DISSERTAÇÃO DE MESTRADO PROFISSIONAL

Proposta de Modelo de Conformidade sobre as Melhores Práticas relativas à Segurança de Cadeias de Suprimentos de Software

Carlos Eduardo Miranda Zottmann

Programa de Pós-Graduação Profissional em Engenharia Elétrica

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Proposta de Modelo de Conformidade sobre as Melhores Práticas relativas à Segurança de Cadeias de Suprimentos de Software

Carlos Eduardo Miranda Zottmann

Orientador: Rafael Rabelo Nunes
Coorientador: João José Costa Gondim

DISSERTAÇÃO DE MESTRADO PROFISSIONAL EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPEE.MP.079
BRASÍLIA/DF, 30 de Dezembro de 2024

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

Proposta de Modelo de Conformidade sobre as Melhores Práticas relativas à Segurança de Cadeias de Suprimentos de Software

Carlos Eduardo Miranda Zottmann

Dissertação de Mestrado Profissional submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Rafael Rabelo Nunes, Ph.D, FT/UnB _____
Orientador

Prof. William Ferreira Giozza, Ph.D, FT/UnB _____
Examinador Interno

Prof. Laerte Peotta, Ph.D, Banco do Brasil _____
Examinador externo

Prof. Geraldo Pereira Rocha Filho, Ph.D, FT/UnB _____
Suplente

FICHA CATALOGRÁFICA

ZOTTMANN, CARLOS EDUARDO MIRANDA
Proposta de Modelo de Conformidade sobre as Melhores Práticas relativas à Segurança de Cadeias de Suprimentos de Software [Distrito Federal] 2024.
xvi, 93 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2024).
Dissertação de Mestrado Profissional - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Cadeia de Suprimentos de Software	2. Segurança
3. Conformidade	4. Melhores Práticas
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

ZOTTMANN, C. (2024). *Proposta de Modelo de Conformidade sobre as Melhores Práticas relativas à Segurança de Cadeias de Suprimentos de Software*. Dissertação de Mestrado Profissional, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 93 p.

CESSÃO DE DIREITOS

AUTOR: Carlos Eduardo Miranda Zottmann
TÍTULO: Proposta de Modelo de Conformidade sobre as Melhores Práticas relativas à Segurança de Cadeias de Suprimentos de Software.
GRAU: Mestre em Engenharia Elétrica ANO: 2024

É concedida à Universidade de Brasília permissão para reproduzir cópias desta Dissertação de Mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Do mesmo modo, a Universidade de Brasília tem permissão para divulgar este documento em biblioteca virtual, em formato que permita o acesso via redes de comunicação e a reprodução de cópias, desde que protegida a integridade do conteúdo dessas cópias e proibido o acesso a partes isoladas desse conteúdo. O autor reserva outros direitos de publicação e nenhuma parte deste documento pode ser reproduzida sem a autorização por escrito do autor.

Carlos Eduardo Miranda Zottmann
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIA

Dedico essa dissertação a meus pais, Luiz e Neuza (ambos *in memoriam*), que valorizavam bastante a boa formação acadêmica e profissional, bem como passaram aos seus filhos sólidos princípios de ética, dedicação e profissionalismo.

Teriam ficado muito felizes com essa etapa alcançada!

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois sem ele não seríamos capazes de alcançar nenhum dos objetivos que traçamos na vida.

A minha esposa Carla, pelo constante incentivo para meu crescimento profissional e pessoal, e que teve que abrir mão de numerosos momentos de convivência ao longo dessa jornada do mestrado, especialmente na fase final de conclusão da pesquisa e preparação da dissertação.

Aos meus filhos, noras, genros e netas, por formarem a base emocional que nos incentiva no dia-a-dia.

Ao meu orientador, Rafael Rabelo Nunes, pela fantástica condução da minha jornada, sempre me incentivando e tranquilizando com relação ao trabalho, especialmente naqueles momentos em que sentimos aquela insegurança sobre a adequação da pesquisa ao escopo de um mestrado profissional, e ao conteúdo final da dissertação.

Ao meu co-orientador, João José Costa Gondim, também pelo constante incentivo com relação ao trabalho, pelo bom-humor nas nossas reuniões de discussão do assunto, e pelas propostas desafiadoras de elaboração dos artigos que terminaram por ser aprovados e publicados, mesmo em um momento em que eu ainda tinha a percepção de que não estava pronto para o desafio.

Aos professores das demais disciplinas que cursei durante o mestrado, William Ferreira Giozza, Demétrio Antônio da Silva Filho, Carlos André de Melo Alves, Robson de Oliveira Albuquerque, Clóvis Neumann e João Souza Neto, pelo rico compartilhamento de conhecimento e pelas trocas de experiências durante as aulas.

Aos amigos do grupo de orientação do professor Rafael Rabelo, com quem convivi mais de perto durante o mestrado, Marcus Aurélio Carvalho Georg, Renato Solimar Alves, Edvan Gomes da Silva, Jady Pâmella, Lucas Vinicius Andrade Ferreira e Leandro Barra Santana de Souza.

Ao amigo Thiago Stuckert, com quem divido diariamente projetos e discussões a respeito de nossos desafios profissionais, o que inclui a implantação de segurança sobre cadeia de suprimentos de software, e a quem agradeço especialmente pela parceria na publicação dos artigos sobre o tema.

E ao amigo Antonio Esio Marcondes Salgado, o Toné, que foi uma das primeiras pessoas do meu círculo profissional, se não a primeira, a me incentivar a enfrentar o desafio de um mestrado, ainda no início da minha carreira. O incentivo finalmente culminou nesse trabalho!

RESUMO

O processo atual de produção de softwares é composto pelos esforços de projeto e codificação das equipes de desenvolvimento, complementado pela reutilização intensiva de código elaborado por terceiros, sendo seu processo de implantação ou de publicação para consumo externo automatizado. Essa estrutura é conhecida como cadeia de suprimentos de softwares, que tem sido alvo de ataques cada vez mais frequentes e sofisticados. Boas práticas de segurança relativas ao tema têm sido publicadas, entretanto, sendo uma preocupação recente da indústria e da academia, ainda não há conjunto único de boas práticas reconhecido com padrão de fato. Assim, o trabalho tem por objetivo consolidar as melhores práticas acerca da segurança de cadeias de suprimentos de software e propor um modelo que reúna as recomendações dos principais *frameworks* voltados ao assunto, baseando-se no método utilizado pelo Center for Internet Security para a realização de comparações entre seus *frameworks* e outros *frameworks* semelhantes. Apresenta como principais contribuições uma comparação extensiva entre os principais *frameworks* voltados à segurança da cadeia de suprimentos de software, com detalhamento em nível de cada um dos controles de segurança recomendados, e seu próprio resultado final, um Modelo de Conformidade estruturado em torno dos pilares da cadeia de suprimentos de software, equivalentes ao código-fonte, dependências de terceiros, ambiente de compilação e ambiente de implantação ou distribuição. O modelo pode ser utilizado tanto para avaliar a conformidade atual dos controles de segurança adotados, como para subsidiar a elaboração de um plano de melhoria destinado a alcançar o nível de segurança desejado pela organização.

ABSTRACT

The current software production process is composed of the design and coding efforts of the development teams, complemented by the intensive reuse of code developed by third parties, and its deployment or publication process for external consumption is automated. This structure is known as the software supply chain, which has been the target of increasingly frequent and sophisticated attacks. Best security practices related to the topic have been published, however, being a recent concern of industry and academia, there is still no single set of best practices recognized as a de facto standard. Thus, this work aims to consolidate the best practices about the security of software supply chains and propose a model that brings together the recommendations of the most significant frameworks on the subject, based on the method used by the Center for Internet Security to make comparisons between its frameworks and other similar frameworks. Presents as main contributions an extensive comparison between the frameworks focused on the security of software supply chains, with a breakdown at the level of each of the recommended security controls, and its own final result, a Compliance Model structured around the pillars of the software supply chain, equivalent to source code, third-party dependencies, build environment, and deployment or distribution environment. The model can be used both to assess the current compliance of the security controls adopted, as well as

to subsidize the elaboration of an improvement plan aimed at achieving the level of security desired by the organization.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

ABNT	Associação Brasileira de Normas Técnicas
ACSC	*Australian Cyber Security Centre*
AHP	*Analytic Hierarchy Process*
AML	Alhazmi-Malaia Logistic
ANP	*Analytical Network Process*
ASC	*Application Security Controls*
ASMP	*Application Security Management Process*
BBN	*Bayesian Belief Network*
BSI	*Germany's Federal Office for Information Security*
BSIMM	*Building Security In Maturity Model*
CCCS	*Canadian Centre for Cyber Security*
CERT NZ	*Computer Emergency Response Team New Zealand*
CISA	*Cybersecurity & Infrastructure Security Agency*
CI-CD	*Continuous Integration and Continuous Delivery*
CMMI	*Capability Maturity Model Integration*
CNCF	*Cloud Native Computing Foundation*
CSA	*Cyber Security Agency of Singapore*
C-SCRM	*Cyber Supply Chain Risk Management*
CSF	*Cybersecurity Framework*
CSIRT	*Cyber Incident Response Team*
CSIRTAMERICAS	*Network of Government Cyber Incident Response Teams (CSIRT) of the Member States of the Organization of American States*
CVSS	*Common Vulnerability Scoring System*
DAST	*Dynamic Application Security Testing*
E-ciber	Estratégia Nacional de Segurança Cibernética
ELECTRE	*Elimination et Choix Traduisant la Réalité*
ENISA	*European Union Agency for Cibersecurity*
ENSEC-PJ	Estratégia Nacional de Cibersegurança do Poder Judiciário
ESF	*Enduring Security Framework*
FBI	*Federal Bureau of Investigation*
FMEA	*Failure Mode and Effect Analysis*
IDS	*Intrusion Detection System*
IPS	*Intrusion Prevention System*
ISO	*International Organization for Standardization*
JPCERT-CC	*Japan Computer Emergency Response Team Coordination Center*
KISA	*Korea Internet & Security Agency*

NCSC	National Counterintelligence and Security Center - US
NCSC-NL	*Nationaal Cyber Security Centrum - Netherland*
NCSC-NO	*Norwegian National Cyber Security Center*
NCSC-NZ	*National Cyber Security Centre - New Zealand*
NCSC-UK	*National Cyber Security Centre - United Kingdom*
NISC-JP	*National Center of Incident Readiness and Strategy for Cybersecurity - Japan*
NIST	*National Institute of Standards and Technology*
NPM	*Node Package Manager*
NSA	*National Security Agency*
NVD	*National Vulnerability Database*
NUKIB	*National Cyber and Information Security Agency - Czech Republic*
MTTR	*Mean Time To Remediate*
ONF	*Organization Normative Framework*
OWASP	*Open Worldwide Application Security Project*
PBOM	*Pipeline Bill of Materials*
Pipy	*Python Package Index*
S2C2F	*Secure Supply Chain Consumption Framework*
SAMM	*Software Assurance Maturity Model*
SAST	*Static Application Security Testing*
SCM	*Supply Chain Management*
SCRM	*Supply Chain Risk Management*
SCVS	*Software Component Verification Standard*
SDLC	*Software Development Life Cycle*
SEI	*Software Engineering Institute*
SLSA	*Supply-chain Levels for Software Artifacts*
SSDF	*Secure Software Development Framework*
TEMAC	Teoria do Enfoque Metaanalítico Consolidado
TI	Tecnologia da Informação
TTP	Técnicas, Táticas e Procedimentos

1 INTRODUÇÃO

O mundo atual conta cada vez mais com soluções de software para a realização de nossas tarefas do dia a dia, sejam elas simples como nos manter informados sobre as notícias mais recentes, consultarmos a programação cultural e esportiva da cidade ou dos canais de televisão e de *streaming*, gerenciarmos nossas informações bancárias, agendarmos consultas médicas, ou mesmo interagirmos com as diversas agências governamentais para o exercício de nossa cidadania [1].

A demanda pela disponibilização de novas soluções de software para atender a essas necessidades, bem como pela contínua atualização dessas soluções de forma a mantê-las competitivas em comparação com soluções semelhantes, fez com que os ambientes de desenvolvimento de softwares adotassem soluções de Integração Contínua e Disponibilização Contínua (conhecidas como ambientes de CI/CD), com o objetivo de agilizar a disponibilização de novos produtos e atualizações. Com efeito, levantamento recente realizado pelo SANS Institute revelou que cerca de 45% das organizações pesquisadas realizam a disponibilização de modificações em suas soluções de forma diária ou semanal, e, para isso, utilizam-se de tais soluções [2].

Além dessa característica, a necessidade de entregas cada vez mais rápidas também guiou a disciplina de desenvolvimento de software na direção da utilização cada vez mais intensa de componentes desenvolvidos por terceiros, muitas das vezes disponibilizadas como código livre ou de uso gratuito.

Levantamento realizado pela empresa Sonatype [3] traz um panorama do incremento da utilização de componentes de terceiros, considerando os principais repositórios de componentes, a saber, Maven [4], repositório de componentes escritos em Java, NPM [5], que congrega componentes escritos em Javascript, Pipy [6], que armazena componentes desenvolvidos na linguagem Python, e Nugget Gallery, referente a pacotes .Net [7].

Os dados do levantamento, referentes ao ano de 2023, indicam que, ao todo, esses repositórios reúnem aproximadamente 3,9 milhões de projetos de software, cada um apresentando, em média, 15 versões distintas (o que equivale a um total de 60 milhões de versões), representando um crescimento de 29% sobre o ano anterior. Em relação ao uso efetivo desses projetos, registrou-se um total de 4 terabytes de dados baixados desses repositórios pela comunidade de desenvolvedores, com uma estimativa de crescimento anual de downloads em torno de 33%.

Uma das plataformas de colaboração para o desenvolvimento de projetos de código livre, Github, publica anualmente seu relatório sobre o cenário de projetos de código livre. Sua última edição, de 2024, informa que a plataforma recebeu 1,4 milhões de novos desenvolvedores, sobre um total que já soma mais de 100 milhões. que contribuem sobre mais de 518 milhões de projetos públicos e privados [8].

Consequentemente, quase todas as soluções de software atuais dependem de códigos desenvolvidos por terceiros para implementar suas funcionalidades. Relatório elaborado pela empresa Synopsys também em 2024, a respeito do cenário de riscos relacionados a componentes de código livre, informa que, dentre as bases de código que avaliou, 96% continham código aberto [9].

A cadeia de suprimentos de software, que, como será detalhado ao longo desta dissertação, é composta

pelos pilares código fonte, dependências de terceiros, ambiente de compilação, e ambientes e processos de implantação (*deploy*) ou distribuição, é cada vez mais explorada como vetor de ataques cibernéticos [10], sendo um dos grandes desafios para a comunidade internacional, especialmente em combinação com a utilização de Inteligência Artifical para a geração de código, conforme destacado pelo relatório "Global Cybersecurity Outlook 2024"do Fórum Econômico Mundial [11].

Inicialmente, esses ataques se concentravam na exploração de vulnerabilidades existentes em produtos já disponíveis. Entretanto, a tendência que atualmente se observa é a de ataques por meio da inserção de código malicioso em um determinado produto de software ainda durante as etapas de sua construção, de forma a intencionalmente inserir vulnerabilidades no produto final, que será utilizado por uma grande diversidade de usuários, seja como componente inserido em um terceiro produto em desenvolvimento, seja como efetivo produto acabado, adquirido gratuitamente ou mesmo comercialmente [12].

A possibilidade de ataques deliberados à cadeia de suprimentos traz alguns benefícios ao atacante frente às práticas de cibersegurança estabelecidas.

O primeiro dos benefícios é relacionado ao fato de que historicamente as organizações estabeleceram suas defesas cibernéticas focadas em soluções de perímetro, compostas por ferramentas como *firewalls*, *web application firewalls*, sistemas de detecção ou prevenção de intrusão (IDS/IPS), antispam, e antivírus, voltadas à proteção de ataques direcionados de fora para dentro do ambiente de TI.

Nesse novo cenário, no, entanto, a própria organização acaba introduzindo a vulnerabilidade em seu ambiente, instalando produtos intencionalmente modificados pelos atacantes, sob a premissa de tratar-se de um produto disponibilizado por um fabricante de sua confiança, facilitando assim a ocorrência de ataques.

Um segundo benefício para os atacantes é o ganho de escala, uma vez que um ataque bem sucedido a um projeto de software comumente utilizado por outros projetos terá seu alcance amplificado, uma vez que a vulnerabilidade será inserida em todos os produtos finais que o utilizem [13].

A literatura especializada já documenta uma série ocorrências de ataques distintos à cadeia de suprimento de softwares. Destacam-se como principais casos a vulnerabilidade da biblioteca de código aberto Log4J, exemplo de exploração de vulnerabilidade presente no produto [14], e o ataque ao produto Orion, da empresa norte-americana Solar Winds [15], objeto de ataque deliberado, por meio de inserção de código malicioso em produto legítimo. Exemplo ainda mais recente é o que atingiu a biblioteca liblzma, conhecida como "vulnerabilidade XZ", que tinha como objetivo inserir um *backdoor* em produtos que a utilizassem, sendo o produto mais relevante o servidor sshd, utilizado para oferecer conexão remota ao ambiente de linha de comando em servidores Linux[16].

Considerando essa tendência, diversas organizações públicas e privadas publicaram *frameworks*, guias, modelos de processos e ferramentas com medidas para proteção contra esse tipo de ameaça. Esses recursos, que serão melhor detalhados no tópico sobre referencial teórico nessa dissertação, devem ser incorporados em diferentes pontos do ciclo de desenvolvimento de software por fornecedores e consumidores de software.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

Ataques à cadeia de suprimentos de software já vêm sendo reconhecidos pelas organizações especializadas como um dos principais riscos de cibersegurança atuais. Relatório publicado pela European Union Agency for Cibersecurity (ENISA) a respeito de ameaças emergentes no cenário de cibersegurança apontam o comprometimento da cadeia de suprimentos em relação a dependências de software como a principal ameaça em cibersegurança em um período que se estende até o ano de 2030 [17]. Quanto à percepção dos profissionais que atuam no ramo de cibersegurança, o assunto já surge também como uma das principais preocupações, conforme aponta o relatório 2024 Cyberthreat Defense Report [18]

Com efeito, a preocupação quanto ao tema já tem chegado a governos de diversos países, conforme demonstram normativos ou recomendações referentes à segurança da cadeia de suprimento de software publicados por agências oficiais dos Estados Unidos, Europa, Reino Unido, Austrália, Canadá, Japão e Nova Zelândia [3].

O caso norte-americano é o mais significativo, uma vez que já conta com iniciativas importantes tais como a Executive Order 14028 - Improving the Nation's Cybersecurity, que define diretivas referentes a cibersegurança, incluindo sua Seção 4, totalmente dedicada à segurança da cadeia de suprimento de software [19], como a National Cybersecurity Strategy [20], publicada em 2023, que também já aborda o tema, bem como com um projeto de lei específico sobre o assunto em tramitação, o Securing Open Source Software Act of 2023, que tem como objetivo definir as atribuições da Cybersecurity and Infrastructure Security Agency (CISA) com relação à segurança de softwares de código aberto [21].

O Brasil ainda carece de recomendações e normativos específicos sobre o tema, sendo que as únicas referências encontradas residem na E-ciber do Poder Executivo, que reconhece dentre as preocupações de segurança cibernética do governo os ataques sofisticados e direcionados às cadeias de suprimentos, e recomenda que as políticas de segurança cibernética incluam requisitos relativos à gestão da cadeia de suprimentos e seja realizado um monitoramento da implementação dos requisitos mínimos de segurança cibernética pelos fornecedores que integram a cadeia de suprimentos, sem, no entanto, relacionar que requisitos seriam aplicáveis [22]. Outros normativos, como a Estratégia Nacional de Cibersegurança do Poder Judiciário - ENSEC-PJ [23] ou mesmo a publicação do Tribunal de Contas da União, "Cinco Controles de Segurança Cibernética para Ontem"[24] ainda não abordam a questão.

O presente trabalho, portanto, tem por motivação oferecer uma contribuição às diversas organizações públicas e privadas quanto à estruturação de seus esforços relativos ao incremento da segurança de suas cadeias de suprimentos de software.

1.2 O PROBLEMA DE PESQUISA

Os principais *frameworks* e artigos acadêmicos sobre o tema são bastante recentes, com as primeiras versões surgindo a partir de 2020, sendo que um deles, o SLSA [25], inclusive, ainda se encontra em construção, não cobrindo todos os pilares da cadeia que se propõe a tratar.

As organizações interessadas em fortalecer suas próprias cadeias de suprimentos de software, por sua

vez, necessitam de fontes confiáveis e completas sobre o tema, que possam ser utilizadas como modelos para a estruturação de seus esforços voltados à questão.

O problema de pesquisa que se coloca, portanto, é o seguinte: dada a multiplicidade de fontes de informações sobre o assunto, que conjunto de recomendações deve ser adotado para o alcance do nível de segurança da cadeia de suprimentos de software apropriado para cada organização?

Dado o problema, a seguinte hipótese foi formulada: É possível estabelecer um conjunto de recomendações específicas, baseado na multiplicidade de fontes que propõem melhores práticas sobre o tema, que permita às organizações conhecerem o nível de segurança de sua cadeia de suprimento de softwares, e alcançarem níveis adequados de segurança na cadeia de suprimentos de software, ajustados às suas particularidades e necessidades?

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Consolidar as melhores práticas de segurança em cadeias de suprimentos de software e propor um modelo que permita que organizações públicas e privadas avaliem seu estado atual e adotem as recomendações necessárias para alcançarem o nível de segurança desejado.

1.3.2 Objetivos Específicos

- Organizar um conjunto de boas práticas de segurança relativa a cadeias de suprimentos de software a partir de seus principais *frameworks*;
- Analisar e avaliar as semelhanças e diferenças entre os diversos *frameworks* e melhores práticas de segurança relativas a cadeias de suprimentos de software;
- Estruturar um Modelo de Conformidade de Segurança para Cadeias de Suprimentos de Software baseado nas recomendações de segurança citadas no item anterior.

1.4 CONTRIBUIÇÕES

O presente trabalho apresenta como principais contribuições uma comparação extensiva entre os principais frameworks voltados à segurança da cadeia de suprimentos de software, com detalhamento em nível de cada um dos controles de segurança recomendados, e seu próprio resultado final, um Modelo de Conformidade estruturado em torno dos pilares da cadeia de suprimentos de software, equivalentes ao código-fonte, dependências de terceiros, ambiente de compilação e ambiente de implantação ou distribuição.

O Modelo de Conformidade pode ser utilizado tanto para avaliar a conformidade dos controles de segurança adotados por uma organização em um determinado momento, como para subsidiar a elaboração

de um plano de melhoria destinado a alcançar o nível de segurança desejado por essa organização.

1.5 PUBLICAÇÕES RELACIONADAS À PESQUISA

- Zottmann, Carlos Eduardo Miranda; Amaral, Thiago Melo Stuckert; Nunes, Rafael Rabelo; Gondim, João José Costa. **Comparing Software Supply Chain Protection Approaches**. Workshop on Communication Networks and Power Systems (WCNPS), Brasilia, Brazil, dezembro de 2023.
- Zottmann, Carlos Eduardo Miranda; Amaral, Thiago Melo Stuckert; Nunes, Rafael Rabelo; Gondim, João José Costa. **Comparação de abordagens de proteção à cadeia de suprimento de software**. Revista Ibérica de Sistemas e Tecnologias de Informação - RISTI, nº E70.
- Zottmann, Carlos Eduardo Miranda; Nunes, Rafael Rabelo; Georg, Marcus Aurélio Carvalho; Alves, Renato Solimar; Silva, Marcelo Antônio da. **Proposta de Metodologia para Avaliação de Riscos de Privacidade para Órgãos do Poder Judiciário no Brasil**. Encontro Nacional de Administração da Justiça - ENAJUS, outubro de 2023.
- Alves, Renato Solimar; Zottmann, Carlos Eduardo Miranda; Georg, Marcus Aurélio Carvalho; Nunes, Rafael Rabelo; Arruda, Luiz Guilherme Schiefler de. **Enfrentando os Ataques Hackers: Controles de Segurança da Informação Prioritários para o Tratamento dos Riscos de Negócio do Poder Judiciário**. Encontro Nacional de Administração da Justiça - ENAJUS, outubro de 2023.
- Godinho, Gabriel Marinho; Fernandez, Beatriz Teles; Alves, Renato Solimar; Zottmann, Carlos Eduardo Miranda; Nunes, Rafael Rabelo; **Não é só trocar a senha: Segurança Cibernética no Tribunal Superior**. Casoteca ADM, janeiro de 2024.

1.6 ESTRUTURAÇÃO DO TRABALHO

A presente dissertação está organizada desta maneira: além deste capítulo de introdução, há um referencial teórico onde encontram-se conceitos sobre cadeias de suprimentos, *frameworks* de desenvolvimento seguro, principais normativos publicados sobre segurança da cadeia de suprimento de software, identificação dos *frameworks*, guias, processos e ferramentas voltados à proteção da cadeia de suprimento de software e conceitos sobre gestão de riscos (Capítulo 2), seguida da metodologia utilizada no trabalho (Capítulo 3), resultados da pesquisa (Capítulo 4) e conclusão (Capítulo 5).

2 REFERENCIAL TEÓRICO

Com o objetivo de consolidar as melhores práticas sobre a segurança de cadeias de suprimentos de software, este referencial teórico abordará os principais conceitos sobre cadeias de suprimentos em geral, cadeias de suprimentos de software, práticas de desenvolvimento seguro, gestão de riscos de cadeias de suprimentos de software e os normativos publicados por organizações internacionais relacionadas à segurança desse contexto.

2.1 CONCEITOS SOBRE CADEIAS DE SUPRIMENTOS

Cadeia de suprimentos, de forma geral, pode ser definida como um processo integrado onde diferentes entidades, tais como fornecedores, fabricantes e distribuidores, trabalham em conjunto com o objetivo de transformar insumos primários em produtos acabados ou serviços, entregá-los aos pontos de venda, e em última instância aos clientes finais, conforme ilustra a Figura 2.1 [26].

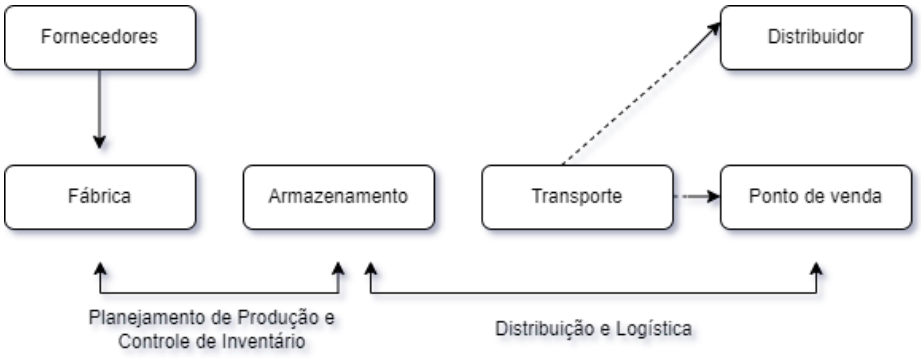


Figura 2.1: Modelo conceitual de cadeia de suprimentos genérica

Fonte: Adaptada de [26]

Outra definição, que destaca o caráter cíclico e recursivo da cadeia de suprimentos, é apresentada pela ISO em seus normativos sobre o tema, que a descreve como sendo um conjunto de organizações com recursos e processos interligados, onde cada uma atua como adquirente, fornecedor, ou ambos, estabelecendo relacionamentos sucessivos por meio da formalização de uma ordem de compra ou outro modelo de acordo [27].

Trata-se de um conceito originalmente ligado à logística empresarial, que, no mundo globalizado em que vivemos hoje, é responsável pela integração de processos e atividades empresariais que muitas vezes são realizados em países diferentes, e compõem o processo de produção de bens e serviços desde a obtenção da matéria prima até sua entrega ao cliente final. Essa integração de bens e serviços produzidos em diferentes localidades busca aproveitar as eficiências de produção de cada tipo de item de acordo com as características de cada local [28].

Em cadeias de suprimentos tradicionais, as funções de armazenamento e transporte são as que representam os maiores custos de logística, sendo responsáveis, respectivamente, por agregar valor com relação ao tempo de disponibilidade dos produtos e aos locais de disponibilização desses mesmos produtos aos clientes finais [28].

Embora usualmente as empresas que realizem a integração da cadeia de suprimentos limitem sua gestão aos elos imediatamente anteriores e posteriores às atividades da própria organização, garantindo a obtenção de insumos a partir de seus fornecedores imediatos e a distribuição de seus produtos aos seus clientes e consumidores, deve-se observar a existência da cadeia de suprimentos estendida, composta pelos fornecedores dos fornecedores imediatos, e pelos clientes dos clientes imediatos da organização [28].

No contexto do desenvolvimento de software, a cadeia de suprimentos refere-se à sequência de etapas realizadas para a criação de soluções de software, representadas por ferramentas de infraestrutura, sistemas de informação, aplicativos móveis, etc., sendo composta pelos pilares relativos a códigos fonte, compilações, dependências e implantação (*deploy*) ou distribuição de pacotes. Assim, a cadeia de suprimentos de um produto é a combinação das cadeias de suprimentos de suas dependências associadas ao seu próprio código fonte, que formam o pacote final compilado, conforme ilustra a Figura 2.2 [25].

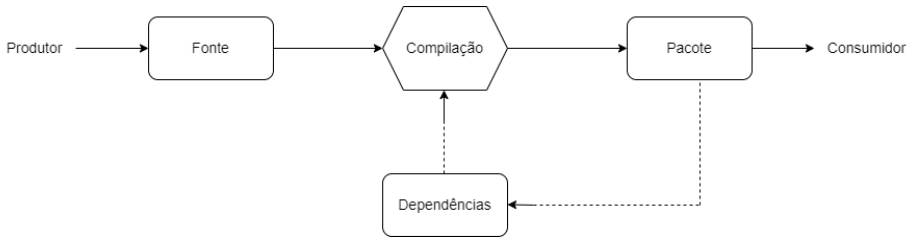


Figura 2.2: Cadeia de Suprimentos de Software

Fonte: Adaptada de [25]

O pilar referente às dependências é onde se observa a recursividade que pode ser depreendida da definição da ISO, equivalente à cadeia de suprimentos estendida. Um determinado desenvolvedor, ao mesmo tempo em que pode ser fornecedor de um componente para certos projetos de software, pode também ser um consumidor de componentes desenvolvidos por outros fornecedores.

Do ponto de vista de um produto final, os componentes explicitamente consumidos por um desenvolvedor em seu projeto são conhecidos como dependências diretas, enquanto as demais dependências utilizadas por esses componentes são conhecidos como dependências transitivas.

Conforme apontado pelo *framework* publicado pela Cloud Native Computing Foundation, a cadeia de suprimentos de software se assemelha à cadeia de suprimentos tradicional, associada a atividades logísticas, conforme ilustra a Figura 2.3 [29].

O *framework* cita como diferenças importantes entre ambos os tipos de cadeias de suprimento o fato de que as cadeias de suprimentos de software são intangíveis (uma vez que são compostas por componentes virtuais), sujeitas a modificações frequentes (pois projetos de software facilmente podem modificar as dependências que utilizam), e envolvem componentes que podem ser iterativamente reutilizados (dado que um mesmo componente pode ser utilizado em diversos projetos de software, sem que seja necessária a produção de novos exemplares desse componente).

Figura 2.3: Cadeia de Suprimentos Tradicional e Cadeia de Suprimentos de Software

Fonte: Adaptada de [29]

Um caso específico da cadeia de suprimentos de software é a dos softwares de código aberto, em virtude do uso amplo e predominante de componentes de software aberto no cenário atual. Entre suas características mais relevantes e distintivas, destacam-se as seguintes:

- Não há mecanismo formal (contrato, acordo, ou documento semelhante) que regule as relações entre adquirente e fornecedor;
- O fornecedor não tem nenhuma obrigação legal junto ao adquirente de atender aos requisitos de segurança definidos por este;
- A verificação por parte do adquirente de que o fornecedor atende aos seus requisitos de segurança deve ser feita por meio da publicação de informações por parte do fornecedor a respeito de seus componentes (tipicamente por meio de metadados associados aos componentes disponibilizados), ou por parte de avaliação de terceiros (atualmente já disponíveis a partir de serviços ou ferramentas como TACOS e OpenSSF Scorecard);
- O fornecedor não tem acesso a informações do adquirente referentes aos projetos ou à forma como o componente será utilizado, exceto às informações públicas coletadas sobre quem faz download de seus componentes tais como endereço IP e outras informações eventualmente exigidas para cadastro no repositório de pacotes onde o componente é disponibilizado.

As características específicas de cada um dos pilares da cadeia de suprimentos de software possibilitam a realização de ataques de naturezas também específicas, não observados nos casos das cadeias de suprimentos tomadas como conceito geral.

2.2 CASOS REAIS DE ATAQUES À CADEIA DE SUPRIMENTOS DE SOFTWARE

Estudo realizado por Geer et al. no ano de 2020 compilou um conjunto de dados relativos a ataques conhecidos publicamente, e produziu estatísticas a respeito de relatos, ataques e incidentes [30].

A Figura 2.4 ilustra os quantitativos de relatos e ataques, onde relatos se referem à divulgação pública de um ou mais ataques, e ataques se referem a uma ação deliberada de comprometimento de cadeias de

suprimentos de software. O estudo ainda diferencia ataques de incidentes, de forma que estes seriam uma única instância de ataque sobre um determinado alvo, e a intenção dos autores, ainda não materializada no estudo, seria a de utilizar a quantidade de incidentes como mensuração do impacto de um ataque.

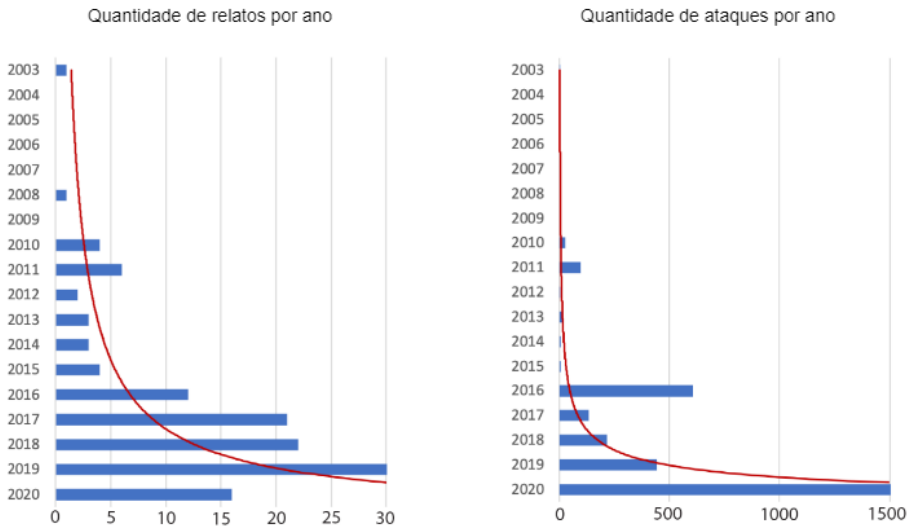


Figura 2.4: Quantidades de Relatos e Ataques à Cadeia de Suprimentos de Software

Fonte: Adaptada de [30]

É possível observar que a tanto a quantidade de relatos quanto a quantidade de ataques vêm em uma tendência crescente (considerando-se que o estudo foi publicado em 2020, e os dados daquele ano ainda eram parciais, bem como exceções relacionadas a ocorrências especiais em 2016, documentadas na publicação).

Uma conclusão interessante alcançada pelo estudo é a de que a frequência de ataques é inversamente proporcional à complexidade de execução desses ataques (i.e., quanto mais simples o ataque, maior é sua frequência), mas que a quantidade efetiva de incidentes segue padrão inverso, ou seja quanto mais complexo o tipo de ataque, maior é a frequência de sucesso com relação ao alcance real de vítimas desses ataques.

Em outro estudo, Ohm et al. desenvolveram um conjunto de dados sobre pacotes maliciosos empregados em ataques a cadeias de suprimentos de software, analisando como esses pacotes são utilizados na execução dos ataques [31].

Resultados importantes do estudo foram a elaboração de uma árvore de ataques, construída a partir da observação das técnicas efetivamente utilizadas para a construção e disponibilização dos pacotes maliciosos, e a identificação dos objetivos reais dos pacotes analisados, representados respectivamente pelas Figuras 2.5 e 2.6.

A Figura 2.5 representa uma árvore de ações que podem ser executadas para o alcançe do objetivo principal do ataque, equivalente à "Injeção de código malicioso (na árvore de dependências)"em um projeto de software. O segundo nível da árvore indica as ações principais que podem ser realizadas, "Criar Novo Pacote"e "Infectar Pacote Existente", e os níveis subsequentes indicam as tarefas que devem ser realizadas, ou os meios que podem ser empregados, para a execução de cada ação. Ilustrando uma das ações de segundo

nível, a ação "Inserir no código"pode ser executada por meio de *Pull Request* por parte do contribuidor, ou de *Commit* por parte do mantenedor.

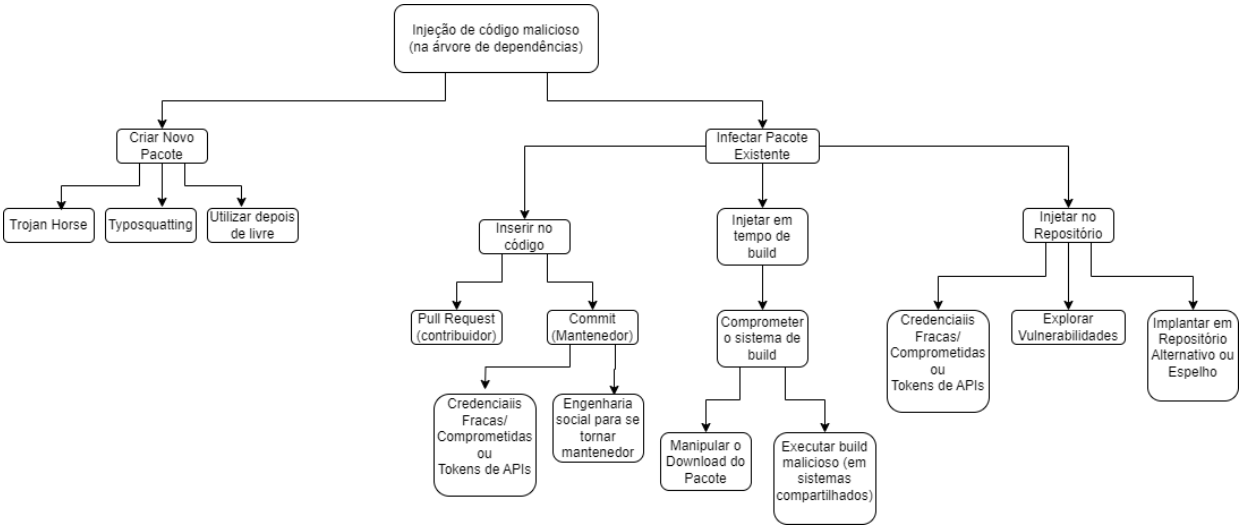


Figura 2.5: Árvore de ataques à cadeia de suprimentos de software

Fonte: Adaptada de [31]

A Figura 2.6 representa os objetivos dos ataques realizados sobre projetos de softwares nas plataformas "npm", "RubyGems"e "Pipy", e a soma dos objetivos dos ataques realizados sobre essas três plataformas (*overall*). Os números indicam as quantidades de ataques detectados.

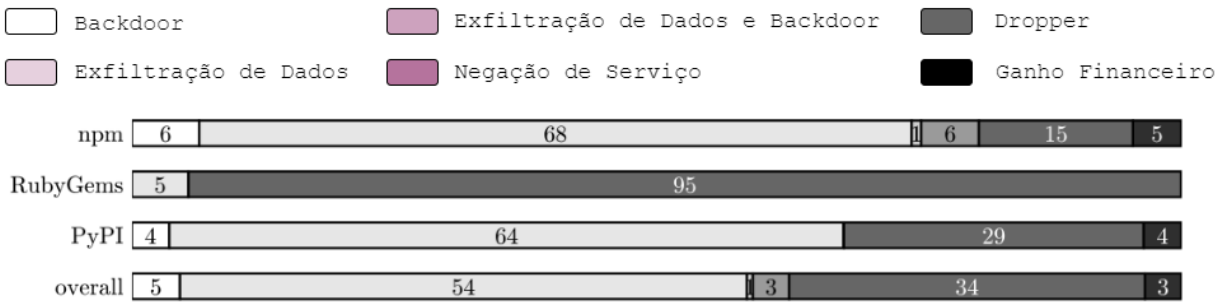


Figura 2.6: Objetivo primário dos pacotes maliciosos

Fonte: Adaptada de [31]

A árvore de ataques e os objetivos primários dos pacotes maliciosos apresentam grande semelhança, respectivamente, com a árvore de ataques e com os objetivos de ataques documentados por Ladisa et al. [32], de forma tal que ambos os trabalhos confirmam esses resultados entre si.

