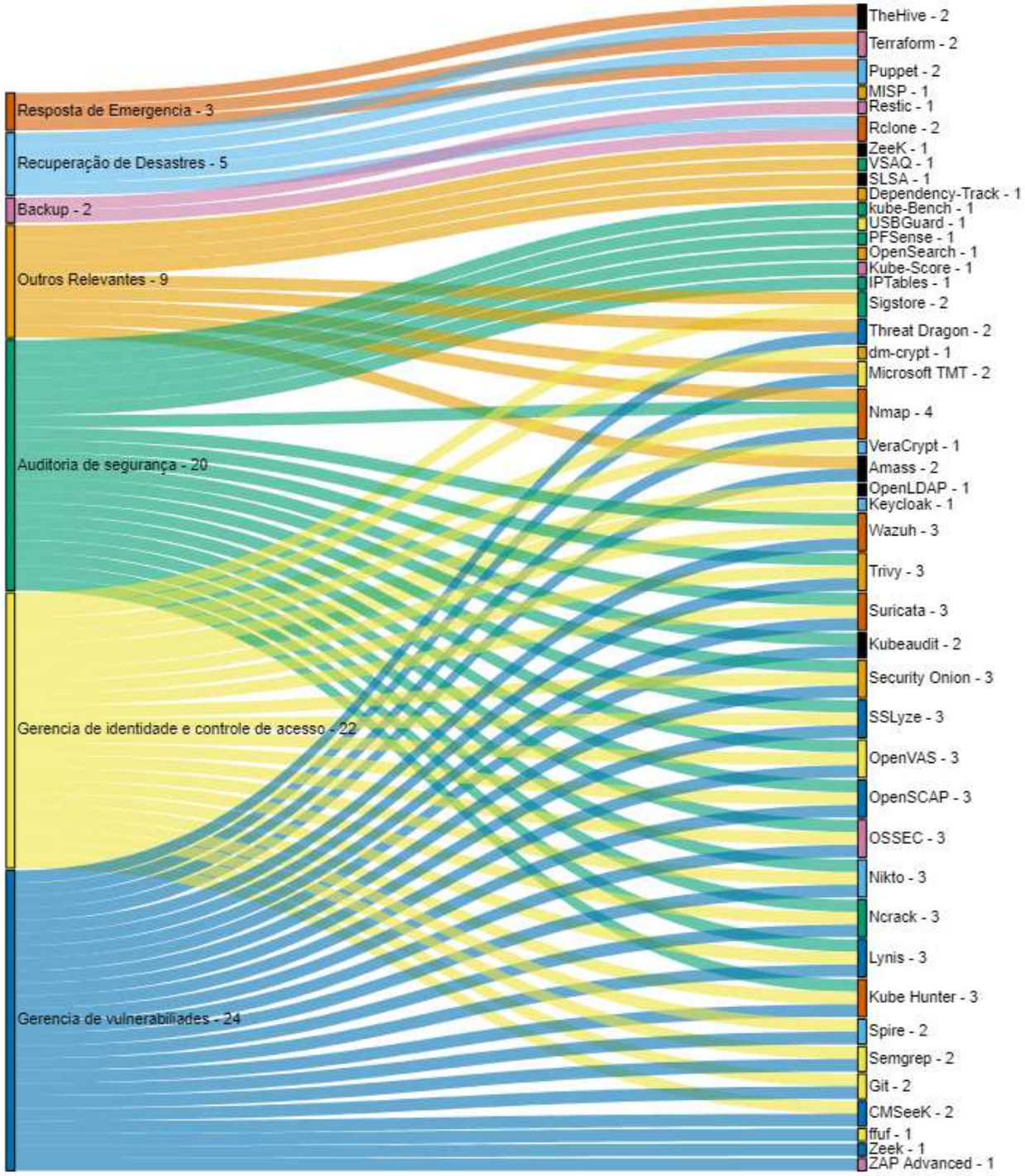


Figura 6.1: Relação entre componentes de gerenciamento de segurança e ferramentas de segurança.

Fonte: De autoria própria.

6.1.3 Escopo de implementação

A lista de recomendações de segurança, somada a ferramentas como disposto na Figura 6.1, abrange amplamente âmbitos imprescindíveis para a segurança. Objetiva-se testar a proposta para implementação de segurança em conformidade com recomendações do CIS e NIST. Desse modo foi estabelecido o escopo de utilização, descrito na Seção 4.3 e fundamentado nas Seções 3.2.2 e 3.2.3. Para atender a implementação de segurança ao domínio de interesse, é preciso identificar recomendações elegíveis para tal. O procedimento que se segue obteve a lista de recomendações de segurança a partir da busca por palavras-chave relacionadas ao escopo de utilização. As palavras-chave definidas foram *vulnerability, threat, asset, configuration, patch, assessment, provider, third-party, supply chain*.

A definição das palavras-chave considerou informações sobre o componente gerenciamento de vulnerabilidades do ITU, bem como palavras relacionadas à segurança da cadeia de suprimentos. No Apêndice F, a Tabela F.1 expõem as recomendações de segurança identificadas. Os resultados foram dispostos de modo a evitar redundância entre recomendações de segurança. Sendo assim, as informações dispostas passaram por critério de descarte ao qual o CIS *Controls* é alternativa prioritária e NIST adicionado complementarmente.

Definir a utilização das recomendações baseado apenas na seleção de recomendações por palavras-chave não pode ser tido como produto final. Utilizou-se os critérios definidos nos níveis de alinhamento descritos na Seção 4.3. Portanto, na Tabela 6.5 estão dispostas as recomendações de segurança alinhadas ao escopo de utilização. Recomendações classificadas por nível de alinhamento ao projeto com aplicação a questões de segurança que contemplam o gerenciamento de vulnerabilidades e a segurança na cadeia de suprimentos.

Classificação	**Controle**		**Recomendações**
	CIS	**NIST**	
AL1	2, 7, 15, 16	ID.RA, ID.SC	2.5, 7.1, 7.5, 7.6, 15.5, 15.6, 16.2, 16.5, 16.6, 16.7, ID.RA-1, ID.RA-2, ID.SC-1
AL2	1, 7, 15, 16	NIST ID.RA	1.2, 2.3, 7.7, 15.2, 16.1, 16.4, ID.RA-5
AL3	2, 7, 16	ID.RA, ID.SC	2.2, 7.3, 7.4, 16.3, ID.RA-3, ID.SC-2

Tabela 6.5: Recomendações alinhadas ao escopo de utilização.

O processo de atribuição entre recomendações e ferramentas de segurança realizado na Seção 6.1.2, permite a formação de uma lista de ferramentas aplicáveis ao escopo definido

na Seção 4.3. Desse modo, as ferramentas que provisionam implementações técnicas a recomendações de segurança direcionadas ao gerenciamento de vulnerabilidades e segurança da cadeia de suprimentos são o Trivy, OpenSCAP, OpenVAS, Lynis, Nmap, ZeeK, KubeHunter, Security Onion, Wazuh, CMSeeK, Ncrack, Nikto, SSLyze, OSSEC, Peass, VSAQ, SLSA, Dependency track e Sigstore.

Reforçando a consistência da lista formada pelas ferramentas identificadas, foi considerada a descoberta de ferramentas de gerenciamento de vulnerabilidades em fontes relevantes da área de segurança, estas fundamentadas na Seção 3.3.1. Desse modo, foram identificadas 17 ferramentas de código aberto da OWASP, 37 ferramentas do Kali Linux e 10 ferramentas da CNCF. Após análise que objetivou identificar ferramentas complementares, evitando redundância e aumentando o poder de verificação e análise aos cenários.

Do total de 64 ferramentas, 11 foram elegíveis para análise mais detalhada, no qual 6 já haviam sido identificadas constando na Tabela 6.2 e 5 foram novas descobertas, como: Peass, Sysdig, Prisma Cloud, Falco e Kubescape. Com isso, 7,8% do total de ferramentas analisadas foram novas descobertas e 10.1% ,corresponde a ferramentas identificadas na busca por ferramentas de segurança realizadas na Seção 6.1.2. Entre as 53 ferramentas restantes, 49 foram desconsideradas para uso por motivos categorizados na Apêndice F, no Tabela G.1, como ferramenta de pentest, fora de contexto definido, poucas informações ou obsoleto e baixa popularidade. Foram identificadas 5 ferramentas de segurança com potencial para atribuição a recomendações de segurança para compor a base de ferramentas como Deepfeence ThreatMapper, Nuclei, Zed Attack Proxy, Legion e Kube clarity.

Em um processo de análise, ferramentas foram instaladas para verificação detalhada considerando aspectos técnicos. Esse processo compreende identificar o cenário de implementação descrito na Seção 4.3, cenário ao qual cada ferramenta deve ser atribuída, bem como identificar os pré-requisitos necessários para execução. Após levantamento e análise dessas informações, cada uma das ferramentas consideradas foi descrita com base em sua disposição para uso e solução de problemas, bem como os princípios funcionais utilizados durante a execução.

- **CMSeeK** (OWASP): é uma ferramenta para exploração de vulnerabilidades em ambientes de *Content Management System* (CMS). Permite exploração de vulnerabilidade de mais de 170 CMS como o WordPress, Joomla e Drupal. As funcionalidades in-

cluem identificação de *plugins* instalados e versões, verificação de configurações incorretas, identificação de vulnerabilidades de versão, usando métodos de detecção em cabeçalhos HTTP, código-fonte de páginas e `robots.txt`. A ferramenta também tem suporte a módulos de força bruta que podem ser usados por equipes de segurança e de testes de penetração.

- **KubeScape:** É uma ferramenta que realiza verificações de inúmeros aspectos da arquitetura do Kubernetes semelhante ao Trivy. Isto abrange a descoberta de vulnerabilidades e configurações incorretas. As funcionalidades da ferramenta incluem verificação de linha base de configuração, sendo possível identificar problemas comparando as configurações identificadas com padrões esperados. A ferramenta possui um repositório de políticas que podem ser utilizadas durante a execução de suas verificações para adotar diferentes abordagens de verificação.

- **Linpeas:** É uma ferramenta que realiza verificações de privilégio no Linux/Unix. Seu nome contém o termo *Privilege Escalation Awesome Script Suite*. A execução do Linpeas realiza a verredura de problemas de privilégio do sistema operacional. Indo além, a ferramenta é capaz de identificar problemas de escalonamento de privilégios e reportar vulnerabilidades conhecidas do CVE em seus relatórios gerado após sua execução.

- **Lynis:** A ferramenta Lynis, é utilizada em auditoria de segurança, desempenha um papel essencial na verificação e relato de problemas de segurança nos sistemas. Isso envolve a análise de concessões inadequadas de permissões, configurações do sistema, integridade do sistema de arquivos, entre outros aspectos relevantes. Um dos objetivos dessa ferramenta é realizar testes e verificações abrangentes das defesas de segurança, com o propósito de fortalecimento do sistema. Dentre as funcionalidades do Lynis, destaca-se o suporte a testes de conformidade, detecção de vulnerabilidades e gerenciamento de *patches* de software. Essas capacidades permitem uma abordagem sistemática e abrangente para aprimoramento da segurança.

- **Nmap** (OWASP): A ferramenta Nmap é utilizada para descoberta de redes e portas abertas em *hosts* e realização de auditorias de segurança e inventário de rede, sendo

muito utilizada por administradores de rede, profissionais de segurança e pesquisadores. Entre possibilidade de utilização do Nmap estão o *scanner* de portas utilizadas como portas TCP, UDP ou portas específicas e listagem de portas abertas em *hosts* remotos, descoberta de *hosts* ativos em rede e identificação de sistemas operacionais.

- **Nikto** (OWASP): Nikto é uma ferramenta de *scanner* de servidores *web*, analisa servidores e aplicativos em busca de vulnerabilidades conhecidas. O Nikto executa testes abrangentes que incluem verificação de arquivos e programas potencialmente perigosos, verificação de versões desatualizadas para inúmeros servidores, incluindo detecção de problemas específicos de versões. Para a resolução de problemas e vulnerabilidades de segurança, a ferramenta também identifica erros de configuração e conduz dicas para testadores humanos.

- **OpenSCAP:** A ferramenta OpenSCAP é composta por um conjunto de ferramentas que estabelece um ecossistema que auxilia na implementação e medição das linhas base de segurança. Essa ferramenta é mantida pela Red Hat e utiliza o protocolo de automação de conteúdo de segurança SCAP para fornecer suporte à configuração automatizada e às atividades de conformidade de controle técnico. Dentre as suas funcionalidades, destacam-se a avaliação de vulnerabilidades por meio de um processo de identificação e classificação, além da verificação da conformidade com a política de segurança estabelecida.

- **Semgrep** (OWASP): A ferramenta Semgrep é um analisador de código para detecção de bugs e vulnerabilidades. Permitir a realização de análises de segurança automatizadas em código com identificação de violações e boas práticas de desenvolvimento. O suporte inclui linguagens como Java, Go, Python, PHP, Ruby, JavaScript além de incluir regras de escaneamento com mais de 2000 orientações da comunidade de desenvolvimento que cobrem segurança, correção e vulnerabilidades de dependências.

- **SSLyze** (OWASP): A SSLyze é uma ferramenta de *Command-line Interface* (CLI, em tradução, Interface de Linha de Comando) e uma biblioteca Python que realiza análise de segurança em servidores. Ao se conectar em um servidor, a ferramenta tem como propósito ser rápida e abrangente em suas operações, auxilia no exame de configu-

rações e certificados SSL/TLS. As principais características da ferramenta incluem a verificação contínua dos servidores para análise de configurações recomendadas, suporte à verificação de diversos tipos de servidor alvo e informações sobre identificação de possíveis vulnerabilidades e criptografia fraca.

- **Trivy** (OWASP): A ferramenta Trivy é um *scanner* de vulnerabilidades, faz verificação em componentes e dependências, capaz de realizar descoberta de problemas nas imagens de contêiner, sistemas de arquivo, repositórios Git, imagens de máquina virtual e Kubernetes. As verificações realizadas pelo Trivy incluem *scanners* de vulnerabilidades conhecidas no CVE, verificação de problemas de configurações incorretas e relatórios de conformidade definidos em arquivos YAML para diretrizes e políticas específicas da organização.

O processo de análise de ferramentas compreende critérios definidos para compor um modelo composto por atividades automatizadas e passíveis de auditoria e mensuração de resultados. As Tabelas 6.6 e 6.7 definem em alto nível as propriedades necessárias para compor um processo de gerenciamento de vulnerabilidades. Isto considera analisar ferramentas mediante a aderência aos cenários determinados na Seção 4.3 e definir a abrangência de atuação no gerenciamento de vulnerabilidades em categorias como: capacidades de *scanner* de vulnerabilidades, indicativos de severidade para suporte a processos de priorização e análise de risco e análise de configurações.

A Tabela 6.6 é constituída por detalhes e resultados do processo de análise, contém ferramentas elegíveis para utilização nos cenários descritos na Seção 4.3. Ao lado de cada cenário de implementação, as colunas referentes à ferramenta informa o nome e versão atribuída. As colunas de Gerenciamento de Vulnerabilidades, sendo elas *Scanner* de Vulnerabilidades e Análise de Configurações, foram baseadas no componente de gerenciamento de vulnerabilidades do ITU. Uso de componentes vulneráveis e configurações incorretas são problemas de segurança presentes no OWASP *top* 10 *risk of security*[1] nos anos 2017 e 2021, exemplificando a relevância de verificações nestes aspectos. A coluna Indicador de Severidade foi inserida após observações sobre as diferentes maneiras em que vulnerabilidades são reportadas em diferentes ferramentas, algumas consideram o escore CVSS, outras o nível de

[1] `https://owasp.org/www-project-top-ten/`

severidade, sendo ele baixo, médio, alto ou crítico. Em alguns casos, apenas a referência e descrição da vulnerabilidade é retornada, sem constar nos resultados as informação sobre indicadores de severidade.

Cenário	**Nome**	**Versão**	**Scanner de Vulnerabilidades**	**Indicador de Severidade**	**Análise de Configurações**
Aplicação	CMSeeK	v1.1.3	X		
	Nikto	v2.5.0	X	X	X
	SSLyze	v5.2.0	X		
	Semgrep	v1.55	X		X
Aplicação / SO	Nmap	v7.80	X		X
Plataforma	kubescape	v2.9.0	X	X	X
	Trivy	v0.47.0	X	X	X
SO	OpenSCAP	v1.2.16	X		X
	Lynis	v2.6.2	X	X	X
	Peass	v2.6.8	X		X

Tabela 6.6: Relação entre cenários e ferramentas de segurança.

Nome	**CLI**	**Automação Nativa**	**Entrada**	**Saída**
CMSeeK	X		-	JSON
Nikto	X		-	HTML, JSON, CSV
SSLyze	X		-	JSON
Semgrep	X		-	JSON, SARIF
Nmap	X		NSE	NMAP, XML, GNMAP
kubescape	X	X	REGO Policy	XML, PDF, prometheus, HTML JSON
Trivy	X	X	-	TABLE, JSON, KBOM, SBOM, SAFIR
OpenSCAP	X		SSG, XCCDF, OVAL	XML, HTML
Lynis	X	X	PRF	DAT
Peass	X		-	TXT

Tabela 6.7: Critérios de automação.

Considerando o processo de execução de forma automática, a Tabela 6.7 contempla critérios observados para suportar a utilização das ferramentas sem intervenção humana como definido na Seção 3.3.3. Foram identificadas ferramentas que possuem funcionalidades assistidas por processos automatizados e nativos, justificadamente a motivação para a coluna

Automação Nativa. A possibilidade de execução de comandos interpretados pela ferramenta via CLI é fundamental para programação de *scripts* de automação. A coluna Entrada indica os pré-requisitos necessários para execução da ferramenta adotando algum comportamento guiado por valores de entrada para execução, por exemplo *scripts default* da ferramenta ou políticas escritas em REGO aplicadas às verificações. Em Saída, constam os possíveis formatos suportados produzidos após um processo de execução.

Os cenários foram populados com ferramentas que solucionam ou mitigam problemas específicos do cenário em questão. No cenário de aplicação, as ferramentas CMSeeK, Nikto, SSLyze e Semgrep, estão relacionadas a solução de problemas como configuração incorreta de servidores, verificação de certificados SSH, exposição a SQL *Injection* e XSS *Cross-Site Scripting*, exposição indevida de componentes e informações de infraestrutura, vulnerabilidades em nível de código e muitos outros problemas pertinentes a este cenário.

No cenário de plataforma, ferramentas de *scanner* de vulnerabilidades e configurações incorretas cobrem a arquitetura de componentes e objetos do Kubernetes, como *cluster, file system,* IaC, *secrets, namespaces*, pods, etc. Desse modo, podem ser identificadas uma ampla gama de vulnerabilidades e problemas potencialmente nocivos à segurança, bem como verificação e apoio na constituição de linhas base de configuração segura através de descoberta e indicação de problemas e implementações de padrões industriais recomendados.

No cenário de SO, ferramentas que executam especificamente verificações do estado de segurança do sistema foram atribuídas. Dessa forma, a verificação contempla configurações do SO, configurações de rede, componentes em versões com vulnerabilidades conhecidas e privilégios desnecessários, bem como problemas relacionados ao escalonamento de privilégios ocasionando riscos à segurança. Tomando como exemplo a ferramenta Peass realiza verificação de problemas relacionados ao escalonamento de privilégios, o Lynis realiza verificações de segurança padrão para sistemas Linux e o OpenSCAP permite a realização de verificações customizadas de acordo com o propósito do SO.

Entre as ferramentas analisadas e definidas nos cenários, Nmap, Kubescape, OpenSCAP e Lynis demandaram pré-requisitos específicos como entrada para execução da ferramenta. Os pré-requisitos são NSE, REGO *Policy*, SSG, XCCDF, OVAL e PRF. Desse modo, o NSE[2] usado na execução do Nmap, não é item obrigatório. A utilização do

[2] `https://nmap.org/book/man-nse.html`

NSE estende a capacidade de análise do Nmap com o uso de *scripts* escritos na linguagem de programação Lua para verificação de explorações específicas. Na versão 7.80 instalada, o Nmap dispõe de 593 *scripts* NSE disponíveis para uso no diretório `/usr/share/nmap/scripts/`. São observados *scripts* de verificação de vulnerabilidade, como o *script* `http-vuln-cve2017-8917.nse`, usado para verificar CVE com pontuação de 9.8 e uma série de outras vulnerabilidades potencialmente exploradas pela porta 80. Dentre os *scripts*, estão verificações em serviços como LDAP, FTP, mysql, verificações de firewall e muitos outros. O uso de *scripts* NSE mostra-se um importante recurso de verificação da postura de segurança para alvos definidos durante uma verificação.

A ferramenta Kubescape é regida pelo mecanismo OPA[3] *Open Policy Agent* aplicando políticas escritas na linguagem Rego. As verificações realizadas pela ferramenta utilizam as políticas dispostas no repositório de políticas[4] da ferramenta. Dessa forma, a arquitetura do Kubernetes é verificada mediante as políticas. Ao executar um simples comando, como `kubescape scan -verbose -format json -format-version v2 -output output_file`, é possível estabelecer uma linha base de segurança através da verificação de configurações recomendadas. O fluxo de execução via CLI é exibido na Figura 6.2.

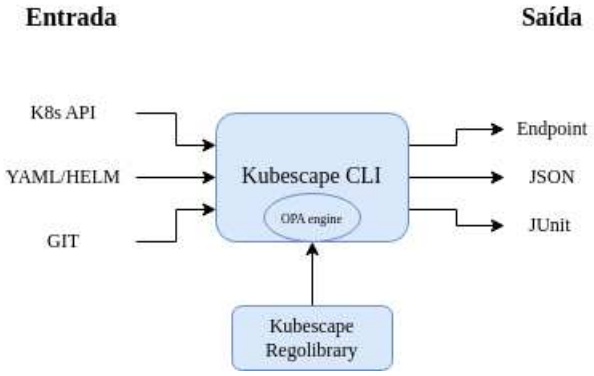


Figura 6.2: Arquitetura do Kubescape CLI.