As duas ações iniciais da árvore proposta por Ohm, "Criar novo Pacote"e "Infectar Pacote Existente", coincidem com duas das três ações iniciais da árvore proposta por Ladisa, respectivamente *"Develop and Advertise Malicious Package from Scratch"* e *"Subvert Existing Package"*. A diferença é a ação *Create Name Confusion with Legitimate Package*, presente na árvore de Ladisa porém ausente da árvore de Ohm, que categoriza sob a ação "Criar novo Pacote"algumas das ações categorizadas por Ladisa sob a ação *Create*

Name Confusion with Legitimate Package. As ações de segundo nível sob "Infectar Pacote Existente", na árvore de Ohm, e *"Subvert Existing Package"*, na árvore de Ladisa, são idênticas.

Quanto aos objetivos dos ataques, ambos os estudos identificaram a instalação de backdoors (classificada por Ladisa como *reverse shell*, exfiltração de dados, instalação de *droppers* (módulos que permitem que outras funções maliciosas sejam baixadas e instaladas após o comprometimento inicial, negação de serviço e ganho financeiro.

Trazem ainda contribuições quanto à compreensão do potencial de alcance de ataques a cadeias de suprimentos de software. Ohm et al. observaram que pacotes maliciosos ficam disponíveis para download por uma média de 209 dias, com casos extremos em que permaneceram acessíveis por mais de 1.000 dias, ilustrando a ampla janela de oportunidade que os atacantes têm para que alvos em potencial os utilizem [31].

Já o estudo de Geer et al. fornece informações sobre o alcance potencial de ataques específicos, relatando que a versão maliciosa do pacote *event-stream* (utilizado para a implementação de fluxos de dados na aplicação) permaneceu disponível para download por 57 dias, período no qual houve mais de 7 milhões de downloads das diversas versões deste pacote (não tendo sido possível, infelizmente, detectar quantos desses downloads foram relativos à versão maliciosa). Traz ainda informações sobre o caso *ShadowHammer*, em que a empresa Kaspersky identificou que 57.000 de seus clientes foram afetados, e estimou que o ataque atingiu um total de cerca de 1 milhão de pessoas [30].

O potencial de impacto de ataques à cadeia de suprimentos tem atraído tipos de atacantes sofisticados, tais como Estados-nação. Artigo publicado pela empresa Checkmarx documenta campanhas de ataques à cadeia de suprimentos de software conduzidas pela Coreia do Norte em 2023, e alerta para a expectativa de aumento desse tipo de atividade pelo país em 2024. De forma semelhante, o ataque à cadeia de suprimentos de software que mais repercussão alcançou até o momento, conhecido como *Solarwinds*, também é atribuído a um Estado-nação, no caso, a Rússia [33][34].

2.2.1 Solarwinds

Solarwinds é o nome de uma empresa que produz e comercializa produtos voltados ao monitoramento e gestão de ambientes de infraestrutura de TI, e seu nome também é utilizado para identificar o mais conhecido ataque à cadeia de suprimentos até o momento. A sofisticação e alcance do ataque foram tão marcantes que provocaram artigos especializados em diversas publicações renomadas no ramo de cibersegurança, tais como IEEE Security and Privacy [35], Wired [36], TechCrunch [37], TechTarget [38] e Reuters [39], assim como análises aprofundadas por parte de grandes empresas de tecnologia, dentre as quais destaca-se empresa FireEye (atualmente chamada Mandiant), que foi a primeira a relatá-lo [40], e a Microsoft, que publicou uma série de quatro artigos detalhando-o [34][41][42][43].

O ataque, também conhecido como *Sunburst*, inseriu código malicioso em uma das bibliotecas que compõem o produto Orion IT (que tem a função de monitorar ambientes de rede), ainda dentro do ambiente de desenvolvimento da empresa Solarwinds, de forma tal que a biblioteca contendo o código malicioso foi digitalmente assinada pela própria empresa, e disponibilizada como atualização do produto para todos os seus clientes por meio de seus canais oficiais. Cerca de 18.000 clientes efetivamente fizeram o download

da atualização e assim foram infectados. Dentre esses clientes destacam-se departamentos governamentais norte-americanos, tais como o Department of Homeland Security, Department of State, Department of Commerce e Department of Treasury, além de grandes empresas privadas, dentre as quais destacam-se FireEye, Microsoft, Intel, Cisco e Deloitte [38][40].

A dinâmica de detecção e resposta ao ataque evidencia boas práticas no monitoramento dos ambientes de TI, como também na união de esforços para o combate a incidentes de grande impacto e escala.

O evento inicial que possibilitou a descoberta do ataque, dos atores envolvidos e das técnicas empregadas, foi a rotina de monitoramento de segurança implementada pela empresa FireEye. Em novembro de 2020, verificando os logs de autenticação no ambiente relativos ao dia anterior, a equipe da FireEye detectou um acesso de um usuário a partir de um equipamento diferente do usualmente utilizado por ele. Ao questionar o usuário sobre o registro de um novo equipamento, a equipe recebeu a confirmação de que isso não havia ocorrido. Com essa informação, o problema foi escalado internamente.

Ao analisar mais aprofundadamente o incidente, a empresa entendeu que precisaria de auxílio para melhor compreendê-lo e para obter uma telemetria mais acurada sobre o ambiente, e assim contactou a Microsoft, iniciando-se um esforço colaborativo global de busca de ameaças, centrada em torno do Microsoft Threat Intelligence Center [41].

Os especialistas envolvidos, atuando de maneira conjunta, após identificarem que o incidente era relacionado ao produto Orion IT, realizaram a descompilação do produto e a identificação das linhas de código maliciosas, além da identificação de cerca de 70 táticas, técnicas e procedimentos utilizadas pelos atacantes contra seus alvos, incluindo a movimentação lateral dentro da rede das organizações atacadas, roubo de credenciais de acesso, e exfiltração de dados. O ataque foi executado com uma sofisticação tal que seus responsáveis envidaram todos os esforços possíveis para encobrir seus rastros, incluindo a adoção de diferentes servidores de comando e controle para ataques a diferentes vítimas [42].

O ataque como classificado como de alta complexidade. A título de exemplo, o presidente da Microsoft declarou em entrevista à rede de notícias CBS que provavelmente foi o maior e mais sofisticado que o mundo já havia observado. Bruce Schneier, renomado especialista em cibersegurança, ao falar sobre o caso, destacou o fato de que software atualmente é um ativo crítico para a segurança nacional, e que mecanismos tradicionais de segurança já não são suficientes para deter ataques às cadeias de suprimentos de software, sendo necessárias novas técnicas de análises mais aprofundadas, verificação detalhada de origem dos softwares que consumimos (*code provenance*), e alertou ainda para o fato de que somente por meio de regulação formal será possível fazer com que as empresas produtoras de software efetivamente venham a elaborar e disponibilizar produtos mais seguros [39][35].

2.2.2 Log4J

Um outro caso de incidentes de segurança relacionados à cadeia de suprimentos de software é o caso da vulnerabilidade que ficou conhecida como *log4shell*, que afetou a biblioteca de software conhecida como *Log4j*.

Log4j é um *framework* que implementa funções de *logging* desenvolvido para ambientes Java pela fun-

dação Apache, sendo composto por uma API, sua implementação, e componentes auxiliares para utilização em vários casos de uso [44].

Log4Shell, por sua vez, é uma vulnerabilidade encontrada no *framework log4j*, catalogada sob o identificador CVE-2021-44228, que permite que um atacante que tenha controle sobre as as mensagens de log ou seus parâmetros possam executar código remotamente no equipamento onde o pacote *log4j* estiver sendo utilizado [45][46].

A vulnerabilidade foi descoberta pela empresa de tecnologia Alibaba em 24 de novembro de 2021, que a atribuiu o nível de criticidade 10 na escala CVSS (que utiliza um intervalo de 1 a 10, sendo 10 o nível mais crítico) [47]. Era uma vulnerabilidade *zero-day*, pois quando foi descoberta e divulgada ainda não havia nenhuma correção publicada pelo fabricante. Há evidências de que atacantes tenham começado a explorá-la já em 1º de dezembro de 2021, ou seja, apenas sete dias depois de ter sido descoberta, e sabe-se que foi utilizada para atividades como vazamento de senhas, exfiltração de informações sensíveis, e distribuição de *ransmoware*, tendo sido explorada até mesmo por Estados-nação tais como China, Irã, Coreia do Norte e Turquia [48].

O que tornou o caso tão representativo, além de seu nível máximo de criticidade e a simplicidade para sua exploração, foi a ampla utilização do Log4j como biblioteca em diversos projetos distintos de software, além da simplicidade da exploração da vulnerabilidade. Estima-se que seja utilizada por milhões de computadores em todo o mundo, e que 93% dos ambientes de nuvem fossem afetados pela vulnerabilidade quando descoberta, uma vez que é utilizada em uma grande diversidade de serviços, websites e aplicações, tanto por consumidores finais quanto por empresas. Estatísticas de sua utilização como bibliotecas em projetos de software, em 2021, indicavam que mais de 35.000 pacotes Java disponibilizados no repositório Maven utilizavam versões do *log4j* afetadas pela vulnerabildade, o que representava cerca de 8% dos pacotes ali existentes [48][49][50].

Outro fator que contribuiu para a criticidade da vulnerabilidade *log4shell* é a persistência a ela associada. Como muitos projetos de software utilizam a biblioteca, por vezes como dependência direta, e por vezes como dependência transitiva (algum componente de terceiros utilizado pelo software tem o pacote *log4j* como dependência), muitas vezes torna-se difícil identificar todas as ocorrências de uso da biblioteca nos softwares desenvolvidos ou utilizados pela organização. Adicionalmente, conforme o ciclo de desenvolvimento se desenrola, há casos em que softwares que já haviam sido corrigidos, por meio da atualização das versões do *log4j* que utilizavam, voltam a apresentar a vulnerabilidade, por meio da reutilização de versões antigas, vulneráveis [48].

Essa característica de persistência pode ser observada pelo fato de que em maio de 2023, cerca de um ano e meio depois da descoberta da vulnerabilidade, estatísticas indicavam que o *log4shell* ainda era a segunda vulnerabilidade mais explorada globalmente [48].

2.2.3 ShadowHammer

ShadowHammer é o nome de um *malware* que foi desenvolvido com o objetivo de funcionar como backdoor, possibilitando a exfiltração de informações de suas vítimas, bem como ser um canal para a instalação de funcionalidades adicionais a partir de seus servidores de comando e controle. Foi descoberto

pelo fabricante de soluções de segurança Kaspersky, que documentou suas principais características em três artigos publicados em seus sites [51] [52] [53].

O *malware* foi inserido como parte de pacotes de atualização de computadores da Asus, um dos maiores fabricantes mundiais de computadores pessoais, assinado por meio de um certificado digital legítimo, e disponibilizado nos canais oficiais de download da empresa. Posteriormente, foi incluído também nos canais oficiais de atualizações de outras três empresas fornecedoras software, as asiáticas Electronics Extreme, Innovative Extremist e Zepetto.

O artefato malicioso ficou disponível para download no site da Asus de junho a novembro de 2018, e estima-se que tenha atingido cerca de 1 milhão de usuários. Considerados apenas os clientes da Kaspersky, foram realizados 57 mil downloads do *malware*.

Apesar do alcance, os atacantes tinham alvos bem específicos, representados por 600 endereços MAC codificados no próprio *malware*, referentes a computadores instalados em diversas localidades, inclusive no Brasil.

Não se encontrou registros de ações específicas que tenham sido realizadas pelos atacantes como resultado do sucesso na distribuição de seus artefatos maliciosos.

2.2.4 Event-stream

O ataque ao componente *event-stream* é um caso interessante a ser analisado pois, diferentemente dos demais casos aqui citados, teve como vetor de ataque a delegação do papel de mantenedor do pacote a um outro usuário que solicitou esse papel (que, na verdade, era o atacante) [54].

No ano de 2018 o pacote *event-stream* era utilizado por cerca de 1.600 projetos de software, e era objeto de cerca de 1,5 milhão de downloads por semana. O atacante ganhou a confiança do mantenedor original, por meio da realização de diversas contribuições significativas para o projeto, e, então, solicitou os direitos de acesso para que se tornasse o mantenedor do pacote. Após recebidos os direitos, inseriu código malicioso no produto, especificamente direcionado ao software "Copay", utilizado como software de carteira digital de *bitcoins*, tanto para computadores pessoais quanto para dispositivos móveis [54].

Embora o pacote *event-stream* fosse utilizado por uma ampla gama de outros softwares, o atacante direcionou o código malicioso utilizando o campo de descrição do software "Copay"como chave para descriptografar suas funções maliciosas. Para todos os demais projetos, como suas descrições seriam diferentes, a descriptografia não funcionaria, e a execução do código malicioso se encerraria normalmente, sem alterar suas funcionalidades originais [54].

O ataque tinha como objetivo roubar as credenciais de acesso às carteiras digitais dos usuários do Copay para um servidor controlado pelo atacante, que, então utilizaria tais credenciais para roubar *bitcoins* de seus proprietários. O pacote malicioso ficou disponível para download entre 4 de setembro e 20 de novembro de 2018 [54].

2.2.5 NotPetya

NotPetya foi um ataque à cadeia de suprimentos executado contra um software de contabilidade largamente utilizado por empresas ucranianas, o M.E.Doc. Pesquisadores da empresa de segurança Eset acreditam que os atacantes tiveram acesso ao código-fonte do software, e assim inseriram código malicioso em módulos legitimos do produto, que foram baixados por seus usuários como parte de atualizações de versão. O código malicioso atuava como *ransomware*, criptografando os arquivos do equipamento infectado e exigindo um resgate de 300 dólares em bitcoins.

Entretanto, a real intenção dos atacantes era efetivamente causar dano, de forma que na prática fizeram com que a descriptografia dos dados se tornasse altamente improvável. Uma característica interessante é que o NotPetya não utilizava um servidor próprio de comando e controle, mas se comunicava diretamente com o servidor oficial de atualização do software M.E.Doc, o que levou os pesquisadores a crerem que o próprio servidor também havia sido comprometido pelos atacantes [55].

2.2.6 CCleaner

O CCleaner é um software para otimização de computadores pessoais que tem ampla aceitação no mercado global, desenvolvido e publicado pela empresa Avast. No ano de 2016, imediatamente anterior ao ataque aqui reportado, o software já havia sido baixado mais de 2 bilhões de vezes, e apresentava um crescimento semanal de 5 milhões de novos usuários por semana [56].

Em 2017, atacantes conseguiram acesso a uma estação de trabalho de um dos desenvolvedores do produto, por meio do software de acesso remoto "TeamViewer", provavelmente utilizando credenciais que haviam sido descobertas anteriormente. A partir dessa estação, acessaram uma segunda estação de trabalho de desenvolvedores, e, assim, foram capazes de inserir código malicioso na versão 5.33 do produto, código esse que tinha por objetivo a exfiltração de informações [57].

A versão infectada ficou disponível para download pelos clientes no período de 2 de agosto a 13 de setembro de 2017, quando foi detectada pela empresa de segurança Cisco Talos. Estima-se que a versão infectada tenha tido cerca de 2,27 milhões downloads, e que os atacantes tenham conseguido instalar o componente de segundo estágio do ataque, a partir dessa infecção inicial, em 40 estações de trabalho selecionadas dentre grandes empresas de tecnologia, dentre as quais Google, Microsoft, Cisco, Intel, Samsung, Sony, HTC, Linksys, D-Link, Akamai e VMware. Esse componente de segundo estágio faria o download do módulo de exfiltração de dados, no terceiro estágio planejado para o ataque, porém não há evidências de que a última fase tenha sido executada [57].

2.2.7 Codecov

Codecov é um produto comercial voltado a testes automatizados de software. Como parte do produto, há um módulo chamado *"Codecov Bash Uploader"*, responsável pela funcionalidade de enviar à console principal informações de configuração sobre o ambiente onde estiver sendo executado, colecionar relatórios e também enviá-los à console, além de realizar a integração do Codecov com produtos de terceiros, tais como o Github [58].

No dia 1º de abril de 2021, a empresa responsável pelo produto descobriu que atacantes haviam obtido acesso ao seu ambiente, alterado o módulo *"Codecov Bash Uploader"*, e disponibilizado este módulo contendo código malicioso para download por parte dos clientes por meio do canal oficial de distribuição. O ataque foi possível devido a uma falha no procedimento de geração de imagens *docker* utilizado pelos desenvolvedores, que possibilitou aos atacantes obterem credenciais de acesso ao ambiente de construção do software. O ataque foi descoberto a partir da observação de um dos usuários do produto de que o código *hash* calculado para o módulo instalado em seu ambiente era diferente do código *hash* publicado no site oficial do fabricante [58][59].

O código malicioso, composto por apenas uma linha de programação dentro de um módulo que continha cerca de 1.800 linhas, fazia com que as informações usualmente enviadas à console do produto fossem também enviadas a um outro servidor controlado pelos atacantes, que, assim, receberiam informações sensíveis dos clientes do Codecov. A versão maliciosa tinha a capacidade de exfiltrar credenciais, tokens ou chaves utilizadas no ambiente de CI/CD do cliente, às quais o módulo comprometido tivesse acesso durante sua operação normal. Quaisquer serviços ou aplicações cujos acessos fossem dependentes dessas credenciais poderiam ser afetados. Sabe-se que, como resultado da obtenção dessas credenciais, os atacantes acessaram repositórios privados no site Github [59].

A versão maliciosa do produto ficou disponível para download entre 31 de janeiro e 1º de abril de 2021, e quaisquer dos mais de 27.000 clientes do produto podem ter sido atingidos. Dentre esses clientes, as empresas Twilio, Hashicorp, Rapid7 e Confluent, todas produtoras de software, publicaram declarações informando como haviam sido afetadas [59].

2.2.8 Kaseya

Kaseya é o nome do fabricante da solução Kaseya VSA, que realiza o gerenciamento de remoto de operações de TI por parte de provedores de serviço gerenciados [60].

Em julho de 2021, o Kaseya VSA foi comprometido, permitindo que os atacantes inserissem código malicioso nos pacotes de atualização disponibilizados através de seus canais oficiais. O código malicioso executava a função de *ransomware*, que alcançou os clientes dos provedores de serviço gerenciados. O ataque foi atribuído ao grupo de *hackers* REvil, também conhecido como Sodikinobi, de origem russa [60].

Foi perpetrado por meio da exploração de uma vulnerabilidade conhecida no produto (identificada como CVE-2021-30116), enquanto o fabricante estava desenvolvendo uma correção para mitigá-la. Estima-se que tenha atingido pouco menos de 60 clientes diretos, provedores de serviços gerenciados, e cerca de 1.500 clientes finais desses provedores [61][62].

Os atacantes exigiram o pagamento de um valor de 570 milhões de dólares do fabricante para descriptografarem os dados de todos os clientes afetados. O fabricante, entretanto, conseguiu adquirir uma chave de descriptografia universal, e a forneceu aos clientes afetados. Entretanto, há notícias de que os atacantes receberam pagamentos de *ransowmare* de alguns clientes finais, que variaram entre 40 mil e 220 mil dólares [62].

2.2.9 XZ Utils

O pacote XZ é um utilitário que executa a compressão de arquivos, e é amplamente utilizado em distribuições Linux e MacOS, incluindo o serviço sshd, que implementa acesso remoto a máquinas Linux [63].

O ataque foi cuidadosamente planejado e executado ao longo de dois anos, o que sugere ser um trabalho de *hackers* associados a Estados-nação, que tipicamente têm recursos para realizar operações longas e sofisticadas. O atacante aparentemente iniciou seu trabalho por meio da utilização de contas falsas que reportavam bugs sobre o pacote, o que pressionou seu mantenedor a procurar ajuda para corrigir as falhas apontadas. O atacante então começou a participar do projeto e passou a enviar vários *pull requests* para corrigir as falhas em questão. Construindo uma boa reputação a partir dessas contribuições, eventualmente obteve o *status* de mantenedor do projeto [63].

A partir do novo nível de acesso obtido, o atacante então inseriu no pacote seu código malicioso, que tinha como objetivo atuar como um *backdoor* nos sistemas onde estivesse presente. Sua versão maliciosa foi eventualmente incluída em versões mais recentes do serviço sshd à época, serviço esse que provê acesso remoto aos equipamentos em que estiver em execução. Por meio do serviço sshd comprometido, o atacante teria acesso remoto a qualquer servidor Linux que estivesse exposto à internet [63].

A vulnerabilidade foi detectada antes que as versões infectadas do pacote XZ e do serviço sshd fossem incluídas nas versões oficiais das principais distribuições Linux. Um engenheiro de software da Microsoft percebeu que acessos que realizava em um servidor Debian Linux estavam ocorrendo de forma mais lenta do que o usual, e, rastreando a origem do comportamento, chegou ao pacote XZ, e reportou o problema aos seus responsáveis. O histórico completo de como a vulnerabilidade foi descoberta encontra-se em uma postagem que o próprio engenheiro realizou em um fórum na Internet [64], e a vulnerabilidade eventualmente foi classificada sob o código CVE-2024-3094 [16].

A serveridade, complexidade e alcance dos ataques a cadeias de suprimentos de software, bem como a tendência observada de incremento desses ataques, chamaram a atenção da indústria de tecnologia da informação, e as organizações que se dedicam à publicação de *frameworks* e modelos de maturidade referentes a desenvolvimento seguro passaram a abordar o tema em suas versões mais atuais.

2.3 PRINCIPAIS NORMAS, PADRÕES E MODELOS DE MATURIDADE DE DESENVOLVIMENTO SEGURO

Neste tópico serão apresentados os *frameworks* e modelos de maturidade identificados na literatura como sendo voltados à implementação de boas práticas em desenvolvimento seguro, com foco na identificação de recomendações referentes à segurança da cadeia de suprimentos de software, resumindo suas principais características.

2.3.1 ISO 27.034 - Information technology - Security techniques - Application security

A International Organization for Standardization - ISO, é uma organização internacional não governamental, composta por representantes de 172 países, voltada à elaboração de padrões sobre boas práticas relativa a uma série de atividades, desde a fabricação de produtos ao gerenciamento de processos [65].

A norma ISO 27.034 tem com objetivo prover orientações quanto à integração de medidas e controles de segurança no processo de gestão de aplicações, podendo ser utilizada nos cenários de desenvolvimento interno de aplicações, terceirização do desenvolvimento, ou de aquisição de aplicações junto a fornecedores externos.

É estruturada em seis normativos distintos, identificados como Parte 1 a Parte 6, sendo a Parte 1 relativa à visão geral e conceitos, a partir da qual são trazidos os trechos de maior interesse para o nosso estudo [66].

Tem como público-alvo equipes relacionadas ao desenvolvimento ou aquisição de soluções de software:

- gerentes de desenvolvimento e de produtos: responsáveis pela gestão das aplicações durante todo o seu cilo de vida, incluindo os estágios de provisionamento e produção, e tipicamente são responsáveis por balancear os custos de implementação de segurança e o valor que as aplicações representam para a organização. Atuam ainda na autorização do Nível de Confiança Desejado para as aplicações, e na determinação de que controles de segurança e procedimentos de verificação devem ser implementados;
- equipes de provisionamento e operação: são as responsáveis pelo projeto, desenvolvimento e manutenção das aplicações durante todo seu ciclo de vida, e devem, dentre outras atribuições, compreender quais controles de segurança devem ser aplicados em cada etapa do ciclo de vida, os controles de segurança que devem ser implementados na aplicação propriamente dita, e garantir que esses controles atendam aos requisitos de segurança definidos;
- equipes de aquisições: são as responsáveis por definir os requisitos de segurança necessários para as aquisições, selecionar os fornecedores que atendam a esses requisitos, verificar as evidências de aplicação de controles de segurança por parte de serviços terceirizados, e verificar as evidências de que os controles de segurança tenham sido efetivamente implementados

Seu conteúdo estabelece conexões diretas com outras normas ISO voltadas à segurança e gestão de riscos de segurança da informação, além de apresentar princípios relacionados à segurança de aplicações. Dentre esses princípios, destacam-se os relacionados aos requisitos de segurança, à dependência do contexto de uso da aplicação, à adequação dos investimentos em segurança e à capacidade de demonstrar que a aplicação é segura.

Adicionalmente, relaciona controles de segurança de aplicações capazes atender às recomendações pertinentes contidas na ISO 27.002, bem como auxilia na implementação do processo de gestão de riscos de segurança da informação proposto pela ISO 27.005 [67][68].

Com relação aos requisitos de segurança de aplicações, estabelece que devem ser tratados da mesma forma que os requisitos de funcionalidade, qualidade e usabilidade, e traz da norma ISO/IEC/IEEE 29.148 [69] a definição de que requisitos devem ter como características serem necessários, abstratos, não ambíguos, consistentes, completos, concisos, viáveis, rastreáveis e verificáveis.

Destaca que a segurança de aplicações está diretamente relacionada aos contextos de negócio, nos quais se incluem a política de segurança da informação, a metodologia de desenvolvimento e as melhores práticas de desenvolvimento aplicáveis às linguagens de programação utilizadas na organização, de contextos regulatórios, onde se inserem leis o outros normativos que devem ser observados pela organização, e de contextos tecnológicos, onde se inserem softwares comerciais, softwares gerenciadores de bancos de dados, infraestrutura tecnológica, e produtos utilizados pela aplicação (tais como componentes desenvolvidos e utilizados por terceiros).

Com respeito à adequação dos investimentos, introduz o conceito de Nível de Confiança Desejado para a aplicação e recomenda que os custos associados à implementação de controles de segurança sejam compatíveis com esse nível de confiança.

A partir desses conceitos, define então o Processo de Gestão de Segurança de Aplicações, dividido em duas etapas principais. A primeira (Organization management processes) define o conjunto normativo que irá orientar o desenvolvimento seguro de aplicações, enquanto a segunda (Application Security Management Process), é onde efetivamente é realizada a análise de riscos, são definidos os requisitos de segurança, aplicados os controles de segurança pertinentes, para que então seja alcançado o Nível de Confiança Desejado para a aplicação.

Aborda questões como gestão de risco das aplicações e de seus componentes, definição de acordos de níveis de serviço e avaliação de fornecedores (que podem ser adaptados para a definição de requisitos de segurança associados também a componentes de terceiros disponibilizados como código aberto), e trata da aplicação de controles de segurança sobre o ciclo de desenvolvimento e sobre o produto final, atividades efetivamente relacionadas à segurança da cadeia de suprimentos de software.

Além disso, pode ser aplicada em conjunto com a norma ISO 27.036, que aborda especificamente a questão da cadeia de suprimentos, e que será detalhada mais adiante [27].

2.3.2 NIST Secure Software Development Framework

O National Institute of Standards and Technology (NIST) é uma organização vinculada ao Departamento de Comércio norte-americano, que tem como missão promover a inovação e competitividade industrial daquele país, por meio do avanço da ciência, padrões e tecnologias de medição [70].

Elabora uma série de padrões a serem observados pela indústria, dentre os quais encontram-se os padrões da série Special Publication 800 (SP 800), voltados à Tecnologia da Informação [71].

Dentre esses, elaborou elaborou o padrão SP 800-218, conhecido como Secure Software Development Framework (SSDF), pois identificou que poucos *frameworks* de ciclo de vida de desenvolvimento de software tratavam de forma aprofundada as questões de segurança de software. O SSDF que traz recomendações de segurança de alto nível a serem integradas a diferentes *frameworks* de desenvolvimento de

software[72].

Define seus públicos-alvo de forma semelhante à ISO 27.034, sendo direcionado a produtores de software (fornecedores comerciais, empresas contratadas para desenvolvimento terceirizado, e equipes internas de desenvolvimento), bem como a organizações que realizem aquisições de software, incluindo agências de governo e outras entidades.

O SSDF defende a adoção de práticas de desenvolvimento seguro de software por três razões principais, que são a redução de vulnerabilidades no software final, a redução do impacto potencial dessas vulnerabilidades e a identificação das causas raiz das vulnerabilidades encontradas visando a redução de novas ocorrências, e advoga que quanto mais no início do ciclo de desenvolvimento de software as práticas de segurança forem aplicadas, menores serão o esforço e o custo associados, em um conceito que identifica como *shifting left*.

Não detalha a implementação de cada prática, mas descreve os resultados esperados de sua aplicação e sugere uma série de referências para auxiliar na sua implementação, dentre as quais destacam-se a própria norma ISO 27.034, o NIST Cybersecurity Framework (CSF), o Microsoft Security Development Lifecycle (SDL), o Building Security In Maturity Model (BSIMM), o OWASP Software Assurance Maturity Model (SAMM), o OWASP Software Component Verification Standard (SCVS), o CNCF Software Supply Chain Best Practices, sendo os dois últimos diretamente relacionados à cadeia de suprimentos de software. Todos serão detalhados em outros tópicos dessa dissertação.

O SSDF organiza as práticas de desenvolvimento seguro em quatro grupos principais, a saber:

1. Preparar a Organização (*Prepare the Organization* - PO): preparar as pessoas, processos e tecnologia para executar as tarefas de desenvolvimento seguro
2. Proteger o Software (*Protect the Software* - PS): proteger todos os componentes de software contra modificações e acessos não autorizados
3. Produzir Software Seguro (*Produce Well-Secured Software* - PW): produzir software com baixa quantidade de vulnerabilidades
4. Responder a Vulnerabilidades (*Respond to Vulnerabilities* - RV): identificar e corrigir vulnerabilidades residuais em seus softwares, e tomar providências para evitar que se repitam em futuros projetos ou versões

Cada grupo contém uma série de práticas associadas. Cada prática é descrita em termos de sua definição, tarefas que devem ser executadas, exemplos de implementação e referências. Em sua versão atual (1.1), traz uma relação total de 19 práticas subdivididas em um total de 42 tarefas, das quais destacam-se, como mais diretamente associadas à segurança da cadeia de suprimentos de software, as seguintes (em tradução livre):

- PO.1 - Definir Requisitos de Segurança para o Desenvolvimento de Software;
  - PO.1.3 - Comunicar os requisitos a todas as terceiras partes que irão prover componentes comerciais de software para reutilização nos softwares desenvolvidos pela própria organização;

- PO.4 - Definir e utilizar critérios para a verificação de segurança de software;
- PO.5 - Implementar e manter ambientes seguros para o desenvolvimento de software;
- PS.2 - Prover um mecanismo para a verificação de integridade de software;
- PS.3 - Arquivar e proteger cada versão de software;
  - PS 3.2 - Coletar, proteger, manter e compartilhar informações de origem (*provenance*) sobre cada componente de cada versão de software (por exemplo, em uma software bill of materials [SBOM]).
- PW.4 - Reutilizar software já existente e seguro sempre que possível, ao invés de duplicar funcionalidades;
  - PW.4.1 - Adquirir e manter componentes de software seguros (tais como bibliotecas de software, módulos, *middleware* e *frameworks*) de fornecedores comerciais, código aberto, e outros tipos de terceiros, para utilização em projetos de software da organização;
- RV.1 - Identificar e confirmar vulnerabilidades de forma contínua;
  - RV 1.1 - Obter informações a partir de adquirentes, usuários e fontes públicas sobre potenciais vulnerabilidades nos softwares e em componentes de terceiros que esses softwares utilizarem, e investigar todos os relatos confiáveis;
- RV.2 - Avaliar, priorizar e remediar vulnerabilidades.

O NIST SSDF é utilizado como base para atendimento às recomendações do governo americano relativas à segurança da cadeia de suprimentos de softwares, emanadas por meio da Ordem Executiva (*Executive Order*) nº 14028, intitulada *"Executive Order in Improving the Nations Cybersecurity"* [19].

A esse respeito, traz um apêndice onde faz um mapeamento entre as recomendações da Ordem Executiva e as práticas e tarefas preconizadas pelo *framework*, bem como uma publicação associada contendo orientações para agências de governo, produtores e fornecedores de software com relação ao seu atendimento [73].

2.3.3 OWASP Software Assurance Maturity Model

O OWASP Software Assurance Maturity Model (OWASP SAMM) é um modelo de maturidade relativo ao desenvolvimento de software seguro, que tem por objetivo prover um modelo mensurável para análise e melhoria do ciclo de vida do desenvolvimento seguro de software, estabelecendo níveis de maturidade que variam de 0 a 3 [74].

É dividido em cinco funções de negócio (Governança, Projeto, Implementação, Verificação e Operação), e para cada função define um conjunto de práticas de segurança, em um total de 15 práticas distribuídas pelas cinco funções. Cada prática, por sua vez, é subdividida em fluxos de execução (*streams*) que agrupam atividades correlacionadas, e os *streams* são efetivamente classificadas nos níveis de maturidade acima mencionados, conforme estrutura representada pela Figura 2.7.

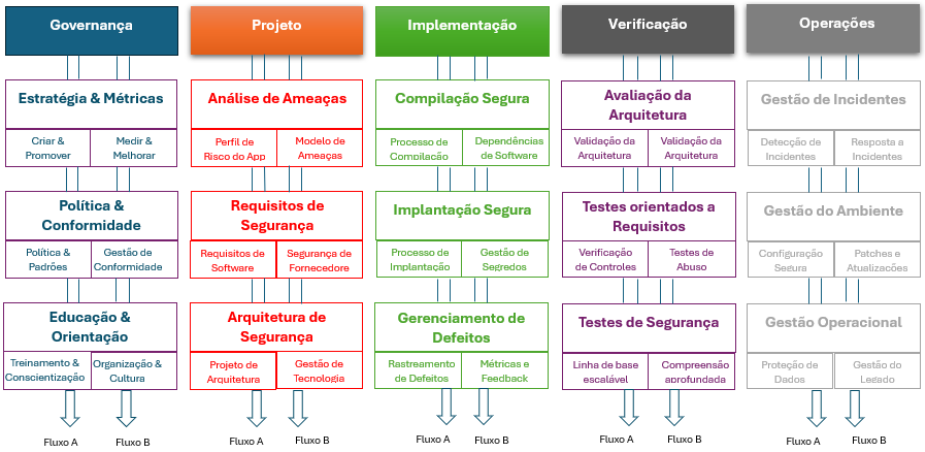


Figura 2.7: OWASP SAMM - Estrutura

Fonte: Adaptada de [74]

Embora não mencione explicitamente a segurança da cadeia de suprimentos de software, recomenda práticas de segurança que são diretamente aplicáveis a esse contexto.

Tomando como base os pilares da cadeia de suprimentos de software, mencionados na Seção 2.1, as práticas de *Secure Build* (Compilação Segura) e *Secure Deployment* (Implantação *Deploy* Segura) são diretamente mapeáveis sobre os pilares "Dependências", "Compilação"e "*Deploy* ou Distribuição".

A prática de *Secure Build* (Compilação Segura), é subdividida nos *streams Build Process* (Processo de Compilação) e *Software Dependencies* (Dependências de Software), com tarefas atribuídas aos três diferentes níveis de maturidade, conforme observa-se na Tabela 2.1.

Tabela 2.1: OWASP SAMM - Níveis de maturidade para a prática de Compilação Segura

Nível de Maturidade	**Processo de Compilação**	**Dependências de Software**
1 - Processo de compilação é repetitível e consistente	Criar uma descrição formal do processo de compilação de forma que seja consistente e repetitível	Criar Listas de Componentes (SBOMs - Software Bill of Materials) de suas aplicações e avaliá-las oportunamente
2 - O processo de compilação é otimizado e completamente integrado ao fluxo de trabalho	Automatizar seu pipeline de compilação e torná-lo seguro por meio do uso de ferramentas. Adicionar verificações de segurança à pipeline de compilação	Avaliar as dependências e garantir reação em tempo hábil a situações que tragam risco a suas aplicações
3 - O processo de compilação auxilia na prevenção da implantação de problemas conhecidos no ambiente de produção	Definir verificações de segurança obrigatórias no processo de compilação e garantir que a compilação de artefatos não conformes apresente erro	Analisar dependências utilizadas quanto a problemas de segurança de forma comparável ao código da própria organização

Fonte: Adaptada de [74]

Secure Deployment é subdividida nos *streams Deployment Process* (Processo de *Deploy* e *Secret Management* (Gerenciamento de Credenciais). Trata exclusivamente da função de *deploy*, que executa a implantação dos artefatos nos ambientes de desenvolvimento, testes, homologação e produção da organização, porém não aborda a questão de distribuição de artefatos finais para terceiros. Seu detalhamento pode ser observado na Tabela 2.2.

O *framework*, entretanto, não trata especificamente da segurança do código-fonte, embora sua prática de Design (Projeto) aborde aspectos relacionados à análise de ameaças, requisitos de segurança e arquite-

Tabela 2.2: OWASP SAMM - Níveis de maturidade para a prática de Implantação Segura

Nível de maturidade	Processo de deploy	Processo de gerenciamento de credenciais
1 - O processo de deploy é completamente documentado	Formalizar o processo de deploy e implementar segurança nas ferramentas e processos utilizados	Utilizar medidas básicas de proteção para limitar o acesso às credenciais de produção
2 - O processo de deploy contém marcos de verificação de segurança	Automatizar o processo de deploy em todas as etapas e utilizar testes de verificação de segurança apropriados	Injetar as credenciais dinamicamente durante o processo de implantação a partir de repositórios seguros, e auditar os acessos a esses repositórios
3 - O processo de deploy é completamente automatizado e incorpora verificação automatizada de todos os marcos críticos	Verificar automaticamente a integridade de todo o software implantado, independentemente de ter sido desenvolvido internamente ou externamente	Melhorar o ciclo de vida das credenciais, gerando-os regularmente e garantindo o uso adequado.

Adaptada de [74]

tura de segurança.

2.3.4 Building Security In Maturity Model

O *Building Security in Maturity Model* (BSIMM) é um *framework* proposto pela empresa Synopsys, sendo um modelo descritivo sobre as melhores práticas efetivamente utilizadas por organizações em todo o mundo relativas ao tema de desenvolvimento de software seguro. De acordo com o documento BSIMM 14 - Report 2023 [75], sua primeira versão, BSIMM 1, foi elaborada em 2009, com base em um levantamento inicial realizado junto a 9 organizações, e tem evoluído continuamente, alcançando, em 2023, sua 14ª versão, que reúne as principais práticas de segurança para aplicações adotadas por 130 organizações.

O modelo é organizado em 4 domínios, Governança, Inteligência, Pontos de contato com o SDLC (*Software Development Lifecicle*), e Implantação, subdivididos em 12 práticas, que congregam um total de 126 atividades distintas, atuando como controles preventivos, detectivos, corretivos ou compensatórios. As atividades, por sua vez, são agrupadas em três níveis de frequência de utilização observada nas organizações entrevistadas. No Nível 1 encontram-se as atividades mais praticadas, seguidas pelas atividades um pouco menos frequentes, que compõem o Nível 2 e, por fim, as atividades mais raras, onde normalmente se encontram as mais recentemente adotadas, que integram o Nível 3.

A estrutura básica de domínios e práticas e ilustrada na Tabela 2.3.

Tabela 2.3: BSIMM - Estrutura de Domínios e Práticas

Governança	Inteligência	Pontos de Contato no SSDL	Implantação
Estratégia e Métricas	Modelos de Ataque	Análise de Arquitetura	Testes de Invasão
Conformidade e Políticas	Funcionalidades de Segurança e Projeto	Revisão de Código	Ambiente de Sofware
Treinamento	Padrões e Requisitos	Testes de Segurança	Gerência de Configuração e de Vulnerabilidades

Adaptada de [75]

O documento sobre o BSIMM 14 traz um tópico sobre tendências observadas no campo da segurança de software. Dentre as tendências observadas, menciona a Gestão de riscos da Cadeia de suprimentos de software, que tem ganhado importância crescente desde a publicação da Ordem Executiva nº 14028 pelo governo americano [19] e em função de ocorrências efetivas de ataques à cadeia de suprimentos.

Pondera que, embora a utilização de componentes de código aberto desenvolvidos por terceiros seja uma prática adotada há décadas, apenas há cerca de 5 anos a gestão de riscos da cadeia de suprimentos de software vem sendo adotada de forma mais ampla pelas organizações. Com efeito, observa que no último ano uma maior quantidade de organizações passou a incluir formalmente essa disciplina em seus programas de segurança, de forma a gerenciar riscos tanto de softwares comerciais eventualmente adquiridos como riscos oriundos de componentes de código aberto.