Fonte: Adaptado de Kubescape [5].

O padrão SCAP usado pela ferramenta OpenSCAP segue uma síntese de especificações. OVAL e XCCDF são especificações SCAP contendo políticas aplicadas nas verificações du-

[3] `https://github.com/open-policy-agent/opa`

[4] `https://github.com/kubescape/regolibrary`

rante a execução do OpenSCAP. Algumas fontes podem ser usadas para consumo de arquivos nesse padrão. O SSG é um exemplo de fonte que é disponibilizado para instalação usando através do `yum install scap-security-guide` em algumas distribuições Linux. Também está disponível através do projeto *ComplianceAsCode*[6] contendo atualizações de arquivos SCAP para diversas plataformas. Especificações SCAP também podem ser obtidas através dos sites oficiais do SO como em nossos testes realizados utilizando Ubuntu[7]. Desse modo, configurações ou vulnerabilidades de segurança podem ser identificadas e posteriormente corrigidas.

Assim como o Nmap, a instalação do Lynis provisiona um arquivo usado durante a execução da ferramenta. No diretório do Lynis em `/etc/lynis/default.prf` o arquivo de extensão `.prf` é um perfil padrão disponibilizado pela ferramenta para a realização de verificações no SO. Foi observada a possibilidade de criar o arquivo `custom.prf` a partir do arquivo `default.prf` e alterar as verificações padrão realizadas nos componentes *kernel*, processos, rede, permissões de arquivo e *plugins*.

Dessa forma, são definidas verificações direcionadas para determinado uso do sistema, podendo ser observado para compor um servidor *web*, servidor de email, entre outros. *Plugins* disponíveis na ferramenta são extensões do seu núcleo escritas em Shell *script*. O propósito dos *plugins* envolve verificações de *firewall*, integridade de arquivos, autenticação, conformidade e *kernel*, entre outras possibilidades. Os resultados são reportados através de saídas como indicadores de status *ok* ou *warning* para componentes analisados.

6.2 Ambiente de testes

Os ambientes de teste motivados para execução das ferramentas foram idealizados para atender propósitos pré-definidos. Inicialmente, é crucial a validação da aderência da ferramenta de segurança ao cenário propost. Neste sentido, o objetivo não é estabelecer uma métrica para estimar, mas indicar se o uso da ferramenta é válido ou não para o cenário de implementação. Para suportar essa validação, é preciso compreender a motivação e aproveitamento em usar a ferramenta para fortalecer questões de segurança provisionada pela implementação.

[6] `https://github.com/ComplianceAsCode/content/releases/`

[7] `https://ubuntu.com/security/oval`

Após isso, o processo de teste permite mapear a execução da ferramenta para identificar procedimentos em comum acordo com recomendações de segurança. Por fim, são verificados artefatos de saída para compor evidências de conformidade, bem como abstrair funcionalidades que compreendam o processo de automação descrito na Seção 6.4.

6.2.1 Execução em ambiente controlado

A execução em ambiente controlado para teste do cenário para SO, compreendeu a instanciação de uma VM usando o SO Ubuntu 20.04 LTS para executar as ferramentas OpenSCAP, Lynis e Peass e Nmap. Desse modo o primeiro passo realizado foi a instalação das ferramentas e pré-requisitos necessários.

OpenSCAP

A ferramenta OpenSCAP é executada com base nos arquivos SCAP. Para execução, foram obtidos os arquivos SCAP do projeto *ComplianceAsCode* para fornecer o SSG e site oficial do Ubuntu como fornecedor do arquivo OVAL usado na verificação. A execução é baseada na proposta de uso do sistema operacional, estão disponíveis verificações como servidor ou estação de trabalho em diferentes níveis usando a linha base alinhada a definição do CIS *Benchmark* aplicada para Ubuntu 20.04 LTS.

Em execução realizada no SO, o arquivo `ssg-ubuntu2004-ds.xml` fornece os seguintes perfis para verificação: CIS Ubuntu 20.04 *Level* 1 *Server Benchmark*; CIS Ubuntu 20.04 *Level* 1 *Workstation Benchmark*; Ubuntu 20.04 *Level* 2 *Server Benchmark*; e CIS Ubuntu 20.04 *Level* 2 *Workstation Benchmark*. Por padrão, o arquivo `ssg-ubuntu2004-ds.xml`, é um tipo *DataStream* de origem SCAP[8], este faz referência a arquivos como `ssg-ubuntu2004-oval.xml` e `ssg-ubuntu2004-ocil.xml` que fornece capacidades de verificação automática. Estes arquivos constituem a base que alinha-se ao *Center for Internet Security* Ubuntu 20.04 LTS *Benchmark* v1.0.0. É possível a customização de verificações, usando o arquivo `ssg-ubuntu2004-xccdf.xml` configurando nas propriedades *Referenced check files* os arquivos que devem ser usados durante a análise.

[8] `https://static.open-scap.org/openscap-1.3/oscap_user_manual.html`

Postas as considerações, o SO foi analisado com uso do arquivo `ssg-ubuntu2004-ds.xml` para verificação de linha base de segurança usando o CIS *benchmark*. Usando o perfil de verificação CIS Ubuntu 20.04 *Level* 1 *Workstation Benchmark* a verificação produziu o resultado na Figura 6.3. Foram verificadas 229 regras, destas 112 foram aprovadas e 109 falharam. Das falhas, 3 regras são classificadas como gravidade alta, são elas: Impedir *login* em contas com senha vazia; Definir a senha do carregador de inicialização no grub2; e Desativar o acesso SSH por meio de senhas vazias.

Figura 6.3: Verificação OpenSCAP para perfil de estação de trabalho nível 1.

Fonte: Captura de tela, artefado da ferramenta OpenSCAP.

A verificação produz um relatório e apresenta informações sobre todas as 229 regras testadas. Para as regras de alta gravidade, além da severidade da regra, apresenta os documentos de controles de segurança que são referência para criação da regra, a descrição, justificativa e alertas relacionados. Para as 3 regras de alta gravidade que falharam. O relatório evidencia as seguintes descrições:

- **Impedir *login* em contas com senha vazia:** Se uma conta estiver configurada para autenticação por senha, mas não tiver uma senha atribuída, poderá ser possível fazer *login* na conta sem autenticação. Remova todas as instâncias do `nullok` em `/etc/pam.d/common-password` para evitar *logins* com senhas vazias.

- **Definir a senha do carregador de inicialização no grub2:** O carregador de inicialização grub2 deve ter uma conta de superusuário e proteção por senha habilitadas para proteger as configurações de inicialização. Como as senhas em texto simples são

um risco à segurança, gere um *hash* para a senha executando o seguinte comando: `# grub2-mkpasswd-pbkdf2`. Quando solicitado, digite a senha que foi selecionada. Usando o *hash* da saída, modifique o arquivo `/etc/grub.d/40_custom` com o seguinte conteúdo: `set superusers="boot"password_pbkdf2 boot grub.pbkdf2.sha512.VeryLongString`.

NOTA: a conta e senha do superusuário do `bootloader` devem ser diferentes da conta e senha root. Depois que a senha de superusuário for adicionada, atualize o grub.cfg executando: `update-grub`.

- **Desative o acesso SSH por meio de senhas vazias:** Proibir *login* SSH com senhas vazias. A configuração SSH padrão desativa *logins* com senhas vazias. A configuração apropriada será usada se nenhum valor for definido para `PermitEmptyPasswords`. Para proibir explicitamente o *login* SSH de contas com senhas vazias, adicione ou corrija a seguinte linha em `/etc/ssh/sshd_config`: `PermitEmptyPasswords no` . Quaisquer contas com senhas vazias devem ser desativadas imediatamente e a configuração do *Pluggable Authentication Modules* PAM deve impedir que os usuários possam atribuir senhas vazias a si mesmos.

A conformidade é estabelecida quando a regra atribuída a controles de segurança é aprovada. A pontuação indicando percentual de conformidade foi de 61,53% de regras aprovadas após a verificação.

Foi testada a execução usando o perfil Ubuntu 20.04 *Level* 2 *Server Benchmark* com o arquivo `ssg-ubuntu2004-ds.xml` no mesmo SO. Desse modo, o resultado encontrou 123 falhas, contra 109 da verificação anterior. As falhas de severidade alta se mantiveram as mesmas do primeiro relatório. Foram encontradas 6 novas falhas de severidade média e 8 de saveridade baixa. As falhas de severidade média foram:

- Certifique-se de que `/var/tmp` esteja localizado em uma partição separada;
- Certifique-se de que o subsistema de auditoria esteja instalado;
- Desativar suporte *Datagram Congestion Control Protocol* (DCCP);
- Desativar suporte *Stream Control Transmission Protocol* (SCTP);

- Desativar carregamento `Modprobe` do driver de armazenamento *Universal Serial Bus* (USB);
- Desativar encaminhamento SSH TCP.

Nas falhas de severidade baixa o relatório reportou:

- Certifique-se de que `/home` esteja localizado em uma partição separada;
- Certifique-se de que `/var` esteja localizado em uma partição separada;
- Certifique-se de que `/var/log` esteja localizado em uma partição separada;
- Certifique-se de que `/var/log/audit` esteja localizado em uma partição separada;
- Desativar suporte *Reliable Datagram Sockets* (RDS);
- Desativar suporte *Transparent Inter-Process Communication* (TIPC); Estender o limite do *backlog* de auditoria para o daemon de auditoria;
- Habilitar auditoria para processos iniciados antes do *daemon* de auditoria.

A justificada análise mais criteriosa para o perfil Ubuntu 20.04 Level 2 *Server Benchmark* fez com que o percentual de conformidade caísse em comparação com a verificação anterior, ficando em 60,02% como na Figura 6.4.

Figura 6.4: Verificação OpenSCAP para perfil de servidor nível 2.

Fonte: Captura de tela, artefado da ferramenta OpenSCAP.

Outros testes realizados com outros perfis retornaram os seguinte resultados: CIS Ubuntu 20.04 *Level* 1 *Server Benchmark* retornou 62.88% de regras aprovadas e CIS Ubuntu 20.04 *Level* 2 *Workstation Benchmark* retornou 59.65% de regras aprovadas.

O arquivo OVAL fornecido pela Canonical foi usado em execução no OpenSCAP e relatou 1578 verificações CPEs. A verificação contempla a busca por CVEs conhecidos relacionados aos componentes que constituem o SO e verificações LSN *Linux Security Notification* são realizadas como exemplo da LSN-0092-1[9], observada na figura 6.5. Em posterior análise, o mesmo SO foi verificado com arquivo OVAL obtido da mesma fonte. Observa-se na Figura 6.6 que novas vulnerabilidades foram identificadas para o SO. Portanto as Figuras 6.5 e 6.6 refletem a importância de um procedimento contínuo de gerenciamento de vulnerabilidades.

OVAL System Characteristics Generator Information

Schema Version	Product Name	Product Version	Date	Time
5.11.1	cpe:/a:open-scap:oscap	1	2024-01-09	08:13:31

OVAL Definition Results

✕ ✓ Error Unknown Other

ID	Result	Class	Reference ID	Title
oval:com.ubuntu.focal:def:100	true	inventory		Check that Ubuntu 20.04 LTS (focal) is installed.
oval:com.ubuntu.focal:def:991000000	false	patch	[LSN-0099-1], [CVE-2023-42752], [CVE-2023-3777], [CVE-2023-3609], [CVE-2023-42753], [CVE-2023-4623], [CVE-2023-3567], [CVE-2023-40283], [CVE-2023-5197], [CVE-2023-3776], [CVE-2023-4622], [CVE-2023-4004], [CVE-2023-34319], [CVE-2022-3643], [CVE-2023-31436]	LSN-0099-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:981000000	false	patch	[LSN-0098-1], [CVE-2023-3776], [CVE-2023-3609], [CVE-2023-21400], [CVE-2023-4004], [CVE-2023-3777], [CVE-2023-40283], [CVE-2023-3090], [CVE-2023-3567]	LSN-0098-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:971000000	false	patch	[LSN-0097-1], [CVE-2023-31248], [CVE-2023-32629], [CVE-2023-3090], [CVE-2023-3390], [CVE-2023-35788], [CVE-2023-35001], [CVE-2023-3389]	LSN-0097-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:961000000	false	patch	[LSN-0096-1], [CVE-2023-31436], [CVE-2023-35001], [CVE-2023-30456], [CVE-2023-31248], [CVE-2023-1380]	LSN-0096-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:951000000	false	patch	[LSN-0095-1], [CVE-2023-32233], [CVE-2023-2612], [CVE-2023-0386], [CVE-2023-1872], [CVE-2023-1380], [CVE-2023-31436]	LSN-0095-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:941000000	false	patch	[LSN-0094-1], [CVE-2023-1281], [CVE-2023-0468]	LSN-0094-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:931000000	false	patch	[LSN-0093-1], [CVE-2023-0461], [CVE-2023-0179]	LSN-0093-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:921000000	false	patch	[LSN-0092-1], [CVE-2022-42896], [CVE-2022-4378], [CVE-2022-43945]	LSN-0092-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:911000000	false	patch	[LSN-0091-1], [CVE-2022-42719], [CVE-2022-41222]	LSN-0091-1 -- Kernel Live Patch Security Notice

Figura 6.5: Verificação OpenSCAP OVAL Canonical.

Fonte: Captura de tela, artefado da ferramenta OpenSCAP.

[9] `https://ubuntu.com/security/notices/LSN-0092-1`

OVAL System Characteristics Generator Information

Schema Version	Product Name	Product Version	Date	Time
5.11.1	cpe:/a:open-scap:oscap	1	2024-04-08	15:05:44

OVAL Definition Results

✕ ✓ Error Unknown Other

ID	Result	Class	Reference ID	Title
oval:com.ubuntu.focal:def:67021000000	true	patch	[USN-6702-1], [CVE-2023-23000], [CVE-2023-23004], [CVE-2024-1086], [CVE-2024-24855]	USN-6702-1 -- Linux kernel vulnerabilities
oval:com.ubuntu.focal:def:66811000000	true	patch	[USN-6681-1], [CVE-2021-44879], [CVE-2023-22995], [CVE-2023-4244], [CVE-2023-51779], [CVE-2023-51780], [CVE-2023-51782], [CVE-2023-6121], [CVE-2024-0340]	USN-6681-1 -- Linux kernel vulnerabilities
oval:com.ubuntu.focal:def:66481000000	true	patch	[USN-6648-1], [CVE-2023-51781], [CVE-2023-6915], [CVE-2024-0565], [CVE-2024-0646]	USN-6648-1 -- Linux kernel vulnerabilities
oval:com.ubuntu.focal:def:100	true	inventory		Check that Ubuntu 20.04 LTS (focal) is installed.
oval:com.ubuntu.focal:def:991000000	false	patch	[LSN-0099-1], [CVE-2023-42752], [CVE-2023-3777], [CVE-2023-3609], [CVE-2023-42753], [CVE-2023-4623], [CVE-2023-3567], [CVE-2023-40283], [CVE-2023-5197], [CVE-2023-3776], [CVE-2023-4622], [CVE-2023-4004], [CVE-2023-34319], [CVE-2022-3643], [CVE-2023-31436]	LSN-0099-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:981000000	false	patch	[LSN-0098-1], [CVE-2023-3776], [CVE-2023-3609], [CVE-2023-21400], [CVE-2023-4004], [CVE-2023-3777], [CVE-2023-40283], [CVE-2023-3090], [CVE-2023-3567]	LSN-0098-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:971000000	false	patch	[LSN-0097-1], [CVE-2023-31248], [CVE-2023-32629], [CVE-2023-3090], [CVE-2023-3390], [CVE-2023-35788], [CVE-2023-35001], [CVE-2023-3389]	LSN-0097-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:961000000	false	patch	[LSN-0096-1], [CVE-2023-31436], [CVE-2023-35001], [CVE-2023-30456], [CVE-2023-31248], [CVE-2023-1380]	LSN-0096-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:951000000	false	patch	[LSN-0095-1], [CVE-2023-32233], [CVE-2023-2612], [CVE-2023-0386], [CVE-2023-1872], [CVE-2023-1380], [CVE-2023-31436]	LSN-0095-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:941000000	false	patch	[LSN-0094-1], [CVE-2023-1281], [CVE-2023-0468]	LSN-0094-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:931000000	false	patch	[LSN-0093-1], [CVE-2023-0461], [CVE-2023-0179]	LSN-0093-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:921000000	false	patch	[LSN-0092-1], [CVE-2022-42896], [CVE-2022-4378], [CVE-2022-43945]	LSN-0092-1 -- Kernel Live Patch Security Notice
oval:com.ubuntu.focal:def:911000000	false	patch	[LSN-0091-1], [CVE-2022-42719], [CVE-2022-41222]	LSN-0091-1 -- Kernel Live Patch Security Notice

Figura 6.6: Verificação atualizada OpenSCAP OVAL Canonical.

Fonte: Captura de tela, artefado da ferramenta OpenSCAP.

As capacidades de verificação de padrões de configuração recomendados pela indústria, bem como configurações incorretas e *scanner* de vulnerabilidades tornam a ferramenta OpenSCAP elegível para satisfazer recomendações de segurança no escopo definido. A execução da ferramenta permite a obtenção de arquivos de saída com potencial para evidenciar o cumprimento de recomendações de segurança. Por ser operada por CLI, comandos como o exemplo abaixo podem ser executados mediante *scripts* que compõem processos de execução automatizada.

```
oscap xccdf eval --profile xccdf_org.ssgproject.
    content_profile_cis_level2_server --results-arf report.xml --report
    report.html ssg-ubuntu2004-ds.xml
```

Lynis

Com a execução da ferramenta Lynis usando o arquivo `default.prf` padrão de sua instalação. O resultado exibe o que é checado durante a execução do *scanner* de segurança, a ferramenta verificou detalhes como 224 testes de desempenho, 54 verificações de *hardening*, verificações de *firewall*, bem como execução do módulo Lynis que executa auditoria de segurança e *scanner* de vulnerabilidades como mostra a Figura 6.7, informações exibidas ao final de uma execução.

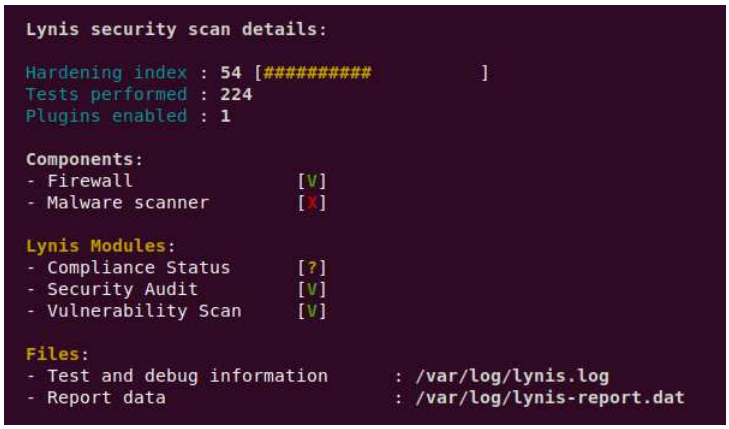


Figura 6.7: Verificação Lynis no Ubuntu 20.04 LTS.

Fonte: Captura de tela, execução da ferramenta Lynis.

A execução do Lynis reportou status como *ok* ou *warning* para maioria dos componentes verificados. Desse modo, os *status warning* verificados apontam que devem ser analisados itens como identificação da presença de GRUB2 com aviso para verificação da proteção por senha, relacionados às descobertas em componentes boot e serviços, bem como verificando a autenticação no modo de usuário único do Linux relacionado à análise em usuários, grupos e autenticação. Também há avisos para realização de checagem nas configurações relacionadas a rede, reportando especificamente os componentes que devem ser observados.

As verificações realizadas pelo Lynis permitem identificar componentes do sistema que estão alinhadas com o esperado em definições de segurança, bem como os que demandam análise para adequação. A execução contempla análise em sistemas de arquivo, dispositivos USB, *kernel*, rede, *firewall*, servidor *web*, configuração e suporte SSH, Suporte SNMP *Simple Network Management Protocol*, base de dados, Servidor LDAP, permissões de arquivo entre muitos outros testes. Sua natureza de execução em modo CLI, bem como práticas que executam verificações importantes de segurança no sistema operacional, contemplando a descoberta de vulnerabilidades e direcionando para configurações mais adequadas de segurança, permitem implementar o Lynis como alternativa válida no suporte a condições estabelecidas pelas recomendações de segurança.

Peass

As verificações realizadas através do Peass, em sua versão para Linux conhecida como Linpeass. Poderam fornecer detalhes sobre a saúde do SO pela ótica de problemas de segurança relacionados a escalonamento de privilégios. Desse modo, a verificação identificou 7 CVEs que podem ser potencialmente exploradas no SO analisado. A seguir são expostos os detalhes relacionados aos CVEs, embora a ferramenta identifique vulnerabilidades conhecidas, o escoreCVSS não é fornecido pelo Linpeass. Os CVEs encontrados foram:

- CVE-2019-7304: (Escore CVSS 9.8) Canonical snapd antes da versão 2.37.1 executava incorretamente a validação do proprietário do soquete, permitindo que um invasor executasse comandos arbitrários como root. Este problema afeta: Versões canônicas ajustadas anteriores a 2.37.1.

  1. `-rwsr-xr-x 1 root root 129K Sep 15 20:13 /snap/snapd/20290/usr/lib/snapd/snap-confine --> Ubuntu_snapd<2.37_dirty_sock_Local_Privilege_Escalation (CVE-2019-7304)`
  2. `-rwsr-xr-x 1 root root 129K Nov 29 14:54 /snap/snapd/20671/usr/lib/snapd/snap-confine --> Ubuntu_snapd<2.37_dirty_sock_Local_Privilege_Escalation (CVE-2019-7304)`
  3. `-rwsr-xr-x 1 root root 144K May 29 2023 /usr/lib/snapd/snap-confine --> Ubuntu_snapd<2.37_dirty_sock_Local_Privilege_Escalation (CVE-2019-7304);`

- CVE-2022-2586: (Escore CVSS 7.8) Foi descoberto que um objeto ou expressão NFT *Netfilter Table* poderia fazer referência a um conjunto NFT em uma tabela NFT diferente, levando a um uso após liberação assim que a tabela fosse excluída.
  `Tags: [ ubuntu=(20.04) ]kernel:5.12.13;`

- CVE-2021-4034: (Escore CVSS 7.8) vulnerabilidade de escalonamento de privilégios locais foi encontrada no utilitário pkexec do polkit.

`Tags: ubuntu=10|11|12|13|14|15|16|17|18|19|20|21 ],debian=7|8|9|10|11,fedora,manjaro;`

- CVE-2021-3156: (Escore CVSS 7.8) Sudo antes de 1.9.5p2 contém um erro *off-by-one* que pode resultar em um *buffer overflow* baseado em *heap*, que permite escalonamento de privilégios para root via "`sudoedit -s`"e um argumento de linha de comando que termina com um único caractere de barra invertida.
  `Tags: mint=19,[ ubuntu=18|20 ], debian=10;`
  `Tags: centos=6|7|8,[ ubuntu=14|16|17|18|19|20 ], debian=9|10;`

- CVE-2021-22555: (Escore CVSS 7.8) Uma gravação fora dos limites de *heap* afetando o Linux desde a v2.6.19-rc1 foi descoberta em `net/netfilter/x_tables.c`. Isso permite que um invasor obtenha privilégios ou cause um DoS (por meio de corrupção de memória *heap*) por meio do espaço de nome de usuário.
  `Tags: [ ubuntu=20.04 ]kernel:5.8.0-*;`

- CVE-2022-32250: (Escore CVSS 7.8) `net/netfilter/nf_tables_api.c` no kernel Linux até 5.18.1 permite que um usuário local (capaz de criar namespaces de usuário/rede) aumente privilégios para root porque uma verificação `NFT_STATEFUL_EXPR` incorreta leva a um uso após liberação.
  `Tags: ubuntu=(22.04)kernel:5.15.0-27-generic;`

- CVE-2017-5618: (Escore CVSS 7.8) A tela GNU anterior à versão 4.5.1 permite que usuários locais modifiquem arquivos arbitrários e, consequentemente, ganhem privilégios de root, aproveitando a verificação inadequada de permissões de arquivos de *log*.

Além da verificação de vulnerabilidades CVE, o Linpeass estende a verificação em uma série de componentes imprescindíveis. A análise tem potencial de detectar problemas de escalonamento de privilégios verificando processos, Crons, Serviços, *Sockets*, sistema de arquivos, usuários, super usuários, softwares instalados e muitas outras análises. Linpeass se mostra uma alternativa para uso em *hardening* de SO. Para o cenário de implementação,

e proposta, sua capacidade de detecção está alinhada com recomendações de segurança que embasam o gerenciamento de vulnerabilidades. É uma ferramenta usada em CLI, permite que a verificação seja salva em um arquivo `.txt`, o que provisiona os atributos necessários para um processo automatizado de utilização e verificação de resultados.

Trivy

As capacidades desempenhadas em verificações realizadas no cenário de plataforma consideraram executar verificações realizadas em um *cluster* Kubernetes com as ferramentas Trivy e Kubescape. Em análise, as ferramentas têm funcionalidades similares de verificações, no qual trata importantes componentes da arquitetura do Kubernetes. Portanto, a julgar por critérios de popularidade definidos na Seção 4.2.2, a ferramenta Trivy foi tida como prioritária e Kubescape atuando como ferramenta complementar, estendendo funcionalidades não praticadas por Trivy.

O Kubernetes tem, como parte de sua arquitetura, alguns componentes, como o *control plane*, kube-apiserver, etcd, kubelet e kube-proxy. Também inclui elementos Kubernetes *Objects*, como Pods, *Namespaces, Services, Secrets* e *Jobs*. As ferramentas de segurança atuam na verificação de vulnerabilidades conhecidas e *scanner* de configurações, detectando problemas a serem tratados e dando suporte ao processo de gerenciamento de vulnerabilidades. Trivy atua sincronizando seu banco de dados local com informações de fontes como o CVE. Após sincronizadas as informações sobre vulnerabilidades, as verificações ocorrem de acordo com a amplitude e profundidade necessárias a uma organização que realiza verificações em seus sistemas.

Dentre as capacidades do Trivy, podemos identificar o conjunto importante de componentes utilizados pelo Kubernetes gerando o Kubernetes *Bill Of Materials* (KBOM, em tradução, Lista de Materiais do Kubernetes). Semelhante ao SBOM, essa alternativa permite identificar todos os componentes do *cluster*, como kubelet, kubectl, etcd, api-server, entre outros. Com o arquivo detalhando a composição do Kubernetes, podemos verificar informações como o tipo de CNI utilizado e as versões de cada componente. Indo além, a ferramenta possui capacidades testadas em verificações de imagens de contêineres, repositórios, *cluster*, *namespaces*, configurações incorretas, *filesystem* e muitas outras formas de verificação. Como exemplo, os resultados abaixo destacam verificações executadas no

cluster Kubernetes com o *control plane* e dois *nodes*. `Client Version: v1.28.2, Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3` e `Server Version: v1.28.4`.

- usr/bin/kube-controllers (gobinary)
  - Total: 8 (UNKNOWN: 0, LOW: 0, MEDIUM: 5, HIGH: 3, CRITICAL: 0)
- toolautomation2 (gobinary)
  - Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 1, HIGH: 0, CRITICAL: 0)
- toolautomation3 (gobinary)
  - Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 1, HIGH: 0, CRITICAL: 0)

O comando `Trivy k8s cluster –scanners vuln –report all` executado retorna uma série de resultados referentes a vulnerabilidades identificadas na verificação do *cluster*, porém, também podem ser executadas verificações de configurações. O comando sudo `trivy config /etc/kubernetes/manifests/` é um exemplo de verificação de configurações. Como resultado, um exemplo de retorno é observado na configuração do kube-scheduler: "*High*: Pod 'kube-scheduler' *should* `not` *set* `'spec.template.spec.hostNetwork'` *to* `true`. *Sharing the host's network namespace permits processes in the pod to communicate with processes bound to the host's loopback adapter. See* `https://avd.aquasec.com/misconfig/ksv009`"

Kubescape

As capacidades de verificação da ferramenta Kubescape estão dispostas de dois modos. Após a instalação da ferramenta, alguns comandos e funções já estão disponíveis para utilização, porém outros, requerem a instalação de um operador Kubescape dentro do *cluster*. Após a instalação da ferramenta, é possível realizar as seguinte operações:

- Não requer um operador instalado no *cluster*
  - Verificação de linha base de segurança.
  - Verificação da estrutura (requer controles disponíveis em regolibrary).

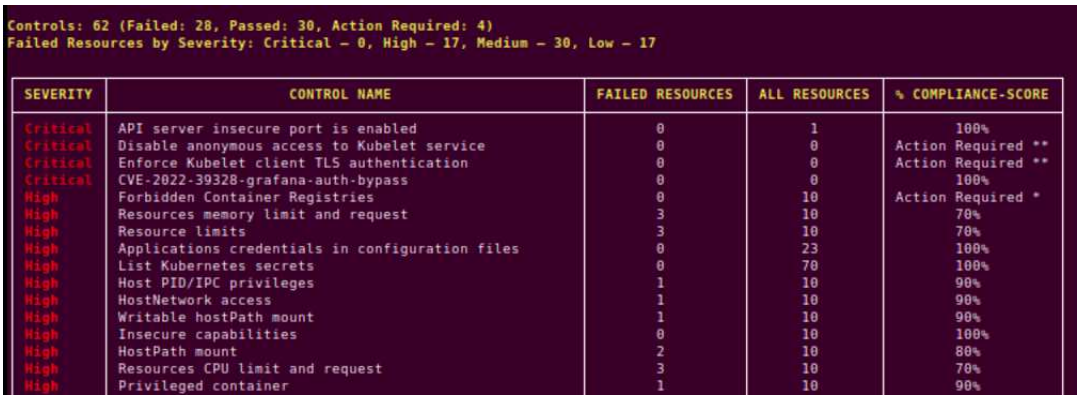
  - Verificação de controles (requer controles disponíveis em regolibrary).
  - Scan de JSON e YAML.
  - Scan de repositório.
  - Scan de diretórios.

- Requer instalação de operador no *cluster*
  - Verificador de *host* validar configurações executadas nos nós.
  - Configurações incorretas.
  - Scanner de vulnerabilidades de imagem retornando CVEs.

Um detalhe sobre o Kubescape define que o *scanner* de vulnerabilidades da ferramenta utiliza os mecanismos do Grype[10] para verificar bancos de dados de vulnerabilidades conhecidas. O Kubescape permite a verificação geral do cluster executando um *scanner* de segurança de linha de base. Dessa forma, são identificadas configurações que demandam alterações para o fortalecimento da segurança. O comando usado para realizar a verificação é `kubescape scan`. Podemos implementar parâmetros para customizar a saída do scanner.

Como visto anteriormente na Seção 6.1.3, o Kubescape possui um repositório de políticas utilizadas nas verificações, porém com comandos básicos podem ser realizadas verificações que podem ser extenções usadas em conjunto com Trivy para potencializar o poder de verificação em um processo de gerenciamento de vulnerabilidades. O comando `kubescape scan -verbose -format json -format-version v2 -output scanner-kubescape.json` realiza o *scanner* detalhado utilizando o parâmetro `--verbose` e define o formato e nome do arquivo de saída e tem seu resultado exposto na Figura 6.8.

[10]Um *scanner* de vulnerabilidades para imagens de contêiner e sistemas de arquivos. Disponível em: `https://github.com/anchore/grype`



```
Controls: 62 (Failed: 28, Passed: 30, Action Required: 4)
Failed Resources by Severity: Critical — 0, High — 17, Medium — 30, Low — 17
```

SEVERITY	CONTROL NAME	FAILED RESOURCES	ALL RESOURCES	% COMPLIANCE-SCORE
Critical	API server insecure port is enabled	0	1	100%
Critical	Disable anonymous access to Kubelet service	0	0	Action Required **
Critical	Enforce Kubelet client TLS authentication	0	0	Action Required **
Critical	CVE-2022-39328-grafana-auth-bypass	0	0	100%
High	Forbidden Container Registries	0	10	Action Required *
High	Resources memory limit and request	3	10	70%
High	Resource limits	3	10	70%
High	Applications credentials in configuration files	0	23	100%
High	List Kubernetes secrets	0	70	100%
High	Host PID/IPC privileges	1	10	90%
High	HostNetwork access	1	10	90%
High	Writable hostPath mount	1	10	90%
High	Insecure capabilities	0	10	100%
High	HostPath mount	2	10	80%
High	Resources CPU limit and request	3	10	70%
High	Privileged container	1	10	90%

Figura 6.8: Verificação de linha base de configuração.

Fonte: Captura de tela, execução da ferramenta Kubescape.

O resultado também retorna o sumário quantificando a conformidade sobre o estado de configuração após a verificação. Isto é verificado considerando padrões observados por NSA e MITRE. Os valores para conformidade obtidos na verificação foram `FRAMEWORKS: All Controls (Compliance: 79.49%), NSA (Compliance: 68.27%), MITRE (Compliance: 78.21%).`

6.2.2 Execução em sistema em produção

O teste de ferramentas no ambiente em produção permite que este seja verificado e posteriormente estimado com métricas quantitativas de conformidade. Desse modo, a adequação a recomendações de segurança pode ser implementada de maneira a satisfazer ações e condições impostas pelas recomendações de segurança, adequando o sistema à conformidade com as recomendações do CIS *Controls* V8 e NIST *Cybersecurity Framework*.

O ambiente utilizado para execução de testes do cenário de aplicação fez uso de serviços do Laboratório de Sistemas Distribuídos (LSD) da Universidade Federal de Campina Grande (UFCG), bem como o site próprio da UFCG disponível em `https://portal.ufcg.edu.br`. Sendo assim, os primeiros testes foram realizados no site do LSD disponível em `https://www.lsd.ufcg.edu.br`. Classificamos as ferramentas utilizadas para verificação de serviços como DAST, onde a superfície de ataque é inicialmente mapeada para que testes realizados nos sistemas em uso possam ser executados. As ferramentas Nikto, SSLyze, CMSeek e Nmap são utilizadas seguindo esse princípio

de execução. A ferramenta Semgrep segue a abordagem SAST onde o teste de caixa branca realiza testes estáticos em nível de código.

CMSeeK

A execução da ferramenta CMSeek possibilitou a descoberta do *content management system* (CMS) utilizado para construção do site. A partir disto, é possível direcionar o *scanner* para o CMS detectado. A coleta de informações possibilitou identificar o uso do WordPress v4.4, plugin wp-filebase v3.4.0, theme lsd-website v4.4 e WordPress Deepscan reportando a licença disponível em `https://www.lsd.ufcg.edu.br//license.txt`. CMSeeK realizou a descoberta de *usernames*.

Após a detecção de um formulário de *login*, foram verificados os componentes wp-json e jetpack *public* API que não retornaram resultados. A verificação no componente WordPress *Author Parameter* retornou 15 *usernames*. A coleta permitiu a execução do ataque de força bruta com o módulo de ataque padrão do CMSeeK. A ferramenta prosseguiu com o ataque iterando entre os usuários testando todas as possibilidades disponíveis no módulo padrão, sem que o servidor impedisse a execução do ataque após um determinado número de tentativas falhas.

```
[i] Harvesting usernames from wordpress author Parameter
[*] Found user from redirection: andr***
[*] Found user from redirection: admin
[*] Found user from redirection: franc***iro
[*] Found user from redirection: naz***
...
[*] 15 Usernames were enumerated
```

Em outras verificações de segurança, não foram identificados problemas adicionais. Sendo assim, CMSeeK obteve a evidência sobre a vulnerabilidade que permite a descoberta de *usernames* e a inexistência de bloqueios por parte do servidor para tratar o grande número de requisições que evita ataques de força bruta.

Nikto

A ferramenta Nikto executada nas portas 80 e 443 permitiu a identificação dos seguinte itens com inadequação e potencial risco a segurança:

- Apache/2.4.41 parece estar desatualizado (atual é pelo menos 2.4.57),
- Falta do *Header x-Frame-Options*,
- Falta do *Header X-Content-Type-Options*,
- Redirecionamento encontrado,
- Falta do *Header Strict-Transport-Security*,
- Incompatibilidade do Nome do *Host* com o Certificado,
- Exposição de informações pelo arquivo `teste.php`,
- Identificação de arquivo `license.txt`.

O poder de verificação executado pela ferramenta Nikto contempla verificações relevantes de segurança, podemos destacar verificações como: descoberta de arquivos, configurações incorretas, injeção SQL e XSS, descoberta de vulnerabilidades, upload de arquivos e identificação de softwares.

Em execução realizada ao portal da UFCG disponível em `https://portal.ufcg.edu.br`. A ferramenta Nikto identificou alguns problemas de configuração incorretas, exposição de dados potencialmente comprometedores para segurança e 28 CVEs. O extenso relatório gerado após a execução aponta para necessidade de realização de diversas correções. As correções estão relacionadas a configurações incorretas, uso de componentes desatualizados, CWEs, exposição de informações da arquitetura potencialmente ofensiva à segurança e injeção de dados. Dentre os 28 CVEs, 3 seguem detalhados a seguir:

- CVE-2019-2725 (Escore CVSS 9.8) Vulnerabilidade no componente Oracle WebLogic Server do Oracle Fusion Middleware. A vulnerabilidade facilmente explorável permite que um invasor não autenticado com acesso à rede via HTTP

comprometa o Oracle WebLogic Server. Impacta na confidencialidade, integridade e disponibilidade. `https://portal.ufcg.edu.br:443/_async/AsyncResponseServiceJms?WSDL`;

- CVE-2002-1560 `index.php` no gBook 1.4 permite que invasores remotos ignorem a autenticação e obtenham privilégios administrativos definindo o parâmetro *login* como *true*. `https://portal.ufcg.edu.br:443/gb/index.php?login=true`;

- CVE-2001-1198 Permite que usuários locais sobrescrevam arquivos arbitrários e obtenham privilégios. `/admin/admin.php?adminpy=1: PY-Membres` 4.2 *may allow administrator access*. `https://portal.ufcg.edu.br:443/admin/admin.php?adminpy=1`.

A ferramenta mostrou potencial de detecção de problemas em ambiente real[11]. Sua utilização contempla a verificação de vulnerabilidade e configurações incorretas requeridas por recomendações de segurança. Além disso, a ferramenta é totalmente operada por CLI permitindo seu uso de forma automatizada.

SSLyze

Com a execução do SSLyze, é possível a verificação de SSL e TLS de modo rápido e eficiente. No site do LSD, o teste indicou a confiança do certificado reconhecido em várias CAs conhecidas. O teste indicou não suscetibilidade a vulnerabilidades OpenSSL, *Change Cipher Spec* CCS, Injection, Heartbleed e ROBOT. Inconformidades foram reportadas sobre a validade do certificado que está definido em 399 dias, a verificação é baseada na recomendação da Mozilla[12] que indica um número inferior a 365 dias deve ser praticado. A Mozilla ainda recomenda que cifras como *Advanced Encryption Standard* AES, CAMELLIA, ARIA, em combinação com *Rivest-Shamir-Adleman* (RSA), *Diffie-Hellman Ephemeral* (DHE) RSA, e *Elliptic Curve Diffie-Hellman Ephemeral* (ECDHE) RSA sejam rejeitadas. Na verificação o servidor as trata como suportadas.

[11] Relatório reportado ao Núcleo de Tecnologia da Informação (NTI) responsável pelo suporte.

[12] `https://ssl-config.mozilla.org/`

Nmap

A ferramenta Nmap é utilizada para descoberta de redes, portas abertas em *hosts*, auditorias de segurança e inventários de rede. Como extensão de suas capacidades de operação, a instalação convencional da ferramenta traz o diretório `/usr/share/nmap/scripts`. São disponibilizados *scripts* NSE para automatizar a realização de operações de verificação que fogem do uso convencional da ferramenta. Uma série de verificações podem ser realizadas em servidores *web*, DNS, *firewalls*, bancos de dados, verificações CVE entre outras.

Em execução de *scanner* utilizando os *scripts* disponíveis do Nmap para verificar a instância que executa o site do LSD, foi observada diferença de resultados obtidos através do uso da ferramenta Nikto no mesmo serviço. Para o site do LSD, a verificação realizada por Nikto aponta problemas como Servidor Apache desatualizado, sendo Apache/2.4.41 *appears to be outdated* (*current is at least* 2.4.57). Apache 2.2.34 *is the* EOL *for the* 2.x *branch*, falta do *header* `x-frame-options`, falta do *header* `Strict-Transport-Security` e arquivo `teste.php` expondo informações. Enquanto a verificação executada pelo Nmap retorna resultados complementares às verificações realizadas por Nikto.

A execução do Nmap retornou resultados como CVE-2005-3299 indicando vulnerabilidade no phpMyAdmin, além de permitir coleta de informações do WordPress e diversos componentes e versões de software utilizadas. Em outros aspectos de verificação, os *scripts* NSE permitem coleta de informações de servidores DNS e LDAP. Em testes realizados com uso do `dns-brute.nse`, é possível a identificação de hosts após execução do DNS *Brute-force hostnames* e direcionar verificações com *scripts* específicos para servidores específicos.

Verificações com uso de *scripts* NSE específicos do LDAP como ldap-rootdse, ldap-search, ldap-brute, ldap-novell-getpass, permitem coleta de informações e testes de segurança, como descoberta de nomes de usuários com uso do ldap-rootdse e ataque de força bruta com *script* próprio para a finalidade, como o exemplo no comando:

```
nmap -Pn -d --scan-delay 0.5s --script ldap-brute --script-args 'ldap.
   base="cn=****,dc=**,dc=**,dc=**,dc=**",passdb=rockyou.txt' 192.168.1.2
```

6.3 Coleta de evidências e determinação de conformidade

Definimos na Seção 5.2 que cada recomendação de segurança foi mapeada em ações e condições a serem satisfeitas. Partindo deste princípio, a coleta de evidências e definição de conformidade está diretamente relacionada às ações e condições que compõem um requisito. Definimos que a execução das ferramentas de segurança gera artefatos a no qual cada artefato relaciona-se ao predicado A "atende". Também foi definida uma evidência e, mais detalhes sobre evidências são encontrados na Seção 5.1. Em e é definida a atividade necessária para atender pelo menos uma condição c. Desta forma, para todo artefato a, existe pelo menos uma evidência e em que a evidência e é atendida pelo artefato a e para toda evidência e, existe pelo menos uma condição c em que a condição c é atendida pela evidência e.

$$\forall a \exists e A_{a,e} \wedge \forall e \exists c A_{e,c} \tag{6.1}$$

Uma condição que constitui um requisito é a diretriz que deve ser seguida e satisfeita. Neste sentido, estabelecemos o processo de execução das ferramentas para suporte ao gerenciamento de vulnerabilidades. Empregamos para o cumprimento de recomendações as condições de descoberta de vulnerabilidades r e configurações s dentro dos cenários observados e detalhados na Seção 4.3. Definimos então que as condições possuem fatores x para determinar a detecção de r e s. Determinamos o predicado P "implementar"para tratar dos fatores identificados nas condições e o predicado Q "desempenha gerenciamento de"para tratar do processo de gerenciamento de vulnerabilidades requerido, no qual P_x desempenha gerenciamento de r ou p_x desempenha gerenciamento de s

$$P_x \rightarrow (Q_{xr} \vee Q_{xs}) \tag{6.2}$$

Para definir uma evidência, consideramos refletir sentenças descritivas de atividades diretamente mencionadas ou a conduta para realizar uma atividade. Sendo assim, ao serem identificados os fatores relacionados às condições impostas, as evidências tornam-se claras e compreensíveis. Categorizamos evidências comuns a diversas condições, pelas quais são definidas como: *Scanner* de vulnerabilidade, *Scanner* de configurações, Frequência do *scanner*, Processo de automação, Especificações SCAP e Classificação de severidade.