Dentre as 126 atividades identificadas pelo BSIMM, aquelas alinhadas diretamente à questão da segurança da cadeia de suprimentos de software que vêm observando incremento na frequência de adoção são:

- Integrar a gestão de riscos da cadeia de suprimentos de software (Governança)
- Utilizar componentes e serviços baseados no conceito de "segurança por projeto- *security by design*"(Funcionalidades de Segurança e Projeto)
- Identificar código aberto - opensource (Padrões e Requisitos)
- Controlar o risco de código aberto (Padrões e Requisitos)
- Incluir testes de segurança na automação de testes de qualidade (Testes de Segurança)
- Definir parâmetros e configurações seguras para a implantação de software (Ambiente de Software)
- Proteger a integridade do código (Ambiente de Software)
- Utilizar proteção para o código (Ambiente de Software)
- Criar listas de componentes para os softwares implantados em produção - SBOMs (Ambiente de Software)
- Executar análise de componentes de terceiros nos repositórios de software (Ambiente de Software)
- Proteger a integridade das ferramentas da esteira de desenvolvimento (Ambiente de Software)

2.3.5 Microsoft Security Development Lifecycle

O *Microsoft Security Development Lifecycle* (Microsoft SDL), é um *framework* de desenvolvimento seguro elaborado pela Microsoft para sua utilização em projetos internos, composto por orientações, melhores práticas, ferramentas e processos, e compartilhado de forma pública desde 2008 e constantemente atualizado [76].

As práticas preconizadas pelo Microsoft SDL são as seguintes:

- Prover treinamento
- Definir requisitos de segurança
- Definir métricas e relatórios de conformidade

- Executar modelagem de ameaças
- Estabelecer requisitos de projeto
- Definir e utilizar padrões de criptografia
- Gerenciar os riscos de segurança sobre o uso de componentes de terceiros
- Utilizar ferramentas aprovadas
- Executar testes de segurança sobre o código fonte (SAST)
- Executar testes de segurança dinâmicos (DAST)
- Executar testes de invasão
- Estabelecer um processo de resposta a incidentes

O *framework*, em seu tópico relativo ao gerenciamento de riscos segurança sobre o uso de componentes de terceiros, destaca a importância da compreensão a respeito do impacto de segurança que o uso desses componentes (comerciais ou de código aberto) pode trazer para o projeto de software, devendo ser realizado um inventário de todos os componentes de terceiros bem como ser estabelecido um plano para responder às vulnerabilidades neles encontradas.

Com relação à gestão do uso de componentes de terceiros de uma forma geral recomenda a utilização de um conjunto de boas práticas identificadas pela iniciativa Safecode, publicadas em um documento intitulado *"Managing Security Risks Inherent in the Use of Third party Components"* [77], que define um processo de gestão de componentes de terceiros composto por quatro elementos principais: Manter uma lista de componentes de terceiros, Avaliar os riscos de segurança, Mitigar ou aceitar os riscos, e Monitorar mudanças nos componentes de terceiros.

Quanto à gestão de riscos do uso de componentes de código aberto especificamente, a Microsoft, em conjunto com a organização OpenSSF, elaborou e mantém atualizado o Secure Supply Chain Consumption Framework, conhecido como S2C2F, que será mais bem detalhado em tópicos mais adiante nesta dissertação [78].

2.4 PRINCIPAIS NORMATIVOS SOBRE SEGURANÇA DA CADEIA DE SUPRIMENTOS DE SOFTWARE

Além dos *frameworks* e normativos sobre o desenvolvimento seguro de software, há também *frameworks* e normativos específicos para a segurança da cadeia de suprimentos de software, que trazem melhor especificação e detalhamento sobre o tema.

Neste tópico abordaremos os normativos publicadas pelas instituições que têm maior aceitação no cenário nacional.

2.4.1 ISO 27036 - Cybersecurity - Supplier relationships

A norma ISO 27.036 provê orientações para que as organizações garantam a segurança de suas informações, e de seus sistemas de informação, em suas relações com fornecedores de uma forma geral, sendo composta por 4 partes:

1. *Overview and concepts* [27]
2. *Requirements* [79]
3. *Guidelines for hardware, software and services supply chain security* [80]
4. *Guidelines for security of cloud services* [81]

A Parte 3 da norma aborda especificamente a cadeia de suprimentos de software (além das cadeias de suprimentos de hardware e de serviços), e se propõe a auxiliar no incremento de segurança dessa cadeia com relação aos seguintes aspectos:

- incrementar a visibilidade e rastreabilidade das cadeias de suprimentos de hardware, software e serviços
- melhorar a compreensão por parte dos adquirentes com relação à proveniência dos componentes adquiridos, bem como das práticas utilizadas em sua elaboração
- prover informações sobre que componente pode ter sido comprometido em um caso de incidente de segurança que tenha afetado o produto ou serviço

Traz os controles de segurança que considera essenciais para as cadeias de suprimentos de hardware, software e serviços, inicialmente baseadas na nas normas ISO voltadas ao ciclo de vida dos produtos (ISO 15.288 [82] e 12.207 [83]), bem como dos normativos ISO 27.001 [84] e ISO 27.002 [67]. Dentre os controles citados, os mais diretamente aplicáveis à cadeia de suprimentos de software são os seguintes:

- cadeia de custódia: comprovações de que as transferências (ex: *downloads* de software são autorizadas, transparentes e verificáveis)
- proteção contra modificações não autorizadas (*tamper resistance*)
- verificação de código, para identificação de código malicioso
- existência de lista de componentes de software relativa ao produto envolvido (SBOM - *Software Bill of Materials*)
- treinamento referente à cadeia de suprimentos de hardware, software e serviços
- identificação e reposta a vulnerabilidades por parte dos fornecedores
- requisitos de segurança bem definidos

- evitar componentes não autenticos e não verificados

Adicionalmente, distribui os controles de segurança entre os processos de aquisição e fornecimento dentro da cadeia de suprimentos, bem como de processos que a norma classifica como habilitadores da cadeia de suprimentos, a saber, gestão do ciclo de vida, gestão de infraestrutura, gestão de portfolio, gestão de recursos humanos, gestão de qualidade, gestão de conhecimento, planejamento de projetos, conrole de projetos, gestão do processo decisório, gestão de riscos, gestão de configurações, gestão da informação, mensuração, garantia de qualidade, análise de negócios e da missão, definição de requisitos das partes interessadas, definição dos requisitos de sistema, definição da arquitetura de sistemas, definição de projetos de software, análise dos sistemas, implementação, integração, transição, validação, operação, manutenção e, por fim, descarte.

2.4.2 NIST - Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations

O padrão, identificado como NIST SP 800-61r1, aborda de forma ampla a gestão de riscos cibernéticos sobre cadeias de suprimento de uma forma geral, da qual os riscos da cadeia de suprimentos de software são um caso específico [85].

É estruturado em torno de outros dos padrões previamente estabelecidos pelo NIST, adaptando-os ao cenário da cadeia de suprimentos, tais como os padrões SP 800-39 [86] e SP 800-37 [87], que definem, respectivamente, o processo de gestão de riscos de segurança da informação, distribuído em 3 níveis de responsabilidade, e as tarefas de análise de riscos aplicáveis a sistemas de informação. Apoia-se ainda no NIST Cybersecurity Framework (NIST CSF), seu *framework* de gestão de cibersegurança [88], e no padrão SP 800-53, um conjunto detalhado de controles de segurança cibernética recomendados [89].

A partir desses padrões, propõe uma estrutura organizacional completa para a gestão de riscos cibernéticos da cadeia de suprimentos, adotando o mesmo conceito de três níveis utilizado na definição do processo de gestão de riscos de segurança da informação pelo padrão SP 800-39, a saber, Nivel 1 - Organizacional, Nível 2 - Missão e Processos de Negócios e Nível 3 - Operacional.

Adota as seguintes definições a respeito de cadeia de suprimentos e riscos de cibersegurança da cadeia de suprimentos:

- cadeia de suprimentos: Recursos e processos interrelacionados, entre diversos níveis da organização, cada um deles sendo responsável pela aquisição de produtos e serviços que se estendem ao longo do ciclo de vida do produto ou serviço
- riscos de cibersegurança da cadeia de suprimentos: Potencial de dano ou comprometimento a partir de fornecedores, suas cadeias de suprimento, seus produtos e seus serviços

Traz uma esquematização dos riscos de cibersegurança ao longo da cadeia de suprimentos, que leva em consideração os possíveis impactos ao negócio, a probabilidade de ocorrência do risco, e a relação entre ameaças e vulnerabilidades, conforme ilustrado pela Figura 2.8.

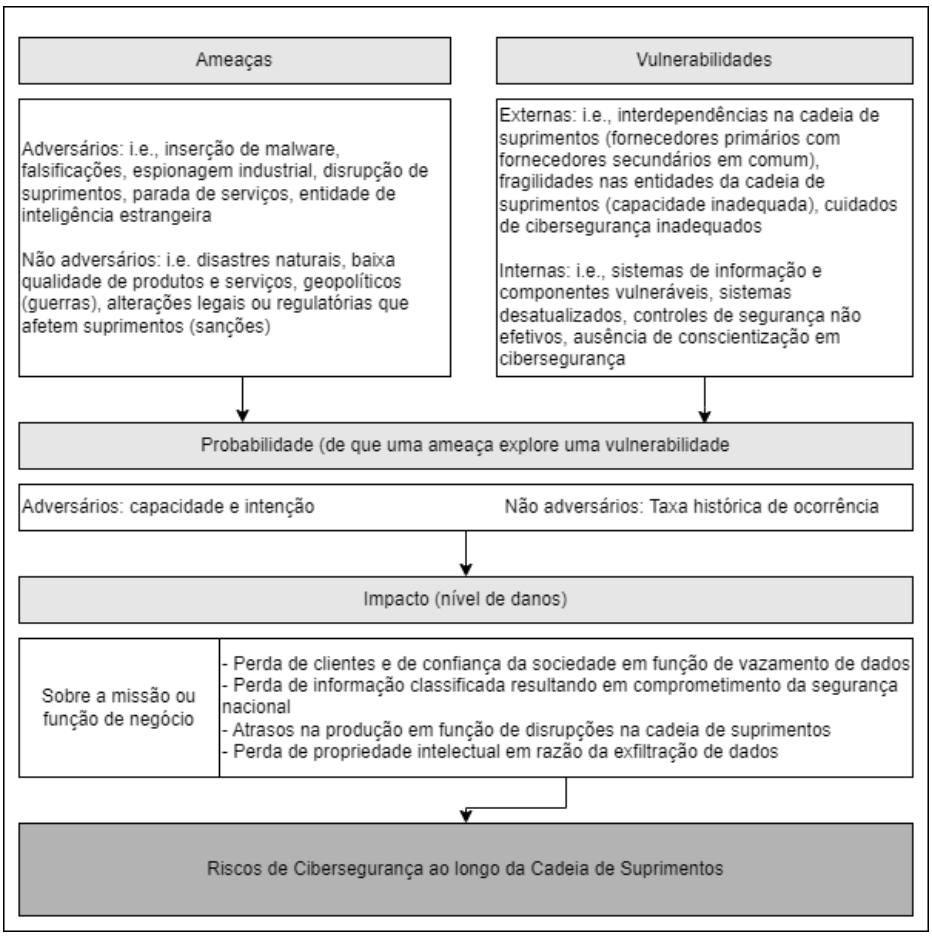


Figura 2.8: Riscos de Cibersegurança sobre a Cadeia de Suprimentos

Fonte: Adaptada de [85]

Esse conjunto de características dá origem a atividades que compõem o processo de avaliação de riscos cibernéticos da cadeia de suprimentos, detalhado a partir das quatro etapas do processo avaliação de riscos definido pelo padrão NIST SP 800-39, equivalentes ao enquadramento dos riscos (*frame*), avaliação dos riscos (*assess*), resposta aos riscos (*respond*) e monitoramento dos riscos (*monitor*).

O processo de avaliação de riscos cibernéticos pode ser visualizado na Figura 2.9.

Como parte da estratégia de gestão de riscos sobre a cadeia de suprimentos, destaca o tratamento de riscos nos processos de aquisição (onde se enquadram também os casos de aquisição de softwares de código aberto, não comerciais), e traz recomendações diretamente aplicáveis à obtenção de componentes de software de terceiros, que equivalem às dependências no cenário da cadeia de suprimento de software:

- Estabelecer um checklist de requisitos de segurança, de forma a garantir que as proteções necessárias integrem o componente a ser adquirido
- Obter software de código aberto a partir de bibliotecas pré-aprovadas
- Realizar aquisições a partir de listas de produtos ou de fornecedores pré-aprovados que tenham demonstrado conformidade com os requisitos de segurança da organização (que pode ser traduzida na

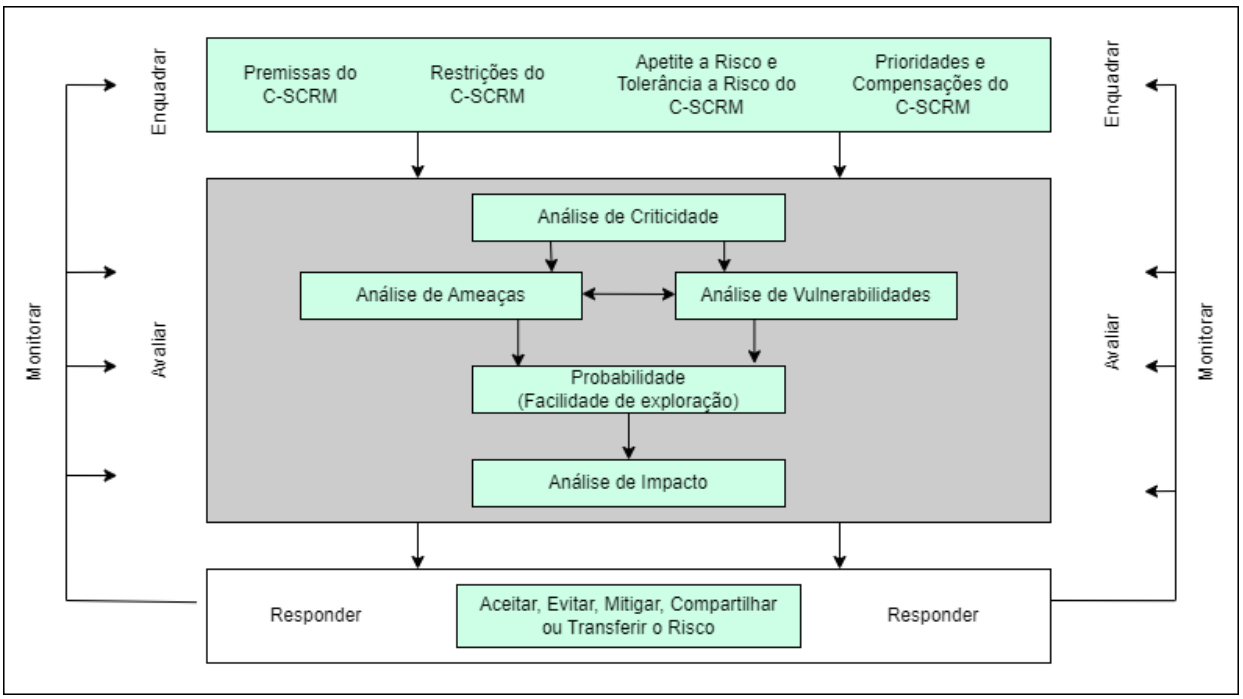


Figura 2.9: Gestão de Riscos Cibernéticos da Cadeia de Suprimentos

Fonte: Adaptada de [85]

recomendação de utilização de software de código aberto produzido por entidades ou desenvolvedores reconhecidos pela sua postura de segurança)

Destaca a importância do compartilhamento de informações sobre riscos entre as organizações que compõem a cadeia de suprimentos, bem como a importância que o uso de ferramentas, técnicas e procedimentos apropriados podem ter para prover proteção contra ataques à cadeia de suprimentos, dentre eles a inserção maliciosa de código malicioso ou de *backdoors* e práticas de desenvolvimento ruins ao longo do SDLC.

Traz um conjunto de práticas que devem ser executadas pela organização para o estabelecimento de um programa de Gerenciamento de Riscos Cibernéticos da Cadeia de Suprimentos (C-SCRM), dentre os quais destacam-se duas mais diretamente ligadas à cadeia de suprimento de software:

- Estabelecer processos internos para validar que fornecedores e provedores de serviço identifiquem e divulguem vulnerabilidades em seus produtos de forma ativa;
- Estabelecer uma capacidade de governança para gerenciar e monitorar componentes de software de forma a gerenciar riscos para a organização (ex: SBOMs acompanhadas de informações sobre criticidade, vulnerabilidade, ameaça e possibilidade de exploração, para possibilitar automação da análise de riscos).

Complementando o conjunto de práticas recomendadas, traz um conjunto de mais de duzentos controles de segurança adaptados para a os riscos cibernéticos à cadeia de suprimentos a partir do padrão NIST SP 800-53, dentre os quais diversos são aplicáveis ao cenário da cadeia de suprimento de software.

Por fim, oferece um *framework* para a determinação de exposição ao risco (*Risk Exposure Framework*), onde define o conceito de Cenário de Ameaça (*Threat Scenario*) como sendo um conjunto de eventos de ameaça com um potencial específico, ou associados a uma fonte de ameaças específica, parcialmente ordenadas ao longo do tempo. A identificação de cenários de ameaças pode facilitar a compreensão, por parte da organização, dos diversos riscos identificados e avaliados ao longo do processo de avaliação de riscos, organizando-os de forma lógica em um escopo mais bem definido, à semelhança de uma estória, que possa ser então melhor avaliada.

Traz alguns exemplos de cenários de ameaça, dentre os quais destaca-se um cenário relativo à Inserção Maliciosa de Código (um dos principais riscos à cadeia de suprimento de software), relativo a uma situação hipotética de inserção de código por parte de um integrador. Ainda que seja um exemplo bastante específico, oferece um panorama sobre como tratar outros cenários de ameaças à cadeia de suprimento de software, a partir de adaptações do cenário exemplificado.

2.5 GESTÃO DE RISCOS

Os normativos e *frameworks* citados nas Seções 2.3 e 2.4, com destaque para o padrão SP 800-161r1, mencionam por diversas vezes a questão da gestão de riscos das cadeias de suprimentos de software. Assim, trazemos aqui os principais conceitos e normativos sobre gestão de riscos em geral e gestão de riscos de segurança da informação.

2.5.1 ABNT NBR ISO 31.000 - Gestão de riscos — Diretrizes

A norma ABNT NBR ISO 31.000, em sua versão de 2018, é a principal das normas brasileiras voltadas ao tema de gestão de riscos, trazendo diretrizes para sua execução, fornecendo uma abordagem comum para gerenciar qualquer tipo de risco e aplicada a qualquer atividade [90].

Define risco como o efeito da incerteza sobre os objetivos e apresenta a gestão de riscos como uma disciplina que organiza atividades coordenadas para direcionar e controlar uma organização em relação aos riscos. Além disso, enfatiza a importância de comunicá-los de forma adequada às partes interessadas, com o objetivo principal de criar e proteger o valor da organização.[90].

Estabelece como principais diretrizes a importância do patrocínio explícito da alta gestão com relação à execução da gestão de riscos pela organização, sendo sua responsabilidade a elaboração de uma política de gestão de riscos, incluindo a atribuição de autoridades e responsabilizações, a atribuição de recursos suficientes para a execução das atividades necessárias, e a definição de parâmetros organizacionais para o balizamento da gestão de riscos, tais como a delimitação da tolerância e do apetite a riscos da organização.

A ISO 31.000 tem como uma de suas principais contribuições a definição de um processo de gestão de riscos, representado por meio da figura 2.10.

Resumidamente, o processo tem início com a definição de escopo, contexto e critérios, quando a organização definirá sobre o tipo de atividades em que o processo de gestão de riscos atuará, podendo ser sobre processos de negócio, sistemas de informação, unidades organizacionais, ou outros, e tem sequência com

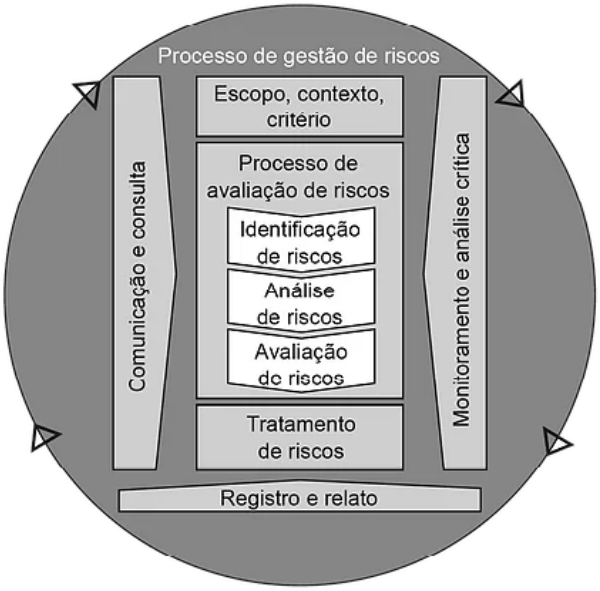


Figura 2.10: Processo de Gestão de Riscos

Fonte: [90]

o processo de avaliação de riscos em si.

O processo de avaliação de riscos, por sua vez, é composto pelas atividades de identificação de riscos, onde esses são levantados e registrados, pela análise de riscos, quando são definidas suas características tais como probabilidade de ocorrência e impacto sobre o escopo, e pela avaliação dos riscos, etapa onde basicamente é definido se são aceitáveis ou não para a organização.

Por fim, há a etapa de tratamento de riscos, quando, em termos gerais, a organização decidirá se irá evitá-los, mitigá-los, aceitá-los ou transferi-los. Caso a opção seja por mitigar os riscos, devem ser avaliadas ainda quais são as opções disponíveis, o nível de risco residual após a eventual implantação, e o custo de cada opção, de forma a subsidiar a seleção da melhor opção.

Todo esse processo deve ainda ser registrado e relatado às partes interessadas, e deve ser objeto de monitoramento, estabelecendo assim um ciclo de melhoria contínua do processo.

2.5.2 ABNT NBR ISO 27005 - Segurança da informação, segurança cibernética e proteção à privacidade - Orientações para gestão de riscos de segurança da informação

A norma ISO 27.005, atualmente em sua quarta edição, datada de 2023, é a especialização da norma ISO 31.000 para o processo de gestão de riscos de segurança da informação. Tem como objetivo trazer recomendações às organizações para que cumpram os requisitos da norma ISO 27.001 quanto a riscos de segurança da informação, em especial com relação às atividades de avaliação e tratamento desses riscos, e, assim como a ISO 31.000, é aplicável a todos os tipos de organização [68].

Os riscos de segurança da informação são os que afetam os objetivos relativos aos seus quatro pilares,

que são a Disponibilidade, Integridade, Confidencialidade e Autenticidade dos dados [91].

Sendo a cadeia de suprimentos de software a forma moderna de organização dos trabalhos de desenvolvimento de software, pode-se concluir que os riscos que a afetam são os que atingem os objetivos de Disponibilidade, Integridade, Confidencialidade e Autenticidade dos dados e informações tratados por artefatos de software.

Quanto ao processo de avaliação dos riscos, a norma informa que avaliações de risco podem ser realizadas sobre diversos tipos de processos distintos, incluindo a gestão de projetos (tais como projetos de desenvolvimento de software) gestão de vulnerabilidades, gestão de incidentes e gestão de problemas.

Sugere critérios para o processo, dentre os quais destacam-se o nível de classificação das informações a serem tratadas por um determinado software, a quantidade e concentração dessas informações, o valor estratégico dos processos de negócio associados, a criticidade dessas informações e dos ativos envolvidos em seu tratamento, e a importância operacional e empresarial da disponibilidade, confidencialidade e integridade dessas informações.

Como detalhamentos adicionais à norma ISO 31.000, a ISO 27.005 traz ainda sugestões de escalas qualitativas e quantitativas de consequência (que levam em conta a preservação ou perda de confidencialidade, integridade e disponibilidade de informações) e probabilidade de ocorrência de um risco. A combinação das definições de consequência e probabilidade permite estabelecer o nível do risco.

Com relação à probabilidade, a ISO 27.005 nos alerta que devem ser consideradas, dentre outros fatores, a motivação e as capacidades de uma determinada fonte de risco (por exemplo, as habilidades técnicas de eventuais atacantes), bem como fatores que possam influenciar erros humanos e mau funcionamento do produto. Faz-se importante observar aqui que essas questões são diretamente relacionadas aos dois principais tipos de ataque à cadeia de suprimentos mencionadas na introdução desta dissertação, a saber, a inserção de código malicioso em um determinado produto de software, e a exploração de vulnerabilidades em produtos já existentes.

Com relação à identificação dos riscos, primeira atividade do processo de análise de riscos, esclarece que pode ser realizada com base em duas abordagens, a primeira com base em eventos, que identifica cenários estratégicos, suas fontes de risco e seus impactos, e a segunda com base em ativos, onde são identificados cenários operacionais, em termos de ativos, ameaças e vulnerabilidades.

Segundo a ISO 27.005, os eventos e suas consequências podem ser identificados a partir das preocupações da alta direção, dos proprietários de riscos e dos requisitos com relação ao contexto da organização, enquanto na abordagem com base em ativos os riscos podem ser identificados a partir da enumeração desses ativos, de suas vulnerabilidades, e das ameaças que podem explorá-las. Caso todos os ativos envolvidos possam ser mapeados, bem como suas vulnerabilidades e ameaças associadas, em tese todos os riscos seriam identificados.

Quanto à atividade de tratamento dos riscos identificados, a ISO 27.005 nos traz um detalhamento sobre os tipos de controles que podem ser adotados para a mitigação dos riscos identificados, classificando-os em controles preventivos (que visam evitar a ocorrência de um evento), detectivos (que visam detectar a ocorrência de um evento), e corretivos (que visam limitar as consequências de um evento).

A norma ISO 27.005, portanto, oferece importantes contribuições para a execução do processo de

gestão de riscos voltado especificamente para o tratamento de riscos de segurança da informação.

2.5.3 ABNT NBR ISO 31.010 - Gestão de Riscos - Técnicas para o processo de avaliação de riscos

A norma ISO 31.010:2021 documenta uma série de técnicas existentes para aplicação nas diversas etapas do processo de avaliação de riscos, fornecendo orientação para a seleção das técnicas mais apropriadas para diferentes cenários de gestão de riscos [92].

As técnicas descritas pela norma abordam aspectos relacionados à implementação do processo de avaliação de riscos, fornecendo subsídios para tarefas como a decisão sobre a aceitação de riscos, decisão entre diferentes opções de tratamento de riscos, identificação de riscos, determinação de fontes, causas e fatores de risco, identificação da eficácia de controles existentes, entendimento das consequências e probabilidades, análises das interações e dependências entre riscos, e monitoramento e análise crítica.

Traz breves descrições sobre cada uma das técnicas, seus cenários de aplicação, pontos fortes e fracos, além de quadros comparativos que indicam as etapas do processo de avaliação de riscos aos quais as técnicas apresentadas melhor se enquadram, o nível de conhecimento requerido para empregá-las, e outras informações.

Oferece, portanto, insumos para a seleção das técnicas mais apropriadas para utilização por parte da organização nas diversas etapas do ciclo de análise de riscos.

2.5.4 NIST - Managing Information Security Risk Organization, Mission, and Information System View

O padrão, identificado como SP 800-39, tem como objetivo oferecer orientações a respeito da gestão de riscos de segurança da informação sobre sistemas de informação utilizados em nível federal norte-americano, com exceção de sistemas classificados como sendo de segurança nacional.

Trata-se de um padrão de nível mais amplo, que trata de aspectos gerais a respeito da estruturação da gestão de riscos na organização, sendo complementado por outros padrões no tocante ao detalhamento das atividades de avaliação, resposta e monitoramento de riscos [86].

Seu próprio texto cita as normas ISO 31.000, ISO 31.010, ISO 27.001 e ISO 27.005, e informa que seus conceitos são similares aos dessas normas, de maneira a reduzir o impacto sobre organizações que tenham que atender tanto aos padrões do NIST quanto às normas da ISO.

Estabelece um processo de gestão de riscos estruturado em quatro etapas, a saber: *frame risk* (enquadramento dos riscos), *assess risk* (avaliação dos riscos), *respond to risk* (resposta aos riscos) e *monitor risk* (monitoramento dos riscos).

Na etapa de enquadramento dos riscos é definida a estratégia a ser utilizada para o gerenciamento de riscos, definindo-se como serão realizadas as atividades de avaliação, resposta e monitoramento dos riscos, tornando transparentes a percepção de riscos utilizada pela organização para decisões tanto relativas a investimento quanto operacionais.

Na etapa de avaliação de riscos são avaliadas as ameaças, vulnerabilidades, impactos e possíveis consequências, resultando na determinação do risco (probabilidade de que um dano ocorra, e o grau de impacto deste dano).

Na terceira etapa são definidas respostas consistentes, em nível organizacional, aos riscos identificados, incluindo a definição de alternativas de resposta, a avaliação dessas alternativas, determinação das alternativas que atendam à tolerância da organização a riscos, e a implementação das alternativas escolhidas.

Já na etapa de monitoramento a organização deve verificar se as alternativas selecionadas encontram-se efetivamente implementadas e se atendem aos requisitos de segurança definidos, a efetividade das medidas implementadas, e a eventual ocorrência de mudanças nos cenários interno e externo que venham a impactar os riscos sobre o ambiente em que os sistemas de informação operam.

Adicionalmente, o padrão de gestão de riscos de segurança da informação definido pelo NIST adota uma abordagem multinível, onde as atividades necessárias são executadas nos níveis organizacional, de missão e processos de negócio, e de sistemas de informação, conforme ilustra a Figura 2.11.

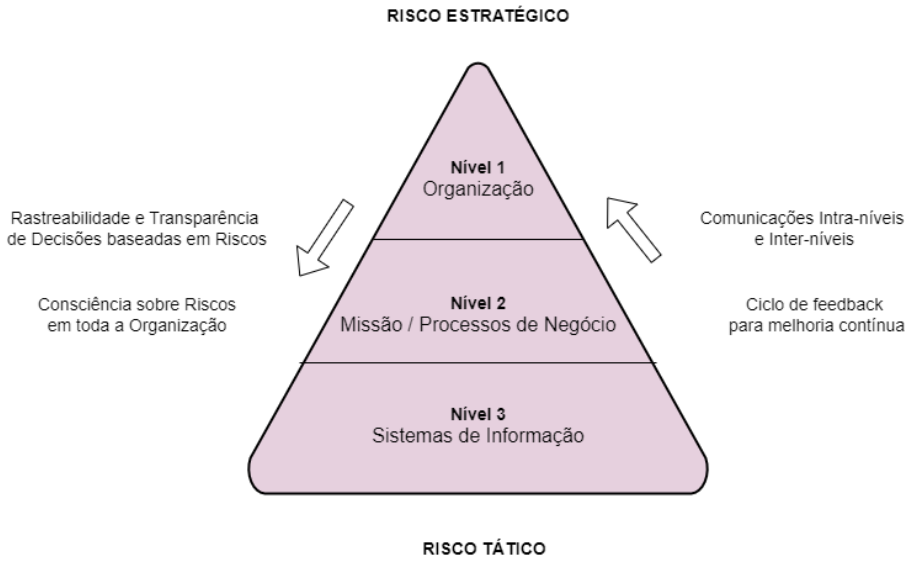


Figura 2.11: NIST - Gestão de Riscos de Segurança da Informação Multinível

Fonte: Adaptada de [86]

Cada um dos níveis tem atribuições específicas no processo de gestão de riscos.

O Nível 1 - Organização (*Tier 1*), define a gestão de riscos a partir de uma perspectiva organizacional, e é responsável pela execução das tarefas relativas ao enquadramento do risco.

O Nível 2 (*Tier 2*) executa o tratamento de riscos em nível de missão da organização e de processos de trabalho, obedecendo as definições sobre contexto do tratamento de riscos, e demais decisões a respeito da forma de tratamento de riscos definidas pelo Nível 1, e dentre suas atribuições destacam-se a definição dos processos de negócios necessários para suportar a missão e os as diversas funções da organização, a priorização desses processos, e a incorporação de requisitos de segurança da informação nos processos de negócio.

Por fim o Nível 3 (*Tier 3*) é encarregado do tratamento de riscos sob a perspectiva dos sistemas de

informação, e tem como atividades principais a categorização dos sistemas de informação, a alocação de controles de segurança sobre esses sistemas e sobre o ambiente em que operam, e a gestão da seleção, implementação, avaliação, autorização e monitoramento desses controles.

2.5.5 NIST Risk Management Framework

O NIST Risk Management Framework, ou NIST RMF, identificado como SP 800-37, define um processo de gestão de riscos de informação e de privacidade em relação a sistemas de informação, aplicável ao longo de todo o ciclo de vida do desenvolvimento de software (SDLC - *Software Development Life Cycle*), sendo assim o detalhamento das funções de gestão de riscos associadas ao Nível 3 da abordagem multiníveis definida pelo padrão NIST SP 800-39 [87].

Destaca o incremento e complexidade de sistemas interconectados tanto no setor público quanto no setor privado, incluindo a infraestrutura crítica norte-americana, e alerta para o fato da exploração da cadeia de suprimentos como um vetor de ataque para a invasão de sistemas, com o objetivo de comprometer a integridade de elementos do sistema e assim obter acesso a ativos críticos, e adota como um de seus objetivos a integração de conceitos relacionados à gestão de riscos da cadeia de suprimentos (*Supply Chain Risk Management* - SCRM) às atividades de gestão de riscos de segurança da informação.

O NIST RMF é organizado em sete etapas, conforme ilustrado na Figura 2.12.

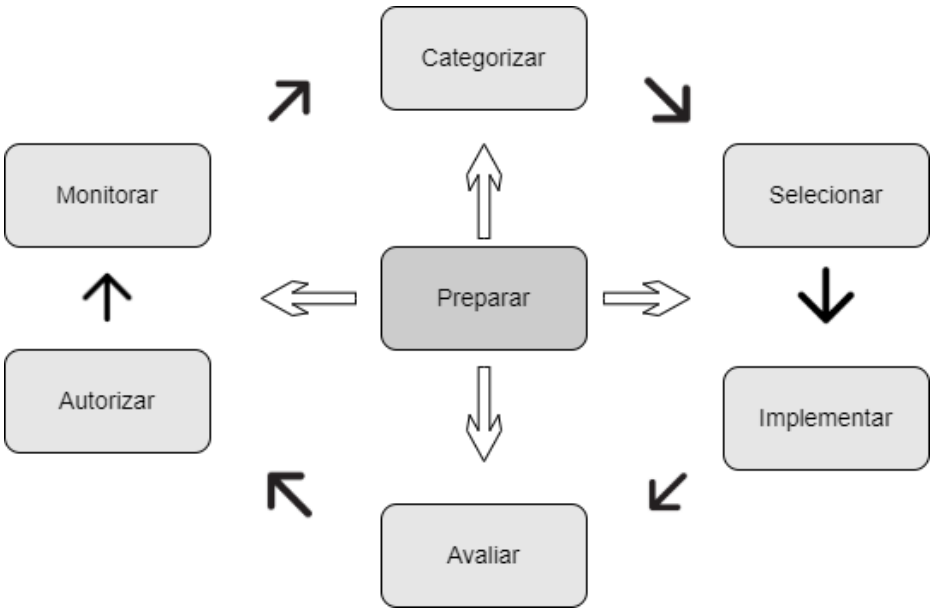


Figura 2.12: Etapas do NIST RMF

Fonte: Adaptada de [87]

A etapa inicial, Preparação (*Prepare*) é equivalente às ações executadas pelos Níveis 1 e 2 definidos no padrão SP 800-39, onde são estabelecidos os parâmetros gerais que conduzirão as demais etapas. Seu posicionamento como ponto central do processo indica que as definições da etapa de preparação influenciam todas as demais etapas.

A próxima etapa é Categorizar (*Categorize*) o sistema e as informações por ele tratadas, sob o ponto de

vista do impacto de potenciais incidentes.

A etapa de Seleção (*Select*) é onde os controles de segurança iniciais serão selecionados, e, se necessário, adaptados ao sistema em questão.

Durante a etapa de Implementação (*Implement*) os controles são efetivamente implementados e sua utilização correta é documentada.

A seguir, a etapa de Avaliação (*Assess*) abriga as atividades de verificação da corretude da implementação dos controles, quando são checados seu correto funcionamento e atendimento aos requisitos de segurança.

Durante a etapa de Autorização (*Authorize*), são aprovados os controles de segurança aplicáveis ao sistema, ou os controles de utilização comum por diversos sistemas, com base na verificação de que o nível de risco residual se encontra dentro dos limites definidos pela organização.

Já na etapa de Monitoramento (*Monitor*) a efetividade dos controles é avaliada de forma contínua, alterações de cenário são documentadas e novos riscos são avaliados, e a postura de segurança e privacidade do sistema é comunicada às partes interessadas.

O NIST RMF traz, para cada uma das etapas, a definição de seus objetivos, resultados esperados, e um conjunto de atividades necessárias à obtenção desses resultados, oferecendo assim informações que facilitam sua implementação.

Por fim, faz-se importante destacar que o NIST RMF pode ser utilizado para auxiliar a implementação do *NIST Cybersecurity Framework* (NIST CSF) a partir de uma abordagem centrada em riscos.

2.5.6 NIST SP 800-30 - Guide for conducting risk assessments

O NIST publicou também o padrão SP 800-30, que oferece orientações sobre a condução da atividade de avaliação de riscos [93].

O documento complementa os padrões SP 800-39 e 800-37, e, diferentemente da norma ISO 31.010, que complementa as normas ISO 31.000 e ISO 27.005, não traz um conjunto de técnicas aplicáveis a cada fase da avaliação de riscos, mas tem como foco oferecer um processo passo a passo aplicável à preparação para avaliações de risco, efetiva condução da avaliação de riscos, comunicação dos resultados às partes interessadas, e manutenção das avaliações de riscos ao longo do tempo.

Nesse contexto, traz o conceito de modelos de riscos, que definem termos essenciais, fatores de risco e os relacionamentos entre esses fatores. A utilização de um modelo de riscos traz como benefícios a possibilidade de reprodução e de repetição das avaliações de risco.

Como fatores de risco típicos o padrão SP 800-30 relaciona ameças, vulnerabilidades, impacto, probabilidade e condições de predisposição ao risco. A Figura 2.13 ilustra um modelo de riscos genérico.

Define ainda um processo de avaliação de riscos, composto pelas atividades de preparação para a avaliação de riscos, condução efetiva da avaliação, comunicação dos resultados e manutenção da avaliação. A segunda etapa, referente à condução da avaliação, é subdividida nas tarefas de identificação de fontes e eventos de ameaças, identificação de vulnerabilidades e condições de predisposição, determinação da

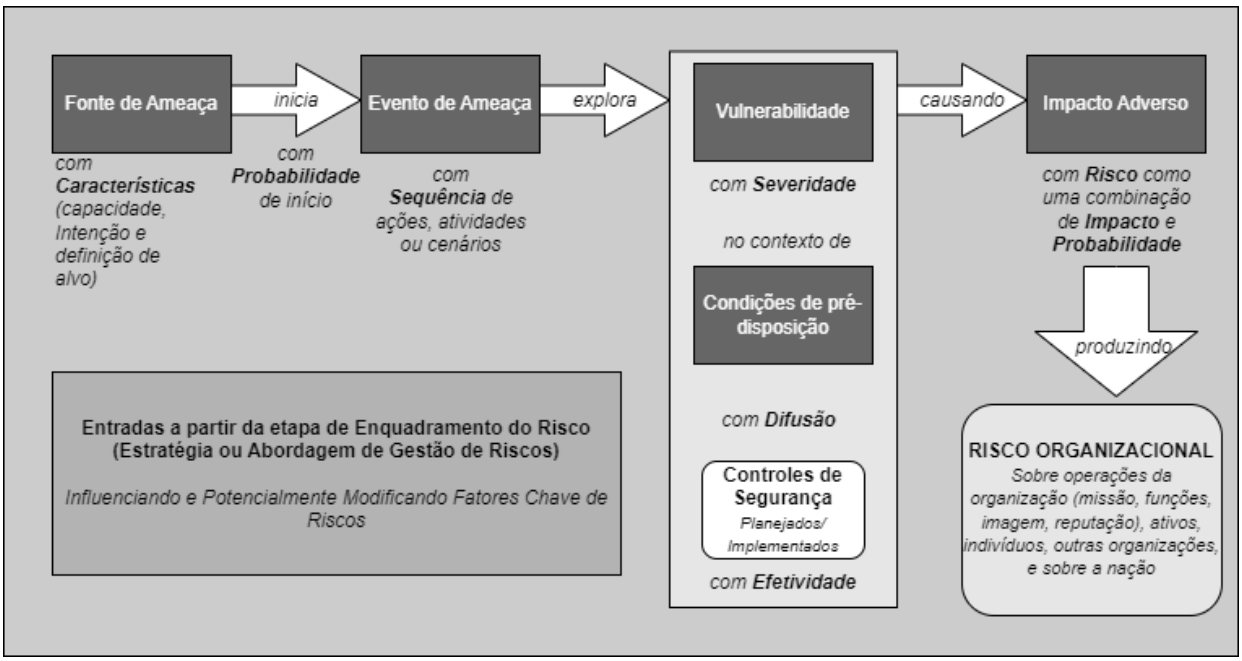


Figura 2.13: NIST - Modelo de Riscos Genérico

Fonte: Adaptada de [93]

probabilidade de ocorrência, determinação da magnitude do impacto, e determinação do risco. O processo é ilustrado pela Figura 2.14.