A Tabela 6.8 demonstra com o exemplo das recomendações 7.1 e 7.5 do CIS *Controls* V8 como são decompostas ações, condições, evidências e atribuição de artefatos a partir de uma recomendação. Aplicando a lógica descrita na sentença (6.2), definimos o fator a x com propriedades relacionadas a condição e a constante individual r que representa vulnerabilidade cidata anteriormente. Desta forma, temos os seguinte resultados corresponentes a cada linha da tabela:

- Implementar processo para gestão de ativos desempenha gerenciamento de vulnerabilidade;
- Implementar documentação atualizada desempenha gerenciamento de vulnerabilidade;
- Implementar varreduras automáticas desempenham gerenciamento de vulnerabilidades;
- Implementar ferramenta de varredura desempenha gerenciamento de vulnerabilidade.

Rec.	**Ação**	**Condição**	**Evidência**	**Artefato**
7.1	Estabeleça e mantenha	Processo de gerenciamento de vulnerabilidades para ativos empresariais	Scanner de Vulnerabilidades	arquivo-scanner
	Revise e atualize	Documentação atualizada anualmente ou em mudanças significativas	Frequência de execução do scanner	arquivo-scanner
7.5	Realize	Varreduras automáticas de vulnerabilidades	Scanner de Vulnerabilidades; Processo de automação (compliance monitor)	arquivo-scanner; config.ini
	Use	Ferramenta de varredura de vulnerabilidades compatível com SCAP	Especificações SCAP. Ex: CVE, XCCDF, OVAL	arquivo-scanner

Tabela 6.8: Mapeamento de evidências de conformidade.

Destacamos que nem sempre um artefato gerado por ferramentas de segurança irá satisfazer uma evidência, neste sentido os possíveis artefatos mapeados foram: arquivo-scanner,

config.ini e padrão SCAP. Portanto, arquivo-scanner se refere ao resultado produzido pelas ferramentas, config.ini faz parte do processo de automação pelo qual *scripts* de automação consultam config.ini para adotar padrões de execução, padrão SCAP como padrões CVE e arquivos XCCDF e OVAL, tratados nas execuções das ferramentas de segurança.

Os artefatos destacados e atribuídos a evidências são os indícios necessários para obtenção de conformidade. Considerando as recomendações classificadas nos níveis de alinhamento observadas na Tabela 6.5 na Seção 6.1.3, as recomendações que tratam de gerenciamento de vulnerabilidades em AL1 com exceção da recomendação 2.5 que trata de controles técnicos para software autorizados, 15.5, 15.6 e ID-SC1 que tratam de cadeia de suprimentos, às 9 demais recomendações se dividem em 21 requisitos compostos por ações e condições. Das 21 condições, 19 são satisfeitas pelas evidências atribuídas. As condições não satisfeitas referem-se a processos para remediação de vulnerabilidades tratada na recomendação 16.2 e aquisição de componentes de fontes confiáveis tratada na recomendação 16.5.

Considerando a classificação de alinhamento das recomendações de segurança nos níveis AL1, AL2 e AL3, para recomendações classificadas em AL1, as evidências satisfazem 90,47% das condições. Deste total, 73,68% das evidências tem como artefato o arquivo-scanner gerado pelas ferramentas de segurança mediante sua execução. Para AL2, 58,33% das condições são satisfeitas pelas evidências. Para AL3 apenas 20% condições são satisfeitas pelas evidências. No sentido de obter maior porcentagem de conformidade para recomendações classificadas em AL2 e AL3, são necessários esforços adicionais que estão fora do escopo de atuação.

Com finalidade de tornar evidente os aspectos praticados para determinar o nível de conformidade, requeremos a aplicação das métricas definidas na Seção 5.2. Inicialmente, a expressão quantitativa de uma recomendação permite mensurar sua observância por meio de evidências que satisfazem as condições postas por uma recomendação. Em outro ponto de verificação, tratamos da capacidade para determinar quais vulnerabilidades têm potencial para comprometimento da segurança em aspectos de confidencialidade, integridade e disponibilidade. Para isto, definimos o emprego do MVN também descrito na Seção 5.2.

Em testes realizados no LSD, um nó especificamente faz o *deploy* de produção. Este nó é responsável por hospedar *frontend*, *backend* e alguns serviços. É determinado que o nó utilize rede para comunicação entre aplicações locais e serviços externos, como Kafka, Mon-

goDB e serviços auxiliares. Obtendo a MVN por meio do formulário definido, obtivemos a *string* `AV:{N,A,L}/AC:H/PR:L/UI:R/S:U/C:H/I:L/A:H` que define os valores de explorabilidade e impacto.

Considerando verificações de SO como cenário de interesse, ferramentas de *scanner* foram executadas para identificar possíveis vulnerabilidades de segurança. O resultado obtido por OpenSCAP e Lipeass reportou vulnerabilidades CVE potencialmente nocivas ao nó. As vulnerabilidades identificadas seguem listada abaixo:

- CVE-2016-9840: AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H,
- CVE-2016-9841: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H,
- CVE-2018-25032: AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H,
- CVE-2022-37434: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H,
- CVE-2022-32250: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H,
- CVE-2019-7304: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H,
- CVE-2019-13272: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H,
- CVE-2011-1485: NULL.

Tomando por base a *string* gerada pela MVN, podemos observar as métricas de explorabilidade e impacto seguindo as regras da Tabela 6.9. São definidas as condições pela qual uma vulnerabilidade é potencialmente prejudicial a um determinado nó verificado. Uma importante distinção para as métricas PR e UI deve ser tida, os valores definidos como *none* atribuem máxima severidade a estes aspectos para uma vulnerabilidade. Sendo assim, mesmo que a MVN defina valores diferentes em PR e UI, uma vulnerabilidade que possua valor *none* atribuído em PR ou UI a torna potencialmente prejudicial a segurança do nó.

Aplicando as regras definidas na Tabela 6.9 para os CVEs identificados na verificação de SO do nó, quase todas as vulnerabilidades são potencialmente nocivas a segurança deste nó com base no MVN definido. A vulnerabilidade CVE-2011-1485 não possui métrica CVSS definida. Constatamos através de análise amostral que CVEs anteriores a 2016 disponíveis no NVD não possuíam em sua maioria a modelagem CVSS definida para a vulnerabilidade.

Explorabilidade	Impacto	Efeito
AV $\wedge$ (AC $\vee$ PR $\vee$ UI $\vee$ S)	C $\vee$ I $\vee$ A	Explorabilidade $\wedge$ Impacto
0	1	0
1	0	0
1	1	1

Tabela 6.9: Efeitos das métricas de explorabilidade e impacto.

A mesma análise demosntra um crescimento da prática de modelagem CVSS após o ano de 2016, como observado na Figura B.2 no Apêndice B

Considerando a execução das ferramentas e os artefatos obtidos na verificação, temos uma conformidade de 73,68% com as recomendações de segurança. Para aumento do nível de conformidade, é preciso ampliar o processo de gerenciamento de vulnerabilidades para contemplar aspectos de nível organizacional. Propomos então convergir o processo de gerenciamento para atividades automatizadas. O propósito é abranger condições organizacionais para definir o tempo de identificação de vulnerabilidades, processo de relatar à parte responsável pelo tratamento dos relatórios de vulnerabilidade e varreduras automáticas de vulnerabilidades com capacidades autonomas para rastreamento de vulnerabilidades, classificações e estimativas de comprometimento da segurança do nó. Os procedimentos automatizados são definidos na proxima Seção.

6.4 Processo de automação

Procedimentos automatizados devem ser considerados para suporte a verificações contínuas de conformidade. A conformidade contínua, discutida na Seção 3.2.1, é requerida da perspectiva de verificações constantes para garantias de conformidade sem a utilização de intervenção humana em procedimentos de auditoria. Desse modo, propomos um modelo para implementação do *compliance monitor*, que modelo irá associar recomendações, ferramentas de segurança e processos automatizados de checagem. O conhecimento agregado ao longo deste documento, proporciona a elaboração de um modelo de gerenciamento de vulnerabilidades compatível com recomendações de segurança dos *frameworks* do CIS e NIST, para entregar conformidade contínua auditável e sem intervenção humana.

A Figura 6.9 representa um modelo que atua em ambiente de computação na nuvem

abrangendo o processo para manutenção do gerenciamento de vulnerabilidades. São observadas duas alternativas para iniciar uma determinada instância que seguem as especificações do modelo proposto. A alternativa 1 representa uma forma isolada de *boot* utilizando o *systemd*, a alternativa 2 é uma inicialização convencional de instância em um *cloud provider*. Para ambos os casos de inicialização, parâmetros essenciais para o funcionamento da instância são passados através de *cloud-init*. Sendo assim, para a alternativa 1 temos a customização da inicialização do SO. Com o uso do *Systemd*, propomos a criação de um *custom-target* para atender um determinado estado do SO durante sua inicialização. O *target* customizado pode ser definido como padrão do sistema com o comando `systemctl set-default custom-target`. O *custom-target* pode invocar *services* criados para realizar operações enquanto é executado.

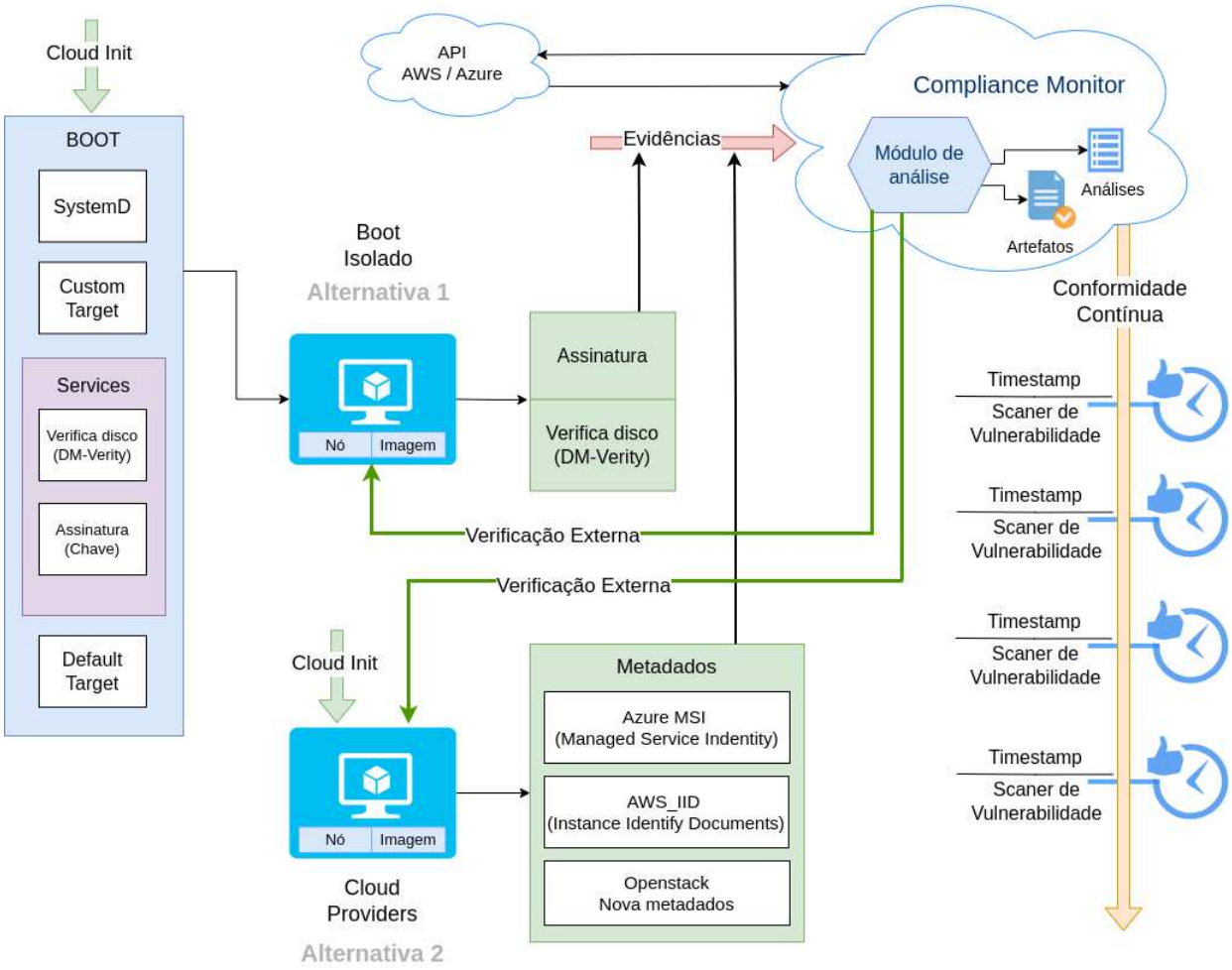


Figura 6.9: Modelo de automação com implementação do *Compliance Monitor*.

Fonte: De autoria própria.

Mesmo alterando o *target* padrão com o objetivo de obter um estado isolado no processo de boot, por exemplo sem conexões de rede, sem interação com usuários e sem os serviços de CLI ativos, o *systemd* inicializa alguns dos serviços essenciais para o funcionamento do SO. Consideramos que uma determinada imagem deve ser utilizada na instância, um *service* criado pode realizar assinaturas com o uso de chaves obtidas através de *cloud-init* e usadas em um momento de estado isolado no boot. Para garantir que serviços de conexão de rede e demais serviços citados anteriormente não devam estar operando durante o estado isolado, os parâmetros utilizados no *unit* do *service*, podem determinar *conflicts* para que serviços sejam interrompidos caso já estejam inicializados. Como exemplo temos `conflicts=NetworkManager-dispatcher.service`,

`NetworkManager-wait-online.service` e `NetworkManger.service`, indicando que os serviços de *network* devem ser interrompidos quando o serviço chamado pelo *custom-target* estiver em execução.

Ao assinar evidências que comprovem que a instância está utilizando a imagem adequada, o *service* deve executar a assinatura realizada com uso da chave passada por *cloud-init* e excluí-la ainda durante o estado isolado com *custom-target*. Durante o estado isolado, o *service* executa *dm-verity* para verificação de disco e utiliza a chave recebida para assinar o resultado. Logo após todos os *services* necessários serem executados, o *default-target* do SO pode ser chamado, assim, o processo de boot do sistema irá continuar após o fim da execução do *custom-target*. A verificação de disco assinada é passada para o *compliance monitor* na primeira comunicação da instância para que seja realizada a verificação da imagem que deve corresponder à uma imagem do inventário de imagens definido no *compliance monitor*. O *compliance monitor* irá verificar a assinatura para garantir a autenticidade da operação e o *hash* correspondente da imagem no seu inventário.

A inicialização de instâncias na alternativa 2 segue o processo de boot convencional, sem alterações no processo de boot do SO. A coleta de evidências sobre a instância é realizada através de serviços de metadados. As informações são passadas da instância para o *compliance monitor*, que irá comprovar se a instância está usando uma imagem do inventário consultando a API do *cloud provider*.

Para inicialização de instâncias utilizando a alternativa 2, propomos passar os parâmetros necessários através de *cloud-init*. A coleta de evidências fica a cargo da utilização de metadados disponibilizados por *cloud providers*, obtidos através das próprias funções disponibilizadas pelo ambiente de nuvem. Tomando como exemplo a AWS, no AWS_IID[13] (*Instance Identity Document*) os dados de identificação da instância são obtidos através do IMDS (*Instance Metadata Service*)

O *compliance monitor* é o responsável por verificar as imagens utilizadas nas instâncias, armazenar um inventário de imagens verificadas e realizar *scanners* de vulnerabilidade continuamente como parte do processo de gerenciamento de vulnerabilidades. Por via de regra, as instâncias devem utilizar apenas imagens do inventário de imagens. Este é composto por

[13] `https://docs.aws.amazon.com/pt_br/AWSEC2/latest/WindowsGuide/instance-identity-documents.html`

imagens verificadas por ferramentas de segurança do SO OpenSCAP, Lynis e Linpeas. Uma instância iniciada deve enviar as evidências necessárias para o *compliance monitor*. Isto inclui a chave pública correspondente a chave enviada para a instância com *cloud-init*, que é usada pelo *compliance monitor*, para constatação da autenticidade da verificação de disco. Constatada a autenticidade como uma declaração válida, o resultado da verificação de disco da instância é comparado com as informações correspondentes no inventário de imagens.

Após verificação das evidências enviadas pela instância, o *compliance monitor* é responsável pela verificação contínua de vulnerabilidades. O *compliance monitor* deve armazenar o endereço IP da instância junto ao registro de imagem utilizada. A execução automática das verificações é guiada por pré-configurações estabelecidas para operação do *compliance monitor*, estas seguirão a frequência das verificações de acordo com a demanda necessária. A verificação de vulnerabilidades deve considerar métricas nocivas à instância definidas pelo MVN via questionário estimativo. Durante verificações de segurança, os CVEs identificados por ferramentas de segurança são comparados com MVN para definir as vulnerabilidades potencialmente nocivas à instância em explorabilidade e impacto.

O processo de conformidade contínua realizado pelo *compliance monitor* nas instâncias é realizado através de verificação externa. A execução deve levar em conta as ferramentas de segurança para descoberta de vulnerabilidades nas camadas de aplicação, plataforma e SO. O *compliance monitor* opera a verificação nas instâncias conhecidas de acordo com a pré-configuração de frequência como definida no exemplo a seguir com o arquivo `config.ini`. Este é um arquivo de configuração para definir o comportamento dos *scripts* de automação das ferramentas de segurança. A frequência das verificações é determinada através de um CronJob no SO com sua frequência determinada no arquivo `config.ini`.

Código Fonte 6.1: Arquivo config.ini.

```
[network]
expected-port = 80, 443
target = 192.168.1.1, 192.168.1.5
server-dns = 0
server-web = 1
server-ldap = 0
database = 1
[scanner]
```

```
MVN = CVSS:3.1/AV:{N,A,L}/AC:H/PR:L/UI:R/S:U/C:H/I:L/A:H
cronjob = 0 0,12 * * *
[alert]
email = 1
email-address = raiff.silva@lsd.ufcg.edu.br
[cve]
exception = cve-2017-8976, cve-2018-4030
```

O arquivo de config.ini determina pré-configurações para tratar de *network*, scanner, alertas e vulnerabilidades. Em *network* são definidos os alvos de verificação, bem como a lista permissiva de portas em operação e detalhes sobre os serviços especificando tipos de servidores observados. Em scanner, uma análise utilizando a MVN determina quais vulnerabilidades são nocivas a uma instância em métricas de explorabilidade e impacto. Isto é realizado comparando a *string* MVN com as *strings* CVSS das vulnerabilidades identificadas.

Definições de alerta podem ser emitidos por email quando verificações são realizadas e vulnerabilidades são identificadas. Para as vulnerabilidades, uma lista de exceções CVE pode ser definida quando um operador humano julgar que uma vulnerabilidade identificada pode ser um falso positivo e ignorada mesmo com indicação de potencial risco à instância. Destacamos que operações desta natureza demandam consentimento de pessoal responsável pela segurança em uma organização que devem autorizar e confirmar a inclusão de uma exceção.

Ao realizar verificações nas instâncias, o *compliance monitor* armazena os artefatos resultantes da verificação. Cada ferramenta executada na análise compõe o conjunto das evidências sobre a segurança do sistema computacional. Partindo desta perspectiva, os resultados irão direcionar os esforços necessários para potencial correção de problemas de segurança detectados. Tratando os artefatos produzidos como unidades verificáveis, são tidas conclusões sobre os diferentes cenários analisados e discutidos na Seção 4.3. Cada artefato atribuído a um dos cenários reflete uma verificação de segurança crucial realizada pelas ferramentas de segurança.

Considerando o aspecto de usabilidade das ferramentas, sua execução parte do princípio da automação pelo qual o processo de verificação realizado é expresso no artefato resultante. Sendo fator significativo para o modo de operação desempenhado pelo *compliance monitor*, a análise dos artefatos deve ser realizada através de *scripts* que inferem informações sobre

problemas identificados. Ferramentas que adotam padrões de saída para expressar um resultado têm *scripts* compostos por poucas linhas de código para uma análise automatizada. Desta forma, o padrão SCAP contribui no cenário de gerenciamento de vulnerabilidades quando as especificações CVE, OVAL ou XCCDF estão sendo aplicadas por ferramentas de segurança.

Em ferramentas que reportam vulnerabilidades CVE como resultado das verificações de vulnerabilidade, observamos que *scripts* são relativamente simples com possível reuso de código para verificações em artefatos de diferentes ferramentas. Isso diminui a necessidade de reescrita e complexidade dos *scripts*. Este fator determinante para a automação de verificações é observado nas ferramentas Nikto, Nmap, Trivy, Kubescape, Linpeass, Semgrep e OpenSCAP. Por outra perspectiva, ferramentas como CMSeeK, SSLyze e Lynis, não assumem padronização nos artefatos de sua análise. Isto interfere na complexidade dos procedimentos automatizados, consequentemente verificações realizadas por estas ferramentas tornam inviável o processo de automação e reuso de código e demandam *scripts* mais elaborados para interpretação dos resultados.

Por consequência, ferramentas que não assumem uma padronização como as especificações SCAP em suas respostas, demandam maior esforço de interpretação dos artefatos, isto leva a possibilidade de verificações realizadas a partir de operadores humanos. A automação nestes casos limita-se apenas à execução da ferramenta, cabendo a atribuição de análise manual a um responsável por verificações de segurança. O modelo de automação deve considerar um prazo para realização da verificação manual por um operador humano. Este deve sinalizar a análise dos resultados antes que o prazo de validade da última verificação expire e comprometa a confiabilidade do gerenciamento de vulnerabilidades. O *compliance monitor* para estes casos deve verificar continuamente se os artefatos foram verificados manualmente e estão dentro do prazo de validade pré-determinado.

A proposta do *compliance monitor* atuando com verificações automáticas tem potencial benefício em ambientes que demandam monitoramento, como por exemplo integrar um sistema *watchdog* que executa detecção de intrusão, análise de *log*, análise tráfego de redes e detecção de vulnerabilidades, com uso do *compliance monitor*. Por outra perspectiva, a detecção de vulnerabilidades reportada pelo *compliance monitor* pode agregar com informações para tomada de decisão em mecanismos de segurança, como revogar permissões de

execução de softwares vulneráveis.

Tratando-se da expansão da conformidade com as recomendações, reforçar a segurança com a concepção de recomendações para controle de execução e suporte a softwares autorizados, definidos por recomendações do CIS *Controls* V8 na Tabela 6.10, tornando o SPIRE oportunamente agregado à proposta do *compliance monitor*. O SPIRE possibilita o provisionamento de identidades de agentes através do processo de atestação de nó, conforme definido na Seção 3.1.4.

Convenientemente, possuir informações a respeito dos nós na infraestrutura pode aumentar a robustez desse processo. Propõe-se estender a conformidade com as recomendações utilizando o modelo de automação proposto e ilustrado na Figura 6.9, em combinação com a atestação de nó viabilizada pelo SPIRE para garantir a confiabilidade de nós em ambientes na nuvem. Com essa colaboração, a utilização automatizada de ferramentas de segurança coopera com o provisionamento de identidades periodicamente, para atingir a conformidade contínua.

Referência	**Requisito**	**Descrição**
1.2, 2.3	Aborde ativos e softwares não autorizados	Garanta que exista um processo para lidar com ativos não autorizados e que softwares não autorizados sejam removidos do uso em ativos empresariais ou recebam uma exceção documentada. Reveja mensalmente ou com mais frequência.
2.2	Garanta que o software autorizado esteja atualmente suportado	Garanta que apenas softwares atualmente suportados sejam designados como autorizados no inventário de software. Documente uma exceção detalhando controles mitigadores e aceitação de risco residual para softwares não suportados necessários. Revise a lista de softwares para verificar o suporte pelo menos mensalmente ou com mais frequência.
2.5	Lista de permissões para softwares autorizados	Use controles técnicos para garantir que apenas softwares autorizados possam ser executados ou acessados.

Tabela 6.10: Definição de conformidade para uso do SPIRE.

Nessa proposta, as evidências coletadas através do *Compliance Monitor* são transformadas em seletores, que servem para, através da cooperação mútua entre agente e servidor SPIRE, alcançar uma atestação de nó bem sucedida. É importante destacar que a dinâmica entre agente e servidor, na atestação de nó, ocorre por meio da colaboração entre *plugins agent-side* e *server-side*. No lado do agente, são coletadas provas da identidade do nó, que

ainda precisam ser validadas. Enquanto isso, no lado do servidor, há descoberta de novos seletores e atribuição do SPIFFE ID adequado para o agente.

Os seletores produzidos nesse processo estão relacionados à instância onde o agente está sendo executado, e correspondem a informações sobre a identidade do nó e evidências de conformidade, na forma de verificações de vulnerabilidade e problemas de configuração. Para identificação do nó é utilizado o *hash* da imagem gerado pelo dm-verity, e assinado pela chave descartável usada no processo de boot isolado, possibilitando verificar se esse identificador único possui alguma correspondência no inventário de imagens. Para seletores sobre vulnerabilidade, é necessário não somente verificar a existência de vulnerabilidades no nó, mas também definir limites de tolerância.

Para traduzir esses limites em seletores, houve distinção na avaliação dos relatórios das ferramentas, separando-as em dois grupos: as que apresentam CVEs e as que utilizam análise própria para tratar de vulnerabilidades como descrito anteriormente. Para ferramentas que retornam CVEs, a proposta é utilizar o formulário definido na Seção 5.2 cujas respostas produzem o MVN. Essas definições são utilizadas para estabelecer o vetor CVSS de referência, que representa o limite de severidade não tolerável e será comparado (valor por valor) aos vetores CVSS das vulnerabilidades encontradas para produzir um seletor.

Em vista de atestar a confiabilidade do nó baseado em evidências coletadas pelo *compliance monitor*, foi elaborada uma lista de seletores suportados que resultaram dessa análise. A Tabela 6.11 a seguir contém os seletores, com um exemplo de utilização e descrição de cada um. Os seletores são do tipo “cm”, em virtude do papel fundamental do *compliance monitor* na sua definição.

Seletor	**Exemplo**	**Descrição**
Node IP	cm:node-ip:150.165.15.83	Endereço IP do nó no qual o agente está sendo executado
Image ID	cm:image-id:a2a8fd07889deb10b4cdf53c01637...	Hash raiz do sistema de arquivos da imagem calculado pelo dm-verity
Application tools	cm:app-tools:sslyze,nikto,cmseek,semgrep,nmap	Lista de ferramentas do cenário de aplicação que foram executadas
Platform tools	cm:platform-tools:trivy,kubescape	Lista de ferramentas do cenário de plataforma que foram executadas
SO *tools*	cm:so-tools:openscap,lynis,linpeas	Lista de ferramentas do cenário de sistema operacional que foram executadas
Acceptable CVSS	cm:unacceptable-cvss:false	Verificação de ocorrência de vulnerabilidade mediante MVN
Successful NSE *script*	cm:nse-success:false	Verificação de ocorrência de script NSE de vulnerabilidade bem-sucedido
CMSeeK *report analyzed*	cm:cmseek-analyzed:true	Confirmação da análise dos resultados do CMSeeK por um operador humano
Lynis *report analyzed*	cm:lynis-analyzed:true	Confirmação da análise dos resultados do Lynis por um operador humano
SSLyze *report analyzed*	cm:sslyze-analyzed:true	Confirmação da análise dos resultados do SSLyze por um operador humano

Tabela 6.11: Seletores baseados em conformidade contínua para atestação de nó no SPIRE.

Capítulo 7

Conclusão

7.1 Considerações finais

O presente estudo expõem uma abordagem com foco em processos automatizados para implementação de controles de segurança. Para este fim, consideramos os recursos necessários para operacionalização da conformidade com a finalidade de conter a necessária compreensão dos fatores envolvidos na definição de conformidade com base em recomendações de segurança. Propomos recomendações classificadas para abarcar integralmente a segurança em IaaS baseadas em *frameworks* internacionalmente reconhecidos e pertinentes à área de segurança. Nos estudos e análises realizadas as recomendações, detalhamos como são estruturalmente concebidas, atentando-se também ao potencial de abrangência para benefício da conformidade, considerando o mapeamento de diversos documentos realizados pelos *frameworks*.

Destacamos que, em essência, as recomendações de segurança refletem diretrizes para ações necessárias que conduzem ao fortalecimento da postura de segurança. Isso consiste em definições para condutas organizacionais com potencial auxílio à composição de políticas de segurança de uma organização e as práticas dirigidas à cibersegurança. Logo, implementações de cibersegurança exigem meios técnicos para efetivação. Concluímos que a natureza detalhada das recomendações revela informações necessárias para conduzir implementações de cibersegurança indicando o que deve ser feito, entretanto, é limitada às informações sobre como proceder com as implementações. Dessa forma, dedicamos esforços a implementação de cibersegurança requerida nas recomendações por intermédio de ferramentas de segurança

de código aberto.

Concluímos que dados extraídos dos textos descritivos e dos títulos das recomendações, utilizando algoritmos de aprendizado de máquina, podem ser usados para definir palavras-chave necessárias para a detecção de ferramentas por meio de processos de busca. Neste ponto, contribuímos com a indicação de ferramentas de segurança usadas para alcance da implementação de ciberseburança requerido por recomendações de segurança, também aprimoramos nossa proposta de recomendações para IaaS com nossa lista de ferramentas identificadas e aprovadas com base em critérios de inclusão e exclusão.

Em testes conduzidos nas ferramentas de segurança, direcionadas exclusivamente aos interesses em conduzir processos de gerenciamento de vulnerabilidades, foram analisados os critérios essenciais as ferramentas que contribuem com processos de verificação automática. Definimos que é possível estabelecer um processo de gerenciamento de vulnerabilidades a partir do uso de ferramenas de código aberto. Nos testes, detectamos o potencial das ferramentas na entrega de soluções para detecção de vulnerabilidades e constatamos que, embora as ferramentas não entreguem de verificação e descoberta em tempo real, verificação reiteradas cumprem com aspectos de verificação consistentes com recomendações de segurança. Comumente, o termo frequência é citado em recomendações que determinam estabelecimento de prazos para realização dos requisitos de segurança que convertem-se em conformidade quando acatados.

Inferimos que a conformidade é observada pela produção de artefatos resultantes da execução de ferramentas de segurança. Foi determinado que um artefato corresponde a evidências de efetivação para um determinado requisito condicional. A consumação de uma recomendação obedece a estrutura resultante da decomposição de uma recomendação de segurança resultando em ações, condições, evidência e por fim, o artefato que determina a efetivação para refletir conformidade.

A estrutura decorrente da decomposição de uma recomendação de segurança permite formalização dos recursos mensuráveis para a obtenção da conformidade. Consideramos então, mapear os artefatos gerados em um processo de gerenciamento de vulnerabilidades e consequentemente definir a porcentagem de satisfação das recomendações de segurança observadas para determinar os números que representam quantitativamente as métricas de conformidade. Para a linha das verificações, concluímos que vulnerabilidades potencial-

mente prejudiciais a um sistema, podem ser detectadas através de métodos que determinam métricas para explorabilidade e impacto de ativos organizacionais com nossa proposta de MVN, em específico uma instância é tornada verificável através de métricas para tornar viável a definição de vulnerabilidades potencialmente nocivas aos ativos considerados em uma averiguação.

Concluímos que conceber uma proposta de conformidade para acomodar procedimentos continuamente verificáveis, requer estender automação ao máximo de processos relacionados ao cumprimento de recomendações. Dessa forma, identificamos procedimentos que consistem em modelar uma série de resultados esperados para comportar procedimentos de verificação automática. Desenvolvemos o design de um modelo automatizado baseado na implementação do componente denominado *compliance monitor*, este componente acomoda a execução automática de ferramentas de segurança e detecção automática dos potenciais problemas relatados nos artefatos produzidos por verificações executadas.

Considerar a execução do *compliance monitor* tem como pretensão sumarizar os esforços necessários para extração de evidências do estado de segurança da arquitetura. As evidências mencionadas têm benefícios potencialmente aplicáveis a fortalecimento de sistemas de *watchdog* para executar detecção de vulnerabilidades de segurança e a conversão de evidências em seletores com aplicação em procedimentos de controle de execução de softwares com uso do SPIRE.

Detectamos limitações para automação relacionadas a falta de padronização nos resultados produzidos por ferramentas de segurança. Nossa lista de ferramentas de gerenciamento de vulnerabilidades reflete de forma amostral, o atual cenário de transição entre a adoção de especificações SCAP e sua contribuição para estabelecer procedimentos automatizados. Evidenciamos a demanda por maiores esforços em definir processos automatizados quando não existe uma padronização dos resultados reportados por uma ferramenta de segurança.

7.2 Trabalhos futuros

Este trabalho envolve uma série de esforços direcionados para comprovação de conformidade com recomendações de segurança, alguns passos podem ser melhorados com procedimentos adicionais para fortalecimento de processos de verificação.

Assinatura de artefatos

Os artefatos mapeados e atribuídos às evidências são um fator determinante da conformidade obtida com uso de ferramentas de segurança. Seria interessante que os artefatos produzidos fossem assinados para comportar um processo de auditoria em casos de investigação, tomando como exemplo o que é definido na LGPD no Art. 48 e § 1° que aborda a necessidade de reportar medidas técnicas de segurança utilizadas para proteção de dados. Em outras palavras, é determinado que cabe aos responsáveis pela organização a comprovação de que existem procedimentos de segurança.

Método para definição de riscos

Obter conformidade é fundamental para cumprir e estar alinhado com as melhores práticas definidas para a segurança. No entanto, procedimentos adicionais para determinar o risco podem ser implementados contribuindo para o entendimento de possíveis lacunas na conformidade. A adoção de modelos como FAIR *Factor Analysis of Information Risk* pode auxiliar na determinação de risco da estrutura, bem como quantificar a exposição potencial a perdas em termos monetários e auxiliar os processos de gerenciamento de vulnerabilidades em termos de amplitude e profundidade de verificações.

Remediação automática de vulnerabilidades

A automação foi abordada no uso do SCAP, em específico utilizamos XCCDF e OVAL para detectar configurações incorretas e vulnerabilidades em SO usando a ferramentas OpenSCAP. No entanto, o padrão o projeto *ComplianceAsCode* que acomoda arquivos SSG, disponibiliza Ansible *Playbooks* e *Scripts* Bash para atuar com processos automatizados que executam operações para colocar máquinas em conformidade. Esta é uma importante abordagem para aumentar a conformidade com recomendações que tem a solução de vulnerabilidades e configurações incorretas como condições a serem atendidas.

Bibliografia

[AAKJ21] Asad Arfeen, Saad Ahmed, Muhammad Asim Khan, and Syed Faraz Ali Jafri. Endpoint detection & response: A malware identification solution. In *2021 International Conference on Cyber Warfare and Security (ICCWS)*, pages 1–8. IEEE, 2021.

[AAN21] Amiruddin Amiruddin, Hafizh Ghozie Afiansyah, and Hernowo Adi Nugroho. Cyber-risk management planning using nist csf v1. 1, nist sp 800-53 rev. 5, and cis controls v8. In *2021 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS*, pages 19–24. IEEE, 2021.

[ABD+22] Vikas Agarwal, Chris Butler, Lou Degenaro, Arun Kumar, Anca Sailer, and Gosia Steinder. Compliance-as-code for cybersecurity automation in hybrid cloud. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pages 427–437. IEEE, 2022.

[AIE22] Arwa Alromaih, Yasser Ismail, and Wael Elmedany. Continuous compliance to ensure strong cybersecurity posture within digital transformation in smart cities. In *6th Smart Cities Symposium (SCS 2022)*, volume 2022, pages 464–479. IET, 2022.

[AJAO20] Basheer Husham Ali, Ahmed Adeeb Jalal, and Wasseem N Ibrahem Al-Obaydy. Data loss prevention by using mrsh-v2 algorithm. *Int. J. Electr. Comput. Eng*, 10:3615–3622, 2020.

[ASKWS+21] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study

of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.

[ASMK20] Aide Alanda, Deni Satria, HA Mooduto, and Bobby Kurniawan. Mobile application security penetration testing based on owasp. In *IOP Conference Series: Materials Science and Engineering*, volume 846, page 012036. IOP Publishing, 2020.

[BASY19] Surayahani Hasnul Bhaharin, Umi Asma'Mokhtar, Rossilawati Sulaiman, and Maryati Mohd Yusof. Issues and trends in information security policy compliance. In *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)*, pages 1–6. IEEE, 2019.

[BK19] Seungjin Baek and Young-Gab Kim. Efficient vulnerability management process in the military. In *2019 International Conference on Platform Technology and Service (PlatCon)*, pages 1–5. IEEE, 2019.

[BOA+21] Sururah A Bello, Lukumon O Oyedele, Olugbenga O Akinade, Muhammad Bilal, Juan Manuel Davila Delgado, Lukman A Akanbi, Anuoluwapo O Ajayi, and Hakeem A Owolabi. Cloud computing in construction industry: Use cases, benefits and challenges. *Automation in Construction*, 122:103441, 2021.

[BS20] Blake D Bryant and Hossein Saiedian. Improving siem alert metadata aggregation with a novel kill-chain based classification model. *Computers & Security*, 94:101817, 2020.

[BS22] Ivan Bashofi and Muhammad Salman. Cybersecurity maturity assessment design using nistcsf, cis controls v8 and iso/iec 27002. In *2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, pages 58–62. IEEE, 2022.

[BSS+20] Andrey Brito, Clenimar Souza, Fábio Silva, Lucas Cavalcante, and Matteus Silva. Processamento confidencial de dados de sensores na nuvem. *Sociedade Brasileira de Computação*, 2020.

[CCCC20] Artur Potiguara Carvalho, Fernanda Potiguara Carvalho, Edna Dias Canedo, and Pedro Henrique Potiguara Carvalho. Big data, anonymisation, and governance to personal data protection. In *Proceedings of the 21st Annual International Conference on Digital Government Research*, pages 185–195, 2020.

[Cha14] K. Chandrasekaran. *Essentials of Cloud Computing*. CRC Press, Boca Raton, FL, 1st edition, 2014.

[Clo22] Cloud Native Computing Foundation. Cncf annual survey 2022. `https://www.cncf.io/reports/cncf-annual-survey-2022/`, 2022. Acessado em: 04/01/2024.

[CVCLC18] Danny C Cheng, Jod B Villamarin, Gregory Cu, and Nathalie Rose Lim-Cheng. Towards compliance management automation thru ontology mapping of requirements to activities and controls. In *2018 Cyber Resilience Conference (CRC)*, pages 1–3. IEEE, 2018.

[CVS23] Gregory Coppola, Aparna S Varde, and Jiacheng Shang. Enhancing cloud security posture for ubiquitous data access with a cybersecurity framework based management tool. In *2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0590–0594. IEEE, 2023.

[CZK22] Alladean Chidukwani, Sebastian Zander, and Polychronis Koutsakis. A survey on the cyber security of small-to-medium businesses: challenges, research focus and recommendations. *IEEE Access*, 10:85701–85719, 2022.

[DD21] Aleksandar Dimov and Vladimir Dimitrov. Classification of software security tools. *Information Systems and Grid Technologies*, 2021.

[DEM17] Kelley Dempsey, Paul Eavy, and George Moore. Automation support for security control assessments. *Vol. 1: Overview*, pages 8011–1, 2017.

[dOGLN19] Nairobi Spiecker de Oliveira, Moises Alexandre Gomes, Ronaldo Lopes, and Jéferson C Nobre. Segurança da informaçao para internet das coisas (iot):

uma abordagem sobre a lei geral de proteçao de dados (lgpd). *Revista Eletrônica de Iniciação Científica em Computação*, 17(4), 2019.

[FAH+18] Robert Filepp, Constantin Adam, Milton Hernandez, Maja Vukovic, Nikos Anerousis, and Guan Qun Zhang. Continuous compliance: Experiences, challenges, and opportunities. In *2018 IEEE World Congress on Services (SERVICES)*, pages 31–32. IEEE, 2018.

[Fer21] Jéssica Pereira Ferreira. *Hardening de Sistemas com CIS Benchmark*. PhD thesis, 2021.

[FF15] Rodrigo Fazenda and Leonardo Fagundes. Análise dos desafios para estabelecer e manter sistema de gestão de segurança da informação no cenário brasileiro. In *Anais do XI Simpósio Brasileiro de Sistemas de Informação*, pages 307–314. SBC, 2015.

[GFG17] Napoleão Verardi Galegale, Edison Luiz Gonçalves Fontes, and Bernardo Perri Galegale. Uma contribuição para a segurança da informação: um estudo de casos múltiplos com organizações brasileiras1. *Perspectivas em Ciência da Informação*, 22:75–97, 2017.

[Ham19] Ashraf Hamadi. Investigating vulnerabilities in a home network with kali linux, 2019.

[Hat] Red Hat. IaaS vs PaaS vs SaaS, acessado em maio. `https://www.redhat.com/pt-br/topics/cloud-computing/iaas-vs-paas-vs-saas`.

[HBSH18] Mouna Hedjeres, Abdellah Boukerram, Samir Sebahi, and Mohand-Said Hacid. Temporal event based compliance monitoring. In *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pages 17–22. IEEE, 2018.

[HFF+20] Muhamad Agreindra Helmiawan, Esa Firmansyah, Irfan Fadil, Yanvan Sofivan, Fathoni Mahardika, and Agun Guntara. Analysis of web security using

open web application security project 10. In *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–5. IEEE, 2020.

[HS22] Muhammad Hafiz and Benfano Soewito. Information security systems design using siem, soar and honeypot. *Jurnal Pendidikan Tambusai*, 6(2):15527–15541, 2022.

[Int16] International Telecommunication Union. Guidelines for the operational security of cloud computing. Recommendation ITU-T X.1642, International Telecommunication Union, March 2016. [Online]. Available: http://handle.itu.int/11.1002/1000/12616.

[Int20] International Telecommunication Union. Security Requirements of Public Infrastructure as a Service (IaaS) in Cloud Computing. Recommendation ITU-T X.1605, International Telecommunication Union, March 2020. [Online]. Available: https://www.itu.int/rec/T-REC-X.1605-202003-I/en.

[ITW21] Nasif Imtiaz, Seaver Thorn, and Laurie Williams. A comparative study of vulnerability reporting by software composition analysis tools. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11, 2021.

[JAH23] Abassi Haji Juma, Arry Akhmad Arman, and Fadhil Hidayat. Cybersecurity assessment framework: A systematic review. In *2023 10th International Conference on ICT for Smart Society (ICISS)*, pages 1–6. IEEE, 2023.

[KB19] Dorian Knoblauch and Christian Banse. Reducing implementation efforts in continuous auditing certification via an audit api. In *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 88–92. IEEE, 2019.

[KMM+21] Tatyana Y Khashirova, Ibragim I Mamuchiev, Madina I Mamuchieva, Marina I Ozhiganova, Alexandr D Kostyukov, and Irina Shumeiko. Assessment

of information security in integrated systems. In *2021 International Conference on Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS)*, pages 201–205. IEEE, 2021.

[KSLZ20] Marija Kovačević, Goran Sladić, Nikola Luburić, and Miroslav Zarić. Software vulnerability management system, 2020.

[KSTE20] Martin Kellogg, Martin Schäf, Serdar Tasiran, and Michael D Ernst. Continuous compliance. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 511–523, 2020.

[LAT] Antonio Lioy, Andrea Atzeni, and Massimiliano Torchio. Security assessment and threat response through scap.

[LCFX21] Zhe Li, Bo Chen, Wu-chang Feng, and Fei Xie. Concolic execution of nmap scripts for honeyfarm generation. In *Proceedings of the 8th ACM Workshop on Moving Target Defense*, pages 33–42, 2021.