Destacam-se ainda ainda diversos apêndices que trazem exemplos a respeito de fatores de risco e outros componentes do modelo de riscos, dentre os quais os relacionados a fontes de ameças e eventos de ameaças.

Como fontes de ameaças trazidas pelo documento que, em nossa avaliação, podem ser ligadas a ataques à cadeia de suprimentos de software, podemos citar adversários individuais (tanto internos quanto externos), grupos organizados ou aleatórios que se reúnem para a execução de ataques (tais como os grupos documentados pelo *framework* Mitre Att&ck), organizações tais como competidores, fornecedores, parceiros ou mesmo consumidores, e Estados Nação.

Com relação a ameaças o padrão SP 800-30 cita algumas claramente ligadas à segurança da cadeia de suprimentos de software, tais como a inserção de *malware* de forma direcionada ou não em produtos de software comerciais ou de código aberto, a exploração de vulnerabilidades em sistemas de informação (que usualmente contém pacotes de terceiros), inclusive de forma sincronizada com eventos importantes para a organização alvo e o comprometimento do projeto, produção e distribuição de componentes de sistemas de informação em fornecedores selecionados.

Tratando-se de cadeias de suprimentos de software, o processo de gestão de riscos pode ser auxiliado em grande medida a partir dos *frameworks* que recomendam boas práticas de segurança, que podem atuar como insumos para algumas das fases de gestão de riscos, com destaque para a identificação de fontes de ameaças e identificação dos riscos, estimativa de consequências e probabilidades, e seleção de controles aplicáveis ao nível de segurança desejado pela organização.

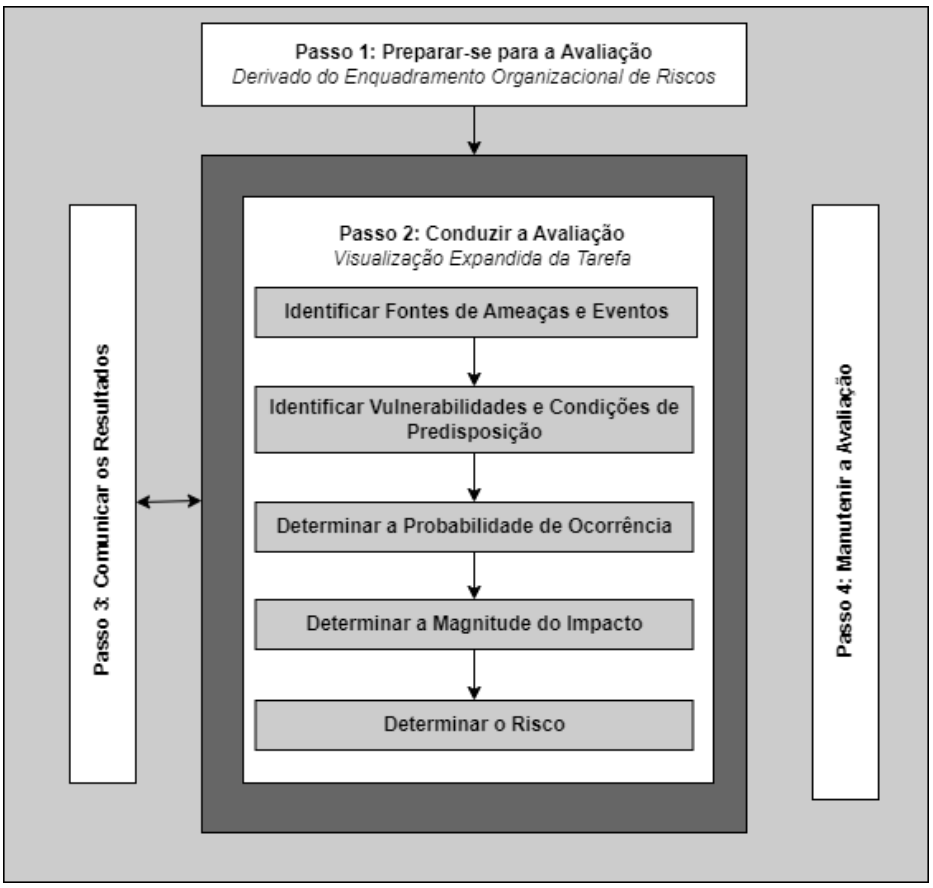


Figura 2.14: NIST - Processo de avaliação de riscos

Fonte: Adaptada de [93]

2.6 PRINCIPAIS FRAMEWORKS RELACIONADOS À SEGURANÇA DA CADEIA DE SUPRIMENTO DE SOFTWARE

Por se tratar de um dos riscos cibernéticos mais críticos atualmente, a segurança de cadeias de suprimentos de software tem recebido atenção significativa de diversas organizações, que vêm desenvolvendo *frameworks* e guias de boas práticas.

Detalharemos a seguir, em ordem cronológica de suas publicações, esses *frameworks* e guias de boas práticas.

2.6.1 OWASP Secure Component Verification Standard (SCVS)

O SCVS é um esforço de comunidade, liderado pela organização OWASP e publicado em 2020, que tem como objetivo definir um *framework* para identificar atividades, controles e melhores práticas que contribuam com o incremento da segurança da cadeia de suprimentos de software, sendo inspirado em uma iniciativa anterior chamada OWASP Application Security Verification Standard [94].

Os controles de segurança que recomenda são projetados para permitir automação e estruturados de forma que possam ser implementados gradualmente, conforme as necessidades de cada organização. Esses

controles são organizados em três níveis crescentes de verificação, detalhados a seguir.

- Nível 1: constitui-se na base a partir do qual o *framework* é elaborado, e recomenda práticas como a criação de SBOMs e a realização de inventários completos de software, o uso de integração contínua com vistas à produção de compilações repetitíveis, e a análise de componentes de terceiros
- Nível 2: expande as capacidades previstas no nível 1, sendo aplicável a organizações que tenham atividades intensivas de desenvolvimento de software, e que estejam expostas a requisitos regulatórios ou contratuais
- Nível 3: adiciona novos requisitos em relação aos dois níveis anteriores, sendo aplicável a cenários de infraestrutura crítica, exigindo auditabilidade e transparência fim-a-fim na cadeia de suprimentos de software

Suas recomendações são distribuídas em seis famílias distintas, com indicações sobre aplicabilidade aos Níveis 1, 2 e 3 do modelo. Embora abordem três das quatro etapas da cadeia de suprimentos de softwares, não contam com descrições detalhadas sobre os controles de segurança recomendados, diferentemente de outros *frameworks*, como o CIS e o CNCF, detalhados mais adiante, constituindo-se apenas de *checklists* de controles a serem implementados.

- Inventário
- Lista de componentes de software (Software bill of materials – SBOM)
- Ambiente de compilação
- Gerenciamento de Pacotes
- Análise de Componentes
- Origem e linhagem

Sua estrutura permite que seja utilizado tanto para a avaliação da capacidade interna de uma organização no tocante à segurança de seu processo de desenvolvimento com relação à cadeia de suprimentos de software, como também para a avaliação de fornecedores, com base em sua documentação e metadados dos projetos, permitindo que clientes avaliem a garantia de segurança de projetos fornecidos por terceiros, quer sejam comerciais, quer sejam de código aberto.

Possibilita ser utilizado como *framework* para a certificação de organizações quanto à segurança da cadeia de suprimentos de softwares, embora a própria OWASP não atue como acreditadora de entidades certificadoras.

2.6.2 Cloud Native Computing Foundation (CNCF) - Software Supply Chain Best Practices

A *Cloud Native Computing Foundation* (CNCF), entendendo que havia pouco material disponível relativo à mitigação ou arquitetura referente a riscos sobre a cadeia de suprimentos de software, elaborou,

em 2021, um guia intitulado *Software Supply Chain Best Practices*, que traz recomendações relativas a práticas, ferramentas e considerações sobre projeto com vistas à redução de impactos derivados de potenciais ataques. Entretanto, informa que não traz orientações sobre a análise de riscos relativas à cadeia de suprimentos de softwares, e que está fora de seu escopo o tratamento de vulnerabilidades em hardware ou sistemas operacionais [29].

Identifica o conjunto de etapas necessárias à segurança da cadeia de suprimentos de software como *Software Factory*, e estrutura suas recomendações em torno de temas principais:

- Verificação: verificações devem ser realizadas em cada etapa da cadeia de suprimentos de software, e essas verificações devem ser atestadas, preferencialmente por procedimentos baseados em primitivas criptográficas
- Automação: auxilia na garantia de que os processos sejam determinísticos, viabilizando os mecanismos de atestação e posterior verificação
- Autorização em Ambientes Controlados: as permissões devem ser concedidas com base no conceito do menor privilégio, de forma a reduzir o impacto de um possível comprometimento
- Autenticação Segura: métodos de autenticação forte devem ser utilizados para todas as interações na cadeia de suprimentos, e as atividades devem ser monitoradas

Define cinco estágios para a segurança da cadeia de suprimentos de software, classificando suas recomendações como sendo aplicáveis a cenários em que a garantia de segurança seja classificada como de nível médio ou alto, e o risco associado ao ambiente também seja classificado como médio ou alto. Ambientes que demandem baixo grau de garantia de segurança e baixo grau de risco não são endereçadas pelo *framework*. Os cinco estágios são:

- Segurança do código-fonte
- Segurança dos componentes
- Segurança dos processos de compilação *(build pipelines)*
- Segurança dos artefatos
- Segurança das implantações

Observa-se, portanto, que traz recomendações referentes a todos os pilares da cadeia de suprimentos de software, e aborda com ênfase a questão da automação dos controles que recomenda.

2.6.3 Center for Internet Security (CIS) Software supply chain security guide

Em 2022, o *Center for Internet Security* (CIS) lançou seu Guia para Proteção da Cadeia de Suprimentos de Software, desenvolvido por uma comunidade global de especialistas em diversas áreas, incluindo

consultoria, desenvolvimento de software, auditoria e conformidade, pesquisa em segurança, operações, governo e questões legais [95].

Não é direcionado a produtos específicos que componham a cadeia de suprimentos, porém traz recomendações de segurança aplicáveis às diversas etapas da cadeia de suprimentos de software, independentemente dos produtos específicos que sejam utilizados nas suas diversas etapas.

Assim, propõe controles relacionados ao código-fonte, dependências, esteira de desenvolvimento, implantação e aos artefatos, e tem como visão geral impulsionar padrões emergentes no mercado, tais como o SLSA. Sua representação da cadeia de suprimentos de software e dos pontos de atenção relativos à aplicação de controles de segurança é ilustrada pela Figura 2.15, que, conforme podemos observar, é bastante semelhante às representações utilizadas por outros *frameworks*.

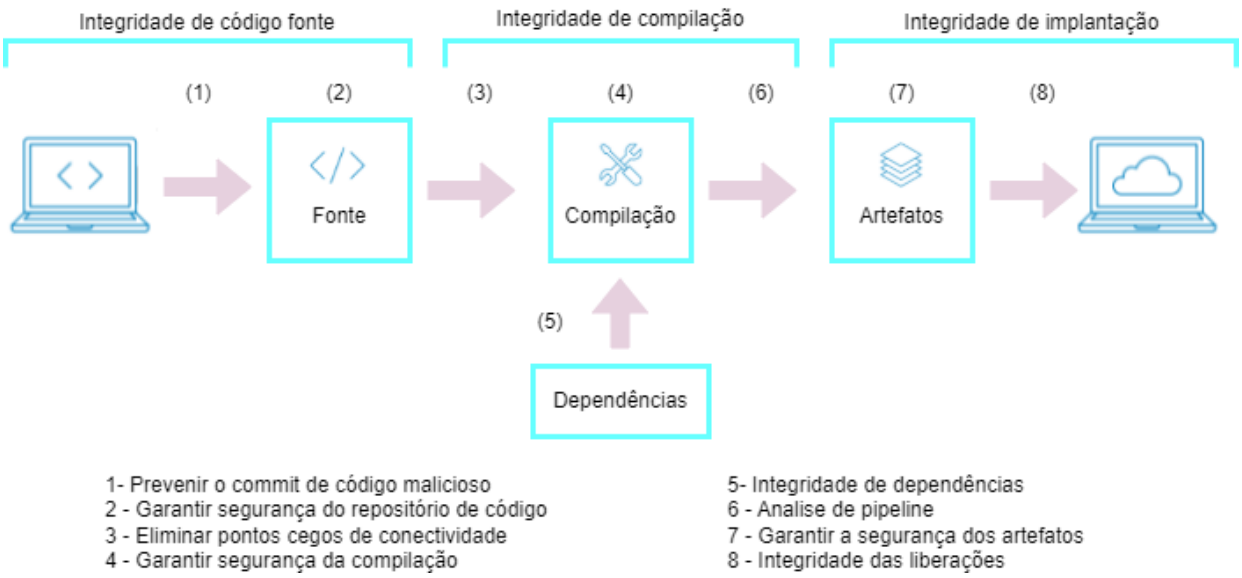


Figura 2.15: Cadeia de Suprimentos de Software

Fonte: Adaptada de [95]

O Guia traz 118 recomendações de segurança distintas, distribuídas entre as principais etapas da cadeia de suprimentos de software:

- Proteção do código, onde são elencadas medidas como gestão de mudanças, gerenciamento de repositórios, controle de acesso dos *commits* e controle de fornecedores
- Esteira de desenvolvimento, onde destacam-se a proteção do ambiente de compilação, dos agentes envolvidos, das etapas e a integridade do pipeline
- Dependências de terceiros, etapa em que aconselha que sejam conduzidas verificações nos pacotes de terceiros
- Ambiente de implantação, para o qual são sugeridas configurações seguras
- Medidas de rastreabilidade de origem e verificação dos acessos aos artefatos entregues

É também um *framework* que oferece recomendações de segurança para todas as etapas da cadeia de suprimentos de software, embora não defina métodos de automação da aplicação dessas recomendações.

2.6.4 Enduring Security Framework (ESF) – Securing the Software Supply Chain

A *Enduring Security Framework* é uma iniciativa de diversas agências do governo norte-americano, organizada pela *National Security Agency* (NSA), que atua em parceria com o mercado privado. É dedicada a tratar riscos à Infraestrutura crítica. Produz documentos contendo recomendações sobre diferentes aspectos da Infraestrutura crítica, passando por redes 5G, redes de rádio abertas, e também sobre a segurança da cadeia de suprimentos de software. Nesse contexto, produziu, em 2022, guias voltados a diferentes partes interessadas na cadeia de suprimentos de software:

- *Securing the Software Supply Chain for Customers* [96]
- *Securing the Software Supply Chain for Developers* [12]
- *Securing the Software Supply Chain for Suppliers* [97]

Esses documentos trazem os cenários de ameaças mais comuns ao ciclo de desenvolvimento de software, bem como ações de mitigação recomendadas para cada etapa das atividades ligadas à cadeia de suprimentos de software, sendo segregados para os diferentes público-alvo. No caso de clientes, as etapas foram definidas em avaliação, aquisição, implantação e manutenção do produto adquirido. Para desenvolvedores, as etapas são as definidas pela organização em seu ciclo de desenvolvimento de software (SDLC). Já para fornecedores, as etapas são idênticas às definidas no NIST SSDF: Preparar a organização, Proteger o Software, Produzir Software Seguro e Responder às Vulnerabilidades. Para cada etapa, define uma série de boas práticas a partir dos cenários de ameaça definidos, em sua maioria derivadas dos *frameworks* NIST SSDF e SLSA.

2.6.5 OWASP Top 10 CI/CD Security Risks

O documento OWASP Top 10 CI/CD Security Risks consolida os esforços de diversas organizações que utilizam fortemente ambientes de CI/CD para automatizarem o fluxo de suas cadeias de suprimentos de software de forma a garantir a segurança adequada ao mesmo tempo em que permitem agilidade às equipes de desenvolvimento [98].

Foi elaborado por especialistas da empresa Cider Security, especializada em segurança de aplicações e em segurança de cadeias de suprimentos de software (eventualmente adquirida pela Palo Alto Networks), com a colaboração de especialistas de empresas como a própria Palo Alto, Mozilla, Rapid7, Atlassian e Netflix, dentre outras.

Utiliza como insumos para a definição dos 10 principais riscos que afetam o ambiente de CI/CD a análise da arquitetura, projeto e postura de segurança de centenas de ambientes reais de CI/CD, discussão com especialistas, e publicações contendo detalhamentos de incidentes que atingiram esses tipos de ambiente.

Os 10 riscos principais apontados pelo documento são os seguintes:

1. Mecanismos de controle de fluxo insuficientes (Insufficient Flow Control Mechanisms): possibilita que atacantes insiram código malicioso nos artefatos devido à falta de revisão e aprovação nas várias etapas da esteira de CI/CD
2. Gerenciamento Inadequado de Identidades e de Acessos (Inadequate Identity and Access Management): a gestão inadequada de identidades tanto de colaboradores quanto de processos automatizados aumenta o potencial de danos derivados do compromtimento dessas identidades
3. Abuso da cadeia de dependências (Dpendency Chain Abuse): resulta na utilização indevida de pacotes de terceiros maliciosos
4. Execução de Pipeline Envenenada (Poisoned Pipeline Execution: atacantes com acesso ao ambiente de códigos-fonte, porém sem acesso à pipeline, inserem códigos maliciosos que posteriormente serão processados pela esteira de CI/CD
5. Controles de Acesso à Pipiline Insuficientes (Insufficient PBAC (Pipeline-Based Access Controls: atacantes exploram os controles de acesso à pipeline deficientes para executarem movimentação lateral dentro do ambiente de CI/CD, ou mesmo para outros ambientes da organização
6. Higiene de Credenciais Insuficiente (Insufficient Credential Hygiene: refere-se à possibilidade de que um atacante obtenha credenciais, tokens, etc., ao longo da pipeline em razão de falhas no controle dessas credenciais
7. Configuração Insegura do Sistema (Insecure System Configuration): é relacionado a falhas nas configurações de segurança dos vários sistemas que compõem o ambiente de CI/CD (repositório de códigos-fonte, orquestrador, repositório de artefatos)
8. Utilização não Gerenciada de Serviços de Terceiros (Ungoverned Usage of 3rd Party Services): são relacionados à facilidade com que são atribuidos direitos de acesso a serviços de terceiros sobre o ambiente de CI/CD, expandindo a superfício de ataque
9. Validação Imprópria da Integridade de Artefatos (Improper Artifact Integrity Validation): possibilita que um atacante insira na pipeline algum artefato malicioso, embora aparente ser um artefato legítimo
10. Log e Visibilidade Insuficientes (Insufficient Logging and Visibility): possibilita que um atacante execute ações maliciosas sem que seja detectado, e dificulta inclusive a identificação, em uma análise posterior de um eventual incidente, de táticas, técnicas e procedimentos por ele utilizados

Para cada um dos riscos o documento traz uma definição de sua natureza, sua descrição detalhada, o possível impacto, recomendações para a mitigação do risco, e exemplos reais em que foi explorado, de forma a permitir ao leitor uma boa compreensão a respeito do risco.

2.6.6 Secure Supply Chain Consumption Framework - S2C2F

O S2C2F é um *framework* diferente dos demais, na medida em que tem foco específico, referente à garantia de segurança e redução de riscos para o consumo de dependências de Software de Código Aberto,

incluindo código fonte, pacotes de idiomas, módulos, componentes, conteiners, bibliotecas ou binários [78].

Possui três conceitos principais e três objetivos específicos voltados a incrementar a segurança no consumo de de componentes de software de código aberto, conforme ilustra a Figura 2.16.

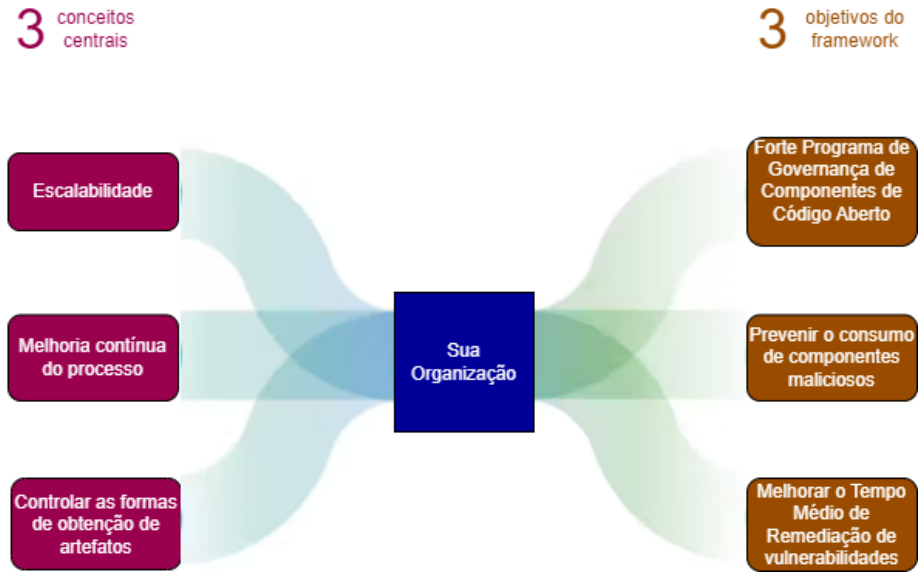


Figura 2.16: S2C2F - Conceitos e Objetivos

Fonte: Adaptada de [78]

Seus três conceitos centrais são melhor detalhados a seguir:

- *Controlar as formas de obtenção de artefatos*. O S2C2F argumenta que é necessária a padronização do processo de consumo de software de código aberto por todas as equipes de desenvolvimento da organização, de forma que todos sigam fluxos de consumo gerenciados, em vez de métodos aleatórios de consumo pelos desenvolvedores, como a clonagem de repositórios disponibilizados na plataforma Git, *download* de pacotes via utilitário "wget", e outros.
- *Melhoria contínua do processo*. Reconhecendo a impossibilidade de implementação simultânea de todas as suas recomendações, o S2C2F foi organizado de forma a auxiliar as organizações na priorização dos requerimentos que devam ser implementados, oferecendo inclusive um modelo de maturidade a partir do qual a organização pode identificar em que estágio se encontra, definir seu estágio alvo, e assim identificar as tarefas necessárias para atingi-lo.
- *Escalabilidade*. As ferramentas desenvolvidas pelo S2C2F têm como objetivo trazer segurança ao consumo de software de código aberto em larga escala, sem requerer uma estrutura de registro central, ou um comitê central de governança

Por meio da aplicação desses conceitos tem como objetivo que as organizações estabeleçam um Programa de Governança para o Consumo de Código Aberto (*Strong OSS Governance Program*), levando à Prevenção do Consumo de Código Aberto Malicioso, bem como à Melhoria do Tempo Médio de Remediação de Vulnerabilidades nos componentes de código aberto (*Improve MTTR for OSS vulnerabilities*).

O modelo traz uma relação de ameaças à cadeia de suprimentos de software, detalhada na Tabela 2.4:

Tabela 2.4: Ameaças à cadeia de suprimentos de software

S2C2F - Ameaças à cadeia de suprimentos de software
Vulnerabilidades acidentais nos códigos fonte ou conteiners herdados pela organização
Vulnerabilidades ou backdoors inseridos intencionalmente a uma base de código aberto
Comprometimento de um espelho de distribuição de pacotes por parte de um ator malicioso
Um ator malicioso compromete um componente de código aberto reconhecido como seguro e insere esse código malicioso no repositório de pacotes
Um ator malicioso cria um pacote malicioso que tem nome similar a um pacote de código aberto de alta popularidade, de forma a induzir os desenvolvedores a utilizarem esse pacote
Um ator malicioso compromete o compilador utilizado pelo pacote de código aberto, adicionando backdoors
Confusão de dependências, ataques de substituição de pacotes
Um componente de código aberto adiciona novas dependências que sejam maliciosas
A integridade de um pacote de código aberto é comprometida após a compilação, porém antes do consumo
A fonte original do pacote pode se tornar indisponível, o que causaria erros na compilação de softwares que tenham aquele pacote como dependência
Componentes de código aberto chegam ao fim de seu ciclo de vida ou de suporte, não recebendo mais correções de vulnerabilidades
Vulnerabilidades não são corrigidas em prazo aceitável pelo mantenedor do pacote
Ator malicioso compromete credenciais de um repositório de pacotes e, sem nenhuma correspondência com o respectivo código fonte, disponibiliza uma versão maliciosa do pacote

Fonte: Adaptada de [78]

De forma a combater as ameaças identificadas, determina oito práticas de segurança, relacionadas na Tabela 2.5:

Finalmente, apresenta um modelo de maturidade com quatro níveis, permitindo que as organizações avancem gradualmente do seu estágio atual de segurança até o nível desejado. O modelo considera ainda diferentes ameaças e temas em cada nível de maturidade.

- Nível 1: Programa mínimo de Governança de Software de código aberto
- Nível 2: Consumo seguro e melhorias no MTTR
- Nível 3: Defesas contra *malwares* e detecção de vulnerabilidades de dia zero
- Nível 4: Defesa contra ameaças avançadas persistentes

2.6.7 Secure Level Software Artifacts - SLSA

O *framework* SLSA [25] foi originalmente elaborado pelo Google, para utilização em seus projetos internos, e posteriormente passou a ser aperfeiçoado em conjunto com a organização OpenSSF e disponibilizado de forma pública, sendo que sua atualização mais recente data de 2023. Trata-se de um conjunto de

Tabela 2.5: Práticas de segurança sobre a cadeia de suprimentos de software

Prática	**Descrição**
Internalize os artefatos (Ingest It)	Devo preservar a capacidade de produção e distribuição de qualquer artefato mesmo que o repositório original desses artefatos se torne indisponível
Verifique as vulnerabilidades (Scan It)	Devo conhecer vulnerabilidades e malwares existentes em minha esteira de integração e entrega contínuas
Inventarie (Inventory It)	Devo conhecer todos os artefatos utilizados em produção
Atualize (Update It)	Devo ser capaz de atualizar artefatos assim que novas versões sejam disponibilizadas
Audite (Audit It)	Devo ser capaz de comprovar que cada artefato em produção possui uma cadeia de custódia desde sua origem, sendo consumido por meio da cadeia de suprimentos oficial
Garanta a segurança (Enforce It)	Posso confiar no consumo seguro e confiável de artefatos de código aberto em minha organização
Recompile (Rebuild It)	Devo ser capaz de recompilar a partir do código fonte qualquer artefato que minha organização utilize
Corrija e Comunique (Fix It + Upstream)	Devo ser capaz de produzir correções, compilar e implantar em produção qualquer artefato externo após a notificação de algum risco, e de submeter a correção ao mantenedor oficial do artefato de forma confidencial

Fonte: Adaptada de [78]

recomendações orientadas à cadeia de suprimentos de software, que podem ser utilizadas tanto por desenvolvedores, quer de projetos disponibilizados publicamente, quer de projetos internos de uma determinada organização, quanto por consumidores de software.

Os primeiros podem usar as recomendações para efetivamente tornarem seus produtos mais seguros, criando confiança a respeito das diferentes etapas do ciclo de desenvolvimento, e os últimos podem utilizar as evidências de confiança produzidas e disponibilizadas na etapa de desenvolvimento para avaliarem a segurança de produtos de software que desejem adquirir e executar em seus ambientes. A Figura 2.17 ilustra esse conceito.

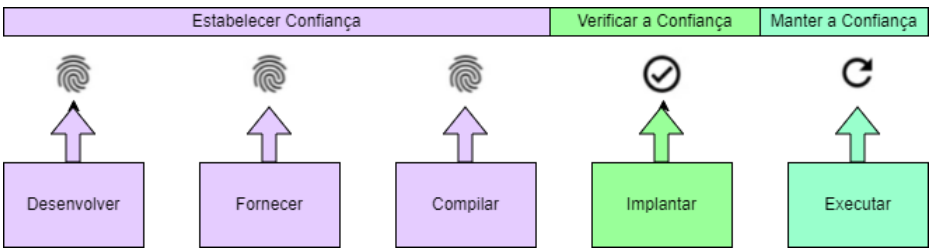


Figura 2.17: SLSA - Objetivos de confiança

Fonte: Adaptada de [25]

O *framework* SLSA é automatizável com relação às tarefas relacionadas à cadeia de suprimentos de software, desde a manipulação de código fonte até a etapa de distribuição, oferecendo proteções contra a modificação não autorizada dos artefatos gerados em cada etapa. A Figura 2.18 ilustra as etapas da cadeia de suprimentos conforme entendidas pelo SLSA (que são idênticas às etapas citadas na Seção 2.1 deste trabalho e ilustradas pela Figura 2.1), e as fontes de ameaças a essas etapas, que são agrupadas em ameaças ao código fonte, ameaças às dependências, e ameaças à compilação, essas últimas atingindo também o pilar de implantação (*deploy*) ou distribuição, por atingirem os repositórios de pacotes finais (*registries*).

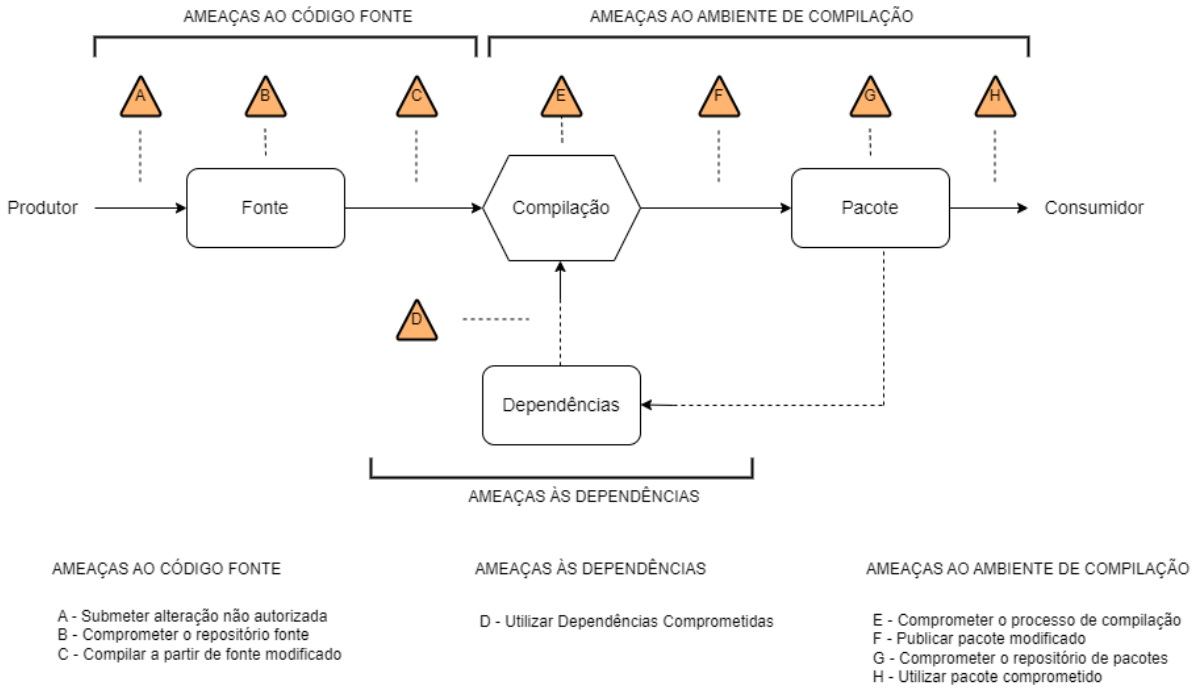


Figura 2.18: SLSA - ameaças à cadeia de suprimentos de software

Fonte: Adaptada de [25]

É estruturado em trilhas, que são subdivididas em níveis crescentes de segurança. Até o momento encontra-se especificada apenas a trilha de compilação *(build)* e seus quatro níveis de segurança, porém o *framework* já prevê as trilhas de código fonte e de plataformas de compilação *(build platforms)*. Os quatro níveis definidos para a trilha de compilação são os seguintes:

- L0: não há qualquer requisito de segurança para este nível
- L1: há verificação de origem dos componentes, demonstrando como o pacote foi compilado
- L2: a compilação é executada em uma plataforma própria, que assina as informações de origem e
- L3: a plataforma de compilação implementa controles fortes para prevenir que compilações influenciem uma à outra, e controle de acesso ao material de assinatura com a prova de origem do artefato

Sugere três casos de uso que ilustram bem seu potencial de usabilidade com relação à segurança da cadeia de suprimentos de software.:

1. Utilização por uma organização com o objetivo de redução de riscos oriundos da equipe interna (*insiders*) e credenciais comprometidas
2. Utilização para a redução do risco de consumo de projetos de código aberto, por meio do mapeamento dos pacotes a serem utilizados às fontes primárias de seu código e dependências
3. Utilização para a redução de riscos no consumo de softwares comerciais. O SLSA pode ser utilizado para certificar que os fornecedores efetivamente implementem as boas práticas relativas à cadeia de suprimentos de software

2.6.8 Resumo dos *frameworks* apresentados

Os tópicos 2.6.1 a 2.6.6 detalham diversos *frameworks* voltados à segurança para cadeias de suprimentos de software, abordando diferentes conjuntos de pilares de cadeias de suprimentos de software, e publicados por diferentes organizações.

A Tabela 2.6 oferece um resumo, de forma a oferecer ao leitor uma fácil visualização do panorama a respeito dos *frameworks* publicados sobre o tema.

Tabela 2.6: Resumo dos frameworks

Framework	**Responsável**	**Pilares abordados**	**Data da publicação ou atualização**
SCVS	OWASP	Dependência, Compilação, Deploy-Distribuição	2020
Software Supply Chain Best Practices	CNCF	Código-fonte, Dependências, Compilação, Deploy-Distribuição	2021
Software Supply Chain Security Guide	CIS	Código-fonte, Dependências, Compilação, Deploy-Distribuição	2022
Securing the Software Supply Chain	ESF	Código-fonte, Dependências, Compilação, Deploy-Distribuição	2022
OWASP Top 10 CI/CD Security Risks	OWASP	Código-fonte, Dependências, Compilação, Deploy-Distribuição	2022
S2C2F	Microsoft	Dependências	2023
SLSA	Google	Compilação (até o momento)	2023

Fonte: Elaborada pelo autor

2.7 TAXONOMIAS SOBRE ATAQUES A CADEIAS DE SUPRIMENTOS DE SOFTWARE

Além dos *frameworks* e guias de boas práticas mencionados, esforços também têm sido direcionados para a definição de taxonomias de ataques, bem como das técnicas, táticas e procedimentos (TTPs) empregados.

De forma a possibilitar o correto entendimento a respeito dessas taxonomias e TTPs, é fundamental estabelecer uma definição clara do que são ataques à cadeia de suprimentos de software. De acordo com a *Cloud Native Computing Foundation* (CNCF), esse tipo de ataque pode ser definido como o comprometimento dos componentes ou processos utilizados na produção de software, tendo como resultado a inserção de vulnerabilidades nos produtos finais, que irão comprometer o ambiente dos clientes dos software produzidos [29].

Ataques à cadeia de suprimento de software podem ocorrer de algumas formas distintas, e taxonomias de ataque podem auxiliar na compreensão tanto do cenário de ameaças quanto nas abordagens de proteção. Nesta Seção serão apresentadas as taxonomias e TTPs sobre o tema que foram identificadas.

2.7.1 Taxonomia proposta por Ladisa et al.

Com relação a propostas de taxonomias, destaca-se o trabalho *"Taxonomy of Attacks on Open-Source Software Supply Chains"*, que propõe uma detalhada taxonomia de ataques à cadeia de suprimentos de software a partir de uma revisão sistemática da literatura científica e especializada, validada por pesquisas realizadas junto a especialistas no domínio da segurança da cadeia de suprimentos de softwares de código

aberto e a desenvolvedores de software [99].

O trabalho tem sido ampliado, resultando em uma segunda publicação, *"Journey to the Center of Software Supply Chain Attacks"* [32]. O primeiro estudo identificou 107 vetores de ataque únicos, número que aumentou para 117 na segunda publicação. Em ambos os trabalhos, foram destacados 33 controles de segurança aplicáveis à prevenção desses vetores.

O primeiro e segundo níveis da árvore de ataques documentam os principais tipos de ataques conhecidos contra cadeias de suprimento de software, conforme ilustra a Tabela 2.7.

Tabela 2.7: Níveis 1 e 2 da árvore de ataques

Primeiro Nível	**Segundo Nível**
Develop and Advertise Distinct Malicious Package from Scratch	
Create Name Confusion with Legitimate Package	Combosquatting
	Altering Word Order
	Manipulating Word Separators
	Typosquatting
	Built-in Package
	Brandjacking
	Similarity Attack
	Omitting Scope or Namespace
	AI Package Hallucination
Subvert Legitimate Package	Inject Into Sources of Legitimate Package
	Inject During the Build of Legitimate Package
	Distribute Malicous Version of Legitimate Package

Adaptada de [32]

Quanto aos 33 controles de segurança relacionados aos 117 vetores de ataque, indica seus níveis de utilidade de custo, o tipo do controle (diretivo, preditivo, detectivo ou corretivo), e as partes interessadas envolvidas na aplicação de cada controle (Mantenedor dos pacotes, Provedor de serviço, Consumidor dos pacotes). Os autores compilaram uma tabela relacionando os 33 controles aos respectivos vetores de ataque, indicando ainda uma relação de custo-benefício referente à aplicação de cada controle, sua classificação entre controle diretivo, preventivo, detectivo e corretivo, bem como os responsáveis por sua aplicação.

O conjunto completo de vetores de ataque, e seus respectivos controles de segurança, foram organizados em uma árvore hierárquica, que vem sendo mantida e atualizada em um repositório online, de forma que possa ser dinamicamente consultada por qualquer interessado [100].

O estudo traz ainda uma relação dos principais objetivos observados em casos reais de ataque à cadeia de suprimentos de software:

- ativação de *Shell* reverso (de forma a prover ao atacante acesso interativo ao ambiente do alvo)
- instalação de *droppers* (como primeiro estágio de ataque, para a implantação de outros artefatos maliciosos)
- exfiltração de dados, que o artigo indica como sendo o comportamento mais comum nos ataques observados

- negação de serviço, normalmente por meio da exaustão de recursos computacinais ou pela exclusão de arquivos no ambiente da vítima
- ganho financeiro, por meio da execução de mineração de criptomoedas no sistema alvo

2.7.2 Taxonomia proposta por Melara

O artigo *"What is Software Supply Chain Security?"* [101] propõe uma taxonomia quanto aos objetivos das soluções de segurança para a cadeia de suprimentos de software, dividida em dois níveis, conforme Tabela 2.8.

Tabela 2.8: Níveis 1 e 2 da árvore de ataques

Objetivos de 1º Nível	**Objetivos de 2º Nível**
Objetivos para estabelecimento de confiança	Verificar a identidade de um participante
	Verificar os componentes de um artefato
	Verificar as propriedades ou o comportamento de um artefato
	Verificar as propriedades de um processo
Objetivos para Ferramentas resilientes	Identificar vulnerabilidades em uma ferramenta existente
	Implementar ferramentas de alta confiabilidade
Objetivos para Processos resilientes	Implementar um processo que reduza a superfície de ataque
	Automatizar processos manuais sujeitos a erros

Adaptada de [101]

As duas taxonomias apresentadas proporcionam uma compreensão mais aprofundada do cenário de segurança da cadeia de suprimentos de software, classificando os tipos de ataques conhecidos e alinhando-os aos objetivos dos *frameworks* e soluções de segurança disponíveis.

2.8 MODELOS DE TÁTICAS, TÉCNICAS E PROCEDIMENTOS (TTPS) UTILIZADOS PARA ATAQUES A CADEIAS DE SUPRIMENTOS

2.8.1 Mitre Att&ck

Mitre é uma organização sem fins lucrativos baseada nos Estados Unidos, fundada em 1958, por meio de patrocínio da Força Aérea norte americana, tendo como objetivo inicial atuar como consultores em engenharia de sistemas para agencias governamentais militares e civis, porém hoje atua em uma diversidade de temas, desde pesquisas sobre câncer a veículos autônomos, passando também pelo tema de cibersegurança, e tem como missão a resolução de problemas para um mundo mais seguro [102]

Nesse contexto, desenvolveu o Mitre Att&ck, uma base de conhecimento a respeito de táticas, técnicas e procedimentos (TTPs) utilizadas por adversários, detectadas com base em observações sobre o mundo real. Tem como foco principal descrever como os adversários atuam durante uma operação de ataque, identificando as várias fases utilizadas, bem como as plataformas que costumam eleger como alvos [103].

Entre as técnicas identificadas pelo ATT&CK, destaca-se a técnica de Comprometimento da Cadeia de

Suprimentos (*Supply Chain Compromise*), identificada pelo código T1195.

De acordo com o Mitre, o comprometimento da cadeia de suprimentos pode ocorrer em qualquer ponto dessa cadeia, conforme relação a seguir, pontos esses que podem ser entendidos como uma taxonomia de ataques à cadeia de suprimentos de software.