[LPD+13] Wes Lloyd, Shrideep Pallickara, Olaf David, Jim Lyon, Mazdak Arabi, and Ken Rojas. Service isolation vs. consolidation: Implications for iaas cloud application deployment. In *2013 IEEE International Conference on Cloud Engineering (IC2E)*, pages 21–30. IEEE, 2013.

[LZZ+20] Si Liao, Chenming Zhou, Yonghui Zhao, Zhiyu Zhang, Chengwei Zhang, Yayu Gao, and Guohui Zhong. A comprehensive detection approach of nmap: Principles, rules and experiments. In *2020 international conference on cyber-enabled distributed computing and knowledge discovery (CyberC)*, pages 64–71. IEEE, 2020.

[MBHR20] Rizki Agung Muzaki, Obrina Candra Briliyant, Maulana Andika Hasditama, and Hamzah Ritchi. Improving security of web-based application using modsecurity and reverse proxy in web application firewall. In *2020 International Workshop on Big Data and Information Security (IWBIS)*, pages 85–90. IEEE, 2020.

[MCP17] Håvard Myrbakken and Ricardo Colomo-Palacios. Devsecops: a multivocal literature review. In *Software Process Improvement and Capability Determination: 17th International Conference, SPICE 2017, Palma de Mallorca, Spain, October 4–5, 2017, Proceedings*, pages 17–29. Springer, 2017.

[MG11] Peter Mell and Tim Grance. The NIST definition of cloud computing. 2011. Special Publication 800-145.

[MPB+13] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Avi Patel, and Muttukrishnan Rajarajan. A survey on security issues and solutions at different layers of cloud computing. *The journal of supercomputing*, 63:561–592, 2013.

[MSM+21] Mohammad Mehrtak, SeyedAhmad SeyedAlinaghi, Mehrzad Mohsseni-Pour, Tayebeh Noori, Amirali Karimi, Ahmadreza Shamsabadi, Mohammad Heydari, Alireza Barzegary, Pegah Mirzapour, Mahdi Soleymanzadeh, et al. Security challenges and solutions using healthcare cloud computing. *Journal of medicine and life*, 14(4):448, 2021.

[MZ21] Chnar Mustafa Mohammed and Subhi RM Zeebaree. Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review. *International Journal of Science and Business*, 5(2):17–30, 2021.

[Nat14] National Institute of Standards and Technology. NIST SP 800-53A Rev. 4: Assessing Security and Privacy Controls in Federal Information Systems and Organizations. Technical report, National Institute of Standards and Technology, December 2014. Available online at https://doi.org/10.6028/NIST.SP.800-53Ar4.

[Nat20] National Institute of Standards and Technology. NIST SP 800-53 Rev. 5: Security and Privacy Controls for Information Systems and Organizations. Technical report, National Institute of Standards and Technology, September 2020. Available online at https://doi.org/10.6028/NIST.SP.800-53r5.

[RDP22] Ashley Rosilier, Mevlut A Demir, and John J Prevost. Automated consul-

ting for cloud native architectures. In *2022 17th Annual System of Systems Engineering Conference (SOSE)*, pages 472–477. IEEE, 2022.

[RH22] Md Abdur Rahman and M Shamim Hossain. A deep learning assisted software defined security architecture for 6g wireless networks: Iiot perspective. *IEEE Wireless Communications*, 29(2):52–59, 2022.

[RW16] Karen Renaud and George RS Weir. Cybersecurity and the unbearability of uncertainty. In *2016 Cybersecurity and Cyberforensics Conference (CCC)*, pages 137–143. IEEE, 2016.

[San21] Amanpreet Kaur Sandhu. Big data with cloud computing: Discussions and challenges. *Big Data Mining and Analytics*, 5(1):32–40, 2021.

[SCW+23] Jeremiah Trent Stoddard, Michael Adam Cutshaw, Tyler Williams, Allan Friedman, and Justin Murphy. Software bill of materials (sbom) sharing lifecycle report. Technical report, Idaho National Lab.(INL), Idaho Falls, ID (United States), 2023.

[SIA23] Bagus Jati Santoso, Royyana Muslim Ijtihadie, and Gusti Ngurah Satria Aryawan. Vulnerability data assessment and management based on passive scanning method and cvss. In *2023 14th International Conference on Information & Communication Technology and System (ICTS)*, pages 325–330. IEEE, 2023.

[Son21] Sonatype. Introduction to the state of the software supply chain report. `https://www.sonatype.com/state-of-the-software-supply-chain/introduction`, 2021. Acessado : 01-01-2024.

[SV21] Manoj Kumar Sasubilli and R Venkateswarlu. Cloud computing security challenges, threats and vulnerabilities. In *2021 6th international conference on inventive computation technologies (ICICT)*, pages 476–480. IEEE, 2021.

[The21] The White House. Executive order on improving the nation's cybersecurity. `https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-`

order-on-improving-the-nations-cybersecurity/, May 2021. Acessado : 01-01-2024.

[THH+20] Francesc Mateo Tudela, Juan-Ramon Bermejo Higuera, Javier Bermejo Higuera, Juan-Antonio Sicilia Montalvo, and Michael I. Argyros. On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications. *Applied Sciences*, 10(24):9119, 2020.

[TKR20] Hamed Tabrizchi and Marjan Kuchaki Rafsanjani. A survey on security challenges in cloud computing: issues, threats, and solutions. *The journal of supercomputing*, 76(12):9493–9532, 2020.

[Tsi23] Nikolaos Tsilikas. Open-source security orchestration, automation, and response (SOAR) platform deployment and use. Master's thesis, Panteion University of Social and Political Sciences, 2023.

[Yao] Kota Tsuchie Taketsugu Yao. Iot real-time threat detection system at network edge.

[ZOE23] Mohamed Zelmati, Zakariae Oulqaid, and Abdelmajid Elouadi. Real-time tracking of auditing process progress with a customizable application for cybersecurity standards compliance: A case study on iso 27001 and tisax. In *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–7. IEEE, 2023.

Glossário

Glossário

CCE *Common Configuration Enumeration:* Padrão destinado a fornecer um identificador único para um determinado estado de configuração de segurança. 37

CIS *Center for Internet Security:* Organização sem fins lucrativos que trabalha para melhoria da segurança cibernética de sistemas privados e públicos. xiii, 2–4, 9, 10, 15, 22, 25–27, 29, 36, 42–44, 48, 56, 57, 70, 71, 77, 86, 87, 90, 98, 104, 107, 114, 144, 145, 148

CISA *Cybersecurity e Infrastructure Security Agency:* Uma agência federal dos Estados Unidos que fornece orientações de segurança cibernética e de infraestrutura. 42

CNCF *Cloud Native Computing Foundation:* Fundação que promove e mantém um ecossistema de projetos open source, como Kubernetes, para facilitar a computação em nuvem nativa. 32–34, 78

COBIT *Control Objectives for Information and Related Technology:* Framework de governança de TI da ISACA para alinhar objetivos de TI com estratégias de negócios, garantindo conformidade e eficiência. 42

CPE *Common Platform Enumeration:* Padrão para descrever e identificar classes de aplicações, sistemas operacionais e dispositivos de hardware conhecidos. 13, 37, 90

CSA *Cloud Security Alliance:* Organização dedicada à promoção de segurança na nuvem, envolvendo educação, pesquisa e padrões de segurança em ambientes de computação em nuvem. 22, 41, 42

CSPM *Cloud Security Posture Managemen:* Processos e tecnologia para identificar e remediar riscos associados à configuração de segurança em ambientes de nuvem. 137,

138

CVE *Common Vulnerabilities and Exposures:* Lista que identifica e cataloga vulnerabilidades conhecidas em software ou hardware. xii, xiii, 14, 30, 37, 58, 79, 81, 90, 93–95, 97, 100, 102, 104–106, 111–113, 115, 140–143

CVSS *Common Vulnerability Scoring System:* Sistema que fornece uma maneira de capturar as características principais e a severidade das vulnerabilidades de software. xii, 14, 30, 37, 58, 59, 62, 81, 93, 94, 100, 106, 107, 112, 115, 141, 142

DAST *Dynamic Application Security Testing:* Processo que testa aplicações em execução para encontrar vulnerabilidades de segurança em tempo real. 34, 35, 53, 98

DLP *Data Loss Prevention:* Estratégias e ferramentas utilizadas para prevenir a perda ou vazamento de dados sensíveis. 137, 139

EDR *Endpoint Detection and Response:* Soluções de segurança projetadas para detectar, investigar e mitigar atividades suspeitas em endpoints de uma rede. 137, 138

ENISA *European Union Agency for Cybersecurity:* A agência da União Europeia dedicada a melhorar a segurança cibernética entre os Estados membros. 22

GDPR *General Data Protection Regulation:* Regulamento da União Europeia que estabelece diretrizes para a coleta e processamento de informações pessoais de indivíduos dentro da UE. 2, 4, 22

HTTP *Hyper Text Transfer Protocol:* O protocolo usado na transferência de dados na World Wide Web. 36, 100

IaaS *Infrastructure as a Service:* Um serviço de computação em nuvem que fornece recursos de computação virtualizados pela internet. xii, 14, 18, 20, 22, 43, 50–52, 117, 118

IaC *Infrastructure as Code:* O processo de gerenciar e provisionar infraestrutura de TI através de código e ferramentas de desenvolvimento, em vez de processos manuais. 53, 83

IAST *Interactive Application Security Testing:* Ferramentas que combinam SAST (Static Application Security Testing) e DAST para fornecer uma análise de segurança durante todas as fases do desenvolvimento de software. 34, 35

ISO *International Organization for Standardization:* Organização internacional que desenvolve e publica padrões internacionais para uma ampla gama de indústrias. 2, 22, 26, 27, 41, 42, 56

ITU *International Telecommunication Union:* Agência das Nações Unidas responsável por questões que concernem às tecnologias da informação e comunicação. 2, 3, 22–25, 29, 43, 45, 50–52, 70, 71, 75, 77, 81

JSON *JavaScript Object Notation:* Formato leve de troca de dados, fácil de ler e escrever para seres humanos e fácil de analisar e gerar por máquinas. 38, 55

LGPD *Lei Geral de Proteção de Dados:* Lei brasileira que regula o tratamento de dados pessoais de indivíduos no Brasil. 2, 4, 22, 44, 56, 120

MDR *Managed Detection and Response:* Serviço de segurança que combina tecnologia e expertise humana para monitorar redes, detectar ameaças e responder a incidentes de segurança. 137

MFA *Multi-Factor Authentication:* Método de controle de acesso em que um usuário só é autorizado após apresentar duas ou mais peças de evidência à um mecanismo de autenticação. 23

NDR *Network Detection and Response:* Soluções focadas na detecção de ameaças e na resposta a incidentes dentro de redes de computadores. 137–139

NIDS *Network Intrusion Detection System:* Sistemas projetados para detectar atividades maliciosas ou violações de políticas dentro de uma rede. 137, 139

NIST *National Institute of Standards and Technology:* Agência do Departamento de Comércio dos EUA que desenvolve padrões e tecnologias para melhorar a segurança cibernética. 2–4, 9, 10, 12, 15, 17, 22, 26, 27, 29, 36, 38, 42–44, 56, 58, 70, 71, 77, 98, 107, 138

NSA *National Security Agency:* Uma agência de inteligência nacional dos Estados Unidos especializada em segurança cibernética e criptoanálise. 22, 98

NVD *National Vulnerability Database:* Base de dados governamental que lista vulnerabilidades de segurança, mantida pelo National Institute of Standards and Technology (NIST) dos EUA. 13, 106, 141

OCIL *Open Check List Interactive Language:* Linguagem que fornece uma estrutura para definição de questionários de segurança e avaliação de configurações manuais. 37

OSCAL *Open Security Controls Assessment Language:* Linguagem desenvolvida pelo NIST para padronizar a documentação de controles de segurança e avaliações. 12, 15

OVAL *Open Vulnerability and Assessment Language:* Linguagem padronizada para avaliar o estado de segurança de sistemas e comunicar detalhes sobre vulnerabilidades. xii, 37, 83, 84, 86, 90, 91, 104, 105, 113, 120

OWASP *Open Web Application Security Project:* Uma organização sem fins lucrativos que trabalha para melhorar a segurança de software através da divulgação de conhecimento e ferramentas na área de segurança de aplicações web. 32–34, 46, 53, 71, 73, 78–81, 144

PaaS *Platform as a Service:* Um serviço de computação em nuvem que fornece uma plataforma que permite aos clientes desenvolver, executar e gerenciar aplicações sem a complexidade de construir e manter a infraestrutura. 18

PII *Personally Identifiable Information:* Informações que podem ser usadas por si só ou com outras informações para identificar, contatar ou localizar uma única pessoa, ou para identificar um indivíduo no contexto. 1, 2, 51

PME Pequenas e Médias Empresas: Empresas com limites de funcionários e receita, menores que grandes corporações. 1, 2

SaaS *Software as a Service:* Distribuição de software via internet, geralmente por assinatura. 18, 19

SAST *Static Application Security Testing:* Teste de segurança que analisa código-fonte ou bytecode sem execução do programa. 34, 35, 99

SBOM *Software Bill of Materials:* Documento detalhado que lista todos os componentes em um software, geralmente usado para gerenciar e garantir a segurança do software. 24, 31, 74, 95

SCA *Software Composition Analysis:* Avaliação de riscos em software de terceiros usados em aplicações. 34, 35

SCAP *Security Content Automation Protocol:* Conjunto de padrões para automatizar a verificação, medição e correção de configurações de segurança. 15, 36, 38, 58, 80, 84–86, 103–105, 113, 119

SGSI *Sistema de Gestão de Segurança de Informação:* Sistema para proteger e gerir a segurança da informação, geralmente baseado em padrões como ISO/IEC 27001. 3

SIEM *Security Information Event Management:* Tecnologia para monitoramento em tempo real da segurança da informação, combinando gestão de eventos e informações de segurança. 137

SO *Sistema Operacional:* Software que, após ser inicialmente carregado no computador pelo um programa de inicialização, gerencia todos os outros programas em um computador. 14, 20, 21, 38, 52, 53, 61, 83, 85–88, 90, 93, 94, 106, 108–111, 120

SOAR *Security Orchestration, Automation and Response:* Tecnologia para coleta de dados de segurança e resposta automatizada a incidentes cibernéticos. 38, 137, 138

SPIFFE *Secure Production Identity Framework for Everyone:* Padrões para identificação segura de software em ambientes de nuvem. 21, 115

SPIRE *SPIFFE Runtime Environment:* Sistema que implementa o SPIFFE para autenticação de serviços de software. xiv, 21, 114, 116

SVID *SPIFFE Verifiable Identity Document:* Documento digital para autenticação e autorização de serviços em TI. 21

TI Tecnologia da Informação: Setor focado em gerenciar informações usando computadores e software. 10, 19, 26–28, 39, 42

VM *Virtual Machine:* Sistema de computador isolado operando dentro de um hardware físico. 14, 18, 19, 21, 51, 53, 86, 142

WAF *Web Application Firewalls:* Firewall para proteção de aplicações web contra ataques cibernéticos. 35, 36

XCCDF *Extensible Configuration Checklist Description Format:* Padrão usado para descrever listas de verificação de segurança, benchmarks de segurança, e configurações de segurança relacionadas. 36, 37, 83, 84, 104, 105, 113, 120

XDR *Extended Detection and Response:* Segurança cibernética que integra várias tecnologias para detecção e resposta a ameaças. 137, 138

Apêndice A

Termos de classificação de ferramentas de segurança

Assim como a classificação de ferramentas de segurança, alguns termos podem estar relacionados a ferramentas no contexto de segurança. Estes termos servem tanto para designar diretamente grupos de ferramentas de segurança, como também métodos ou abordagens que estão associadas a utilização destas em sistemas de informação. A implementação de medidas de segurança no sistema de computação envolve um processo que requer compreensão e análise de uma ampla gama de soluções disponíveis e populares na área de segurança. É essencial ter conhecimento sobre os termos, jargões e abreviações utilizados nesse contexto, bem como estar atualizado com as "*buzzwords*" do mercado. Foram listados alguns termos relacionados a segurança, os termos relevantes são o SIEM *Security Information Event Management*, SOAR *Security Orchestration, Automation and Response*, XDR *Extended Detection and Response*, CSPM *Cloud Security Posture Managemen*, EDR *Endpoint Detection and Response*, NDR *Network Detection and Response*, MDR *Managed Detection and Response*, DLP *Data Loss Prevention* e NIDS *Network Intrusion Detection System*. Os termos são detalhados a seguir:

- **SIEM:** Permite a detecção de atividades potencialmente danosas à segurança. Análises de *logs* são realizadas para descoberta de problemas nocivos a uma organização. As atividades envolvem a coleta de *logs*, centralização de *logs* e monitoramento em tempo real. Atividades dessa natureza permite descobrir e alertar atividades

anormais e potenciais ameaças provenientes de vários componentes de um sistema [HS22, BS20, Tsi23];

- **SOAR:** Seu objetivo é definir tecnologias que permitam organizações realizarem coleta de informações. Isto irá possibilitar o monitoramento por parte de equipes de segurança em suas operações cotidianas. Além do monitoramento, existe o auxílio na definição de análises de incidentes e procedimentos de resposta. Uma das principais propostas é determinar o ponto final para intervenção humana através da automação. SOAR relaciona orquestração, organização de tarefas, processos e execução de políticas como parte de sua proposta de uso [HS22] [Tsi23];

- **XDR:** Utiliza ferramentas que irão permitir a identificação de comportamentos suspeitos, isso independente do sistema que esteja sendo utilizado. Atualmente as organizações dispõem seus recursos para serem acessados de diversos ambientes, como celulares, *Personal Computer* (PCs, em tradução, Computador Pessoal), infraestrutura da organização, etc. Comportamentos suspeitos irão ser identificados e reportados. Isso provê um ambiente em que as atividades e eventos executados estão alinhados com o comportamento esperado [AAKJ21].

- **CSPM:** A proposta do CSPM é de utilizar ferramentas para atuar como auxilio no gerenciamento da postura de segurança na nuvem. Desse modo, possibilita a identificação, avaliação e mitigação de riscos de segurança nesse contexto. A utilização de CSPM pode ser baseada em documentos como o NIST *Cybersecurity Framework*, sua utilização proporciona maneiras de lidar com a complexidade dos desafios de segurança no ambiente de nuvem [CVS23];

- **EDR:** É uma solução que opera em diversas camadas, seu objetivo é detectar e responder a ameaças em dispositivos finais. As ferramentas realizam o monitoramento, coleta de dados de diferentes conjuntos de dados, geração de relatórios confiáveis permitem que o trabalho dos agentes de segurança seja suportado pelo EDR, dessa forma usuários e dados se mantêm seguros em todos os pontos do sistema [AAKJ21];

- **NDR:** Essa é uma alternativa que utiliza ferramentas dispostas para redes, em especial NDR define medidas de segurança para a borda da rede. Suas possibilidades de se-

gurança garantem que o tráfego anormal, por meio da detecção de padrões, possa ser identificado dentre as operações regulares da rede. Uma das possíveis configurações de sistema NDR envolve inserir dispositivos de captura de tráfego na borda da rede. Informações coletadas são enviadas para um servidor em nuvem para processamento e análise. Embora esta seja uma excelente implementação de segurança, o uso com grandes volumes de dados pode sobrecarregar a largura de banda. [RH22, Yao];

- **DLP:** Solução que possui ferramentas que irão auxiliar na prevenção de perda ou vazamento indevido de dados sensíveis. DLP integra o uso de ferramentas e processos que irão auxiliar na mitigação de atividades maliciosas. A utilização do DLP não pode ser considerada uma solução final para a violação de dados, mas o uso de boas ferramentas somados a técnicas como a correspondência por aproximação citado por Ali et al. [AJAO20], irá favorecer a execução de atividades que fortalecem questões de segurança envolvendo proteção de dados sensíveis das organizações [AJAO20];

- **NIDS:** Esta solução é um mecanismo que realiza detecção de ataques, além de fornecer segurança e monitoramento constante. NIDS tem como propósito analisar o tráfego da rede em busca de comportamentos maliciosos e possivelmente suspeitos. Desse modo, possíveis falhas de segurança dentro de uma rede, caso exploradas, podem gerar padrões suspeitos que são detectados indicando uma possível intrusão [ASKWS+21].

O conhecimento de termos de segurança, em especial os relacionados a ferramentas de segurança, leva a uma rápida compreensão sobre a finalidade e disposição para solução de problemas de cada ferramenta quando consideradas para compor processos de conformidade com recomendações.

Apêndice B

Estudo de vulnerabilidades CVE do OpenStack

A plataforma disponível do CVE permite o *download* da base de dados de vulnerabilidades, dessa forma é possível o processamento de informações e descoberta de vulnerabilidades relacionadas ao OpenStack. Para o processamento dos dados e a geração de gráficos, foi utilizada a linguagem de programação Python, junto com as bibliotecas pandas, numpy e matplotlib. Foram identificadas 291 vulnerabilidades para o OpenStack. A figura B.1 representa a distribuição das vulnerabilidades nos diferentes componentes da arquitetura do OpenStack.

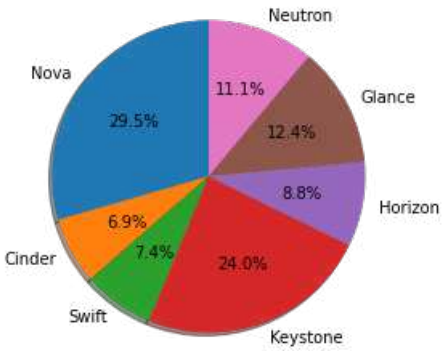


Figura B.1: Vulnerabilidades cadastradas no CVE por componente do OpenStack.
Fonte: Elaborado pelo autor.

Cada vulnerabilidade CVE é representada por um id único chamado CVE-ID, esse identificador é composto em parte pelo ano de registro da vulnerabilidade, desse modo, a Figura

B.2 ilustra o número de registros e, destes, quantos registros têm uma pontuação CVSS associado. É importante informar que os dados CVE baixados não tem informações sobre CVSS, o desenvolvimento de um *script* para buscar essas informações disponibilizadas pelo NVD[1] *National Vulnerability Database*, foi necessário. Nos anos 2013 a 2014 há uma quantidade de vulnerabilidades superior aos demais anos, porém muitas não possuíam informações sobre a severidade nem as métricas do CVSS. Nos anos anteriores também se mantém baixo o número de vulnerabilidades que possuíam as informações sobre a modelagem CVSS. A partir de 2015, apesar de terem sido normalmente registrados números menores de vulnerabilidades, o uso do padrão CVSS é mais presente. Este é um indício de um maior cuidado com a análise das vulnerabilidades, já que o registro passa a incluir métricas de risco (por exemplo, complexidade do ataque e nível de permissão exigido para o ataque) e de impacto (por exemplo, incluindo métricas que descrevem o impacto da vulnerabilidade na confidencialidade, integridade e disponibilidade do sistema).

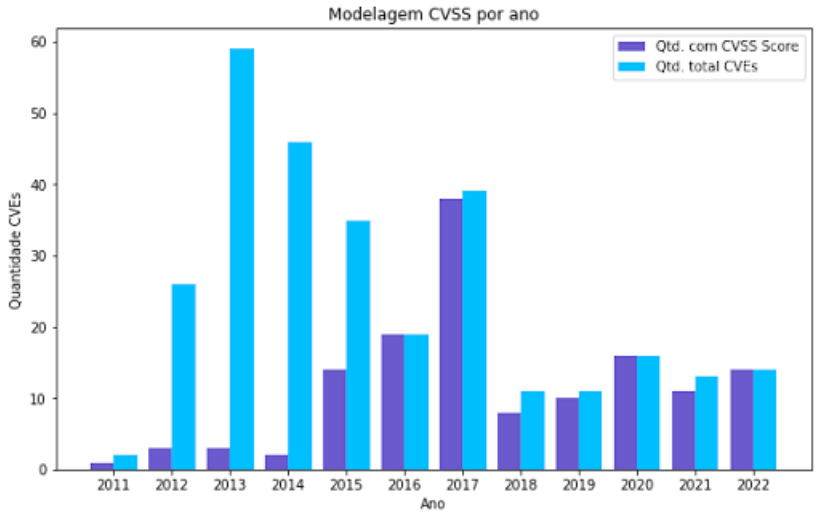


Figura B.2: Registros CVE para o OpenStack com avaliação CVSS separados por ano.

Fonte: Elaborado pelo autor.

As métricas consideradas no CVSS incluem uma série de informações adicionais relevantes em relação aos tipos de ameaça, como por exemplo: a complexidade de ataque, que possui os valores baixo e alto; o privilégio necessário com três parâmetros possíveis, nenhum, baixo ou alto; a confidencialidade, integridade e disponibilidade, que possuem os

[1] `https://nvd.nist.gov/vuln`

parâmetros nenhum, baixo ou alto. Estes fatores puderam ser analisados para os dados dos CVEs referentes aos componentes do OpenStack. A Figura B.3 ilustra a quantidade de registros CVE com alto impacto. O eixo horizontal indica a métrica considerada no CVSS, o eixo vertical indica o componente do OpenStack. O tamanho dos pontos no gráfico representa o número de registros que tiveram um alto impacto: para o caso de complexidade de ataque, o impacto é alto quando a complexidade é baixa; para o privilégio necessário para o ataque, o impacto é alto quando o privilégio necessário é baixo ou nenhum; para confidencialidade, integridade e disponibilidade, o impacto é alto quando o CVSS indica um alto impacto nos respectivos fatores. Finalmente, a cor de cada ponto é a mediana dos valores do escore CVSS dos registros que foram considerados com alto impacto. Por exemplo, o componente Keystone possui um grande número de vulnerabilidades com alto impacto: os pontos atribuídos a ele indicam 12 registros de baixa complexidade, 16 com baixo privilégio para exploração e 12 deles têm um alto impacto em confidencialidade, 10 na integridade de dados ou aplicações, e 9 em disponibilidade.

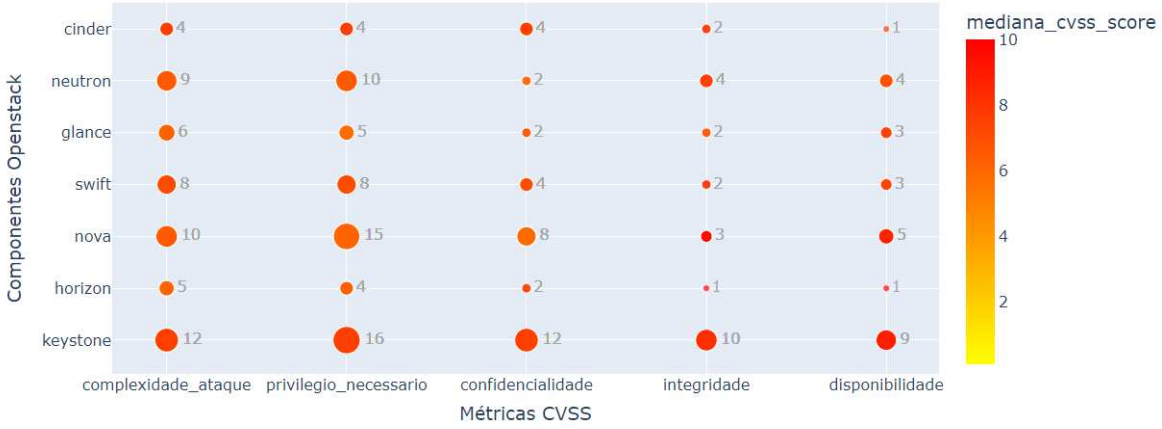


Figura B.3: Número de registros CVE classificados como alto impacto para diferentes componentes do OpenStack.

Fonte: Elaborado pelo autor.

As informações ilustradas na Figura B.4 destacam a importância da análise e monitoramento do Nova (gerência de VMs) e Keystone (autenticação). De forma semelhante à figura anterior, a Figura B.4 ilustra a quantidade de registros CVE com baixo impacto.

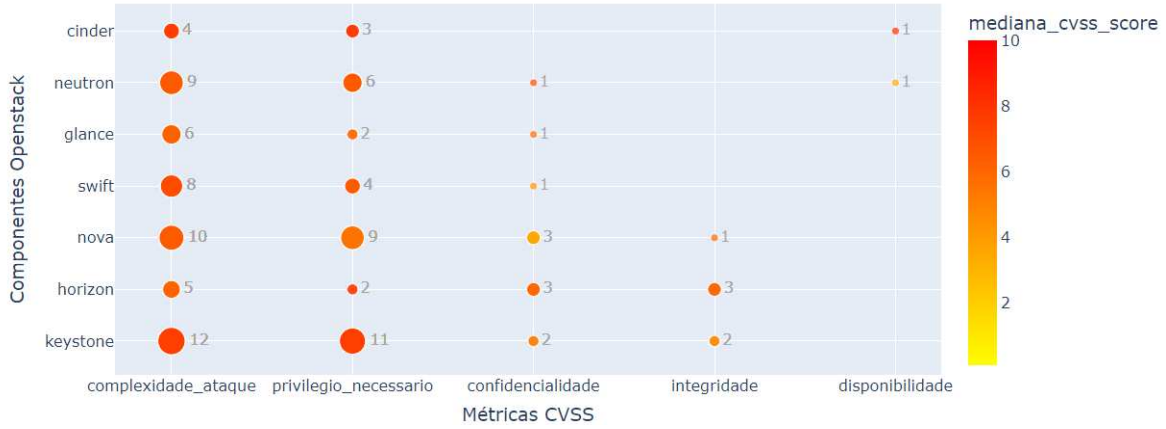


Figura B.4: Número de registros CVE classificados como baixo impacto para diferentes componentes do OpenStack.

Fonte: Elaborado pelo autor.

Apêndice C

Busca por ferramentas de segurança

Definimos uma série de parâmetros usados no processo de identificação das ferramentas. Estes parâmetros irão determinar: onde realizar a pesquisa por ferramentas; como realizar a pesquisa; levantamento de fontes reconhecidas de distribuição de ferramentas de segurança; e como utilizar recomendações de segurança para guiar o processo de identificação. Dessa forma, para definir onde será realizada a pesquisa, foi considerado o maior motor de buscas utilizado atualmente, o Google. O modo de pesquisa precisa de palavras-chave relevantes para potencializar a identificação de ferramentas de segurança nos resultados da pesquisa. A fonte reconhecida de ferramentas considerada foi o projeto *Secure Code Box* da OWASP detalhado na Seção 3.4.1. O *framework* CIS *Controls* V8 foi selecionado como fonte das recomendações para o processo de identificação, este possui 153 recomendações de segurança distribuidas entre 18 controles de segurança.

Mediante o grande número de recomendações, desenvolveu-se um processo automático para extração de informações relevantes da recomendação de segurança, este pode ser aplicado integralmente em todas as recomendações que compõe o CIS *Controls* V8. Levando em conta a estrutura do documento na versão V8, cada recomendação de segurança pertence a um controle de segurança, também dispõe do texto padrão que a descreve e uma descrição detalhando especificamente o propósito de segurança da recomendação. Dessa forma, foram identificados os campos de texto relevantes que juntos contém todas as informações necessárias sobre a recomendação de segurança. Os campos são controle, recomendação e descrição. Na Figura C.1 temos um exemplo de recomendação detalhando a estrutura considerada. No contexto do CIS *Controls* V8, controle é definido como CIS *Control* e recomendação de

segurança é definida como salvaguarda.

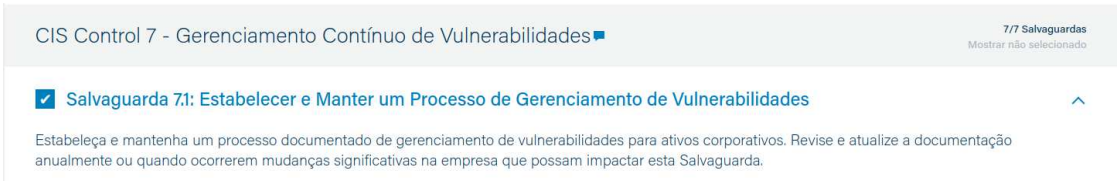


Figura C.1: Recomendação de segurança CIS *Controls* V8.

Fonte: CIS *Controls Navigator*[1] (2023).

Os campos de dados que compõem as informações foram então utilizados para criação de um *dataset*, o objetivo é utilizar todas as recomendações de segurança para compor a base de conhecimento para a busca das ferramentas. Cada um dos campos é uma coluna do *dataset*. Com todas as informações necessárias agora no *dataset*, realizou-se o processamento dos dados. O processamento de dados requer o uso de algoritmos. Dessa forma, foi utilizada a linguagem de programação Python e os algoritmos TF-IDF e Kmeans, ambos disponíveis na biblioteca de aprendizado de máquina SKLearn. O primeiro passo então é identificar palavras que são relevantes palavras-chave para compor uma string de buscas. O algoritmo TF-IDF recebeu os dados das colunas controle, recomendação e descrição e criou uma lista de palavras ordenadas por relevância. Para obter esse resultado o algoritmo utiliza dois valores, TF *Term Frequency*, conta a quantidade de vezes que a palavra aparece em um documento e IDF *Inverse Document Frequency*, onde foram atribuídos pesos para cada palavra sendo os maiores pesos atribuídos a palavras que não se repetem em vários documentos, sendo consideradas exclusivas. Esses dois valores são usados para definir a relevância da palavra, sendo o valor da relevância (TF + IDF / 2). Como resultado, a nota abaixo contém as 200 palavras mais relevantes extraídas do CIS *Controls* V8.

data, enterprise, software, network, management, assets, maintain, establish, security, inventory, process, audit, service, access, application, secure, control, incident, use, malware, log, include, based, review, annually, provider, perform, logs, accounts, ensure, configuration, vulnerability, infrastructure, penetration, devices, user, update, recovery, anti, automated, manage, safeguard, supported, training, workforce, testing, minimum, changes, example, deploy, sensitive, impact, occur, significant, frequently, collect, email, centralize, response, documentation, implementations, train, end, conduct, protection, awareness, configure, monitoring, account, providers, mfa, authorized, vulnerabilities, asset, filtering, party, authentication, intrusion, monthly, date, requirements, members, address, unauthorized, remote, defense, securely, components, policy, systems, tool, services, dns, encrypt, appropriate, solution, require, servers, web, remediation, applications, skills, reviews, protections, prevention, firewall, media, removable, browser, host, continuous, sensitivity, implement, defenses, basis, disable, frequent, operating, architecture, traffic, retention, including, test, scans, unnecessary, administrative, tools, enforce, program, event, discovery, using, resources, detection, updates, code, practices, automatic, design, includes, logging, block, patch, controls, encryption, internal, layer, classification, role, disposal, report, protocols, information, validate, utilize, storage, authorization, scope, connecting, development, best, portable, mechanisms, remediate, administrator, risk, analysis, externally, level, reporting, incidents, verify, url, version, identify, execute, active, weekly, separate, retain, exception, documented, computing, events, thresholds, currently, production, trusted, mobile, aaa, designate, default, bi, personnel, scanning, possible, file, alerting, known, flows.

Com a formação da lista de palavras relevantes em ordem decrescente, o grande número de palavras pode afetar negativamente os resultados obtidos nas buscas. De modo semelhante, uma amostra reduzida de palavras pode não representar as recomendações mediante as 153 recomendações de segurança usadas no algoritmo TF-IDF. Algumas alternativas são então elaboradas para identificar a forma ideal para modelagem das *strings* de busca. Considerando o número de 100 palavras-chave para compor as strings de busca, foram propostas duas alternativas de montagem das *strings*. Esta é uma definição arbitrária, uma vez que limitar a quantidade de palavras da lista pode ocasionar na eliminação de palavras importantes, isso tem potencial de afetar a representação de determinadas recomendações de segurança. Desse modo, as alternativas são:

- **Alternativa 1:** Considerar 100 palavras-chave em uma única *string* de buscas;
- **Alternativa 1.1:** Criar grupos de 25 palavras-chave aleatórias e compor 4 *strings* de busca.

Em análise realizada a lista de palavras relevantes, foi observado que a ordem de relevância não expressa relação entre as palavras. Tomando como exemplo, a relação entre duas ou mais palavras como, controle e acesso, recuperação e dados ou auditoria e log, são palavras que constituem informação quando relacionadas. Dessa forma, foi utilizado o algoritmo Kmeans para gerar agrupamentos de palavras próximas entre si. O algoritmo recebe como entrada os valores de TF e IDF dispostos em duas dimensões, eixos x e y. Através de análises é possível a descoberta do número ótimo de *clusters* (agrupamento das palavras), isso garante que o agrupamento seja consistente em todos os *clusters*. As palavras são agrupadas por proximidade calculando as distâncias euclidianas entre os pontos em duas dimensões. A Figura C.2 mostra o resultado do processamento do algoritmo Kmeans considerando as 25 palavras mais relevantes em cada um dos *clusters*.

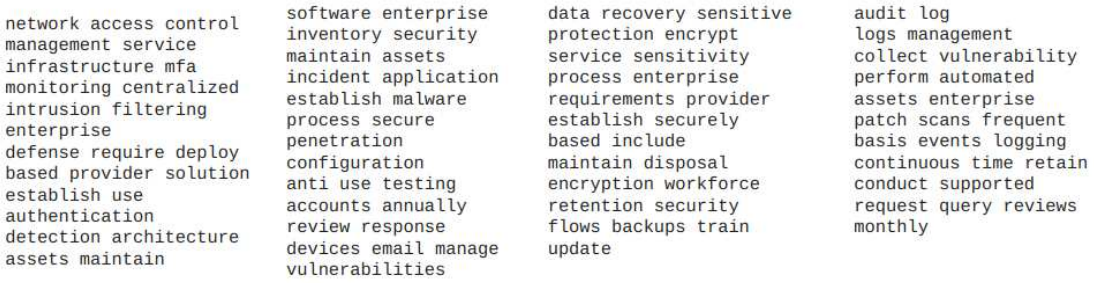


Figura C.2: *Clusters* do Kmeans.

Fonte: Elaborado pelo autor (2023).

O agrupamento resultante do algoritmo Kmeans, permite a construção de informações que não são observadas na lista de palavras relevantes resultante do algoritmo TF-IDF. Em observações realizadas nos *clusters* gerados pelo Kmeans e na lista de palavras relevantes do TF-IDF, foram extraídas duas conclusões importantes. A primeira indica que embora sejam consideradas 100 palavras da lista de Palavras relevantes do TF-IDF e 25 palavras relevantes de cada *cluster* do Kmeans, o *cluster* contém palavras que estão após a centésima posição na lista de palavras relevantes. A segunda conclusão considera que após a centésima posição na lista de palavras relevantes, existem palavras-chave que podem ser relevantes para a descoberta de ferramentas nas pesquisas. Para exemplificar, nós temos as seguintes palavras acompanhadas de sua posição na lista de palavras relevantes.

- **Grupo 1:** [*basis*:115, *frequent*:117, *retention*:121, *unnecessary*:125, *event*:130]

- **Grupo 2:** [*architecture*:119, *scans*:124, *detection*:134, *patch*:143]

Foram criados dois grupos de palavras, o grupo 1 foi classificado como grupo de palavras com baixa relevância para identificação de ferramentas de segurança. O grupo dois foi classificado como grupo de palavras com alta relevância para identificação de ferramentas de segurança. Após essas considerações, foi realizada a comparação da utilização da lista de palavras relevantes do TF-IDF com o agrupamento dos *clusters* do Kmeans para a construção de *strings* de busca. Para inserir a palavra detection do grupo 2, que está na 134ª posição, é preciso considerar 134 palavras da lista de palavras relevantes, isso inclui as palavras do grupo 1. Na abordagem utilizando o agrupamento dos *clusters* do Kmeans, as palavras do grupo 2 são consideradas sem ser necessário aumentar o número de palavras para compor a *string* de buscas. Desse modo, o agrupamento dos *clusters* do Kmeans considerando a proximidade das palavras, permite que palavras consideradas mais relevantes para a identificação de ferramentas de segurança tenham maior potencial de serem incluídas nas *strings* de busca, aumentando significativamente a representação de recomendações de segurança através de palavras-chave na *string* de busca.

Além das palavras-chave extraídas do documento CIS *Controls* V8, a *string* de buscas requer a utilização de operadores lógicos para aprimorar as buscas e a definição de palavras-chave padrão. Portanto foram utilizados os operadores lógicos *AND* e *OR* e as palavras-chave padrão irão direcionar as buscas para resultados específicos, são elas: *Security*; *Tool* e *Open Source*. A *string* de buscas considera as palavras-chave padrão obrigatoriamente presentes nos resultados, estas são separadas pelo operador lógico *AND*. As palavras-chave relevantes do documento CIS *Controls* V8 são agrupadas e separadas pelo operador lógico *OR*. Foram elaboradas *strings* de busca para todas as alternativas 1, 2 e 3. Estas podem ser verificadas no Apêndice D, cada alternativa é subdividida e atende palavras que representam parte das recomendações de segurança. De um modo geral, todas as subdivisões de uma determinada alternativa irão representar todas as recomendações de segurança do CIS *Controls* V8. A alternativa 2 e alternativa 3 foram elaboradas da seguinte forma:

- **Alternativa 2:** Considerar 25 palavras-chave de cada *cluster* para a *string* de buscas;
- **Alternativa 3:** Considerar 25 palavras-chave de cada *cluster* para a *string* de buscas e definir uma das palavras-chave como imprescindível adicionando as *strings* padrão

com o operador *AND*.

Foi elaborado um método de medição do desempenho das *strings* de busca, sendo assim foram analisados os resultados de cada *string* considerando proporcionalmente o número de palavras-chave e o número de resultados considerados na análise. Para a análise dos resultados, os anúncios retornados nas buscas foram desconsiderados. A Tabela C.2 sumariza as informações da análise.

Alternativa	***String***	**Palavras-chave**	**Resultados Considerados**	**Palavras-chave Identificadas**
TF-IDF				
Alt 1	*String* 1	100	80	20%
1.1	*String* 2	25	20	40%
1.2	*String* 3	25	20	48%
1.3	*String* 4	25	20	28%
1.4	*String* 5	25	20	36%
TF-IDF Kmeans				
Alt 2	*String* 1	25	20	48%
	String 2	25	20	36%
	String 3	25	20	28%
	String 4	25	20	36%
Alt 3	*String* 1	25	20	45%
	String 2	25	20	70%

Tabela C.2: Desempenho das *strings* de busca.

Analisando individualmente cada um dos resultados, a alternativa 1 considera a lista de palavras relevantes incluindo 100 palavras-chave na busca, como resultado, apenas 20% das palavras-chave incluídas na *string* 1 estavam presentes nos 80 primeiros resultados. As *strings* de busca *string* 2, *string* 3, *string* 4 e *string* 5, são referentes a quebra da *string* 1 de 100 palavras-chave em 4 partes iguais aleatórias. A busca dessas *strings* considerou 25 palavras-chave e 20 primeiros resultados da pesquisa. Dessa forma, os resultados variaram entre 28% e 48% das palavras-chave incluídas nas *strings* estavam presentes entre os 20 primeiros resultados. As *strings* de busca da alternativa 2 contém as 25 palavras mais relevantes de cada *cluster* do Kmeans. Os resultados variaram entre 28% e 48% das palavras-chave incluídas nas *strings* de busca estavam presentes entre os 20 primeiros resultados. Por fim,

a Alternativa 3 contém as palavras-chave de dois *clusters* do Kmeans e destacou uma das palavras-chave relevantes da lista de palavras do TF-IDF obrigatoriamente presente nos resultados. Os resultados da *string* 1 e *string* 2 ficaram 45% e 70% respectivamente.