- Manipulação de ferramentas de desenvolvimento
- Manipulação do ambiente de desenvolvimento
- Manipulação dos repositórios de código fonte (públicos ou privados)
- Manipulação de código fonte em dependências de código aberto
- Manipulação de mecanismos de distribuição ou atualização de software
- Comprometimento ou infecção de imagens nas midias de distribuição nas próprias dependências do fabricante
- Venda de produtos modificados ou falsificados a distribuidores legítimos
- Impedimento do envio dos produtos aos destinatários

A descrição ressalta que os agentes mal-intencionados, com o objetivo de obter acesso aos ambientes das potenciais vítimas, frequentemente direcionam seus esforços para a inserção de código malicioso em softwares legítimos, explorando a confiança inerente nesses sistemas, ou em seus canais de distribuição, sendo que o alvo do ataque pode tanto ser uma organização específica ou a distribuição do ataque para uma grande diversidade de consumidores, e informa ainda que projetos de código aberto que tenham grande utilização como componentes pelos times de desenvolvimento muitas vezes são escolhidos como alvos para a inserção de códigos maliciosos por parte dos atacantes.

2.8.2 Open Software Supply Chain Attack Reference - OSC&R

Inspirado pelo exemplo do Mitre Att&ck, um grupo de especialistas em cibersegurança oriundos de diversas grandes empresas do setor, tais como Microsoft, Google, GitLab, Check Point, OWASP e Fortinet, sob a liderança da empresa OX Security, elaborou a iniciativa *Open Software Supply Chain Attack Reference* (OSC&R), que também é uma compilação de Técnicas, Táticas e Procedimentos, porém especializada em ataques à cadeia de suprimentos de software [104].

A iniciativa OSC&R define uma "*Pipeline Bill of Materials* (PBOM)", a partir da qual procura avaliar os possíveis ataques sobre toda a esteira de desenvolvimento e de distribuição de software. Até o momento, os elementos indicados como componentes da PBOM, que podem ser entendidos como uma taxonomia de ataques à cadeia de suprimentos de software, são os seguintes:

- Segurança de Containers
- Segurança de projetos de Código Aberto

- Postura de gerenciamento da cadeia de suprimentos (SCM Posture)
- Cuidados no tratamento de credenciais
- Segurança de código
- Segurança de nuvem
- Postura do ambiente de CI/CD (Integração Contínua/Implantação Contínua)
- Segurança de Artefatos
- Infraestrutura como código

Além desses componentes, relaciona doze etapas envolvidas em ataques a cadeias de suprimentos de software:

1. Reconhecimento
2. Desenvolvimento de Recursos
3. Acesso inicial
4. Execução
5. Persistência
6. Escalação de privilégios
7. Evasão das defesas
8. Acesso a credenciais
9. Movimento lateral
10. Coleção de informações
11. Exfiltração
12. Impacto

A partir dos componentes e das etapas, disponibiliza uma matriz bidimensional indicando as técnicas utilizadas para a realização de ataques, provendo descrições detalhadas sobre cada uma das técnicas, bem como maneiras de mitigar e detectar suas eventuais execuções.

2.9 TRABALHOS CORRELATOS

Com o objetivo de identificar propostas de modelos para a estruturação de esforços e processos organizacionais voltados à segurança da cadeia de suprimentos de software, foram realizadas pesquisas em bases de indexação acadêmica, como Scopus, Web of Science e Google Scholar, além de buscas na internet. As investigações abordaram modelos de maturidade, conformidade e gestão de riscos relacionados ao tema.

Identificou-se dois modelos de maturidade em segurança da cadeia de suprimentos de software.

O primeiro foi disponibilizado pela empresa Sonatype em junho de 2023, e é intitulado *"Software Supply Chain Maturity Framework"* [105].

Propõe-se a ser um instrumento de medição de maturidade das práticas executadas pela organização com relação à gestão de sua cadeia de suprimentos de software, definindo níveis de maturidade pelos quais a organização pode avançar enquanto incrementa suas práticas e procedimentos.

É dividido em oito temas principais, a saber:

- Inventário: a organização deve catalogar e rastrear os componentes que utiliza em suas atividades de desenvolvimento de software, e deve adotar processos de identificação de componentes, controle de versões e gestão de vulnerabilidades desses componentes;
- Fornecedores: os fornecedores de componentes de software devem ser cuidadosamente avaliados e selecionados, a partir de critérios como práticas de segurança que utilizam, padrões de qualidade e aderência a padrões de código aberto;
- Gestão de riscos: os riscos que envolvem a cadeia de suprimentos de software devem ser gerenciados, envolvendo medidas proativas como varredura de vulnerabilidades, análises de ameaças e avaliações de segurança, e devem envolver todas as partes interessadas relativas à segurança, arquitetura e aspectos legais;
- Consumo: a organização deve adotar políticas de consumo de componentes de terceiros, definindo padrões para a utilização desses componentes tanto do ponto de vista de seus licenciamentos, quanto de sua qualidade e segurança;
- Compilação e liberação: o processo de compilação e liberação de artefatos deve ser automatizado, reproduzível e consistente, reduzindo a probabilidade de erros e vulnerabilidades, por meio da utilização de esteiras de integração contínua e entrega contínua (CI/CD);
- Remediação: é importante o estabelecimento de processos de remediação de problemas e vulnerabilidades detectadas, tais como gestão de *patches*, vulnerabilidades e conformidade de licenciamento;
- Plano de execução: a organização deve elaborar um plano de execução de suas atividades de gestão da cadeia de suprimentos de software, englobando pessoas, processos e tecnologias, e envolvendo as equipes de segurança, desenvolvimento e jurídica;
- Contribuição: a colaboração com outras organizações participantes da cadeia de suprimentos de software, notadamente os desenvolvedores de componentes de código aberto, contribui para sua

resiliência e robustez, levando, em última instância, ao desenvolvimento de softwares de maior qualidade.

A empresa disponibiliza ainda um questionário de auto-avaliação, como ponto de partida para que as organizações possam conhecer seu estado atual de maturidade quanto às práticas de gestão de suas cadeias de suprimentos de software, e possam traçar seus planos de aprimoramento.

A RedHat, por sua vez, em conjunto com a empresa SlashData, elaborou seu próprio modelo de maturidade com relação às práticas de segurança sobre a cadeia de suprimentos de software, e publicou, em fevereiro de 2024, uma avaliação de maturidade de desenvolvedores e suas organizações com relação ao tema [106].

A avaliação foi conduzida no último trimestre de 2023, por meio de uma pesquisa online realizada junto a mais de 800 desenvolvedores, localizados em mais de 80 países.

A pesquisa continha questões relacionadas a garantias, transparência, conformidade, consistência e resiliência, a partir das quais foram calculados os níveis de maturidade para as práticas relativas a eficácia e integridade de software, desempenho de entregas, automação de processos, e práticas de remediação.

Foram então calculados os níveis de maturidade relativos a cada respondente, e elaborados gráficos comparativos que ilustram os níveis de maturidade gerais para a totalidade dos entrevistados.

Os resultados indicam que, no grupo entrevistado, com exceção das práticas de remediação, mais da metade dos participantes relatou adoção das demais práticas nos níveis de maturidade "High"(alto) ou "Elite". Entretanto, a prática de remediação possui níveis de maturidade inferiores, o que traz o nível de maturidade geral para índices próximos a 50% para a soma dos níveis "Low"(baixo) e "Medium"(médio), e para a soma dos níveis "High"e "Elite".

Quanto aos modelos de gestão de riscos da cadeia de suprimentos de software, foi realizada uma revisão sistemática inicial da literatura, utilizando como base a metodologia TEMAC [107], a partir da qual foram identificados artigos científicos voltados à gestão de riscos da cadeia de suprimentos de software. A revisão, entretanto, não foi estruturada como um artigo finalizado nem submetida para publicação. As principais contribuições dos artigos identificados são apresentadas neste tópico em ordem cronológica.

Ainda no ano de 2003, sob uma inciativa do *Software Engineering Institute* (SEI), da Universidade Carnegie Mellon, Dorofee et al. propuseram uma abordagem sistêmica para a avaliação de riscos da cadeia de suprimentos de software. Nessa abordagem, os autores definem que a cadeia de suprimentos de software é um complexo sistema socio-tecnológico onde elementos técnicos e sociais interagem para o alcance de um comportamento orientado a um objetivo. Ou seja, consideram que a cadeia de suprimentos de software também é composta pela interação entre diferentes sistemas, e entre sistemas e as equipes que os utilizam, representando assim um conceito de cadeia de suprimentos de software distinta do conceito analisado neste trabalho [108].

Os autores esclarecem que há duas abordagens possíveis para a avaliação de riscos em sistemas socio-tecnológicos, a primeira baseada na decomposição dos sistemas e análise de eventos, que é util para prevenir falhas de componentes críticos do sistema, e a segunda baseada na análise sistêmica, em que o sistema é avaliado como um todo. A segunda abordagem seria mais eficiente para a avaliação da cadeia de supri-

mentos de software, pois trata melhor o alto grau de incerteza associado a esse tipo de sistema.

Já em 2010, Woody e Ellison (dois dos autores participantes do estudo conduzido por Dorofee), abordaram a questão da gestão de riscos da cadeia de suprimentos de software recomendando a incorporação de segurança no desenvolvimento de software. Suas principais recomendações são voltadas à segurança do software em seu estado final, tendo sido recomendados o mapeamento da superfície de ataques, realização da modelagem de ameaças e a execução de testes do tipo *fuzzing*, onde a aplicação é submetida a entrada de dados mal formados, de forma a avaliar sua reação a esses casos [109][110].

Du et al. propõem um conjunto de melhores práticas referentes à gestão de riscos da cadeia de suprimentos de software. Dividem os riscos em externos e internos, sendo os primeiros compostos pelos fatores de desastres naturais, políticos, econômicos e sociais, sendo argumentado que não há como prevê-los ou evitá-los, e, no segundo caso, definem seus fatores como sendo associados aos participantes da cadeia de suprimentos de software, componentes de software, operação e manutenção, e logística [111].

Como melhores práticas para enfrentar os riscos internos, propõem as seguintes atividades:

- Projetar uma boa cadeia de suprimentos, estabelecendo canais eficientes de entrega de informações, incrementar a transparência das informações compartilhadas entre clientes e fornecedores, simplificar a cadeia de suprimentos (uma vez que é mais fácil garantir a segurança de uma cadeia de suprimentos mais simples), e, por fim, incrementar a flexibilidade da cadeia de suprimentos, por meio da escolha de uma quantidade apropriada de fornecedores, e da flexibilidade do processo de produção, considerando que uma mesma entidade pode ser consumidora e fornecedora em diferentes pontos da cadeia de suprimentos
- Gerenciar os componentes de código aberto, sugerindo quatro etapas para isso: coletar informações sobre os componentes, analizar as vulnerabilidades da aplicação, estabelecer controle sobre o ciclo de desenvolvimento de software, e gerenciar os componentes com base em uma compreensão mais aprofundada sobre cada um
- Utilizar métodos de desenvolvimento seguro de software, incluindo avaliar os métodos de desenvolvimento seguro utilizados pelos fornecedores
- Testar a segurança dos softwares produzidos, utilizando, por exemplo, ferramentas de análise estática de código

Brennan aborda o conceito de governança de código, disciplina que teve como objetivo original garantir a qualidade e segurança de código desenvolvido internamente nas organizações, e que evoluiu para melhorar a responsabilização e eficiência de equipes de desenvolvimento distribuídas e para obter melhor visibilidade e controle sobre o código de terceiros [112].

Recomenda a implementação de alguns procedimentos, como o estabelecimento de critérios de aceitação de código (que podem ser automatizados por meio de ferramentas de análise estática, por exemplo), de forma que vulnerabilidades no código sejam tratadas como requisito para a aceitação do código, a realização de auditorias sobre o código fornecido por terceiros, idealmente como parte da etapa de integração do software, e, por fim, a auto-certificação, por meio da qual fornecedores certificariam a qualidade de seus códigos antes de sua disponibilização na cadeia de suprimentos de software.

Em uma abordagem diferente, Benthall propõe uma forma automatizada de estimativa da quantidade de vulnerabilidades ainda desconhecidas em um projeto de software, por meio de um cruzamento entre as vulnerabilidades sobre o projeto publicadas na *National Vulnerability Database* (NVD) e as informações sobre as versões dos projetos publicadas em seus sistemas de controle de versão (a exemplo do Git), utilizando o software Openssl como estudo de caso [113].

A proposta é baseada no modelo de ciclo de vida de software chamado *Alhazmi-Malaiya Logistic* (AML), que o divide em três etapas: a etapa de aprendizado, quando o interesse pelo uso do software ainda é pequeno, de forma que há pouco interesse na descoberta de vulnerabilidades; a etapa linear, onde o uso do software e seus esforços de desenvolvimento encontram-se em alta, e a quantidade de vulnerabilidades aumenta de forma linear; e a última etapa, de saturação, quando a grande maioria das vulnerabilidades já foram descobertas, e então a quantidade cumulativa de vulnerabilidades associadas ao software atinge seu platô [114].

De acordo com esse modelo, testado em projetos como o RedHat Linux, Windows, servidores web Apache e Microsoft IIS, e navegadores web Internet Explorer, Firefox e Mozilla, a partir de uma base histórica composta por vários projetos seria possível prever a quantidade de vulnerabilidades existentes em um determinado projeto a partir do índice médio de descoberta de vulnerabilidades. O próprio caso do Openssl, entretanto, representa uma exceção, uma vez que após o alcance da fase de platô houve a descoberta de uma vulnerabilidade crítica, a vulnerabilidade *hartbleed*, que desencadeou investimentos significativos em segurança em torno do projeto, quando foi descoberta uma quantidade significativa de novas vulnerabilidades. Este caso evidenciou empiricamente uma nova característica relativa à detecção de vulnerabildades em um dado projeto: um platô temporário da quantidade acumulada de vulnerabilidades descobertas.

Librantz et al. propõem uma abordagem para a avaliação dos riscos primários da cadeia de suprimentos de software, identificando, modelando e estimando os principais riscos e apontando interdependências e propagações de riscos [115].

A proposta é baseada em *Bayesian Belief Networks* (BBNs), que, segundo os autores, pode representar o conhecimento de especialistas em domínios onde tal conhecimento seja probabilístico, e que tem a vantagem de não apresentar limitações quanto à análise da propagação de riscos ao longo da cadeia de suprimentos, quando comparado a outros métodos utilizados por outros autores, tais como Questionários, *Analytic Hierarchy Process* (AHP), Análise de clusters, *Failure Mode and Effect Analysis* (FMEA), *Fuzzy set theory* (para a modelagem de processos decisórios baseados em informações vagas), *Fuzzy Topsis*, *Fuzzy AHP*, e uma combinação entre *Analytical Network Process* (ANP) e ELECTRE (um método para análise baseada em multicritérios).

A partir da classificação de riscos externos e internos proposta por Du et al. [111], com a adição de riscos culturais propostos por Lee et al.[116], elaboram uma BBN que esquematiza tais riscos, e complementam o método proposto com o uso dos métodos AHP e noisy-OR, como forma de diminuir a quantidade de perguntas que devem ser respondidas por especialistas para a modelagem da BBN. Como forma de validação do modelo, analisam e quantificam os riscos da cadeia de suprimento de softwares de três projetos desenvolvidos em três países distintos, efetivamente calculando os índices de riscos relativos a esses projetos.

Em outro trabalho, Liang et al. analisam os riscos envolvendo a utilização de componentes de código aberto em soluções de software no sistema de transporte chinês. Identificam como principais riscos os relacionados a conformidade com questões de propriedade intelectual, vulnerabilidades de segurança e riscos técnicos e operacionais, e propõem um conjunto de seis etapas como medidas para mitigação desses riscos: pesquisar a utilização de componentes de código aberto na indústria de transportes, construir uma base de dados local de componentes de código aberto, construir e manter uma base de vulnerabilidades conhecidas, atribuir um nível de risco para cada componente, identificar as dependências de cada componente por meio de SBOM, e selecionar vulnerabilidades para análise de riscos [117].

Em uma abordagem não diretamente ligada à gestão de riscos, mas à seleção de controles de segurança para a cadeia de suprimentos de software, Amaral elaborou uma proposta de integração de controles de segurança à cadeia de suprimentos a partir de conceitos de confiança zero (*zero trust*) [118].

O trabalho estrutura os controles de segurança em domínios, e oferece um *checklist* para a realização de análise da situação existente na organização em relação aos controles sugeridos, possibilitando, a partir daí, a elaboração de uma estratégia de aplicação incremental dos controles ainda não adotados.

Por fim, há o trabalho de Zahan, que consolida a primeira etapa de sua pesquisa que tem como objetivo a proposição de um método para Avaliação de Riscos na Cadeia de Suprimentos de Software (*Software Supply Chain Risk Assessment Framework* - SSCRAF) [119].

Parte da premissa de que para diminuir os vetores de ataque à cadeia de suprimentos de software é necessário elaborar métricas de segurança que indiquem que o desenvolvimento dos componentes não tenha sido apoiado em práticas de segurança, e que detectem fragilidades no gráfico de dependências que exponham o pacote a maiores riscos de ataques, e, para tanto, parte de três questões de pesquisa:

1. Que práticas de segurança devem ser adotadas para melhorar a segurança do pacote?
2. De que forma métricas de segurança podem auxiliar na detecção de fragilidades nos ecossistemas de software?
3. De que forma métricas auto-detectáveis podem ser usadas como evidência do uso de práticas de segurança?

O autor estabelece como problema o fato de que os diferentes *frameworks* e projetos da OpenSSF trazem diversas recomendações de segurança, mas que, como as organizações muitas vezes não têm os recursos para implementá-las em sua totalidade, se faz necessário validar em que medida essas recomendações efetivamente trazem melhorias de segurança, de forma a guiar as organizações quanto à adoção daquelas que trazem melhorias mais significativas.

Um segundo problema que aponta é o de que os estudos atuais são eficazes na prevenção de distribuição de código malicioso, porém não previnem a execução com sucesso de estratégias de ataque inovadoras. Assim, faz-se importante a adoção de uma estratégia capaz de prever a suscetibilidade dos pacotes a possíveis ameaças.

A partir dessas questões e problemas foram avaliadas as métricas de segurança propostas pelo OpenSSF Scorecard e seu efetivo uso nos ecossistemas NPM e Pipy, além de métricas de risco no ecossistema NPM.

Com relação à primeira avaliação identificou que nove das métricas de segurança podem efetivamente ser utilizadas para a medição de segurança de pacotes NPM e Pipy, enquanto 5 métricas ainda necessitam de concordância por parte da indústria, e, com relação à segunda avaliação, identificaram seis sinais indicativos de fragilidades de segurança em pacotes NPM.

Como resultado do estudo, os ambientes NPM, GitHub e Pypi passaram a adotar duplo fator de autenticação para as contas de desenvolvedores, e ferramentas da indústria estão integrando os seis sinais identificados como verificações automatizadas de segurança da cadeia de suprimentos de software.

3 METODOLOGIA

O presente estudo é classificado, quanto à sua finalidade, como pesquisa aplicada, na medida em que se caracteriza pela aquisição de conhecimento relativo às melhores práticas atualmente propostas com relação à cadeia de suprimentos de software, com vistas a disponibilizar a qualquer organização interessada uma ferramenta para guiar seus esforços destinados a compreender ou a evoluir o nível de segurança da cadeia de suprimentos de seus próprios projetos de software, alinhando-se com o conceito definido em [120].

Com relação ao seu propósito, trata-se de uma pesquisa descritiva e exploratória, pois procura descrever as principais características relativas à segurança da cadeia de suprimentos de software, compreendê-las adequadamente, e torná-las mais explícitas a eventuais interessados [120].

Quanto aos procedimentos utilizados foi delineado como uma pesquisa bibliográfica, pois é baseada na avaliação de *frameworks* e taxonomias de ataques à cadeia de suprimentos de software já publicados, principalmente na forma de livros e artigos científicos [120].

Foi realizado a partir de uma abordagem qualitativa, uma vez que os controles de segurança recomendados pelos *frameworks* avaliados, e a respectiva atribuição desses controles a grupos de implementação ordenados de forma crescente evidencia seu caráter subjetivo, que não pode ser traduzido em números por meio de métodos estatísticos [121].

3.1 SELEÇÃO DE FONTES DE CONTROLES DE SEGURANÇA E DEFINIÇÃO DO TIPO DE MODELO A SER ELABORADO

O referencial teórico, contido no Capítulo 2, descreveu normas, padrões e *frameworks* relacionados a desenvolvimento seguro de sistemas, gestão de riscos de cadeias de suprimentos, e segurança de cadeias de suprimentos de software especificamente.

Os documentos voltados ao desenvolvimento seguro de sistemas tratam o processo de desenvolvimento como um todo, não detalhando recomendações de segurança específicas para as cadeias de suprimentos de software, citando os *frameworks* especificamente voltados para o tema como referências.

Os padrões e normas voltados à gestão de riscos são documentos de cunho mais genérico, que buscam moldar as atividades de gestão de riscos, não se aprofundando na recomendação de controles de segurança específicos. A exceção é o padrão NIST SP 800-161 r1, que traz uma extensa lista de controles recomendados, porém seu objeto é o gerenciamento de riscos de cibersegurança sobre cadeias de suprimentos de uma forma geral, e não sobre cadeias de suprimentos de software de forma específica.

Os padrões e normas relacionados à segurança da cadeia de suprimentos de software trazem processos e atividades que devem ser executados para uma boa gestão desse ambiente, porém também não detalham recomendações de segurança mais específicas.

Os *frameworks* com foco em segurança de cadeias de suprimentos de software são os que já relacionam

recomendações e controles de segurança específicos para esse ambiente, elaborados por especialistas em cibersegurança, desenvolvimento seguro, e cadeias de suprimentos de software, a partir da observação de casos reais de ataques e incidentes de segurança.

Considerando-se o objetivo principal deste trabalho, o de consolidar as melhores práticas acerca da segurança de cadeias de suprimentos de software e propor um modelo que reúna as recomendações dos principais *frameworks* voltados ao assunto, adotou-se então o conjunto de *frameworks* voltados especificamente à segurança de cadeias de suprimentos de software como insumo para a elaboração do modelo.

Ao avaliar os possíveis tipos de modelos que pudessem orientar organizações interessadas em iniciar sua jornada com relação à cadeia de suprimentos de software, ou mesmo incrementar seus esforços já iniciados, foram identificadas as possibilidades de utilização de modelos de maturidade ou de modelos de conformidade.

Modelos de maturidade são associados à implementação de processos, conforme se pode observar a partir de normativos ou padrões como a ISO 27.036 [27] e o Capability Maturity Model Integration - CMMI [122]. Processos, por sua vez, necessitam da definição de objetivos, práticas e métricas para medição de sua implementação, que necessariamente devem observar particularidades de cada organização [122].

Modelos de conformidade são baseados no atendimento com relação a medidas ou controles formalmente recomendados, conceito que se adapta bem ao presente estudo, em função de seu objetivo, que é o de fornecer orientação às organizações interessadas com relação à implementação das melhores práticas conhecidas sobre a segurança da cadeia de suprimentos de software.

O referencial teórico identificou a existência de dois modelos de maturidade da cadeia de suprimentos de software, ao passo que não foram encontrados modelos de conformidade que abordassem o conjunto das melhores práticas atualmente publicadas.

A análise inicial dos *frameworks* e normativos relacionados ao tema revelou diferenças significativas entre os documentos, indicando que a criação de um modelo de conformidade que integre essas diferenças e recomendações complementares seria uma contribuição relevante para a área.

Adicionalmente, incluiu-se no modelo também os trabalhos acadêmicos de Ladisa et. al [32, 99], pelo fato de definirem formalmente uma taxonomia de ataques à cadeia de suprimentos de software, proporem controles de segurança para estabelecer defesas contra os tipos de ataques identificados, e serem reconhecidos pela comunidade acadêmica, contando com um total de 45 citações por parte de outros autores, de acordo com o Google Scholar.

O Mitre Att&ck e o OSC&R, citados na Seção 2.7 como coleções de técnicas, táticas e procedimentos (TTPs) voltados ao ataque à cadeia de suprimento de softwares, não foram incluídos no modelo de conformidade pelo fato de trazerem uma abordagem bastante distinta em termos de organização dos tipos de ataques conhecidos e respectivas contramedidas.

Adicionalmente, o OSC&R, se estende para além dos pilares relacionados à cadeia de suprimentos de software definidos pelos *frameworks* considerados, abordando também questões como segurança de conteiners, segurança de ambientes de nuvem, e segurança de infraestrutura como código.

Assim, a relação final de *frameworks* e artigos acadêmicos utilizados para a composição do modelo de

conformidade foi a seguinte:

- *Center for Internet Security (CIS) - Software Supply Chain Security Guide*;
- *Enduring Security Framework (ESF) - Securing the Software Supply Chain for Developers*;
- *Cloud Native Computing Foundation (CNCF) - Software Supply Chain Best Practices*;
- *Secure Supply Chain Consumption Framework (S2C2F)*;
- *Secure Level Software Artifacts (SLSA)*;
- *OWASP Secure Component Verification Standard (SCVS)*;
- *OWASP Top 10 CI/CD (OW T10)*;
- *Journey to the Center of Software Supply Chain Attacks*.

3.2 IDENTIFICAÇÃO DOS CONTROLES RECOMENDADOS PELOS *FRAMEWORKS* SELECIONADOS

Os *frameworks* CIS - *Software Supply Chain Security Guide*, *Secure Supply Chain Consumption Framework* (S2C2F), *Secure Level Software Artifacts* (SLSA), *OWASP Secure Component Verification Standard* (SCVS) e o artigo *Journey to the Center of Software Supply Chain Attacks*, são estruturados de forma tal que cada controle de segurança recomendado é equivalente a um tópico do *framework*. Nesses casos, cada um dos tópicos (ou controles) foi diretamente adotado em nosso modelo de conformidade como um controle de segurança.

Já os *frameworks* CNCF - *Software Supply Chain Best Practices*, ESF - *Securing the Software Supply Chain for Developers* e *OWASP Top 10 CI/CD Risks*, são formatados como texto livre, sendo organizados em torno de tópicos genéricos, de forma que os controles de segurança são citados ao longo dos textos referentes a cada um desses tópicos genéricos. Nesses casos, identificou-se os controles sugeridos nos textos, que foram então adotados como controles para a composição do modelo de conformidade.

3.3 MÉTODO UTILIZADO PARA COMPARAÇÃO DOS *FRAMEWORKS*

Cada um dos *frameworks* e dos trabalhos acadêmicos considerados traz recomendações de controles de segurança voltados aos pilares da cadeia de suprimentos de software, a saber, código-fonte, dependências, ambiente de compilação e *deploy* ou distribuição dos artefatos finais.

Inicialmente, quantificou-se os controles recomendados por cada um dos *frameworks*, e em seguida identificou-se e quantificou-se os controles relativos aos pilares da cadeia de suprimentos de software. Dois dos *frameworks* considerados atribuem controles também ao pilar de "artefatos", porém, como suas recomendações são sempre associadas aos pilares "dependências", "*build* (compilação)"ou "*deploy*-distribuição", redistribuímos esses controles ao pilar pertinente dentre esses, de forma a manter a coerência do modelo.

Como o cerne desses conjuntos de boas práticas são controles de segurança, buscou-se métodos já utilizados na indústria para a realização de comparações semelhantes, e identificou-se o método utilizado pelo *Center for Internet Security* (CIS) para a realização de comparações entre seus *frameworks*, mais notadamente o *CIS Controls*, e outros *frameworks* referentes a controles de segurança.

Já foram realizados e publicados em seu site 29 mapeamentos distintos entre o CIS Controls versão 8.1 e outros *frameworks*, dentre os quais destacam-se os referentes aos normativos ISO 27001 e 27002, NIST CSF 1.0 e 2.0, e NIST SP 800-53, além de mapeamentos realizados a partir de suas versões anteriores, CIS Controls versão 7.1 e 8.0 [123].

O método utilizado baseia-se nas seguintes diretivas, documentadas pelo CIS em seu mapeamento entre o CIS Controls e o NIST CSF [124]:

- Caso os controles recomendados por ambos os *frameworks* sendo comparados representem exatamente o mesmo conceito de segurança, serão considerados como **Equivalentes**;
- Caso o controle recomendado pelo *framework* adotado como base da comparação seja parcialmente ou em grande parte relacionado ao *framework* alvo, porém represente um conceito de segurança mais amplo, será considerado como **Superconjunto**;
- Caso o controle recomendado pelo *framework* adotado como base da comparação seja parcialmente ou em grande parte relacionado ao *framework* alvo, porém represente um conceito de segurança menos amplo, será considerado como **Subconjunto**;
- Caso não haja relacionamentos entre controles indicados por ambos os *frameworks*, não será indicada nenhuma associação entre ambos.

No modelo proposto, incluiu-se diretivas adicionais, de forma a melhor representar o escopo do conjunto de controles existentes relacionados aos *frameworks* sob comparação, adotando também os tipos de relacionamento "interseção", "não mapeado no destino"e "não existente na origem", de forma que o conjunto completo de diretivas utilizadas é o seguinte:

- Caso os controles recomendados por ambos os *frameworks* sendo comparados representem exatamente o mesmo conceito de segurança, serão considerados como **Equivalentes**;
- Caso o controle recomendado pelo *framework* adotado como base da comparação seja parcialmente ou em grande parte relacionado ao *framework* alvo, porém represente um conceito de segurança mais amplo, será considerado como **Superconjunto**;
- Caso o controle recomendado pelo *framework* adotado como base da comparação seja parcialmente ou em grande parte relacionado ao *framework* alvo, porém represente um conceito de segurança menos amplo, será considerado como **Subconjunto**;
- Caso os controles recomendados em ambos os *frameworks* sejam relacionados, porém não possam ser classificados com segurança como superconjunto ou subconjunto, serão classificados como **interseção**;

- Caso o controle seja recomendado pelo *framework* adotado como base da comparação, porém não conte com nenhum controle equivalente ou semelhante no *framework* alvo, será considerado como **não mapeado no destino**;

- Caso o controle não seja recomendado pelo *framework* adotado como base da comparação, porém seja contemplado pelo *framework* alvo, será considerado como **não existente na origem**.

Os mapeamentos foram realizados sempre a partir do *framework CIS Software Supply Chain Security Guide*, por ser o que apresenta a maior quantidade de controles dentre os *frameworks* selecionados originalmente..

Os relacionamentos mapeados os entre os controles foram então representados em um painel elaborado na ferramenta *Microsoft Power BI*, que permite uma visualização clara e interativa dos relacionamentos, incluindo filtros dinâmicos que possibilitam observar, para cada pilar da cadeia de suprimentos de software, os controles associados a cada tipo de relacionamento identificado.

O painel completo encontra-se no Apêndice I.

3.4 ESTRUTURAÇÃO DO MODELO DE CONFORMIDADE BASEADA EM GRUPOS DE IMPLEMENTAÇÃO

Ainda tomando como base os trabalhos do Center for Internet Security, observou-se que os controles recomendados pelo CIS Controls são, em suas versões 7.1, 8.0 e 8.1, estruturados em três Grupos de Implementação, definidos de forma incremental e cumulativa, conforme detalhamento a seguir [125][126]:

- *Implementation Group 1* (IG1): Orientado a organizações de pequeno porte, com equipes limitadas para a gestão do ambiente de tecnologia da informação e de cibersegurança. Os controles de segurança constantes do IG1 objetivam impedir ataques gerais não direcionados;

- *Implementation Group 2* (IG2): Voltado a organizações que já contem com equipes responsáveis por gerenciar e proteger o ambiente de TI, que tipicamente tratem algum conjunto de informações sensíveis, e que suportem curtos períodos de indisponbilidade dos serviços;

- *Implementation Group 3* (IG3): Organizações de maior porte, que contem com especialistas em diferentes disciplinas relacinadas à segurança da informação e segurança cibernética, usualmente tratando de informações sujeitas a supervisão regulatória. Os controles sugeridos são voltados à mitigação de ataques direcionados à organização.

De forma semelhante, alguns dos *frameworks* adotados como insumo para a definição do modelo de conformidade indicam níveis incrementais para a adoção dos controles sugeridos:

- CNCF SSC *Best Practices*: Desconsidera recomendações para o nível mais baixo, e as distribui entre os níveis médio e alto relativos à garantia de segurança (*Assurance*) e ao nível de risco associado ao ambiente (*Risk*);

- OWASP SCVC: Níveis 1, 2 e 3;
- SLSA: Níveis 1, 2 e 3;
- Microsoft S2C2F: Níveis 1, 2, 3 e 4.

Considerando que o método utilizado pelo *Center for Internet Security* divide os controles recomendados pelo *CIS Controls* em 3 grupos de implementação, e que os *frameworks* que indicam níveis de implementação também utilizam, majoritariamente, uma divisão em três níveis, o modelo de conformidade aqui apresentado foi estruturado também em três grupos de implementação, conforme diretrizes detalhadas a seguir:

- Quando os *frameworks* indicarem o mesmo nível para o controle em análise, o grupo de implementação adotado será o de numeração equivalente (por exemplo, caso os *frameworks* indiquem o nível 1, o controle será atribuído ao grupo de implementação 1, e assim sucessivamente);
- Como o *framework* "CNCF"não considera controles de segurança para ambientes de baixo nível de *Assurance* e de Risco, os controles que considera como de nível "medium"serão entendidos como de nível 2, e serão atribuídos ao grupo de implementação 2, e os controles que considera como nível "high"serão entendidos como de nível 3, e serão atribuídos ao grupo de implementação 3;
- O *framework* Microsoft S2C2F indica alguns poucos controles, bastante especializados com relação ao pilar de dependências, como sendo de nível 4. Esses controles serão atribuídos ao grupo de implementação 3, agrupado sob o tema de controles especializados;
- Quando houver indicações de níveis conflitantes entre os *frameworks*, o grupo de implementação proposto dependerá do contexto e da complexidade aparente. Se o contexto e complexidade tiverem características mais sofisticadas, será adotado como grupo de implementação o maior dentre os níveis indicados pelos *frameworks*. Caso contrário, será adotado o menor;
- Quando o controle for subconjunto de outro para o qual há indicação de nível por parte de algum *framework*, este o grupo de implementação equivalente será adotado para o controle em questão;
- Quando o controle tiver interseção com outro controle para o qual houver indicação de nível por parte de algum framework, o grupo de implementação equivalente será adotado para o controle em questão;
- Para os controles que não possuírem indicação de nível de implantação em nenhum dos *frameworks*, os grupos de implementação adotados seguirão as seguintes diretrizes:
  - Controles associados à estruturação do ambiente (utilização de Sistemas de Controle de Versão de Software, Esteira de CI/CD e Repositório de Artefatos), e controles associados à gestão de usuários, serão atribuídos ao Grupo de Implementação 1, uma vez que são os controles básicos para que os pilares referentes à cadeia de suprimentos de software sejam implementados com um mínimo de padronização e segurança;

– Controles associados ao gerenciamento de vulnerabilidades do ambiente serão atribuídos ao Grupo de Implementação 2, uma vez que essa atividade é tipicamente executada como passo seguinte à implantação de ambientes operacionais;

– Controles associados a práticas específicas de cada um dos pilares, e à auditoria do ambiente, serão atribuídos ao Grupo de Implementação 3.

3.5 AGRUPAMENTO DE CONTROLES RECOMENDADOS POR TEMAS DE SEGURANÇA

Os controles propostos pelos diversos *frameworks* considerados foram agrupados em temas de segurança comuns aos quatro pilares da cadeia de suprimentos de software, de maneira a facilitar a compreensão do modelo, e, consequentemente o mapeamento do nível de conformidade da organização, ou o planejamento da implantação desses controles, conforme ilustrado pela Tabela 3.1:

Tabela 3.1: Temas de segurança referentes aos controles recomendados para a cadeia de suprimentos de software

Código-fonte	**Dependências**	**Compilação (Build)**	**Deploy/Distribuição**
Estruturação do ambiente	Estruturação do ambiente	Estruturação do ambiente	Estruturação do ambiente
Gerenciamento de vulnerabilidades do ambiente	Gerenciamento de vulnerabilidades do ambiente	Gerenciamento de vulnerabilidades do ambiente	Gerenciamento de vulnerabilidades do ambiente
Gerenciamento de usuários	Gerenciamento de usuários	Gerenciamento de usuários	Gerenciamento de usuários
Autenticação de usuários	Autenticação de usuários	Autenticação de usuários	Autenticação de usuários
Permissões de usuários	Permissões de usuários	Permissões de usuários	Permissões de usuários
Especificidades do ambiente - Políticas de contribuição de código - Políticas para Pull/Merge requests - Proteção de branches - Testes automatizados - Segurança dos projetos de software	Especificidades do ambiente - Política de uso de componentes - Inventário de componentes - Gestão de vulnerabilidades nos pacotes - Correção de vulnerabilidades	Especificidades do ambiente - Processo de build - Geração de SBOMs	Especificidades do ambiente - Segurança dos artefatos
Gestão da aplicativos de terceiros instalados na solução de versionamento de código	Gestão da aplicativos de terceiros instalados na solução de armazenamento de dependências	Gestão de aplicativos de terceiros integrados à plataforma de build	Gestão de aplicativos de terceiros integrados às plataformas de deploy e distribuição
Auditoria do ambiente	Auditoria do ambiente	Auditoria do ambiente	Auditoria do ambiente

Fonte: Elaborada pelo autor

É importante observar que os temas são idênticos para os quatro pilares da cadeia de suprimentos de software, apresentando variações somente quanto aos detalhamentos existentes sob o tema "Especificidade do ambiente".

3.6 MATERIALIZAÇÃO DO MODELO DE CONFORMIDADE

O Modelo de Conformidade sobre a Segurança da Cadeia de Suprimentos de Software proposto foi materializado na forma de uma planilha *Microsoft Excel*, organizada em abas equivalentes a cada pilar da cadeia de suprimentos de software, e, em cada aba ou pilar, tendo os controles agrupados pelos temas citados no item anterior, indentados de acordo com seus relacionamentos (equivalentes, superconjunto, subconjunto, interseções, ou não existência de relação direta entre os controles, porém sendo pertencentes ao mesmo tema), indicação do Grupo de Implementação de cada controle, e, por fim, indicação da refe-

rência relativa ao controle, composta pela menção ao *frarmework* de origem e respectiva página onde se encontra na documentação do *framework*.

Os controles podem ser filtrados com base no Grupos de Implementação, de forma a permitir sua visualização segmentada, ou por *framework* de origem, de forma a permitir uma visualização apenas dos controles recomendados por um determinado *framework*.

A planilha encontra-se no Apêndice II.

4 RESULTADOS

4.1 ANÁLISE QUANTITATIVA COMPARATIVA DOS *FRAMEWORKS* AVALIADOS

Um primeiro resultado é a quantificação das recomendações de segurança contidas em cada um dos *frameworks* e artigos acadêmicos avaliados, relacionados na Tabela 4.1 em ordem decrescente quanto ao seu total, ilustrando suas amplitudes e coberturas:

Tabela 4.1: Frameworks - Comparativo de recomendações

	OWASP Top 10	**CIS**	**SCVS**	**ESF**	**CNCF**	**Ladisa**	**S2C2F**	**SLSA**	**TOTAL por pilar da SSC**
Código Fonte	41	48	0	19	13	5	0	0	**126**
Dependências	46	12	22	15	9	8	25	0	**137**
Compilação (Build)	55	27	45	41	21	11	0	7	**207**
Deploy ou Distribuição	37	11	19	8	3	6	0	0	**84**
Artefatos	0	14	0	0	8	0	0	0	**22**
Operações	0	0	0	0	0	8	0	0	**8**
TOTAL por framework	**179**	**112**	**86**	**83**	**54**	**38**	**25**	**7**	

Fonte: Elaborada pelo autor

As recomendações associadas a artefatos e a operações foram redistribuídas, no modelo de conformidade final, aos pilares "dependências", "*build* (compilação)"ou "*deploy* ou distribuição".

Faz-se importante destacar que o *framework* SLSA atualmente traz recomendações voltadas apenas ao pilar referente à etapa de Compilação (*Build*) dos artefatos, porém tem previsão de abordar os três pilares restantes.

Da Tabela 4.1 pode-se depreender as seguintes observações:

- Todos os *frameworks*, à exceção do S2C2F e do SLSA, preocupam-se com os quatro pilares da cadeia de suprimentos de software;
- O *framework* S2C2F é o único que se concentra em apenas um dos pilares da cadeia de suprimentos de software, referente ao consumo de dependências externas aos projetos (componentes publicado por terceiros);
- O *framework* OWASP Top 10 CI/CD Security Risks é o que traz uma maior quantidade de recomendações de segurança;
- O *framework* CIS é o que traz maior ênfase às recomendações sobre o ambiente relacionado aos códigos-fonte, podendo ser considerado o mais completo com relação a esse pilar;
- Os pilares de Compilação (*Build*) e de Dependências são os que concentram a maior quantidade de recomendações de segurança na soma de todos os *frameworks*, sugerindo que são os que trazem maiores riscos à cadeia de suprimentos de software.

As diferenças entre as quantidades de controles recomendados pelos diferentes *frameworks* para cada

um dos pilares já começa a evidenciar o caráter de complementaridade entre eles, fortalecendo a ideia de elaboração de um modelo de conformidade que aborde todos os controles as relações existentes.

4.2 ANÁLISE COMPARATIVA DAS SEMELHANÇAS E DIFERENÇAS NAS RECOMENDAÇÕES DE SEGURANÇA DE CADA *FRAMEWORK*

Utilizando-se o método para a comparação de *frameworks* definido como parte da metodologia de pesquisa, realizou-se a avaliação de semelhanças e diferenças entre cada um dos controles ou recomendações.

A avaliação teve início antes da identificação do documento OWASP Top 10 CI/CD Security Risks, de forma que, no início dessa atividade, o *framework* do CIS era o que apresentava maior quantidade de recomendações ou controles de segurança para cadeias de suprimentos de software, e foi utilizado como base para as comparações, de forma tal que cada um de seus controles foi comparado com os controles semelhantes encontrados nos demais *frameworks*.

Esse trabalho de comparação foi realizado por meio de uma planilha Excel, parcialmente representada pela Figura 4.1, estruturada de forma a possibilitar o alinhamento dos controles semelhantes de todos os *frameworks* de forma visual, e assim, facilitar a identificação de controles equivalentes entre todos os frameworks.

A inclusão tardia do documento OWASP Top 10 CI/CD Security Risks no trabalho de comparação,não trouxe impactos negativos, justamente em função deste alinhamento visual dos controles semelhantes, o que permitiu seu correto aproveitamento para a posterior construção do modelo de conformidade.

CIS - Descrição	ESF - Descrição	CIS -> ESF - Relacionamento	CNCF - Descrição	Relacionamento CIS -> CNCF
Ensure each pipeline has a single responsibility		Não mapeado no destino		Não mapeado no destino
Ensure all aspects of the pipeline infrastructure and configuration are immutable	The build pipeline infrastructures should be completely locked down	Equivalente	Validate environments and dependencies before usage	Superconjunto
Ensure all aspects of the pipeline infrastructure and configuration are immutable	Subject build scripts and configuration files to the same code review process	Superconjunto		
Ensure all aspects of the pipeline infrastructure and configuration are immutable	Use version control for pipeline configurations	Superconjunto		
Ensure all aspects of the pipeline infrastructure and configuration are immutable				
Ensure the build environment is logged	Log all access to the build pipeline	Superconjunto	Record the Build Environment	Equivalente

Figura 4.1: Planilha de comparação entre os controles recomendados pelos frameworks

Fonte: Elaborada pelo autor

A planilha foi estruturada em quatro abas, cada uma correspondendo a um dos pilares da cadeia de suprimentos de software. A Figura 4.1 apresenta um recorte da aba referente ao ambiente de compilação, elaborada com base ons controles recomendados pelo *framework* do CIS, dispostos na primeira coluna.

A segunda coluna é dedicada aos controles do primeiro *framework* selecionado para comparação, o ESF, e a terceira coluna, destacada em vermelho, é dedicada à atribuição do relacionamento entre os controles.

A partir da quarta coluna repete-se a estrutura de uma coluna destinada aos controles do *framework* selecionado para comparação, e uma coluna subsequente destinada à atribuição do relacionamento.

Observa-se na Figura 4.1 que os controles semelhantes em todos os *frameworks* foram posicionados em uma mesma linha, com o objetivo de facilitar o agrupamento de controles no modelo de maturidade que seria elaborado, conforme linha destacada em azul.

As células destacadas em amarelo ilustram um mesmo controle do CIS repetido em quatro linhas consecutivas. Isso representa casos em que um único controle de um determinado *framework* tenha algum tipo de relacionamento com mais de um controle de um outro *framework*, ou mesmo de vários *frameworks*.

O controle do CIS foi relacionado a três controles do ESF. Enquanto o controle do ESF constante da segunda linha foi considerado equivalente ao do CIS, no caso dos controles constantes da terceira e quarta linhas o controle do CIS foi considerado superconjunto desses. A quarta ocorrência do controle do CIS possuía relacionamento com um controle de outro *framework*, não ilustrado na parte da planilha que foi recortada.

Esse procedimento de comparação foi realizado para os quatro pilares da cadeia de suprimentos de software, sempre tendo como *framework* base para comparação do CIS.

4.3 ABRANGÊNCIA DOS *FRAMEWORKS* SOBRE CADA UM DOS PILARES

Uma vez estabelecidos os relacionamentos entre as recomendações de segurança de cada um dos *frameworks*, agrupados de acordo com os pilares da cadeia de suprimentos de software, foi elaborado um painel visual construído na ferramenta Microsoft Power BI, de forma a melhor ilustrar a abrangência dos diversos *frameworks* sobre cada um dos pilares. O painel é ilustrado pela Figura 4.2, e é disponibilizado de forma completa no Apêndice I.

A área do painel destacada em vermelho exibe gráficos que indicam os percentuais dos tipos de relacionamentos entre os controles recomendados pelo *framework* adotado como base, o CIS, e os demais *frameworks* que abordam o pilar da cadeia de suprimentos de software em questão. A Figura 4.2também ilustra o pilar relativo ao ambiente de compilação.

A representação visual permite uma compreensão instantânea do nível de abrangência de um *framework* em relação aos demais. Neste caso específico, observa-se que 48,21% dos controles do CIS são superconjuntos dos controles do ESF, evidenciando que para quase a metade dos controles do CIS o ESF traz maior detalhamento.

A área destacada em amarelo disponibiliza uma visualização semelhante à da planilha utilizada para a realização do trabalho de comparação, permitindo a visualização das descrições dos controles que apresentem alguma relação entre si, em todos os *frameworks* que tratem sobre o pilar da cadeia de suprimentos de software em questão.

A área destacada em azul oferece a possibilidade de filtrar o painel com base em qualquer recomendação de segurança de qualquer dos *frameworks*, destacando esse controle nos gráficos apresentados, permitindo visualizar que tipo de relação ele possui com controles semelhantes em outros *frameworks*, bem

Figura 4.2: Painel comparativo entre os frameworks de segurança da cadeia de suprimentos de software

Fonte: Elaborada pelo autor

como o quadro descritivo na parte inferior do painel, permitindo a visualização das descrições dos controles semelhantes.

O painel é interativo, de forma que selecionando-se uma área de qualquer dos gráficos exibidos, os demais gráficos e o painel descritivo na parte inferior são atualizados dinamicamente.

4.3.1 Código fonte

Apenas quatro *frameworks* trazem recomendações de segurança relativas ao pilar referente a código fonte, a saber, o *CIS Software supply chain security* (CIS), o *Cloud Native Computing Foundation Software Supply Chain Best Practices* (CNCF), o artigo acadêmico de Ladisa et al (Ladisa) e o OWASP Top 10 CI/CD Security Risks (OW T10).

O *CIS Software supply chain security* guide contém a maior quantidade de recomendações para este caso, conforme Tabela 4.1.

O comparativo entre os controles de segurança recomendados pelos quatro *frameworks*, conforme método descrito na Seção 3.4, encontra-se ilustrado na Figura 4.3.

Na comparação entre os controles previstos nos *frameworks* CIS e CNCF, 50% das recomendações contidas no CIS são subconjuntos de recomendações contidas no CNCF, enquanto 36% das recomendações do CIS não se encontram mapeadas no CNCF, denotando que as recomendações do primeiro são mais detalhadas do que as recomendações do segundo.

Uma situação similar ocorre no comparativo entre o *framework* CIS e o artigo de Ladisa, onde 91,66%

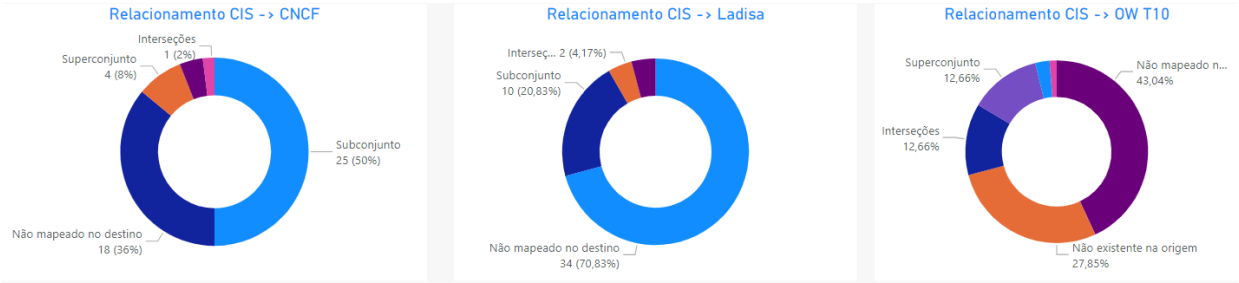


Figura 4.3: Relacionamentos entre as recomendações relativas a código-fonte

Fonte: Elaborada pelo autor

dos controles do CIS não são contemplados por Ladisa ou são um subconjunto dos mesmos.

Já no caso do comparativo entre o CIS e OW T10, 70,89% das recomendações encontram-se em apenas um dos *frameworks*, sendo que 43,04% dos controles do CIS não encontram mapeamento no OW T10, e 27,85% recaem sob o sentido oposto.

Tais percentuais sugerem que os *frameworks* CIS e OW T10 são os mais significativos com relação ao pilar sobre código fonte, apresentam grande quantidade de recomendações complementares entre si, e, assim, sobressaem-se como os mais relevantes para estruturação dos esforços para o fortalecimento do pilar relativos a códigos-fonte.

4.3.2 Dependências

O pilar relativo a dependências traz cenário mais equilibrado em termos de quantidade de recomendações contida em cada *framework*, conforme a Tabela 4.1.

Com relação a esse pilar os *frameworks* SCVS e S2C2F são os que apresentam maior quantidade de recomendações.

Entretanto, uma avaliação da Figura 4.4 nos permite observar que, com exceção do comparativo entre os *frameworks* SCVS e CNCF, em todos os demais comparativos percentuais próximos a 40% ou 50% são de recomendações que existem apenas em um dos *frameworks*, somadas aquelas não mapeadas no destino ou não existentes na origem: SCVS->CIS (45,16%), SCVS->S2C2F (45,94%), SCVS->Ladisa (44,82%) e SCVS-> (57,14%).

Esses números demonstram que os *frameworks* são complementares em relação ao pilar de dependências, cada um trazendo recomendações únicas para o tópico.

4.3.3 Compilação (Build)

O pilar relacionado à compilação (ou build) dos artefatos é o que concentra a maior quantidade de controles de segurança, tanto no conjunto dos *frameworks* quanto na análise individual de cada um. A Figura 4.5 detalha as observações.

A comparação entre os *frameworks* CIS e ESF apresenta um alto percentual de controles do CIS que são

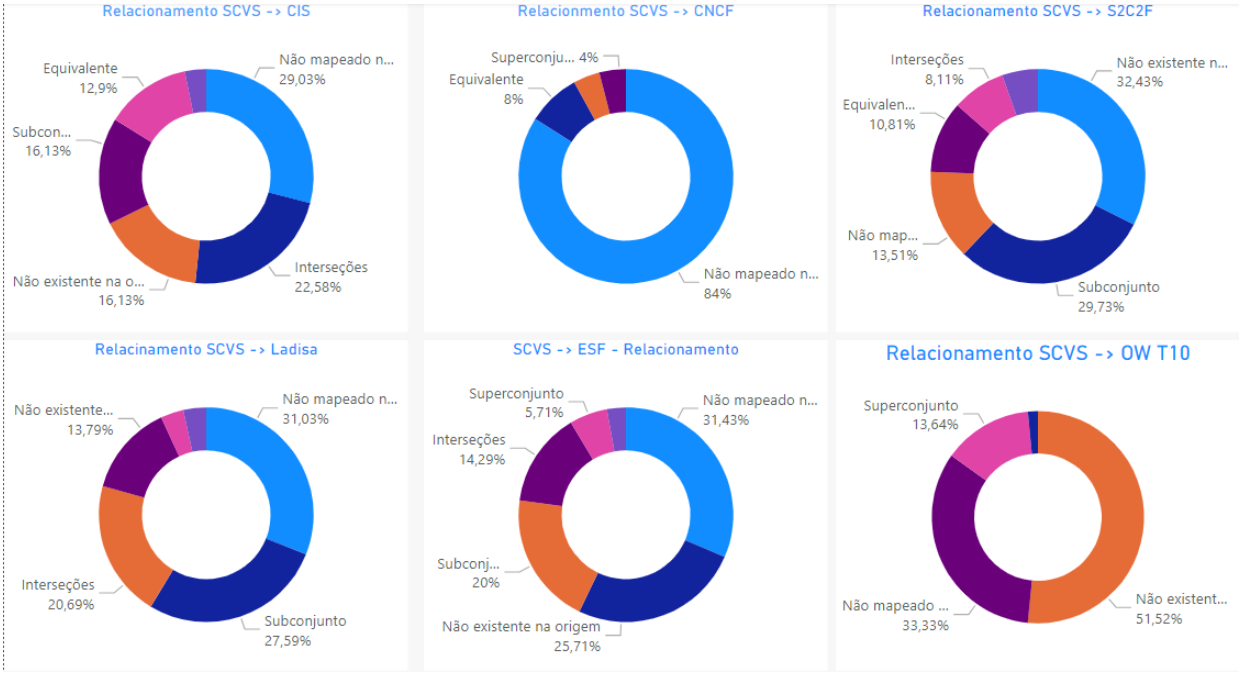


Figura 4.4: Relacionamentos entre as recomendações relativas a dependências

Fonte: Elaborada pelo autor

superconjunto do ESF (48,21%), denotando que o ESF traz um maior detalhamento sobre vários controles do CIS. Pouco mais de um quarto dos controles (26,79%) são equivalentes, interseções ou subconjunto, e o quartil restante é relativo a controles para os quais não há equivalência entre os *frameworks*.

Já os *frameworks* CIS e CNCF nos trazem o maior grau de equivalência entre os controles (35,14%). As interseções respondem por 18,92%, e 16,22% dos controles do CIS são superconjunto dos controles do CNCF, enquanto 27,03% das recomendações do CIS não encontram paralelo no CNCF. Tais números sugerem que o CIS seja um *framework* que aborda de forma mais completa a segurança do ambiente de compilação, contando com algum detalhamento mais aprofundado de parte de seus controles pelo CNCF.

No caso do SCVS, a maior parte dos controles do CIS (56,14%) é um superconjunto de seus controles, significando que o SCVS traz um detalhamento mais aprofundado para esses casos. Entretanto, há um percentual significativo de controles do CIS que não encontram paralelo no SCVS, equivalente a 22,81%, ou quase um quarto. Cerca de 14% dos controles são equivalentes ou apresentam interseções, compondo o conjunto de maior semelhança entre ambos os *frameworks*, e apenas 7%, ou quatro controles do CIS, são subconjuntos de controles do SCVS.

No relacionamento entre o CIS e o SLSA, 56,17% dos controles são subconjuntos, ou seja, os controles do CIS detalham melhor os controles sugeridos pelo SLSA, enquanto 6,9% apresentam interseções, e 37,93% não existem no SLSA, sendo portanto mais um caso em que o *framework* CIS traz boa quantidade de controles complementares ao *framework* sendo comparado.

Com relação ao trabalho acadêmico de Ladisa, 60% dos controles do CIS não são previstos, e 26,67% são equivalentes ou interseções, sugerindo que o trabalho não complemente significativamente o CIS.

Finalmente, o mapeamento entre os *frameworks* do CIS e do OW T10 evidencia que a maioria dos con-

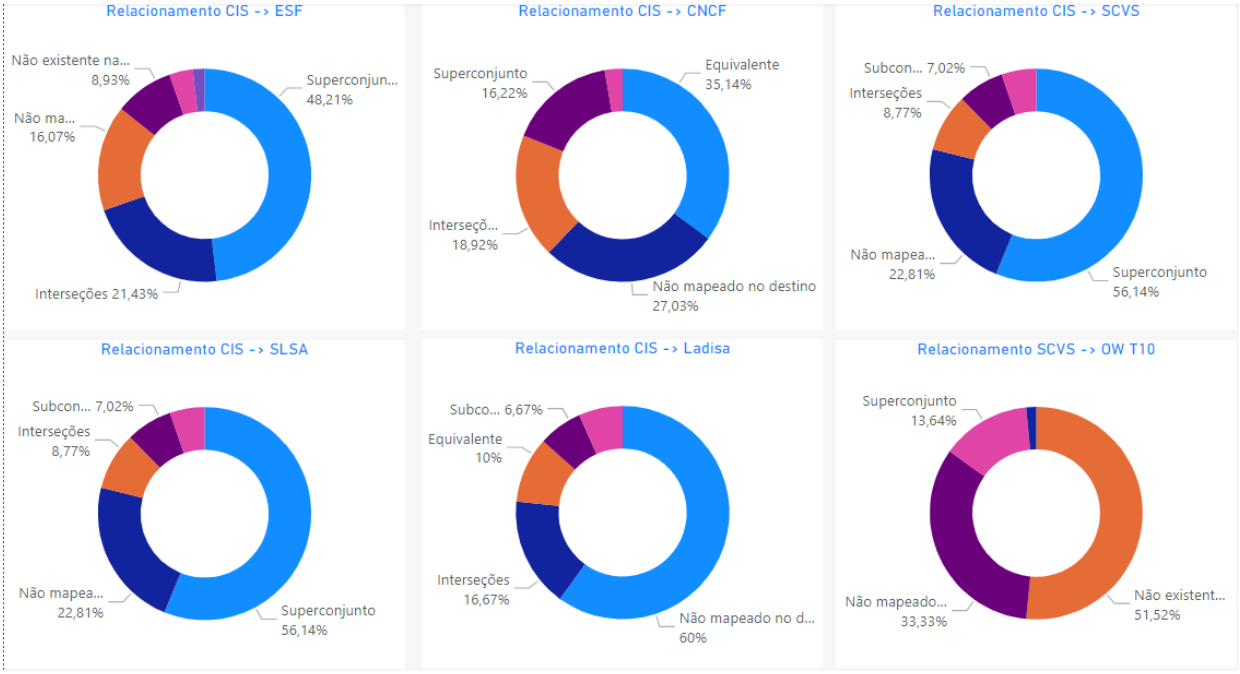


Figura 4.5: Relacionamentos entre as recomendações relativas a build (compilação)

Fonte: Elaborada pelo autor

troles do primeiro são superconjunto dos controles do segundo (56,34%), ou seja, são mais bem detalhados pelo OW T10, e 9,86% dos controles do OW T10 não são previstos pelo CIS. Muito embora 21,13% dos controles do CIS não sejam previstos pelo OW T10, os números evidenciam que este último contém uma quantidade significativa de controles adicionais ao primeiro.

4.3.4 *Deploy* e Distribuição

Por fim, chegou-se à comparação entre os controles de segurança recomendados para o ambiente de *deploy* ou distribuição, ilustrados na Figura 4.6.

Comparando-se o CIS ao CNCF, 61,54% dos controles não são recomendados pelo segundo, e 19,23% dos controles do CIS são subconjunto dos controles do CNCF, ou seja, oferecem um maior grau de detalhamento, o que nos permite concluir que o CIS seja um *framework* mais completo.

Observa-se que 61,54% dos controles do CIS não são mapeados por Ladisa, 15,38% apresentam interseções, e outros 15,38% dos controles propostos pelo CIS são subconjunto do controles propostos por Ladisa, ou seja, os detalham melhor, denotando que o CIS, também neste caso, é um *framework* bastante mais completo.

Temos então três mapeamentos que evidenciam *frameworks* com alto grau de complementaridade.

A comparação entre o CIS e o ESF evidencia que 56,61% não são mapeados no destino, e 16,13% não existem na origem, ou seja, um total de 72,74% de controles que não encontram paralelo entre ambos os *frameworks*.

O relacionamento entre os *frameworks* CIS e SCVS apresenta cenário semelhante, pois 48,65% dos

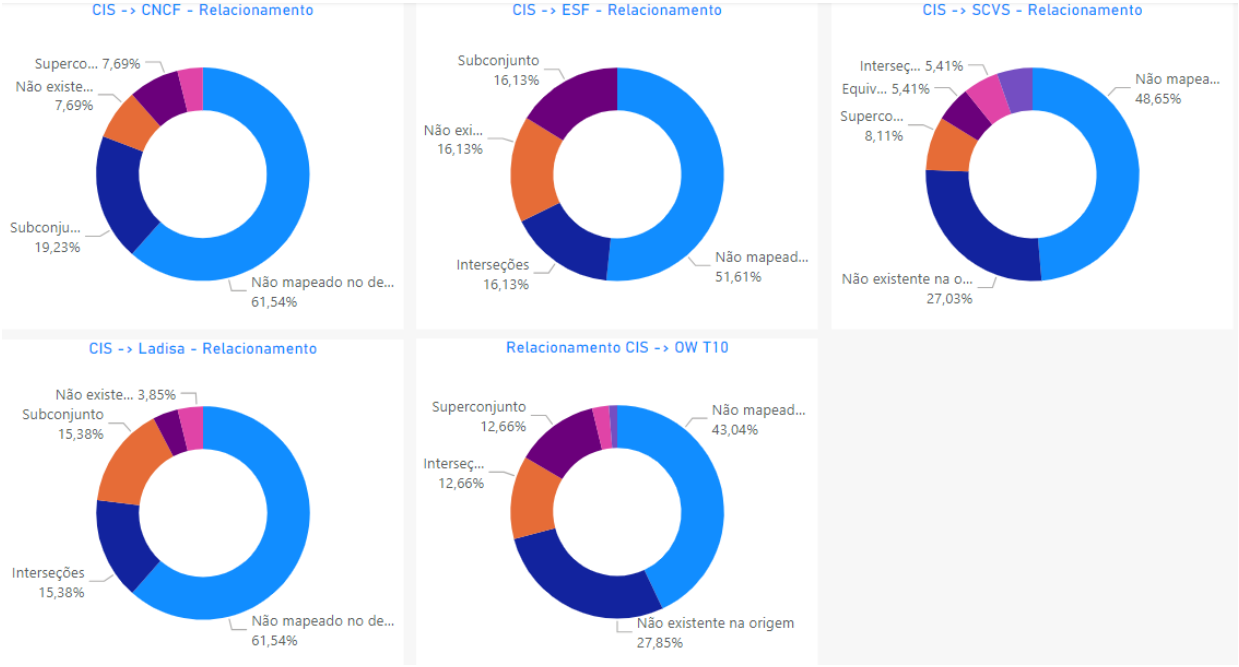


Figura 4.6: Relacionamentos entre as recomendações relativas a *deploy* e distribuição

Fonte: Elaborada pelo autor

controles recomendados pelo CIS não o são pelo SCVS, e 27,03% representam o caso inverso. Ou seja, aproximadamente 3/4 dos controles (75,68%) não apresentam relação entre si.

O mesmo cenário se observa na comparação entre os *frameworks* CIS e OW T10, onde 70,89% dos controles não apresentam qualquer relação entre si.

4.4 SOBRE O MODELO DE CONFORMIDADE

Os resultados dos comparativos mencionados na Seção 4.3 forneceram os insumos para a elaboração do modelo de conformidade.

A partir das relações que representam super/subconjuntos, equivalências, interseções, e ausência de relações entre os controles de segurança propostos pelos diferentes *frameworks*, elaborou-se o modelo de conformidade estruturado em três grupos de implementação, tendo como ponto de partida os itens que representam os principais superconjuntos, detalhando-os com os controles que sejam subconjuntos desses, e assim sucessivamente. Os controles que representem equivalências ou interseções entre esses foram associados aos respectivos grupos de implementação, e, por fim, os controles que não apresentem relações entre si foram atribuídos aos grupos de implementação por similaridade de complexidade de implementação.

O modelo foi materializado em uma planilha elaborada em *Microsoft Excel*, contendo uma aba para cada pilar da cadeia de suprimentos de software, ilustrado pela Figura 4.7, e disponibilizado de forma completa no Apêndice II.

A primeira coluna traz os controles de todos os *frameworks* que tratem o pilar em questão, agrupados

CONTROLES	EQUIVALÊNCIAS	NÍVEL DE MATURIDADE	FONTE	
			FRAMEWORK	PÁGINA
Estruturação do ambiente				
Ensure network access to the systems is aligned with the principle of least access		1	OW	29
Ensure Git access is limited based on IP addresses		1	CIS	22
Ensure two administrators are set for each repository		1	CIS	19
Ensure inactive repositories are reviewed and archived periodically		1	CIS	16
Ensure all public repositories contain a SECURITY.md file		1	CIS	14
Gerenciamento de vulnerabilidades do ambiente				
Maintain an inventory of systems and versions in use, including mapping of a designated owner for each system		2	OW	29
Continuously check for known vulnerabilities in these components		2	OW	29
If a security patch is available, update the vulnerable component. If not, consider removing the component / system, or reduce the potential impact of exploiting the vulnerability by restricting access to the system, or the system's ability to perform sensitive operations		2	OW	29
Establish a process to periodically review all system configurations for any setting that can have an effect on the security posture of the system, and ensure all settings are optimal		2	OW	29
Measures aimed at detecting configuration drifts (e.g. resources in cloud environments which aren't managed using a signed IAC template), potentially indicative of resources that were deployed by an untrusted source or process		2	OW	34
Gerenciamento de usuários de usuários				
User account management		1	LAD	7
Ensure new members are required to be invited using company-approved email	Prevent employees from using their personal email addresses, or any address which belongs to a domain not owned and managed by the organization, against the SCM, CI, or any other CI/CD platform	1	CIS, OW	19, 10
Continuously monitor for non-domain addresses across the different systems and remove non-compliant users		1	OW	10
Ensure an organization's identity is confirmed with a "Verified" badge		1	CIS	20
Ensure Source Code Management (SCM) email notifications are restricted to verified domains		1	CIS	21
Ensure inactive users are reviewed and removed periodically		1	CIS	17
Determine an acceptable period for disabling/removing stale accounts and disable/remove any identity which has surpassed the predetermined period of inactivity		1	OW	9
Refrain from allowing users to self-register to systems, and grant permission on an as-needed basis		1	OW	10

Figura 4.7: Modelo de Conformidade para Cadeias de Suprimentos de Software

Fonte: Elaborada pelo autor

de acordo com os temas definidos na metodologia.

A indentação representa no primeiro nível os controles que não tenham relação direta ou que tenham alguma interseção entre si, no segundo nível os controles que sejam subconjunto do controle exibido na linha imediatamente superior, e no terceiro nível os controles que sejam subconjunto daqueles posicionados no segundo nível.

A segunda coluna traz os controles que sejam equivalentes aos controles representados na primeira coluna, ou sejam, que tenham significado idêntico.

A terceira coluna traz o grupo de implementação sugerido para o controle, permitindo que o leitor identifique aqueles cuja implementação seja mais simples, atribuídos ao Grupo de Implementação 1, e assim possam ser adotados em primeiro lugar, e os de complexidade maior, de acordo com os Grupos de Implementação 2 e 3.

Por fim, as duas últimas colunas trazem a fonte do controle, indicando o *framework* de referência e o local exato na documentação do *framework* onde sua documentação se encontra.

Uma providência que se mostrou interessante foi o agrupamento dos controles nos temas de segurança Estruturação do ambiente, Gerenciamento de vulnerabilidades do ambiente, Gerenciamento de usuários, Autenticação de usuários, Permissões de usuários, Especificidades do ambiente, Gestão de aplicativos de terceiros instalados no ambiente, e Auditoria do ambiente.

Esse agrupamento, aliado aos grupos de implementação, facilita a compreensão do modelo, e contribui para uma melhor organização dos esforços a serem depreendidos pela organização para o alcance do nível de segurança desejado.

Uma organização que esteja inicialmente preocupada com a estruturação dos ambientes relativos aos pilares de sua cadeia de suprimentos, por exemplo, poderá se concentrar nos controles já agrupados para esse tema, tendo também a noção de complexidade da adoção desses controles a partir dos grupos de implementação.

A materialização do modelo em uma planilha *Excel* traz ainda a possibilidade de interação, especialmente quanto à filtragem pelos grupos de implementação ou pelo *framework* de origem dos controles.

A indicação exata da origem dos controles, composta pelo *framework* e pela página onde se encontram, possibilita um aprofundamento quanto compreensão do controle e seu contexto de aplicação, na medida em que permite um rápido acesso ao texto original do *framework* em questão.

Dessa forma, o modelo proposto se mostra um recurso útil para aqueles envolvidos com a implementação de segurança na cadeia de suprimentos de software, contribuindo para a compreensão e organização dos esforços necessários.

5 CONCLUSÃO

O presente trabalho teve como objetivo geral consolidar as melhores práticas de segurança em cadeias de suprimentos de software e propor um modelo que permita que organizações públicas e privadas avaliem seu estado atual e adotem as recomendações necessárias para alcançarem o nível de segurança desejado.

A importância do tema foi demonstrada a partir de informações sobre o aumento da incidência de ataques à cadeia de suprimentos de software, sobre o incremento da complexidade e alcance desse tipo de ataque, e inclusive sobre a tendência de exploração de oportunidades de ataques direcionados a alvos específicos por grupos de atacantes sofisticados, tais como estados-nação.

O material trazido no referencial teórico evidenciou que a questão vem efetivamente ganhando atenção da indústria e da academia recentemente, uma vez que os artigos acadêmicos mais representativos, bem como os *frameworks* voltados à segurança da cadeia de suprimentos de software, foram publicados do ano de 2020 em diante.

Para melhor estruturar o trabalho, partiu-se da conceituação de cadeias de suprimentos de uma forma geral, seguida pela conceituação de cadeias de suprimentos de software em específico, e da comparação entre ambas, mostrando a similaridade e as diferenças significativas entre os conceitos.

A partir dessa conceituação buscou-se alcançar o primeiro objetivo específico, *Organizar um conjunto de boas práticas de segurança relativa a cadeias de suprimentos de software a partir de seus principais frameworks*, tratou-se de apresentar os conjuntos de boas práticas relativos a desenvolvimento seguro, de gestão de riscos associada à cadeia de suprimentos de software, e da segurança da cadeia de suprimentos de software em si, representados por normas e padrões publicados por organismos internacionais, como a ISO e o NIST, e por *frameworks* publicados pelas principais instituições ligadas ao tema, tais como o Center for Internet Security (CIS), Cloud Native Computing Foundation (CNCF), Open Web Application Security Project (OWASP), Enduring Security Framework (ESF), e por grandes empresas voltadas ao segmento de TI em geral e à segurança cibernética em particular, tais como Cisco e Microsoft.

O segundo objetivo específico, *Analisar e avaliar as semelhanças e diferenças entre os diversos frameworks e melhores práticas de segurança relativas a cadeias de suprimentos de software*, foi alcançado a partir da utilização do método definido pelo Center for Internet Security para o mapeamento dos *frameworks* que publica, a exemplo do CIS Controls, sobre *frameworks* semelhantes publicados por outras organizações.

Do método definido pelo CIS utilizou-se ainda a estruturação da recomendação de controles em três grupos de implementação, definidos com base na capacidade relativa à cibersegurança das organizações interessadas na utilização de seus modelos. Assim, o modelo de conformidade aqui proposto também foi estruturado em três grupos de implementação, a partir dos níveis de complexidade de implementação sugeridos por alguns dos *frameworks* utilizados como base.

Por fim, o terceiro objetivo específico da pesquisa, *Estruturar um Modelo de Conformidade de Segurança para Cadeias de Suprimentos de Software baseado nas recomendações de segurança citadas no*

item anterior, que tinha o condão de atender ao objetivo principal, foi alcançado por meio da elaboração do modelo de conformidade proposto, que considerou todas as recomendações de segurança constantes dos *frameworks* utilizados como insumo, bem como as relações entre esses controles.

O alcance do terceiro objetivo especíico permitiu a confirmação de que a hipótese estabelecida era verdadeira: *"É possível estabelecer um conjunto de recomendações específicas, baseado na multiplicidade de fontes que propõem melhores práticas sobre o tema, que permita às organizações conhecerem o nível de segurança de sua cadeia de suprimento de softwares, e alcançarem níveis adequados de segurança na cadeia de suprimentos de software, ajustados às suas particularidades e necessidades?"*

O Modelo de Conformidade elaborado efetivamente estabeleceu esse conjunto de recomendações baseadas nos diversos *frameworks* que consolidam boas práticas sobre o tema, e o levantamento do atendimento às recomendações consolidadas efetivamente permite que as organizações conheçam seu nível de segurança corrente, definam seu nível de segurança desejado, e tracem planos de ação para alcançá-lo.

A utilização de *frameworks* de elaboração e publicação recentes representa a principal limitação da pesquisa, uma vez que são documentos ainda não consolidados ao longo do tempo, e que certamente sofrerão atualizações decorrentes do avanço acelerado dos esforços para o estabelecimento de padrões de segurança para as cadeias de suprimentos de software. O exemplo mais claro do fato de que os *frameworks* pertinentes ainda não estão consolidados é provavelmente o SLSA, que tem como objetivo propor recomendações de segurança para todos os pilares da cadeia de suprimentos de software, mas que, até o momento, conta com recomendações apenas para o pilar referente a compilações (*build*).

Em função dessa característica, é necessário que o modelo de conformidade proposto seja atualizado à medida em que as boas práticas de segurança para a cadeia de suprimentos de software evoluam, e sejam incorporadas aos principais *frameworks*.

Outra limitação foi o fato de que as atribuições dos relacionamentos entre os controles de segurança foram realizadas exclusivamente pelo autor. Por se tratar de atribuições que têm algum grau de subjetividade, um aprimoramento do trabalho seria a revisão dessas atribuições por parte de um grupo focal composto por especialistas segurança da cadeia de suprimentos de software, ou, ao menos, por especialistas em segurança cibernética e desenvolvimento seguro de software.

Iniciativas subsequentes visualizadas a partir da conclusão deste trabalho são a elaboração de uma ferramenta de autoavaliação baseada no modelo de conformidade proposto, que poderia ser disponibilizada publicamente, e, como subproduto dessa autoavaliação, a elaboração de comparativos anonimizados de níveis de conformidade alcançados pelas organizações a partir de segmentações possíveis, tais como porte da organização, setor de atuação, esfera de atuação (federal, estadual ou municipal), e outras segmentações possíveis e pertinentes.

Outra providência seria o estabelecimento de um grupo focal com especialistas em desenvolvimento seguro e em cadeia de suprimentos de software com o objetivo de atribuir valores de custo-benefício com relação à implementação de cada controle, o que possibilitaria melhores critérios de seleção quanto aos controles a serem efetivamente implantados. O trabalho de Ladisa et al. teve essa preocupação, porém, como propõe poucos controles de segurança, não pode ser utilizado para compor o modelo ora proposto.

Por fim, identifica-se ainda a possibilidade de elaboração de um modelo de maturidade voltado à defi-

nição de processos de trabalho relacionados à segurança da cadeia de suprimentos de software, nos moldes dos modelos definidos pelo CMMI - *Capability Maturity Model Integration*, comumente utilizado com relação a processos de desenvolvimento de software, ou pela família de normas ISO 33.000 (relativa a avaliações de processos), possivelmente utilizando como insumo os modelos de maturidade, as normas, padrões e *frameworks*, descrevendo as práticas que devam compor o processo de gestão da segurança de cadeias de suprimentos de software, seus objetivos, e outros aspectos formais conforme sugeridos pelos principais padrões de modelos de maturidade.

REFERÊNCIAS BIBLIOGRÁFICAS

1 CISA, NSA, FBI, ACSC, CCCS, CERT NZ, NCSC-NZ, NCSC-UK, BSI, NCSC-NL, NCSC-NO, NUKIB, KISA, NISC-JP, JPCERTC-CC, CSA, CSIRT AMERICAS. *Security by Design: Shifting the balance of cybersecurity risk*. CISA, 2023. Disponível em: <https://www.cisa.gov/sites/default/files/2023-10/Shifting-the-Balance-of-Cybersecurity-Risk-Principles-and-Approaches-for-Secure-by-Design-Software.pdf>. Acesso em: 27/01/2025.

2 ALLEN, B.; EDMUNDSON, C. *SANS 2023 DevSecOps Survey*. SANS Institute, 2023. Disponível em: <https://www.sans.org/white-papers/2023-survey-event-devsecops/>. Acesso em: 27/01/2025.

3 SONATYPE. *9th Annual State of the Software Supply Chain - Update*. Sonatype Inc., 2023. Disponível em: <https://www.sonatype.com/hubfs/SSC/2023%20Sonatype-%209th%20Annual%20State%20of%20the%20Software%20Supply%20Chain-%20Update.pdf>. Acesso em: 27/01/2025.

4 APACHE. *Maven Repository*. 2024. Disponível em: <https://mvnrepository.com/>. Acesso em: 2025/01/25.

5 NPM. *NPM*. 2024. Disponível em: <https://www.npmjs.com/>. Acesso em: 27/01/2025.

6 PYTHON SOFTWARE FOUNDATION. *Pipy*. 2024. Disponível em: <https://pypi.org/>. Acesso em: 01/04/2024.

7 MICROSOFT. *Nugget Gallery*. 2024. Disponível em: <https://www.nuget.org/>. Acesso em: 04/04/2024.

8 GITHUB. *Octoverse*. 2024. Disponível em: <https://github.blog/news-insights/octoverse/octoverse-2024/>. Acesso em: 27/01/2025.

9 BALS, F. *2024 Open Source Security and Risk Analysis Report*. Sonatype, 2024. Disponível em: <https://www.blackduck.com/resources/analyst-reports/open-source-security-risk-analysis.html>. Acesso em: 27/01/2025.

10 LELLA, I.; THEOCHARIDOU, M.; MALASTRAS, A. *Threat Landscape for Supply Chain Attacks — ENISA*. [S.l.]: ENISA, 2021. ISBN 978-92-9204-509-8.

11 BUEERMANN, G.; ROHRS, M. *Global Cybersecurity Outlook 2024*. World Economic Forum, 2024. Disponível em: <https://www.weforum.org/publications/global-cybersecurity-outlook-2024/>. Acesso em: 27/01/2025.

12 ENDURING SECURITY FRAMEWORK. *Securing the Software Supply Chain - Recommended Practices Guide for Developers*. 2022. Disponível em: <https://media.defense.gov/2022/Sep/01/2003068942/-1/-1/0/esf_securing_the_software_supply_chain_developers.pdf>. Acesso em: 17/09/2023.

13 HUGHES, C.; TURNER, T. *Software Transparency*. [S.l.]: John Wiley and Sons, Inc., 2023. ISBN 978-1-394-15850-8.

14 WETTER, J.; RINGLAND, N. *Understanding the Impact of Apache Log4j Vulnerability*. Google, 2021. Disponível em: <https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html>. Acesso em: 17/09/2023.

15 PEISERT, S.; SCHNEIER, B.; OKHRAVI, H.; MASSACCI, F.; BENZEL, T.; LANDWEHR, C.; MANNAN, M.; MIRKOVIC, J.; PRAKASH, A.; MICHAEL, J. B. Perspectives on the solarwinds incident. *IEEE Security and Privacy*, Institute of Electrical and Electronics Engineers Inc., v. 19, p. 7–13, 3 2021. ISSN 15584046.

16 RED HAT INC. *CVE-2024-3094 - Xz: malicious code in distributed source*. 2024. Disponível em: <https://www.cve.org/CVERecord?id=CVE-2024-3094>. Acesso em: 04/04/2024.

17 MATTIOLI, R.; MALATRAS, A.; HUNTER, E. N.; PENSO, M. G. B.; BERTRAM, D.; NEUBERT, I. *Identifying emerging cybersecurity threats and challenges for 2030.* [S.l.]: ENISA, 2023. ISBN 978-92-9204-634-7.

18 CYBEREDGE. *2024 Cyberthreat Defense Report*. CyberEdge Group, LLC, 2024. Disponível em: <https://cyberedgegroup.com/cdr/>. Acesso em: 27/01/2025.

19 THE WHITEHOUSE. *Executive Order on Improving the Nation's Cybersecurity*. 2021. Disponível em: <https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/>. Acesso em: 06/12/2023.

20 THE WHITEHOUSE. *National Cybersecurity Strategy*. 2023. Disponível em: <https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf>. Acesso em: 01/04/2024.

21 PETERS, G. C. *Securing Open Source Software Act of 2023*. Senate, 2023. Disponível em: <https://www.congress.gov/bill/118th-congress/senate-bill/917/all-actions?s=1&r=1&q=%7B%22search%22%3A%5B%22%5C%22open+source%5C%22%22%5D%7D>. Acesso em: 29/04/2024.

22 BRASIL. *Estratégia Nacional de Segurança Cibernética*. Presidência da República, 2020. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2020/decreto/d10222.htm>. Acesso em: 24/11/2023.

23 BRASIL. *Institui a Estratégia Nacional de Segurança Cibernética do Poder Judiciário (ENSEC-PJ)*. Conselho Nacional de Justiça, 2021. Disponível em: <https://atos.cnj.jus.br/files/original12260820210924614dc3e072cca.pdf>. Acesso em: 29/04/2024.