Conclui-se que a *string* 1 da alternativa 1 teve o pior desempenho, o grande número de palavras-chave na *string* não resultou em bom desempenho se comparada com outras alternativas. A alternativa 1.1 a 1.4 e alternativa 2 possuem desempenho equivalente. Este fato demonstra que palavras-chave identificadas pelo algoritmo TF-IDF e selecionadas aleatoriamente entre as *strings*, têm o mesmo desempenho que as *strings* montadas com base nos *clusters* gerados pelo algoritmo Kmeans. Mesmo sendo agrupadas por proximidade, constituindo informação, o grupo de palavras-chave separadas pelo operador lógico *OR* tem igual desempenho nas buscas que quando feitas com palavras aleatoriamente selecionadas. A alternativa 3 teve o melhor desempenho, porém requer a utilização de palavras-chave padrão, o que obriga a palavras-chave selecionada a estar presente nos resultados.

Após as análises realizadas, a escolha da alternativa de *string* de buscas a ser utilizada foi a alternativa 2, embora não tenha apresentado nenhuma melhoria para o desempenho das *strings* de busca, identificou-se que as *strings* de busca construídas com os *clusters* do Kmeans, auxiliam no processo de atribuição das ferramentas identificadas as respectivas recomendações de segurança descritas na Seção a seguir.

Apêndice D

Strings de busca

Alternativa 1
security AND tool AND “open source” AND (data OR enterprise OR software OR network OR management OR assets OR maintain OR establish OR security OR inventory OR process OR audit OR service OR access OR application OR secure OR control OR incident OR use OR malware OR log OR include OR based OR review OR annually OR provider OR perform OR logs OR accounts OR ensure OR configuration OR vulnerability OR infrastructure OR penetration OR devices OR user OR update OR recovery OR anti OR automated OR manage OR safeguard OR supported OR training OR workforce OR testing OR minimum OR changes OR example OR deploy OR sensitive OR impact OR occur OR significant OR frequently OR collect OR email OR centralize OR response OR documentation OR implementations OR train OR end OR conduct OR protection OR awareness OR configure OR monitoring OR account OR providers OR mfa OR authorized OR vulnerabilities OR asset OR filtering OR party OR authentication OR intrusion OR monthly OR date OR requirements OR members OR address OR unauthorized OR remote OR defense OR securely OR components OR policy OR systems OR tool OR services OR dns OR encrypt OR appropriate OR solution OR require OR servers OR web OR remediation)

Alternativa 1.1
security AND tool AND “open source” AND (data OR enterprise OR software OR network OR management OR assets OR maintain OR establish OR security OR inventory OR process OR audit OR service OR access OR application OR secure OR control OR incident OR use OR malware OR log OR include OR based OR review OR annually)

Alternativa 1.2

security AND tool AND “open source” AND (provider OR perform OR logs OR accounts OR ensure OR configuration OR vulnerability OR infrastructure OR penetration OR devices OR user OR update OR recovery OR anti OR automated OR manage OR safeguard OR supported OR training OR workforce OR testing OR minimum OR changes OR example OR deploy)

Alternativa 1.3

security AND tool AND “open source” AND (sensitive OR impact OR occur OR significant OR frequently OR collect OR email OR centralize OR response OR documentation OR implementations OR train OR end OR conduct OR protection OR awareness OR configure OR monitoring OR account OR providers OR mfa OR authorized OR vulnerabilities OR asset OR filtering)

Alternativa 1.4

security AND tool AND “open source” AND (party OR authentication OR intrusion OR monthly OR date OR requirements OR members OR address OR unauthorized OR remote OR defense OR securely OR components OR policy OR systems OR tool OR services OR dns OR encrypt OR appropriate OR solution OR require OR servers OR web OR remediation)

Alternativa 2.1

security AND tool AND ’open source’ AND (network OR access OR control OR management OR service OR infrastructure OR mfa OR monitoring OR centralize OR intrusion OR filtering OR enterprise OR defense OR require OR deploy OR based OR provider OR solution OR establish OR use OR authentication OR detection OR architecture OR assets OR maintain)

Alternativa 2.2

security AND tool AND ’open source’ AND (software OR enterprise OR inventory OR security OR maintain OR assets OR incident OR application OR establish OR malware OR process OR secure OR penetration OR configuration OR anti OR use OR testing OR accounts OR annually OR review OR response OR devices OR email OR manage OR vulnerabilities)

Alternativa 2.3

security AND tool AND 'open source' AND (data OR recovery OR sensitive OR protection OR encrypt OR service OR sensitivity OR process OR enterprise OR requirements OR provider OR establish OR securely OR based OR include OR maintain OR disposal OR encryption OR workforce OR retention OR security OR flows OR backups OR train OR update)

Alternativa 2.4

security AND tool AND 'open source' AND (audit OR log OR logs OR management OR collect OR vulnerability OR perform OR automated OR assets OR enterprise OR patch OR scans OR frequent OR basis OR events OR logging OR continuous OR time OR retain OR conduct OR supported OR request OR query OR reviews OR monthly)

Alternativa 3.1

security AND tool AND 'open source' AND vulnerability AND (audit OR log OR logs OR management OR collect OR vulnerability OR perform OR automated OR assets OR enterprise OR patch OR scans OR frequent OR basis OR events OR logging OR continuous OR time OR retain OR conduct OR supported OR request OR query OR reviews OR monthly)

Alternativa 3.2

security AND tool AND 'open source' AND data AND (data OR recovery OR sensitive OR protection OR encrypt OR service OR sensitivity OR process OR enterprise OR requirements OR provider OR establish OR securely OR based OR include OR maintain OR disposal OR encryption OR workforce OR retention OR security OR flows OR backups OR train OR update)

Apêndice E

Tabelas de atribuição recomendação ferramenta

Gerência de identidade e controle de acesso		**Ferramenta**
Gestão de contas	Manter um inventário de contas (CIS Control 5.1)	Keycloak
	Usar senhas exclusivas (CIS Control 5.2)	Keycloak
	Desabilitar contas inativas (CIS Control 5.3)	Keycloak
	Gerenciar privilégios de administrador (CIS Control 5.4)	Keycloak
	Centralizar a gestão de contas (CIS Control 5.6)	Keycloak
Gestão do controle de acesso	Estabelecer processo para concessão e revogação de acesso (CIS Controls 6.1 e 6.2)	Keycloak OpenLDAP SPIRE
	Exigir MFA (CIS Control 6.3, 6.4 e 6.5)	Keycloak
	Manter um inventário de sistemas de autenticação e autorização (CIS Control 6.6)	-
	Centralizar o controle de acesso (CIS Control 6.7)	OpenLDAP
	Definir e manter um controle de acesso baseado em funções (CIS Control 6.8)	Keycloak OpenLDAP SPIRE

Continuação da Tabela E.1

Gerência de identidade e controle de acesso		Ferramenta
Proteção de dados	Manter um processo de gestão de dados (CIS Control 3.1)	Keycloak OpenLDAP
	Manter um inventário de dados (CIS Control 3.2)	-
	Configurar listas de controle de acesso de dados (CIS Control 3.3)	Keycloak OpenLDAP
	Aplicar retenção de dados (CIS Control 3.4)	-
	Descartar dados com segurança (CIS Control 3.5)	-
	Criptografar dados sensíveis (CIS Control 3.6, 3.9, 3.10 e 3.11)	VeraCrypt dm-crypt
	Manter um esquema de classificação de dados (CIS Control 3.7)	-
	Documentar fluxos de dados (CIS Control 3.8)	-
	Segmentar o tratamento de dados com base na sensibilidade (CIS Control 3.12)	-
	Implantar uma solução de prevenção contra perda de dados (CIS Control 3.13)	-
	Registrar o acesso a dados sensíveis (CIS Control 5.1)	-
Monitoramento contínuo de segurança	Monitoramento para detecção de eventos de cibersegurança (NIST DE.CM-1)	OSSEC Security Onion OpenVAS Suricata Ncrack Nmap
	Detectar código malicioso (NIST DE.CM-4)	Semgrep
	Monitoramento para detectar atividades não autorizadas (NIST DE.CM-7)	OSSEC Security Onion Suricata

Continuação da Tabela E.1

Gerência de identidade e controle de acesso		Ferramenta
	Realizar varreduras de vulnerabilidade (NIST DE.CM-8)	Kube Hunter OpenSCAP Lynis Security Onion OpenVAS Wazuh CMSeeK Ncrack Nikto SSLyze Trivy

Tabela E.1: Gerência de Identidade e Controle de Acesso.

Auditorias de segurança		Ferramenta
Gestão dos registros de auditoria	Manter um processo de gestão de logs de auditoria (CIS Control 8.1)	-
	Coletar logs de auditoria (CIS Controls 8.2, 8.5, 8.6, 8.7, 8.8 e 8.12)	Lynis OpenVAS Wazuh Kube Hunter Kubeaudit Nikto OpenSearch Security Onion
	Garantir o armazenamento adequado do registro de auditoria (CIS Control 8.3)	OpenSearch Security Onion
	Padronizar a sincronização de tempo (CIS Control 8.4)	-
	Centralizar os logs de auditoria (CIS Control 8.9)	OpenSearch Security Onion
	Reter os logs de auditoria (CIS Control 8.10)	-
	Conduzir revisões de log de auditoria (CIS Control 8.11)	OSSEC Wazuh OpenSearch Security Onion
Detecção de anomalias e eventos	Estabelecer uma referência de operações da rede (NIST DE.AE-1)	OSSEC Security Onion Snort OpenVAS Suricata Ncrack Nmap

Continuação da Tabela E.2

Auditorias de segurança		Ferramenta
	Analisar eventos detectados (NIST DE.AE-2)	OpenSCAP
		Lynis
		Security Onion
		OpenVAS
		Wazuh
		Suricata
		Kube Hunter
		Kubeaudit
		Ncrack
		Nikto
		SSLyze
		Trivy
		Nmap
		Kube-Bench
		Kube-Score

Continuação da Tabela E.2

Auditorias de segurança		Ferramenta
	Coletar dados de eventos de várias fontes (NIST DE.AE-3)	OpenSCAP Lynis OSSEC Security Onion OpenVAS Wazuh Suricata Kube Hunter Kubeaudit Ncrack Nikto SSLyze Trivy Nmap Kube-Bench Kube-Score
	Determinar impacto de eventos (NIST DE.AE-4)	OpenSCAP OpenVAS Wazuh Kube Hunter Nikto SSLyze Trivy
	Estabelecer limiares para alerta de incidente (NIST DE.AE-5)	OSSEC Suricata
Tecnologia de proteção	Proteção de mídia removível (NIST PR.PT-2)	USBGuard OpenSCAP
	Usar o princípio da menor funcionalidade (NIST PR.PT-3)	-

Continuação da Tabela E.2

Auditorias de segurança		Ferramenta
	Proteção de redes de controle e comunicações (NIST PR.PT-4)	IPTables PFSense

Tabela E.2: Auditorias de Segurança.

Gerência de vulnerabilidades		Ferramenta
Inventário e controle de ativos	Manter um inventário de ativos corporativos e de software (CIS Controls 1.1 e 2.1)	Nmap Zeek OSSEC OpenVAS Suricata Ncrack
	Endereçar ativos não autorizados (CIS Controls 1.2 e 2.3)	-
	Usar ferramentas de descoberta (CIS Controls 1.3 e 2.4)	Amass Nmap OSSEC Security Onion OpenVAS Suricata Ncrack
	Assegurar que o software autorizado seja atualmente suportado (CIS Control 2.2)	SPIRE
	Listar permissões de softwares/bibliotecas/scripts autorizados(as) (CIS Controls 2.5, 2.6 e 2.7)	Lynis

Continuação da Tabela E.3

Gerência de vulnerabilidades		Ferramenta
Gestão contínua de vulnerabilidades	Manter um processo de gestão de vulnerabilidade (CIS Control 7.1)	Kube Hunter OpenSCAP Lynis Security Onion OpenVAS Wazuh CMSeeK Ncrack Nikto SSLyze Trivy
	Manter um processo de remediação (CIS Control 7.2)	OpenSCAP OSSEC OpenVAS Wazuh Suricata Kubeaudit
	Executar a gestão automatizada de patches (CIS Control 7.3, 7.4)	-
	Realizar varreduras automatizadas de vulnerabilidade (CIS Control 7.5 e 7.6)	Zeek Lynis OSSEC OpenVAS Wazuh SSLyze
	Corrigir vulnerabilidades detectadas (CIS Control 7.7)	OpenVAS Wazuh

Continuação da Tabela E.3

Gerência de vulnerabilidades		Ferramenta
	Vulnerabilidades de ativos são identificadas e documentadas. (NIST ID.RA-1)	Kube Hunter OpenSCAP Lynis OpenVAS Nmap Wazuh CMSeeK Nikto SSLyze Trivy
	Inteligência de ameaças cibernéticas é recebida de fóruns de compartilhamento de informações e fontes. (NIST ID.RA-2)	Kube Hunter OpenSCAP Lynis OpenVAS Nmap Wazuh CMSeeK Nikto SSLyze Trivy
	Ameaças internas e externas são identificadas e documentadas. (NIST ID.RA-3)	Microsoft TMT Threat Dragon

Continuação da Tabela E.3

Gerência de vulnerabilidades		Ferramenta
	Ameaças, vulnerabilidades, probabilidades e impactos são usados para determinar o risco (NIST ID.RA-5)	Kube Hunter OpenSCAP Lynis OpenVAS Nmap Wazuh CMSeeK Nikto SSLyze Trivy
Estratégia de gerenciamento de risco	Estabelecer e gerenciar processos de gerenciamento de risco (NIST ID.RM-1)	OpenVAS
	Determinar tolerância de risco organizacional (NIST ID.RM-2)	-
Segurança de aplicações	Manter um processo seguro de desenvolvimento de aplicações (CIS Control 16.1)	CMSeeK ffuf Nikto Semgrep
Segurança de aplicações	Estabeleça e mantenha um processo para aceitar e abordar vulnerabilidades de software (CIS Control 16.2)	Kube Hunter OpenSCAP Lynis OpenVAS Nmap Wazuh CMSeeK Nikto SSLyze Trivy

Continuação da Tabela E.3

Gerência de vulnerabilidades		Ferramenta
	Executar análise de causa raiz em vulnerabilidades de segurança (CIS Control 16.3)	Kube Hunter
	Estabeleça e gerencie um inventário de componentes de software de terceiros (CIS Control 16.4)	Trivy
	Usar componentes de software de terceiros atualizados e confiáveis (CIS Control 16.5)	Nikto Trivy Sigstore
	Criar e manter sistema de classificação de vulnerabilidades de aplicações (CIS Control 16.6)	Kube Hunter OpenSCAP CMSeeK Nikto Trivy
	Usar modelos de configurações de segurança padrão (CIS Control 16.7)	Kubeaudit SSLyze Trivy
	Separar sistemas de produção e não produção (CIS Control 16.8)	Git
	Aplicar princípios de design seguro em arquiteturas de aplicações (CIS Control 16.10)	-
	Implementar verificações de segurança em nível de código (CIS Control 16.12)	Semgrep
	Realizar teste de invasão de aplicação (CIS Control 16.13)	ZAP Advanced Lynis CMSeeK ffuf Ncrack
	Conduzir modelagem de ameaças (CIS Control 16.14)	Threat Dragon Microsoft TMT

Continuação da Tabela E.3

Gerência de vulnerabilidades		Ferramenta
	Realizar processos de controle de mudança de configuração (NIST PR.IP-3)	Git

Tabela E.3: Gerência de Vulnerabilidades.

Resposta de Emergência		Ferramenta
Gestão de respostas a incidentes	Designar pessoal para gerenciar tratamento de incidentes (CIS Controls 17.1)	-
	Contato com partes interessadas para relatar incidentes de segurança (CIS Controls 17.2)	-
	Manter um processo corporativo para relatar incidentes (CIS Controls 17.3)	TheHive
	Criar e manter um processo de resposta a incidentes (CIS Controls 17.4)	TheHive Puppet Terraform
	Atribuir funções e responsabilidades chave (CIS Controls 17.5)	TheHive
	Definir mecanismos de comunicação durante a resposta a incidente (CIS Controls 17.6)	TheHive
	Mitigar os efeitos do incidente (NIST RS.MI)	-

Tabela E.4: Reposta de Emergência.

Recuperação de desastres		Ferramenta
Gestão de respostas a incidentes	Estabelecer processos de recuperação (NIST RC.RP)	Rclone Puppet Terraform
	Conduzir exercícios de resposta a incidentes rotineiros (CIS Controls 17.7)	TheHive
	Conduzir análises pós-incidente (CIS Controls 17.8)	TheHive MISP
	Estabelecer e manter limites de incidentes de segurança (CIS Controls 17.9)	-

Tabela E.5: Recuperação de Desastres.

Backup		Ferramenta
Recuperação de dados	Manter um processo de recuperação de dados (CIS Controls 11.1)	Rclone Restic
	Executar backups automatizados (CIS Controls 11.2)	Rclone Restic
	Proteger os dados de recuperação (CIS Controls 11.3)	Rclone Restic
	Manter uma instância isolada de dados de recuperação (CIS Controls 11.4)	Rclone Restic
	Testar os dados de recuperação (CIS Controls 11.5)	Rclone Restic
Processos e procedimentos de proteção de informações	Incorporar princípios de segurança na configuração de sistemas (NIST PR.IP-1)	Rclone Restic
	Conduzir, manter e testar os backups (NIST PR.IP-4)	-
	Destruir dados de acordo com a política (NIST PR.IP-6)	-

Tabela E.6: Backup.

Outros relevantes		Ferramenta
Gestão da infraestrutura de rede	Assegurar que a infraestrutura de rede esteja atualizada (CIS Control 12.1)	-
	Manter uma arquitetura de rede segura (CIS Control 12.2)	Amass ZeeK
	Estabelecer e manter diagrama(s) de arquitetura (CIS Control 12.4)	Nmap Threat Dragon Microsoft TMT
	Centralizar AAA de rede (CIS Control 12.5)	-
	Usar protocolos de comunicação e gestão de rede seguros (CIS Control 12.6)	-
	Manter recursos de computação dedicados para o trabalho administrativo (CIS Control 12.8)	-
Conscientização sobre segurança e treinamento de competências	Manter um programa de conscientização de segurança (CIS Control 14.1)	-
	Treinar membros da força de trabalho (CIS Control 14.2, 14.3, 14.4, 14.5, 14.6, 14.7 e 14.8)	-
	Conduzir treinamento de competências e conscientização de segurança para funções específicas (CIS Control 14.9)	-
Gestão de provedor de serviços	Manter um inventário de provedores de serviços (CIS Control 15.1)	-
	Manter uma política de gestão de provedores de serviços (CIS Control 15.2)	-
	Classificar provedores de serviços (CIS Control 15.3)	VSAQ SLSA
	Garantir que os contratos do provedor de serviços incluam requisitos de segurança (CIS Control 15.4)	-

Continuação da Tabela E.7

Outros relevantes		Ferramenta
	Avaliar provedores de serviços (CIS Control 15.5)	VSAQ SLSA
	Monitorar provedores de serviços (CIS Control 15.6)	VSAQ SLSA
	Descomissionar com segurança os provedores de serviços (CIS Control 15.7)	-
Gerenciamento de riscos da cadeia de suprimentos Gerenciamento de riscos da cadeia de suprimentos	Manter processos para gerenciamento de riscos (NIST ID.SC-1)	Dependency-Track VSAQ SLSA Sigstore
	Manter processos para gerenciamento de riscos da cadeia de suprimentos (NIST ID.SC-2)	Dependency-Track VSAQ SLSA Sigstore
	Estabelecer contratos com fornecedores e parceiros terceirizados (NIST ID.SC-3)	-
	Gerenciamento de fornecedores e parceiros terceirizados (NIST ID.SC-4)	-

Tabela E.7: Outros Relevantes.

Apêndice F

Recomendações do escopo de interesse

Documentação	**Palavra-chave**	**Controle**	**Recomendação**
CIS Controls V8	Vulnerability(ies)	CIS Control 7, 16	7.1, 7.5, 7.6, 7.7, 16.2, 16.3, 16.6
	Threat(s)	CIS Controls 16	16.14
	Asset(s)	CIS Controls 1, 2, 4, 7, 13, 14	1.1, 1.2, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 4.3, 4.6, 4.7, 4.8, 4.9, 7.5, 7.6, 13.5, 14.7
	Configuration	CIS Controls 1, 4, 16	1.4, 4.1, 4.2, 16.7
	Patch(es)	CIS Controls 7	7.3, 7.4
	Provider(s)	CIS Controls 8, 15	8.12, 15.1, 15.2, 15.3, 15.4, 15.5, 15.6, 15.7
	Third-Party	CIS Controls 16	16.4, 16.5
	Other related safeguards	CIS Controls 1, 7, 16	1.3, 7.2, 16.1
NIST Cybersecurity Framework	Assessment	ID.RA, ID.SC	ID.SC-2
	Supply Chain	ID.BE	ID.BE-1
		ID.SC	ID.SC-1, ID.SC-2, ID.SC-3
	Other related Recommendations	ID.RA, ID.SC	ID.RA-1, ID.RA-2, ID.RA-3, ID.RA-5, ID.SC-4, ID.SC-5

Tabela F.1: Definição de recomendações para gerenciamento de vulnerabilidades e cadeia de suprimentos.

Apêndice G

Processo de identificação das ferramentas de segurança

Ferramenta de Pentest	**Fora do contexto definido**	**Poucas informações / Obsoleto**	**Baixa popularidade**
CI Fuzz CLI	Ride (REST JSON Payload Fuzzer)	Grendel_Scan	cisco-global-expl oiter
afl++	Sec-Helpers	bed	GoLismero
yersinia	Vega	cisco-auditing-t ool	OSTE Meta Scanner
unix-privesc-check	gvm	copy-router-config	SecretScanner
cisco-torch	Sonarqube	cisco-ocs	Wapiti
ohrwurm	Prisma Cloud	rtpbreak	purpleteam
siparmyknife	kube-bench	kubescan	
spvcious	kubearmor	W3af	
t50	voiphopper	Grabber	
dhcpig			
enumiax			
iiaxflood			
inviteflood			
protos-sip			

sipp			
slowhttptest			
thc-ssl-dos			
rtpinsertsound			
rtplflood			
rtpmixsound			
sctpscan			
siege			
sipsak			
spike			

Tabela G.1: Ferramentas não consideradas.