24 BRASIL. *Cinco controles de segurança cibernética para ontem*. Tribunal de Contas da União, 2022. Disponível em: <https://portal.tcu.gov.br/publicacoes-institucionais/cartilha-manual-ou-tutorial/5-controles-de-seguranca-cibernetica>. Acesso em: 27/01/2025.

25 OPEN SOURCE SECURITY FOUNDATION. *SLSA - Supply-chain Levels for Software Artifacts*. 2023. SLSA is a specification for describing and incrementally improving supply chain security, established by industry consensus. Disponível em: <https://slsa.dev/>. Acesso em: 20/06/2023.

26 BEAMON, B. M. Supply chain design and analysis:: Models and methods. *International Journal of Production Economics*, Elsevier, v. 55, p. 281–294, 8 1998. ISSN 0925-5273.

27 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 27036 - Cybersecurity - Supplier relationships - Part 1: Overview and concepts*. 2021.

28 BALLOU, R. H. *Gerenciamento da Cadeia de Suprimentos/Logística Empresarial*. 5th. ed. [S.l.]: Bookman, 2006. ISBN 9788536305912.

29 CLOUD NATIVE COMPUTING FOUNDATION. *Software Supply Chain Best Practices*. 2021. Disponível em: <https://project.linuxfoundation.org/hubfs/CNCF_SSCP_v1.pdf>. Acesso em: 17/09/2023.

30 GEER, D.; TOZER, B.; MEYERS, J. S. For good measure - counting broken links: A quant's view of software supply chain security. *;login:*, v. 45, 2020. Disponível em: <www.usenix.org>.

31 OHM, M.; PLATE, H.; SYKOSCH, A.; MEIER, M. Backstabber's knife collection: A review of open source software supply chain attacks. 2020. Disponível em: <https://arxiv.org/pdf/2005.09535.pdf>.

32 LADISA, P.; PONTA, S. E.; SABETTA, A.; MARTINEZ, M.; BARAIS, O. Journey to the center of software supply chain attacks. *IEEE Security & Privacy*, v. 21, 2023. Disponível em: <https://ieeexplore.ieee.org/document/10224821>.

33 GELB, Y.; NACHSHON, G.; ZORENSTEIN, T. *How one country is impacting supply chains*. Checkmarx, 2023. Disponível em: <https://checkmarx.com/blog/how-one-country-is-impacting-supply-chains/>. Acesso em: 05/04/2024.

34 JAKKAL, V. *How nation-state attackers like NOBELIUM are changing cybersecurity*. Microsoft Security, 2021. Disponível em: <https://www.microsoft.com/en-us/security/blog/2021/09/28/how-nation-state-attackers-like-nobelium-are-changing-cybersecurity/>. Acesso em: 24/11/2024.

35 PEISERT, S.; SCHNEIER, B.; OKHRAVI, H.; MASSACCI, F.; BENZEL, T.; LANDWEHR, C.; MANNAN, M.; MIRKOVIC, J.; PRAKASH, A.; MICHAEL, J. B. Perspectives on the solarwinds incident. *IEEE Security and Privacy*, Institute of Electrical and Electronics Engineers Inc., v. 19, p. 7–13, 3 2021. ISSN 15584046.

36 ZETTER, K. *SolarWinds: The Untold Story of the Boldest Supply-Chain Hack*. Wired, 2023. Disponível em: <https://www.wired.com/story/the-untold-story-of-solarwinds-the-boldest-supply-chain-hack-ever/>. Acesso em: 10/06/2023.

37 LORENC, D. *Software supply chain security is broader than SolarWinds and Log4J | TechCrunch*. 2022. Disponível em: <https://techcrunch.com/2022/11/29/software-supply-chain-security-is-broader-than-solarwinds-and-log4j/>. Acesso em: 10/06/2023.

38 OLADIMEJI, S.; KERNER, S. M. Solarwinds hack explained: Everything you need to know. *TechTarget*, 11 2023. Disponível em: <https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know>. Acesso em: 24/11/2024.

39 REUTERS. *SolarWinds hack was 'largest and most sophisticated attack' ever: Microsoft president*. Reuters, 2021. Disponível em: <https://www.reuters.com/article/technology/solarwinds-hack-was-largest-and-most-sophisticated-attack-ever-microsoft-pres-idUSKBN2AF03Q/>. Acesso em: 24/11/2024.

40 FIREEYE. *Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor*. Google, 2020. Disponível em: <https://cloud.google.com/blog/topics/threat-intelligence/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor>. Acesso em: 24/11/2024.

41 LAMBERT, J. *The hunt for NOBELIUM, the most sophisticated nation-state attack in history*. Microsoft Security, 2021. Disponível em: <https://www.microsoft.com/en-us/security/blog/2021/11/10/the-hunt-for-nobelium-the-most-sophisticated-nation-state-attack-in-history/>. Acesso em: 24/11/2024.

42 GOODWIN, C. *Behind the unprecedented effort to protect customers against the NOBELIUM nation-state attack*. Microsoft Security, 2021. Disponível em: <https://www.microsoft.com/en-us/security/blog/2021/12/02/behind-the-unprecedented-effort-to-protect-customers-against-the-nobelium-nation-state-attack/>. Acesso em: 24/11/2024.

43 LEFFERTS, R. *A report on NOBELIUM's unprecedented nation-state attack*. Microsoft Security, 2021. Disponível em: <https://www.microsoft.com/en-us/security/blog/2021/12/15/a-report-on-nobeliums-unprecedented-nation-state-attack/>. Acesso em: 24/11/2024.

44 APACHE. *Apache Log4j*. Apache.org, 2024. Disponível em: <https://logging.apache.org/log4j/2.x/index.html>. Acesso em: 24/11/2024.

45 MITRE. *CVE-2021-44228*. Mitre.org, 2021. Disponível em: <https://www.cve.org/CVERecord?id=CVE-2021-44228>. Acesso em: 24/11/2024.

46 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *CVE-2021-44228 Detail*. nvd.nist.org, 2021. Disponível em: <https://nvd.nist.gov/vuln/detail/CVE-2021-44228>. Acesso em: 24/11/2024.

47 FIRST. *Common Vulnerability Scoring System SIG*. first.org, 2025. Disponível em: <https://www.first.org/cvss/>. Acesso em: 27/01/2025.

48 IBM. *What is the Log4j vulnerability?* IBM. Disponível em: <https://www.ibm.com/topics/log4j>. Acesso em: 24/11/2024.

49 CISA. *Apache Log4j Vulnerability Guidance*. CISA, 2022. Disponível em: <https://www.cisa.gov/news-events/news/apache-log4j-vulnerability-guidance>. Acesso em: 24/11/2024.

50 WETTER, J.; RINGLAND, N. *Understanding the Impact of Apache Log4j Vulnerability*. Google, 2021. Disponível em: <https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html>. Acesso em: 17/09/2023.

51 KASPERSKY. *Update malicioso infecta usuários de notebooks Asus*. Kaspersky, 2019. Disponível em: <https://www.kaspersky.com.br/blog/shadowhammer-asus-malware-update/11552/>. Acesso em: 30/11/2024.

52 KASPERSKY. *ShadowHammer: uma operação em larga escala*. Kspersky, 2019. Disponível em: <https://www.kaspersky.com.br/blog/details-shadow-hammer/11697/>. Acesso em: 30/11/2024.

53 KASPERSKY. *Operation ShadowHammer*. Kaspersky, 2019. Disponível em: <https://securelist.com/operation-shadowhammer/89992/>. Acesso em: 30/11/2024.

54 II, T. H. *Compromised npm Package: event-stream*. Medium, 2018. Disponível em: <https://medium.com/intrinsic-blog/compromised-npm-package-event-stream-d47d08605502>. Acesso em: 24/11/2024.

55 CHEREPANOV, A. *Analysis of TeleBots' cunning backdoor*. We Live Security, 2017. Disponível em: <https://www.welivesecurity.com/2017/07/04/analysis-of-telebots-cunning-backdoor/>. Acesso em: 24/11/2024.

56 BRUMAGHIN, E.; MERCER, W.; WILLIAMS, C. *CCleanup: A Vast Number of Machines at Risk*. Talos, 2017. Disponível em: <https://blog.talosintelligence.com/avast-distributes-malware/>. Acesso em: 24/11/2024.

57 KHANDELWAL, S. *CCleaner Attack Timeline - Here's How Hackers Infected 2.3 Million PCs*. The Hacker News, 2018. Disponível em: <https://thehackernews.com/2018/04/ccleaner-malware-attack.html>. Acesso em: 24/11/2024.

58 CODECOV. *Bash Uploader Security Update*. Codecov, 2021. Disponível em: <https://about.codecov.io/security-update/>. Acesso em: 24/11/2024.

59 JACKSON, M. *Codecov supply chain breach - explained step by step*. Gitguardian, 2021. Disponível em: <https://blog.gitguardian.com/codecov-supply-chain-breach/>.

60 NGUYEN-DUY, J. *New Supply Chain Ransomware Attack Targets Kaseya Platform*. Fortinet, 2021. Disponível em: <https://www.fortinet.com/blog/threat-research/new-supply-chain-ransomware-attack-targets-kaseya-platform>. Acesso em: 25/11/2024.

61 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NVD - CVE-2021-30116*. nvd.nist.org, 2024. Disponível em: <https://nvd.nist.gov/vuln/detail/CVE-2021-30116>. Acesso em: 26/01/2025.

62 NATIONAL COUNTERINTELLIGENCE AND SECURITY CENTER. *Kaseya VSA Supply Chain Ransomware Attack*. NCSC, 2021. Disponível em: <https://www.dni.gov/files/NCSC/documents/SafeguardingOurFuture/Kaseya%20VSA%20Supply%20Chain%20Ransomware%20Attack.pdf>. Acesso em: 25/11/2024.

63 AKAMAI SECURITY INTELLIGENCE GROUP. *XZ Utils Backdoor — Everything You Need to Know, and What You Can Do*. Akamai, 2024. Disponível em: <https://www.akamai.com/blog/security-research/critical-linux-backdoor-xz-utils-discovered-what-to-know>. Acesso em: 25/11/2024.

64 FREUND, A. *Backdoor in upstream xz/liblzma leading to ssh server compromise*. Openwall, 2024. Disponível em: <https://www.openwall.com/lists/oss-security/2024/03/29/4>. Acesso em: 25/11/2024.

65 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *About ISO*. ISO, 2024. Disponível em: <https://www.iso.org/about>. Acesso em: 08/12/2024.

66 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 27034 - Information technology - Security techniques - Application security - Part 1: Overview and concepts*. 2011. Disponível em: <www.iso.org>.

67 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). *ABNT NBR ISO/IEC 27002:2022 - Segurança da informação, segurança cibernética e proteção à privacidade - Controles de segurança da informação*. 2022.

68 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *ABNT NBR ISO 27005 - Segurança da informação, segurança cibernética e proteção à privacidade - Orientações para gestão de riscos de segurança da informação*. 2023.

69 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 29148 - Systems and software engineering — Life cycle processes — Requirements engineering*. 2018.

70 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *About NIST*. NIST, 2024. Disponível em: <https://www.nist.gov/about-nist>. Acesso em: 08/04/2024.

71 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST Glossary*. NIST, 2024. Disponível em: <https://csrc.nist.gov/glossary/term/nist_special_publication#:~:text=Specifically%2C%20the%20SP%20800%2Dseries,%2C%20government%2C%20and%20academic%20organizations.&text=A%20type%20of%20publication%20issued%20by%20NIST.,-Specifically%2C%20the%20Special>. Acesso em: 08/04/2024.

72 SOUPPAYA, M.; SCARFONE, K.; DODSON, D. *Secure Software Development Framework (SSDF) version 1.1*. NIST, 2022. Disponível em: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf>. Acesso em: 27/01/2025.

73 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Software Supply Chain Security Guidance Under Executive Order (EO) 14028 Section 4e*. NIST, 2022. Disponível em: <https://csrc.nist.gov/pubs/other/2022/02/04/software-supply-chain-security-guidance-eo-14028-s/final>. Acesso em: 27/01/2025.

74 OWASP. *OWASP SAMM*. OWASP, 2023. Disponível em: <https://owasp.org/www-project-samm/>. Acesso em: 13/04/2024.

75 BOOTE, J.; ERLIKHMAN, E.; HUTCHISON, B.; LYMAN, M.; MIGUES, S. *BSIMM Report 2023*. Synopsys, 2023. Disponível em: <https://www.blackduck.com/resources/analyst-reports/bsimm.html>. Acesso em: 24/04/2024.

76 MICROSOFT. *Microsoft Security Development Lifecycle*. Microsoft, 2024. Disponível em: <https://www.microsoft.com/en-us/securityengineering/sdl>. Acesso em: 14/04/2024.

77 BISHT, P.; HEIM, M.; IFLAND, M.; SCOVETTA, M.; SKINNER, T. *Managing Security Risks Inherent in the Use of Third-party Components*. Safecode, 2017. Disponível em: <https://safecode.org/wp-content/uploads/2017/05/SAFECode_TPC_Whitepaper.pdf>. Acesso em: 14/04/2024.

78 DIGILO, A.; WANG, J. *Secure Supply Chain Consumption Framework (S2C2F) Simplified Requirements*. [S.l.]: Microsoft, 2023. Acesso em: 01/08/2023.

79 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 27036 - Cybersecurity - Supplier relationships - Part 2 - Requirements*. 2022.

80 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 27036 - Cybersecurity - Supplier relationships - Part 3 - Guidelines for hardware, software and services supply chain security*. 2023.

81 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 27036 - Information technology - Security techniques - Information security for supplier relationships - Part 4: Guidelines for security of cloud services*. 2016. Disponível em: <www.iso.org>.

82 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 15288 - Systems and software engineering - System life cycle processes*. 2023.

83 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). *ABNT NBR ISO/IEC/IEEE 12207 - Engenharia de sistemas e software - Processos de ciclo de vida de software*. 2021.

84 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). *ABNT NBR ISO/IEC 27002:2022 - Segurança da informação, segurança cibernética e proteção à privacidade - Sistemas de gestão da segurança da informação - Requisitos*. 2022.

85 BOYENS, J.; SMITH, A.; BARTOL, N.; WINKLER, K.; HOLBROOK, A.; FALLON, M. *NIST SP 800-161r1 - Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*. NIST, 2022. Disponível em: <https://doi.org/10.6028/NIST.SP.800-161r1>. Acesso em: 25/09/2023.

86 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST SP 800-39 - Managing Information Security Risk Organization, Mission, and Information System View*. NIST, 2011. Disponível em: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-39.pdf>. Acesso em: 04/03/2024.

87 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST SP 800-37 - Risk management framework for information systems and organizations*. NIST, 2018. Disponível em: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf>. Acesso em: 04/03/2024.

88 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *The NIST Cybersecurity Framework (CSF) 2.0*. NIST, 2024. Disponível em: <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf>. Acesso em: 03/04/2024.

89 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST SP 800-53 Rev. 5 - Security and Privacy Controls for Information Systems and Organizations*. NIST, 2020. Disponível em: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf>. Acesso em: 03/04/2024.

90 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). *ABNT NBR ISO 31000 - Gestão de riscos - Diretrizes*. 2018. Disponível em: <www.abnt.org.br>.

91 BRASIL. *Os quatro pilares da segurança da informação – Confidencialidade, Disponibilidade, Integridade e Autenticidade*. Ministério da Ciência, Tecnologia e Inovações, 2024. Disponível em: <https://www.gov.br/lncc/pt-br/centrais-de-conteudo/campanhas-de-conscientizacao/gestao-de-seguranca-da-informacao/os-quatro-pilares-da-seguranca-da-informacao-2013-confidencialidade-disponibilidade-integridade-e-autenticidade>. Acesso em: 01/12/2024.

92 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). *ABNT NBR IEC 31010 - Gestão de Riscos - Técnicas para o processo de avaliação de riscos*. 2021. Disponível em: <www.abnt.org.br>.

93 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST SP 800-30 - Guide for conducting risk assessments*. NIST, 2012. Disponível em: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf>. Acesso em: 03/04/2024.

94 SPRINGETT, S.; RUSSO, D.; FICK, G.; HERZ, J. C.; SCOTT, J.; SYMONS, M.; NALLAPAREDDY, P.; GARCIA, B. *OWASP Software Component Verification Standard v1.0*. OWASP, 2020. Disponível em: <https://owasp.org/www-project-software-component-verification-standard/>. Acesso em: 27/01/2025.

95 CENTER FOR INTERNET SECURITY. *Software Supply Chain Security Guide*. CIS, 2022. Disponível em: <https://www.cisecurity.org/insights/white-papers/cis-software-supply-chain-security-guide>. Acesso em: 04/11/2024.

96 ENDURING SECURITY FRAMEWORK. *Securing the Software Supply Chain - Recommended Practices Guide for Customers*. ESF, 2022. Disponível em: <https://media.defense.gov/2022/Nov/17/2003116445/-1/-1/0/esf_securing_the_software_supply_chain_customer.pdf>. Acesso em: 17/09/2023.

97 ENDURING SECURITY FRAMEWORK. *Securing the Software Supply Chain - Recommended Practices Guide for Suppliers*. ESF, 2022. Disponível em: <https://media.defense.gov/2022/Oct/31/2003105368/-1/-1/0/securing_the_software_supply_chain_suppliers.pdf>. Acesso em: 17/09/2023.

98 KRIVELEVICH, D.; GIL, O. *OWASP Top 10 CI/CD Security Risks*. OWASP, 2022. Disponível em: <https://github.com/OWASP/www-project-top-10-ci-cd-security-risks/blob/main/index.md>. Acesso em: 10/06/2023.

99 LADISA, P.; PLATE, H.; MARTINEZ, M.; BARAIS, O. Taxonomy of attacks on open-source software supply chains. In: . IEEE Computer Society, 2022. Disponível em: <https://www.computer.org/csdl/proceedings-article/sp/2023/933600a167/1OXGW6SQr4s>. Acesso em: 12/06/2023.

100 LADISA, P.; PONTA, S. E.; SABETTA, A.; MARTINEZ, M.; BARAIS, O. *Risk Explorer for Software Supply Chains*. SAP, 2023. Disponível em: <https://sap.github.io/risk-explorer-for-software-supply-chains/#/attacktree>. Acesso em: 2025/01/27.

101 MELARA, M. S.; BOWMAN, M. What is software supply chain security? 2022. Disponível em: <https://arxiv.org/pdf/2209.04006.pdf>. Acesso em: 17/09/2023.

102 MITRE. *Mitre - Our Story*. Mitre, 2024. Disponível em: <https://www.mitre.org/who-we-are/our-story>. Acesso em: 25/04/2024.

103 MITRE. *Mitre Att&ck*. Mitre, 2024. Disponível em: <https://attack.mitre.org/resources/#what-is-attack>. Acesso em: 25/04/2024.

104 ZIV, N.; ARZI, L.; PAZ, E.; CROSS, D.; SUEZAWA, H.; PENSO, N.; SIVAN, S.; SAHNI, D.; KOVALSKY, M.; WANG, C.; FEINTUCH, R.; LAVIE, H. H.; ATIAS, R.; EVRON, G. *Open Software Supply Chain Attack Reference (OSC&R)*. pbom.dev, 2023. Disponível em: <https://pbom.dev/>. Acesso em: 29/09/2023.

105 LINKSKENS, A. *How to measure the maturity of your software supply chain*. Sonatype, 2023. Disponível em: <https://www.sonatype.com/blog/how-to-measure-the-maturity-of-your-software-supply-chain>. Acesso em: 01/09/2024.

106 DODD, L.; KORAKITIS, K. *Maturity of Software Supply Chain Security Practices 2024*. Red Hat, 2024. Disponível em: <https://www.redhat.com/en/resources/software-supply-chain-security-report-overview>. Acesso em: 01/09/2024.

107 MARIANO, A. M.; ROCHA, M. Revisão da literatura: Apresentação de uma abordagem integradora. *AEDM International Conference–Economy, Business and Uncertainty: Ideas for a European and Mediterranean industrial policy*, 2017. Disponível em: <https://www.pesquisatemac.com/_files/ugd/344d4e_63c8f403712b44beacb0e45f3a5a07ec.pdf>. Acesso em: 02/07/2023.

108 DOROFEE, A.; WOODY, C.; ALBERTS, C.; CREEL, R.; ELLISON, R. J. *A Systemic Approach for Assessing Software Supply-Chain Risk*. SEI, 2003. Disponível em: <https://insights.sei.cmu.edu/documents/439/2013_019_001_297385.pdf>. Acesso em: 07/05/2024.

109 WOODY, C.; ELLISON, R. J. Supply-chain risk management: Incorporating security into software development. SEI, 3 2010. Disponível em: <https://resources.sei.cmu.edu/asset_files/whitepaper/2013_019_001_297341.pdf>. Acesso em: 07/05/2024.

110 ELLISON, R. J.; WOODY, C. Considering software supply chain risks. *Crosstalk*, p. 9–12, 10 2010. ISSN 0704-0188. Disponível em: <www.stsc.hill.>

111 DU, S.; LU, T.; ZHAO, L.; XU, B.; GUO, X.; YANG, H. Towards an analysis of software supply chain risk management. In: *World Congress on Engineering and Computer Science 2013*. [S.l.]: Newswood Limited, 2013. p. 162–167. ISBN 978-988-19252-3-7.

112 BRENNAN, K. Managing risk in the software supply chain through software code governance. *Crosstalk*, p. 8–9, 3 2013. ISSN 0704-0188. Disponível em: <https://www.iaeng.org/publication/WCECS2013/WCECS2013_pp162-167.pdf>. Acesso em: 17/09/2023.

113 BENTHALL, S. *Assessing Software Supply Chain Risk Using Public Data*. IEEE Xplore, 2017. 1-5 p. Disponível em: <https://ieeexplore.ieee.org/document/8234461>. Acesso em: 07/05/2024.

114 ALHAZMI, O. H.; MALAIYA, Y. K. *Quantitative vulnerability assessment of systems software*. IEEE Xplore, 2005. Disponível em: <https://ieeexplore.ieee.org/document/1408432>. Acesso em: 27/01/2025.

115 LIBRANTZ, A. F. H.; COSTA, I.; SPINOLA, M. de M.; NETO, G. C. de O.; ZERBINATTI, L. Risk assessment in software supply chains using the bayesian method. *International Journal of Production Research*, Taylor and Francis Ltd., v. 59, p. 6758–6775, 2021. ISSN 1366588X.

116 LEE, S. M.; KIM, S. T.; CHOI, D. Green supply chain management and organizational performance. *Industrial Management 'I&' Data Systems*, v. 112, p. 1148–1180, 8 2012. ISSN 0263-5577.

117 LIANG, L.; WU, X.; DENG, J.; LV, X. Research on risk analysis and governance measures of open-source components of information system in transportation industry. *Procedia Computer Science*, Elsevier, v. 208, p. 106–110, 1 2022. ISSN 1877-0509.

118 AMARAL, T. M. S. do. *Proposta de integração de controles de segurança baseados nos princípios Zero Trust em uma cyber supply chain*. 2022.

119 ZAHAN, N. Software supply chain risk assessment framework. In: *Proceedings - International Conference on Software Engineering*. [S.l.]: IEEE Computer Society, 2023. p. 251–255. ISBN 9798350322637. ISSN 02705257.

120 GIL, A. C. *Como Elaborar Projeto de Pesquisa*. 7ª. ed. [S.l.]: Atlas, 2023. ISBN 978-65-597-7163-9.

121 SILVA, E. L. da; MENEZES, E. M. *Metodologia da Pesquisa e Elaboração de Dissertação*. 4ª. ed. UFSC, 2005. Disponível em: <https://www.academia.edu/download/33206387/.metodologia_da_pesquisa_e_elaboracao_de_dissertacao.pdf>. Acesso em: 28/01/2025.

122 CMMI Product Team. *CMMI ® for Development, Version 1.3 Improving processes for developing better products and services*. SEI, 2010. Disponível em: <https://kilthub.cmu.edu/articles/report/CMMI_for_Development_Version_1_3/6572342/1?file=12057386>. Acesso em: 28/01/2025.

123 CENTER FOR INTERNET SECURITY. *CIS Security Controls Navigator*. 2024. Disponível em: <https://www.cisecurity.org/controls/cis-controls-navigator>. Acesso em: 12/11/2024.

124 CENTER FOR INTERNET SECURITY. *CIS Controls v8 Mapping to NIST CSF 2.0*. Center for Internet Security, 2024. Disponível em: <https://www.cisecurity.org/insights/white-papers/cis-controls-v8-mapping-to-nist-csf-2-0>. Acesso em: 30/11/2024.

125 CENTER FOR INTERNET SECURITY. *CIS Controls v7*. 2019. Disponível em: <https://downloads.cisecurity.org/controls/CIS-Controls-Version-7-1.pdf>. Acesso em: 2025/01/27.

126 CENTER FOR INTERNET SECURITY. *CIS Controls v8*. 2021. Disponível em: <https://www.cisecurity.org/controls/v8>. Acesso em: 2025/01/27.

APÊNDICES

I - Painel - Relacionamentos entre os controles recomendados pelos frameworks

Neste Apêndice encontra-se o painel que exibe os relacionamentos entre os controles de segurança recomendados pelos diversos *frameworks* sobre segurança da cadeia de suprimentos.

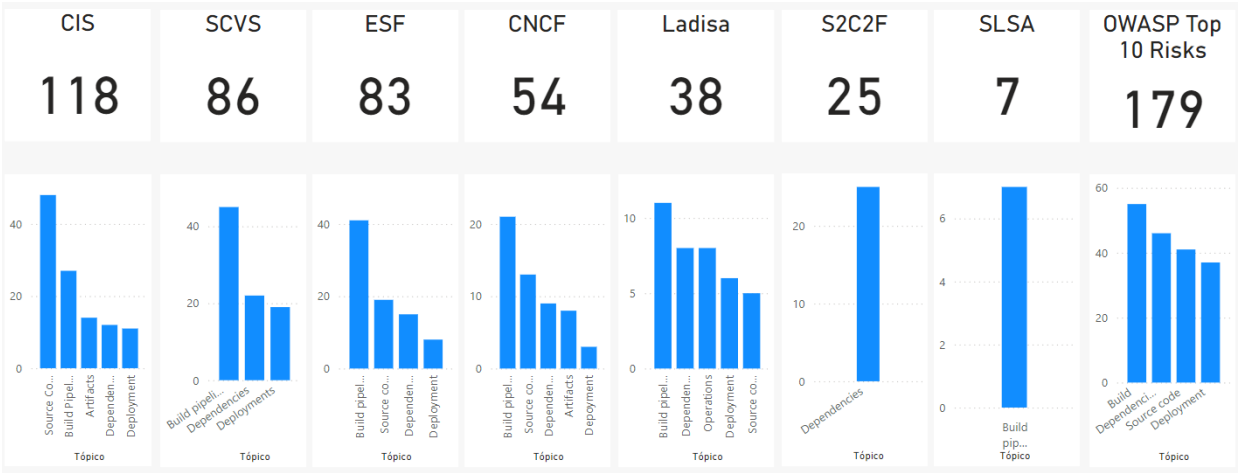


Figura 1: Quantidade de controles, por pilar, por framework
Fonte: Elaborada pelo autor

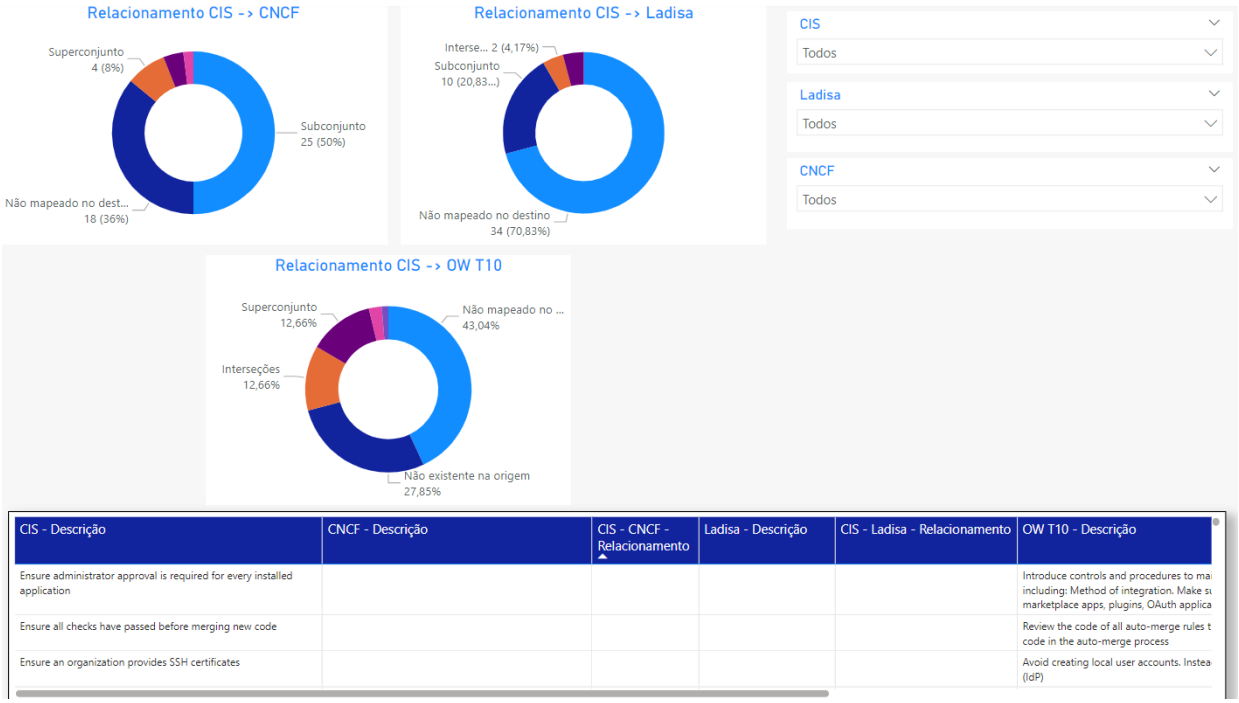


Figura 2: Relacionamentos entre os controles do pilar de código-fonte
Fonte: Elaborada pelo autor

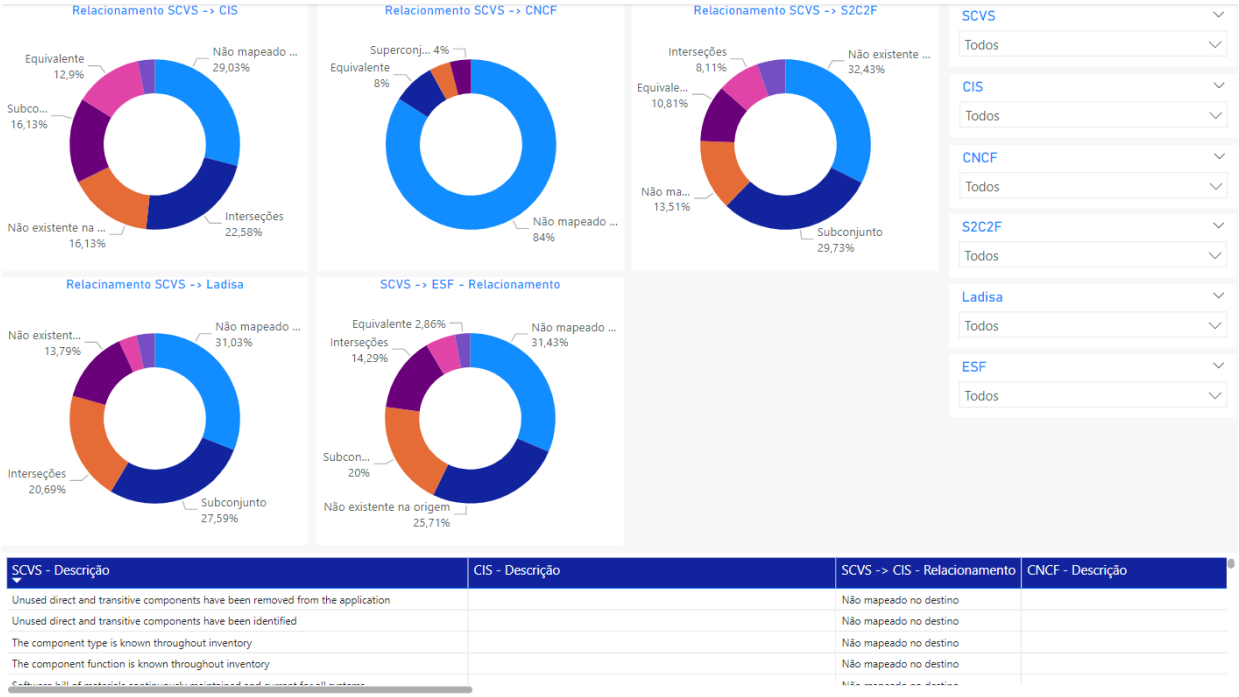


Figura 3: Relacionamentos entre os controles do pilar de dependências
Fonte: Elaborada pelo autor

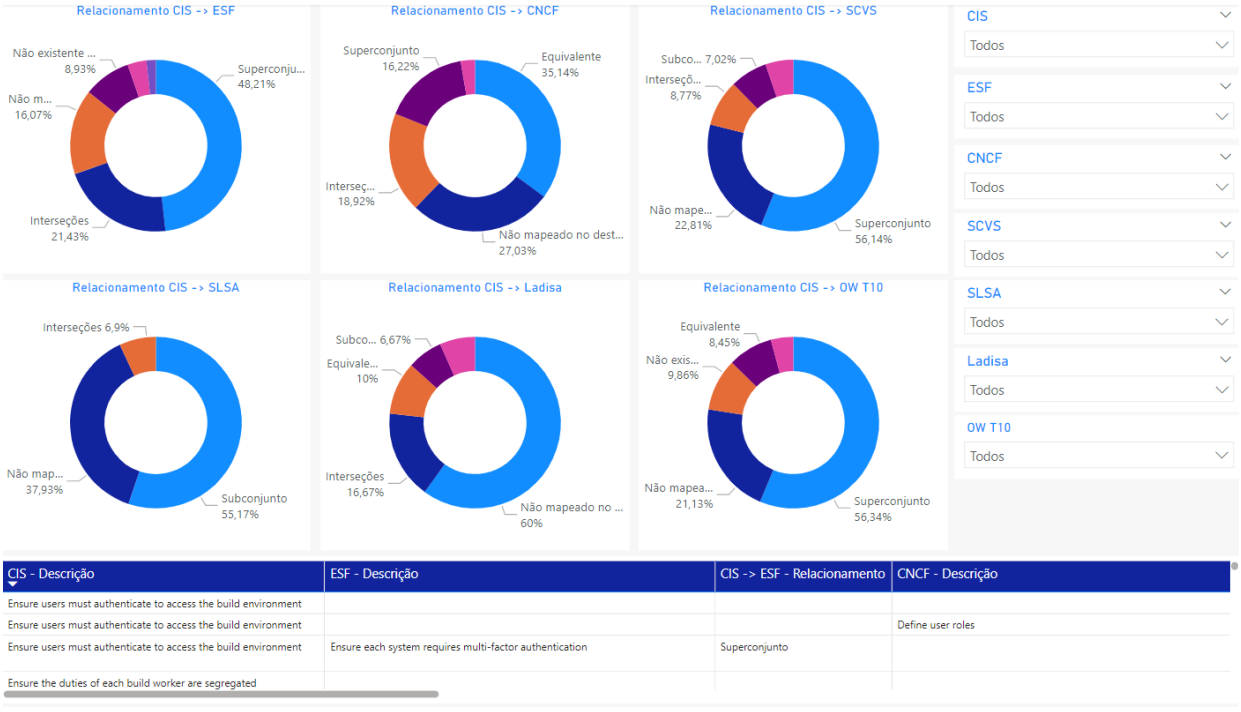
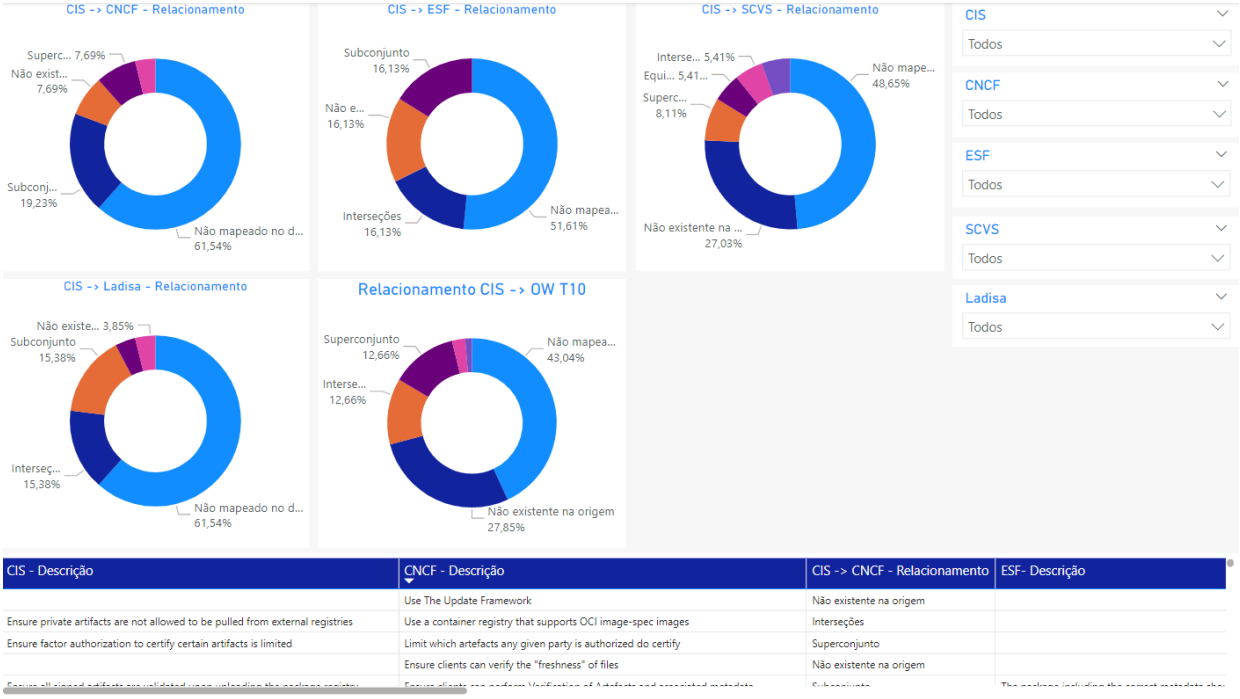


Figura 4: Relacionamentos entre os controles do pilar de compilação
Fonte: Elaborada pelo autor



CIS - Descrição	CNCF - Descrição	CIS -> CNCF - Relacionamento	ESF- Descrição
	Use The Update Framework	Não existente na origem	
Ensure private artifacts are not allowed to be pulled from external registries	Use a container registry that supports OCI image-spec images	Interseções	
Ensure factor authorization to certify certain artifacts is limited	Limit which artefacts any given party is authorized do certify	Superconjunto	
	Ensure clients can verify the "freshness" of files	Não existente na origem	

Figura 5: Relacionamentos entre os controles do pilar de deploy
Fonte: Elaborada pelo autor

II - Modelo de Conformidade

Neste Apêndice encontra-se a planilha que materializa o Modelo de Conformidade sobre as Melhores Práticas relativas à Segurança de Cadeias de Suprimentos de Software.

Introdução

O presente modelo de maturidade foi elaborado com base nos frameworks de mercado referentes à segurança da cadeia de suprimentos de software, com adição de um trabalho acadêmico que propõe uma taxonomia de ataques à cadeia de suprimentos de software, e os respectivos controles de segurança por ele propostos como prevenção a esses ataques, conforme detalhamento a seguir:

CIS Software Supply Chain Security Guide - v1.0
Autoria: Center for Internet Security (CIS)
https://www.cisecurity.org/insights/white-papers/cis-software-supply-chain-security-guide
Identificação no modelo de maturidade: CIS

Software Supply Chain Best Practices
Autoria: Cloud Native Computing Foundation (CNCF)
https://project.linuxfoundation.org/hubfs/CNCF_SSCP_v1.pdf
Identificação no modelo de maturidade: CNCF

OWASP Software Component Verification Standard - Version 1.0
https://owasp.org/www-project-software-component-verification-standard/
Identificação no modelo de maturidade: SCVS

OWASP Top 10 CI/CD Security Risks
Autoria: Open Worldwide Application Security Project (OWASP)
https://github.com/OWASP/www-project-top-10-ci-cd-security-risks/raw/3204d6d181e2a5517ffdcbe208fb536b9cc6c50b/assets/OWASP_Top_10_CICD_Risks.pdf
Identificação no modelo de maturidade: OW

Secure Supply Chain Consumption Framework
Autoria: Microsoft e OpenSSF
https://github.com/ossf/s2c2f/blob/main/specification/framework.md
Identificação no modelo de maturidade: S2C2F

Supply-chain Levels for Software Artifacts
Autoria: The Linux Foundation

https://slsa.dev/spec/v1.0/onepage
Identificação no modelo de maturidade: SLSA

Securing the Software Supply Chain - Recommended Practices Guide for Developers
Autoria: Enduring Security Framework
https://media.defense.gov/2022/Sep/01/2003068942/-1/-1/0/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_DEVELOPERS.PDF
Identificação no modelo de maturidade: ESF

Journey to the Center of Software Supply Chain Attacks
Autoria: Piergiorgio Ladisa, Serena Elisa Ponta, Antonino Sabetta, Matias Martinez, Olivier Barais
https://arxiv.org/abs/2304.05200
Identificação no modelo de maturidade: LAD

CONTROLES - CÓDIGO FONTE	EQUIVALÊNCIAS	NÍVEL DE MATURIDADE	FONTE	
			FRAMEWORK	PÁGINA
Estruturação do ambiente				
Ensure network access to the systems is aligned with the principle of least access		1	OW	29
Ensure Git access is limited based on IP addresses		1	CIS	22
Ensure two administrators are set for each repository		1	CIS	19
Ensure inactive repositories are reviewed and archived periodically		1	CIS	16
Ensure all public repositories contain a SECURITY.md file		1	CIS	14
Gerenciamento de vulnerabilidades do ambiente				
Maintain an inventory of systems and versions in use, including mapping of a designated owner for each system		2	OW 25	29
Continuously check for known vulnerabilities in these components		2	OW 26	29
If a security patch is available, update the vulnerable component. If not, consider removing the component / system, or reduce the potential impact of exploiting the vulnerability by restricting access to the system, or the system's ability to perform sensitive operations		2	OW 27	29
Establish a process to periodically review all system configurations for any setting that can have an effect on the security posture of the system, and ensure all settings are optimal		2	OW 29	29
Measures aimed at detecting configuration drifts (e.g. resources in cloud environments which aren't managed using a signed IAC template), potentially indicative of resources that were deployed by an untrusted source or process		2	OW 37	34
Gerenciamento de usuários de usuários				
User account management		1	LAD	7
Ensure new members are required to be invited using company-approved email	Prevent employees from using their personal email addresses, or any address which belongs to a domain not owned and managed by the organization, against the SCM, CI, or any other CI/CD platform	1	CIS, OW	19, 10
Continuously monitor for non-domain addresses across the different systems and remove non-compliant users		1	OW	10
Ensure an organization's identity is confirmed with a "Verified" badge		1	CIS	20
Ensure Source Code Management (SCM) email notifications are restricted to verified domains		1	CIS	21
Ensure inactive users are reviewed and removed periodically		1	CIS	17
Determine an acceptable period for disabling/removing stale accounts and disable/remove any identity which has surpassed the predetermined period of inactivity		1	OW	9
Refrain from allowing users to self-register to systems, and grant permission on an as-needed basis		1	OW	10
Autenticação de usuários				
Enforce MFA for acessing source code repositories	Secure Authentication (e.g., Multi-Factor Authentication (MFA), password recycle, session timeout, token protection)	2	CNCF, LAD	14, 7
Ensure the organization is requiring members to use Multi-Factor Authentication (MFA)		2	CIS	18
Ensure Multi-Factor Authentication (MFA) is required for contributors of new code		2	CIS	18
Ensure an organization provides SSH certificates		2	CIS	21
Use SSH keys to provide developers access do source code repositories		2	CNCF	14
Have a Key Rotation Policy		2	CNCF	15
Use short-lived/ephemeral credentials for machine/service access		2	CNCF	15
Avoid sharing the same set of credentials across multiple contexts. This increases the complexity of achieving the principle of least privilege as well as having a negative effect on accountability		2	OW	25
Prefer using temporary credentials over static credentials. In case static credentials need to be in use - establish a procedure to periodically rotate all static credentials and detect stale credentials		2	OW	25
Configure usage of credentials to be limited to predetermined conditions (like scoping to a specific source IP or identity) to ensure that even in case of compromise, exfiltrated credentials cannot be used outside your environment		2	OW	25
Avoid creating local user accounts. Instead, create and manage identities using a centralized organization component (IdP)		2	OW	10
Whenever local user accounts are in use, ensure that accounts which no longer require access are disabled/removed and that security policies around all existing accounts match the organization's policies		2	OW	10
Permissões de usuários				
Define roles aligned to functional responsibilities		2	CNCF	13
Ensure there are restrictions on who can dismiss code change reviews		2	CIS	6
Ensure pushing or merging of new code is restricted to specific individuals or teams		2	CIS	11
Ensure repository creation is limited to specific members		2	CIS	14
Ensure repository deletion is limited to specific users		2	CIS	15
Ensure issue deletion is limited to specific users		2	CIS	15
Ensure team creation is limited to specific members		2	CIS	17
Ensure minimum number of administrators are set for the organization		2	CIS	17
Remove permissions not necessary for the ongoing work of each identity across the different systems in the environment		2	OW	9

Conduct a continuous analysis and mapping of all identities across all systems within the engineering ecosystem. For each identity, map the identity provider, level of permissions granted and level of permissions actually used. Ensure all methods of programmatic access are covered within the analysis		2	OW	9
Establish procedures to continuously map credentials found across the different systems in the engineering ecosystem - from code to deployment. Ensure each set of credentials follows the principle of least privilege and has been granted the exact set of permission needed by the service using it		2	OW	25
Remove permissions granted on the SCM repository from users that do not need them		2	OW	19
Continuously map all external collaborators and ensure their identities are aligned with the principle of least privilege		2	OW	10
Whenever possible, grant permissions with a predetermined expiry date - for both human and programmatic accounts - and disable their account once the work is done		2	OW	10
Ensure strict base permissions are set for repositories		1	CIS	20
Refrain from granting base permissions in a system to all users, and to large groups where user accounts are automatically assigned to		1	OW	10
Avoid using shared accounts. Create dedicated accounts for each specific context, and grant the exact set of permissions required for the context in question		2	OW	10
Políticas de contribuição de código				
Estabilish and adhere to contribution policies		2	CNCF	13
Ensure any changes to code are tracked in a version control platform		2	CIS	4
Ensure any change to code can be traced back to its associated task		2	CIS	5
Ensure any change to code receives approval of two strongly authenticated users (Automated)	Enforce an independent four-eyes principle Manual source code review	2	CIS, CNCF, LAD	5, 14, 7
Ensure previous approvals are dismissed when updates are introduced to a code change proposal		2	CIS	6
Prevent commiting secrets to the source code repository		2	CNCF	12
Ensure scanners are in place to identify and prevent sensitive data in code		2	CIS	24
Detect secrets pushed to and stored on code repositories. Use controls such as an IDE plugin to identify secrets used in the local changes, automatic scanning upon each code push, and periodical scans on the repository and its past commits		2	OW	26
Verify that secrets are removed from any type of artifact, such as from layers of container images, binaries, or Helm charts		2	OW	26
Define individuals/teams that are responsible for code in a repository and associated coding conventions		3	CNCF	12
Ensure code owners are set for extra sensitive code or configuration		3	CIS	7
Ensure code owner's review is required when a change affects owned code		3	CIS	7
Implement processes and technologies to validate the integrity of resources all the way from development to production. When a resource is generated, the process will include signing that resource using an external resource signing infrastructure		3	OW	33
Ensure linear history is required		3	CIS	10
Políticas para Pull/Merge requests				
Ensure all checks have passed before merging new code		3	CIS	8
Pull/Merge request review		3	LAD	7
Automate software security scanning and testing		2	CNCF	13
Limit the usage of auto-merge rules and ensure that wherever they are in use - they are applicable to the minimal amount of contexts		3	OW	6
Review the code of all auto-merge rules thoroughly to ensure they cannot be bypassed and avoid importing 3rd party code in the auto-merge process		3	OW	6
Ensure open Git branches are up to date before they can be merged into code base		3	CIS	9
Ensure all open comments are resolved before allowing code change merging		3	CIS	9
Ensure any merging of code is automatically scanned for risks		3	CIS	12
Ensure verification of signed commits for new changes before merging	Require signed commits	3	CIS, CNCF	10
SCM solutions provide the ability to sign commits using a unique key for each contributor. This measure can then be leveraged to prevent unsigned commits from flowing down the pipeline		3	OW	34
Proteção de branches				
Use branch protection rules	Protect production branch	2	CNCF, LAD	14
Configure branch protection rules on branches hosting code which is used in production and other sensitive systems		2	OW	6
Where possible, avoid exclusion of user accounts or branches from branch protection rules		2	OW	6
Ensure branch protection rules are enforced for administrators		2	CIS	11
Ensure force push code to branches is denied		2	CIS	12
Ensure branch deletions are denied		2	CIS	12
Ensure inactive branches are periodically reviewed and removed		2	CIS	8
Ensure any changes to branch protection rules are audited		2	CIS	13
Enforce full attestation and verification for protected branches		3	CNCF	12
Testes automatizados				
Automate software security scanning and testing		2	CNCF	13
Ensure scanners are in place to secure Continuous Integration (CI) pipeline instructions		2	CIS	25
Ensure scanners are in place to secure Infrastructure as Code (IaC) instructions		2	CIS	25
Ensure scanners are in place for code vulnerabilities		2	CIS	26
Ensure scanners are in place for open-source vulnerabilities in used packages		2	CIS	26
Ensure scanners are in place for open-source license issues in used packages		2	CIS	27
Segurança dos projetos de software				
Ensure all copies (forks) of code are tracked and accounted for		3	CIS	15

Ensure all code projects are tracked for changes in visibility status		3	CIS	16
Ensure anomalous code behavior is tracked		3	CIS	22
Gestão da aplicativos de terceiros instalados na solução de versionamento de código				
Ensure administrator approval is required for every installed application		3	CIS	23
Establish vetting procedures to ensure 3rd parties granted access to resources anywhere across the engineering ecosystem are approved prior to being granted access to the environment, and that the level of permission they are granted is aligned with the principle of least privilege		3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Method of integration. Make sure all methods of integration for each system are covered (including marketplace apps, plugins, OAuth applications, programmatic access tokens, etc.).		3	OW	31
Ensure the access granted to each installed application is limited to the least privilege needed		3	CIS	24
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Level of permission granted to the 3rd party		3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Level of permission actually in use by the 3rd party		3	OW	32
Ensure each 3rd party is limited and scoped to the specific resources it requires access to and remove unused and/or redundant permissions		3	OW	32
Ensure stale applications are reviewed and inactive ones are removed	Periodically review all 3rd parties integrated and remove those no longer in use	3	CIS, OW	23
Auditoria do ambiente				
Identify and build an inventory of all the systems in use within the organization, containing every instance of these systems (specifically relevant for self-managed systems e.g. Jenkins)		3	OW	36
Once all relevant systems are identified, the next step is ensuring that all relevant logs are enabled, as this is not the default state in the different systems. Visibility should be optimized around both human access as well as programmatic access through all the various measures it is allowed. It is important to place an equal level of emphasis on identifying all relevant audit log sources, as well as the applicative log sources		3	OW	36
Shipping logs to a centralized location (e.g. SIEM), to support aggregation and correlation of logs between different systems for detection and investigation		3	OW	36
Creating alerts to detect anomalies and potential malicious activity, both in each system on its own and anomalies in the code shipping process, which involves multiple		3	OW	36

CONTROLES - DEPENDÊNCIAS	EQUIVALÊNCIAS	NÍVEL DE MATURIDADE	FONTE	
			FRAMEWORK	PÁGINA
Estruturação do ambiente				
Package managers are used to manage all third-party binary components	Ensure trusted package managers and repositories are defined and prioritized Use package managers trusted by your organization	1	SCVS, CIS, S2C2F	8, ,16
Define and prioritize trusted package managers and repositories		3	CNCF	18
Use an OSS binary repository manager solution		1	S2C2F	16
Any client pulling code packages should not be allowed to fetch packages directly from the internet or untrusted sources		2	OW	12
Whenever 3rd party packages are pulled from an external repository, ensure all packages are pulled through an internal proxy rather than directly from the internet. This allows deploying additional security controls at the proxy layer, as well as providing investigative capabilities around packages pulled - in case of a security incident		2	OW	12
Where applicable, disallow pulling of packages directly from external repositories. Configure all clients to pull packages from internal repositories, containing pre-vetted packages, and establish a mechanism to verify and enforce this client configuration		2	OW	12
Ensure all private packages are registered under the organization's scope		2	OW	13
Ensure all code referencing a private package uses the package's scope		2	OW	13
Ensure clients are forced to fetch packages that are under your organization's scope solely from your internal registry		2	OW	13
When installation scripts are being executed as part of the package installation, ensure that a separate context exists for those scripts, which does not have access to secrets and other sensitive resources available in other stages in the build process		2	OW	13
Ensure that internal projects always contain configuration files of package managers (for example .npmrc in NPM) within the code repository of the project, to override any insecure configuration potentially existing on a client fetching the package		2	OW	13
Avoid publishing names of internal projects in public repositories		2	OW	13
Mirror a copy of all OSS source code to an internal location		3	S2C2F	16
The source code (not binaries alone) should be integrated into the build environment allowing the security scanning process of the build environment	Build libraries based upon source code	3	ESF, CNCF	25, 17
Whenever possible, images should be built from the source and not downloaded from the internet, unless there is an understanding of the provenance and trust of delivery		3	ESF	25
Gerenciamento de vulnerabilidades do ambiente				
Maintain an inventory of systems and versions in use, including mapping of a designated owner for each system		2	OW	28
Continuously check for known vulnerabilities in these components		2	OW	28
If a security patch is available, update the vulnerable component. If not, consider removing the component / system, or reduce the potential impact of exploiting the vulnerability by restricting access to the system, or the system's ability to perform sensitive operations		2	OW	28
Ensure network access to the systems is aligned with the principle of least access		2	OW	28
Establish a process to periodically review all system configurations for any setting that can have an effect on the security posture of the system, and ensure all settings are optimal		2	OW	28
Measures aimed at detecting configuration drifts (e.g. resources in cloud environments which aren't managed using a signed IAC template), potentially indicative of resources that were deployed by an untrusted source or process		2	OW	34
Gerenciamento de usuários				
Determine an acceptable period for disabling/removing stale accounts and disable/remove any identity which has surpassed the predetermined period of inactivity		1	OW	9
Prevent employees from using their personal email addresses, or any address which belongs to a domain not owned and managed by the organization, against the SCM, CI, or any other CI/CD platform		1	OW	10
Continuously monitor for non-domain addresses across the different systems and remove non-compliant users		1	OW	10
Refrain from allowing users to self-register to systems, and grant permission on an as-needed basis		1	OW	10
Autenticação de usuários				
Avoid sharing the same set of credentials across multiple contexts. This increases the complexity of achieving the principle of least privilege as well as having a negative effect on accountability		2	OW	25
Prefer using temporary credentials over static credentials. In case static credentials need to be in use - establish a procedure to periodically rotate all static credentials and detect stale credentials		2	OW	25
Configure usage of credentials to be limited to predetermined conditions (like scoping to a specific source IP or identity) to ensure that even in case of compromise, exfiltrated credentials cannot be used outside your environment		2	OW	25
Avoid creating local user accounts. Instead, create and manage identities using a centralized organization component (IdP)		2	OW	10
Whenever local user accounts are in use, ensure that accounts which no longer require access are disabled/removed and that security policies around all existing accounts match the organization's policies		2	OW	10
Permissões de usuários				
Conduct a continuous analysis and mapping of all identities across all systems within the engineering ecosystem. For each identity, map the identity provider, level of permissions granted and level of permissions actually used. Ensure all methods of programmatic access are covered within the analysis		2	OW	9

Establish procedures to continuously map credentials found across the different systems in the engineering ecosystem - from code to deployment. Ensure each set of credentials follows the principle of least privilege and has been granted the exact set of permission needed by the service using it		2	OW	25
Remove permissions not necessary for the ongoing work of each identity across the different systems in the environment		2	OW	9
Continuously map all external collaborators and ensure their identities are aligned with the principle of least privilege		2	OW	10
Whenever possible, grant permissions with a predetermined expiry date - for both human and programmatic accounts - and disable their account once the work is done		2	OW	10
Refrain from granting base permissions in a system to all users, and to large groups where user accounts are automatically assigned to		2	OW	10
Avoid using shared accounts. Create dedicated accounts for each specific context, and grant the exact set of permissions required for the context in question		2	OW	10
Política de uso de componentes				
Ensure an organization-wide dependency usage policy is enforced		3	CIS	45
Enforce usage of a curated OSS feed that enhances the trust of your OSS	Establish vetting process for Open-Source components hosted in internal/public repositories	3	S2C2F, LAD, ESF	17, 25
	Compile a repository of the third-party components that have been vetted			
Compile a list of known trusted suppliers and their associated artifacts that have been integrated into the company´s products		3	ESF	25
Suppliers should produce artifacts attesting to the development process, quality and security aspects of the component		3	ESF	25
Have a Deny List capability to block known malicious OSS from being consumed		3	S2C2F	16
Version pinning		3	LAD	7
Ensure all packages used are more than 60 days old		3	CIS	44
Avoid configuring clients to pull the latest version of a package. Prefer configuring a pre-vetted version or version ranges. Use the framework specific techniques to continuously “lock” the package version required in your organization to a stable and secure version		3	OW	12
Securely configure your package source files (i.e. nuget.config, .npmrc, pip.conf, pom.xml, etc.)		2	S2C2F	17
Take into account the producer´s country of origin		3	ESF	25
Suppliers should produce artifacts attesting to the development process, quality and security aspects of the component		3	ESF	25
Inventário de componentes				
Maintain an automated inventory of all OSS used in development		1	S2C2F	17
All direct and transitive components and their versions are known at completion of a build	Track dependencies between open source components	1	SCVS, CNCF	8, 17
An accurate inventory of all third-party components is available in a machine-readable format	Ensure dependencies are monitored between open-source components	1	SCVS, CIS	8
Software bill of materials are required for new procurements	Ensure SBOM is required from all third-party suppliers	2	SCVS, CIS, CNCF	8, 17
	Require SBOM from third party supplier			
Ensure a signed SBOM of the code is supplied		2	CIS	43
Validate SBOMs of OSS that you consume into your build (provenance, dependencies, digital signature)		3	S2C2F	17
Components are uniquely identified in a consistent, machine-readable format		1	SCVS	8
The component type is known throughout inventory		3	SCVS	8
The component function is known throughout inventory		3	SCVS	8
The third party´s SBOM can be compared with the SBOM produced by the SCA tools		3	ESF	26
Verify the provenance of your OSS		3	S2C2F	17
Point of origin is known for all components		3	SCVS	8
The organization should verify a third-party´s ownership and control status, and past performance of the suppliers and their upstream suppliers		3	ESF	25
Ensure packages are automatically scanned for ownership change		3	CIS	46
SCA tools should be applied to the software deliverable from the third-party			ESF	26
Use binary scanning and software composition analysis tools to detect unknown files and the open source components in binary packages	Maintain detailed Software Bill of Materials (SBOM) and perform Software Composition Analysis (SCA)	1	ESF, LAD	24, 7
An automated process of identifying non-specified component versions is used		1	SCVS	13
An automated process of identifying out-of-date components is used		1	SCVS	13
An automated process of identifying end-of-life / end-of-support components is used	Scan OSS to determine if its end-of-life	2	SCVS, S2C2F	13, 16
An automated process of identifying component type is used		2	SCVS	13
An automated process of identifying component function is used		3	SCVS	13
An automated process of identifying component quantity is used		1	SCVS	14
An automated process of identifying component license is used	Ensure packages are automatically scanned for license implications	1	SCVS, CIS, S2C2F	14, 45, 16

		Scan OSS for licenses			
Gerenciamento de vulnerabilidades nos pacotes					
SAST/DAST and other appropriate review such as composition analysis must be performed to determine if the risk is acceptable			2	ESF	26
SBOM is analyzed for risk			1	SCVS	9
	Consider the output of the composition analysis in the decision to select and integrate the software component		2	ESF	25
Ensure third-party artifacts and open-source libraries are verified			2	CIS	41
	Enable checksum verification and signature verification for pulled packages	Validate integrity of the OSS that you consume into your build (digital signature or hash)	2	OW, S2C2F	12, 17
Ensure packages are automatically scanned for known vulnerabilities			2	CIS	45
	Component can be analyzed with linters and/or static analysis tools		1	SCVS	13
	Component is analyzed using linters and/or static analysis tools prior to use	Verify third party artefacts and open source libraries	2	SCVS, CNCF	13, 16
	Linting and/or static analysis is performed with every upgrade of a component		2	SCVS	13
	An automated process of identifying all publicly disclosed vulnerabilities in third-party and open source components is used	Monitor available CVE reporting mechanisms and third-party support channles to determine whether vulnerabiliteis identified can impact the products	1	SCVS, ESF, S2C2F, LAD	13, 26, 16, 7
		Scan OSS for known vulnerabilities (i.e. CVEs, GitHub Advisories, etc.)			
		Integrate Open-Source vulnerability scanner into CI/CD pipeline			
	An automated process of identifying confirmed dataflow exploitability is used		3	SCVS	13
Scan OSS for malware			3	S2C2F	16
Perform proactive security review of OSS			3	S2C2F	16
The owner of the third-party component must also notify the product organization of the presence of a vulnerability, the risk associated and the timeframe to address it			3	ESF	26
As a general rule, ensure that an appropriate level of focus is placed around detection, monitoring and mitigation to ensure that in case of an incident, it is identified as quickly as possible and has the minimal amount of potential damage			3	OW	13
Typo guard/Typo detection			3	LAD	7
Preventive squatting the released packages			3	LAD	7
Have an OSS Incident Response Plan			2	S2C2F	17
Correção de vulnerabilidades nas pacotes					
Update vulnerable OSS manually			1	S2C2F	17
Enable automated OSS updates			2	S2C2F	17
Display OSS vulnerabilities as comments in Pull Requests (PRs)			2	S2C2F	17
Rebuild the OSS in a trusted build environment, or validate that it is reproducibly built			3	S2C2F	18
Digitally sign the OSS you rebuild			3	S2C2F	18
Implement processes and technologies to validate the integrity of resources all the way from development to production. When a resource is generated, the process will include signing that resource using an external resource signing infrastructure			3	OW	33
Generate SBOMs for OSS that you rebuild			3	S2C2F	18
Digitally sign the SBOMs you produce			3	S2C2F	18
Remove un-used dependencies			3	LAD	7
Unused direct and transitive components have been identified			3	SCVS	11
Unused direct and transitive components have been removed from the application			3	SCVS	11
Implement a change in the code to address a zero-day vulnerability, rebuild, deploy to your organization, and confidentially contribute the fix to the upstream maintainer			3	S2C2F	18
Gestão da aplicativos de terceiros instalados na solução de versionamento de código					
Establish a process to periodically review all system configurations for any setting that can have an effect on the security posture of the system, and ensure all settings are optimal			3	OW	28
Establish vetting procedures to ensure 3rd parties granted access to resources anywhere across the engineering ecosystem are approved prior to being granted access to the environment, and that the level of permission they are granted is aligned with the principle of least privilege			3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Method of integration. Make sure all methods of integration for each system are covered (including marketplace apps, plugins, OAuth applications, programmatic access tokens, etc.).			3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Level of permission granted to the 3rd party			3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Level of permission actually in use by the 3rd party			3	OW	32

Ensure each 3rd party is limited and scoped to the specific resources it requires access to and remove unused and/or redundant permissions		3	OW	32
Periodically review all 3rd parties integrated and remove those no longer in use		3	OW	32
Auditoria do ambiente				
Identify and build an inventory of all the systems in use within the organization, containing every instance of these systems (specifically relevant for self-managed systems e.g. Jenkins)		3	OW	36
Once all relevant systems are identified, the next step is ensuring that all relevant logs are enabled, as this is not the default state in the different systems. Visibility should be optimized around both human access as well as programmatic access through all the various measures it is allowed. It is important to place an equal level of emphasis on identifying all relevant audit log sources, as well as the applicative log sources		3	OW	36
Shipping logs to a centralized location (e.g. SIEM), to support aggregation and correlation of logs between different systems for detection and investigation		3	OW	36
Creating alerts to detect anomalies and potential malicious activity, both in each system on its own and anomalies in the code shipping process, which involves multiple systems and requires deeper knowledge in the internal build and deployments processes		3	OW	36
Audit that developers are consuming OSS through the approved ingestion method		2	S2C2F	17

CONTROLES - BUILD	EQUIVALÊNCIAS	NÍVEL DE MATURIDADE	FONTE	
			FRAMEWORK	PÁGINA
Estruturação do ambiente				
Software producer follows a consistent build process so that others can form expectations about what a “correct” build looks like.		1	SLSA	41
Cryptographically guarantee policy adherence		3	CNCF	28
Provision a secured orchestration platform do host software factory		2	CNCF	31
Ephemeral build environment		3	LAD	7
Ensure the creation of the build environment is automated	Automate Creation of the Build Environment	3	CIS, CNCF	30, 30
Ensure build workers’ deployment configuration is stored in a version control platform	To prevent the manipulation of the CI configuration file to run malicious code in the pipeline, each CI configuration file must be reviewed before the pipeline runs. Alternatively, the CI configuration file can be managed in a remote branch, separate from the branch containing the code being built in the pipeline. The remote branch should be configured as protected	3	CIS, OW	34, 19
Ensure all build steps are defined as code	Build and related continuous integration/contiuous delivery steps should all be automated through a pipeline defined as code	3	CIS, CNCF	35, 30
Ensure steps have clearly defined build stage input and output		2	CIS	36
All transitive build steps, sources and dependencies should be fully declared up front with immutable references		2	ESF	31
Application build pipeline may only perform builds of source code maintained in version control systems		1	SCVS	10
All build-time manipulations to source or binaries are known and well defined		2	SCVS	11
Find and Eliminate Sources of Non-Determinisms		2	CNCF	29
Ensure output is written to a separate, secured storage repository	Write Output to a Separate Secured Storage Repo	3	CIS, CNCF	36, 33
Build platform runs on dedicated infrastructure, not an individual’s workstation, and the provenance is tied to that infrastructure through a digital signature		2	SLSA	
Application uses a continuous integration build pipeline		1	SCVS	10
Ensure each pipeline has a single responsibility		1	CIS	28
Ensure all aspects of the pipeline infrastructure and configuration are immutable	The build pipeline infrastructures should be completely locked down	3	CIS, ESF	29, 29
Subject build scripts and configuration files to the same code review process		2	ESF	29
Use version control for pipeline configurations	To prevent the manipulation of the CI configuration file to run malicious code in the pipeline, each CI configuration file must be reviewed before the pipeline runs. Alternatively, the CI configuration file can be managed in a remote branch, separate from the branch containing the code being built in the pipeline. The remote branch should be configured as protected	2	ESF, OW	29, 19
Only allow pipeline modifications through "pipeline as code"		2	CNCF	33
Validate environments and dependencies before usage		2	CNCF	28
Application build pipeline prohibits alteration of package management settings		2	SCVS	10
Application build pipeline prohibits alteration of DNS and network settings during build		3	SCVS	10
Application build pipeline prohibits alteration of certificate trust stores		3	SCVS	10
Application build pipeline requires separation of concerns for the modification of system settings		3	SCVS	10
Establish a process to periodically review all system configurations for any setting that can have an effect on the security posture of the system, and ensure all settings are optimal		3	OW	28
Measures aimed at detecting configuration drifts (e.g. resources in cloud environments which aren’t managed using a signed IAC template), potentially indicative of resources that were deployed by an untrusted source or process		3	OW	34
Ensure access to build environments is limited		2	CIS	30
The build pipeline infrastructures should be protected from external and off-local area network (LAN) activity		2	ESF	29
Restrict access to system resources of code executed during each build steps		2	LAD	7

	Application build pipeline prohibits alteration of build outside of the job performing the build		2	SCVS	10
	Application build pipeline prohibits the execution of arbitrary code outside of the context of a jobs build script	Prevent script execution	2	SCVS, LAD	10, 7
	Application build pipeline enforces authorization and defaults to deny		2	SCVS	10
Build platform implements strong controls to prevent runs from influencing one another, even within the same project.		Isolation of build steps	3	SLSA, LAD	43, 7
Ensure build workers are single-used		Revert the execution node to its pristine state after each pipeline execution	3	CIS, OW, CNCF, LAD	31, 22, 31, 7
		Build Workers should be single use			
		Use of dedicated build service			
Do not use a shared node for pipelines with different levels of sensitivity / that require access to different resources. Shared nodes should be used only for pipelines with identical levels of confidentiality			3	OW	22
Distribute Builds Accross Different Infrasctructure			3	CNCF	30
Ensure build worker environments and commands are passed and not pulled		Pass in Build Worker Environment and Commands	3	CIS, CNCF	32, 32
All transitive build steps, sources and dependencies should be fully declared up front with immutable references			3	ESF	31
Ensure the duties of each build worker are segregated		Segregate the Duties of Each Build Worker	3	CIS, CNCF	32, 32
	Protect any secrets associated with the build pipeline		3	ESF	29
	Ensure that pipelines running unreviewed code are executed on isolated nodes, not exposed to secrets and sensitive environments		3	OW	19
	For sensitive pipelines, for example those that are exposed to secrets, ensure that each branch that is configured to trigger a pipeline in the CI system has a correlating branch protection rule in the SCM		3	OW	19
	Each pipeline should only have access to the credentials it needs to fulfill its purpose. The credentials should have the minimum required privileges		3	OW	19
	Ensure secrets that are used in CI/CD systems are scoped in a manner that allows each pipeline and step to have access to only the secrets it requires		3	OW	22
	When installation scripts are being executed as part of the package installation, ensure that a separate context exists for those scripts, which does not have access to secrets and other sensitive resources available in other stages in the build process		3	OW	13
	Use built-in vendor options or 3rd party tools to prevent secrets from being printed to console outputs of future builds. Ensure all existing outputs do not contain secrets		3	OW	26
Ensure build workers have minimal network connectivity		Ensure Software Factory has minimal network connectivity	3	CIS, CNCF	33, 32
	Segregate the engineering network from the corporate network		3	ESF	29
	The build steps should run with no network access		3	ESF	31
	CI and CD pipeline jobs should have limited permissions on the controller node. Where applicable, run pipeline jobs on a separate, dedicated node		3	OW	22
	Ensure network segmentation in the environment the job is running on is configured to allow the execution node to access only the resources it requires within the network. Where possible, refrain from granting unlimited access towards the internet to build nodes		3	OW	22
	Ensure network access to the systems is aligned with the principle of least access		3	OW	28
Ensure access to build process triggering is minimized			3	CIS	37
	Where applicable, prevent accounts from triggering production build and deployment pipelines without additional approval or review		3	OW	6
	Evaluate the need for triggering pipelines on public repositories from external contributors. Where possible, refrain from running pipelines originating from forks, and consider adding controls such as requiring manual approval for pipeline execution		3	OW	19
Gerenciamento de vulnerabilidades do ambiente					
Compilers, version control clients, development utilities, and software development kits are analyzed and monitored for tampering, trojans, or malicious code			2	SCVS	11
Validate runtime security of build workers			2	CNCF	28
Perform periodic vulnerability scans				ESF	34
	Maintain an inventory of systems and versions in use, including mapping of a designated owner for each system		2	OW	28
	Continuously check for known vulnerabilities in these components		2	OW	28

	If a security patch is available, update the vulnerable component. If not, consider removing the component / system, or reduce the potential impact of exploiting the vulnerability by restricting access to the system, or the system's ability to perform sensitive operations		2	OW	28
Ensure build workers are automatically scanned for vulnerabilities			2	CIS	34
Ensure pipelines are automatically scanned for misconfigurations			2	CIS	37
Ensure pipelines are automatically scanned for vulnerabilities			2	CIS	37
Perform periodic penetration testing			2	ESF	34
Ensure the OS user running the pipeline job has been granted OS permissions on the execution node according to the principle of least privilege			2	OW	22
Monitor/perform integrity checks of executables, libraries, configuration files, etc			2	ESF	34
Minimize and regularly audit service accounts			2	ESF	29
Application build pipeline has required maintenance cadence where the entire stack is updated, patched, and re-certified for use		Ensure the execution node is appropriately patched	,	SCVS, OW	10, 22
Ensure scanners are in place to identify and prevent sensitive data in pipeline files (Automated)		Verify that secrets are removed from any type of artifact, such as from layers of container images, binaries, or Helm charts	2	CIS, OW	38, 26
The build pipeline infrastructures should be configured to prevent infiltration and exfiltration			2	ESF	29
The build pipeline infrastructures should be monitored for data leakage, particularly code repositories and engineering workstations			2	ESF	29
Gerenciamento de usuários					
User account management			1	LAD	7
Define user roles			1	CNCF	33
Determine an acceptable period for disabling/removing stale accounts and disable/remove any identity which has surpassed the predetermined period of inactivity			1	OW	9
Refrain from allowing users to self-register to systems, and grant permission on an as-needed basis			1	OW	10
Autenticação de usuários					
Ensure users must authenticate to access the build environment			2	CIS	31
Application build pipeline enforces authentication and defaults to deny			2	SCVS	10
Ensure each system requires multi-factor authentication			2	ESF	29
Secure authentication (e.g., Multi-Factor Authentication (MFA), password recycle, session timeout, token protection)			2	LAD	7
Prefer using temporary credentials over static credentials. In case static credentials need to be in use - establish a procedure to periodically rotate all static credentials and detect stale credentials			2	OW	25
Configure usage of credentials to be limited to predetermined conditions (like scoping to a specific source IP or identity) to ensure that even in case of compromise, exfiltrated credentials cannot be used outside your environment			2	OW	25
Avoid creating local user accounts. Instead, create and manage identities using a centralized organization component (IdP)			2	OW	10
Whenever local user accounts are in use, ensure that accounts which no longer require access are disabled/removed and that security policies around all existing accounts match the organization's policies			2	OW	10
Prevent employees from using their personal email addresses, or any address which belongs to a domain not owned and managed by the organization, against the SCM, CI, or any other CI/CD platform			2	OW	10
Continuously monitor for non-domain addresses across the different systems and remove non-compliant users			2	OW	10
Permissões de usuários					
Conduct a continuous analysis and mapping of all identities across all systems within the engineering ecosystem. For each identity, map the identity provider, level of permissions granted and level of permissions actually used. Ensure all methods of programmatic access are covered within the analysis			2	OW	9
Establish procedures to continuously map credentials found across the different systems in the engineering ecosystem - from code to deployment. Ensure each set of credentials follows the principle of least privilege and has been granted the exact set of permission needed by the service using it			2	OW	25
Remove permissions not necessary for the ongoing work of each identity across the different systems in the environment			2	OW	9
Continuously map all external collaborators and ensure their identities are aligned with the principle of least privilege			2	OW	10
Whenever possible, grant permissions with a predetermined expiry date - for both human and programmatic accounts - and disable their account once the work is done			2	OW	10
Refrain from granting base permissions in a system to all users, and to large groups where user accounts are automatically assigned to			2	OW	10
Avoid using shared accounts. Create dedicated accounts for each specific context, and grant the exact set of permissions required for the context in question			2	OW	10
Ensure permissions to the pipeline execution nodes are granted according to the principle of least privilege. A common misconfiguration in this context is around granting debug permissions on execution nodes to engineers. While in many organizations this is a common practice, it is imperative to take into consideration that any user with the ability to access the execution node in debug mode may expose all secrets while they are loaded into memory and use the node's identity - effectively granting elevated permissions to any engineer with this permission			2	OW	28
Processo de build					
Provenance exists describing how the artifact was built, including the build platform, build process, and top-level inputs. (may be incomplete or unsigned)			3	SLSA	41
Ensure all artifacts on all releases are signed		Sign Every Step in the Build Process Code signing Implement processes and technologies to validate the integrity of resources all the way from development to production. When a	3	CIS, CNCF, LAD, OW	38, 34, 7, 33
	Follow estabilished practices for estabilishing a root of trust from an offline source		3	CNCF	33

	Use short-lived Workload Certificates		3	CNCF	34
	Implement strong authentication methods such as strong passwords, certificates, two factor authentication (MFA), and physical access control to protect the signing infrastructure		3	ESF	33
	Control user access to the signing infrastructure, using least privileges		3	ESF	33
	Conduct code signing on a physically isolated network segment		3	ESF	33
	Use intrusion detection and protection systems in the code-signing environment		3	ESF	33
	Deploy and use a Security Information and Event Management (SIEM) system		3	ESF	33
	Use cryptography in transit and at rest		3	ESF	33
	Apply hardening procedures on the signing environment systems		3	ESF	33
	Use centralized log servers (with append only, encryption, etc.)		3	ESF	33
	Ensure the signing system meets baseline security standards		3	ESF	33
	Require multi-party approval for physical and remote access and log all access		3	ESF	33
	Grant super-user to only a small number of signing system admins		3	ESF	34
	Validate the Signatures Generated at Each Step		2	CNCF	35
	Use TUF/Notary to manage signing of artefacts		2	CNCF	35
	Use a store to manage metadata from in-toto		2	CNCF	35
Use minimal set of trusted build dependencies in the release job			2	LAD	7
Ensure all external dependencies used in the build process are locked		Lock and Verify External Requirements from the Build Process	2	CIS, CNCF	39
The build service must fetch all artifacts in a trusted control plane			2	ESF	31
All build-time manipulations to source or binaries are known and well defined			1	SCVS	11
Ensure dependencies are validated before being used		The build service must verify the integrity of each artifact Integrity check of dependencies through cryptographic hashes Prior to consuming the resource in subsequent steps down the pipeline, the resource's integrity should be validated against the signing authority	2	CIS, ESF, LAD, OW	39, 31, 7, 33
	Usage of tools for signing and verification of code and artifacts provide a way to prevent unverified software from being delivered down the pipeline. Examples for such projects are in-toto, SLSA, and Sigstore, all part of the Linux Foundation		3	OW	34
	3rd party resources fetched from build/deploy pipelines (such as scripts imported and executed as part of the build process) should follow a similar logic - prior to using 3rd party resources, the hash of the resource should be calculated and cross referenced against the official published hash of the resource provider		3	OW	34
	Prefer allowing artifacts to flow through the pipeline only in the condition that they were created by a pre-approved CI service account		3	OW	6
	Prevent artifacts that have been uploaded by other accounts from flowing through the pipeline without secondary review and approval		3	OW	6
Validate environments and dependencies before usage				CNCF	28
Ensure the build pipeline creates reproducible artifacts		Validate Build Artefacts through verifiably reproducible builds Application uses a repeatable build Reproducible builds	3	CIS, CNCF, SCVS, LAD	39, 28, 10, 7
	Rebuild the OSS in a trusted build environment, or validate that it is reproducibly built		3	S2C2F	18
	Documentation exists on how the application is built and instructions for repeating the build		3	SCVS	10
	Estabilish and maintain an authoritative source and repository to provide a trusted source and accountability for approved and implemented system components		3	ESF	31
	Employ automated mechanisms to detect misconfigured or unauthorized system components		3	ESF	32
	Employ automated discovery and management tools to maintain an up-to-date, complete, accurate, and readily available inventory of system components		3	ESF	32

	Identify and authenticate before estabilishing a network connection using bidirectional authentication that is cryptographically based and replay resistant		3	ESF	32
	Employ automated mechanisms for the generation, protection, rotation, and management of passwords for systems and system components that do not support multi-factor authenticationi or complex account management		3	ESF	32
	Employ automated or manual/procedural mechanisms to prohibit system components from connecting to organizational systems unless the components are known, authenticated, ina properly configured state, or in a trust profile		3	ESF	32
	Employ a means to allow the comparison of binaries built from two or more disparate, segmented, protected, and secured environments		3	ESF	32
Geração de SBOMs					
Software producer distributes provenance to consumers, preferably using a convention determined by the package ecosystem.			1	SLSA	41
Ensure pipeline steps produce a Software Bill of Materials (SBOM)		Generate an immutable SBOM fo the code Software bill of materials are generated for publicly or commercially available applications	2	CIS, CNCF, SCVS	40, 18, 8
	Software bill of materials continuously maintained and current for all systems		3	SCVS	8
	A structured, machine readable software bill of materials (SBOM) format is present		1	SCVS	9
	SBOM creation is automated and reproducible		2	SCVS	9
	Each SBOM has a unique identifier		1	SCVS	9
	SBOM is timestamped		1	SCVS	9
	SBOM contains a complete and accurate inventory of all components the SBOM describes		1	SCVS	9
	SBOM contains an accurate inventory of all test components for the asset or application it describes		2	SCVS	9
	SBOM contains metadata about the asset or software the SBOM describes		2	SCVS	9
	Component identifiers are derived from their native ecosystems (if applicable)		1	SCVS	9
	Component point of origin is identified in a consistent, machine readable format (e.g. PURL)		3	SCVS	9
	Components defined in SBOM have accurate license information		1	SCVS	9
	Components defined in SBOM have valid SPDX license ID's or expressions (if applicable)		2	SCVS	9
	Components defined in SBOM have valid copyright statements		3	SCVS	9
	Components defined in SBOM which have been modified from the original have detailed provenance and pedigree information		3	SCVS	9
	Components defined in SBOM have one or more file hashes (SHA-256, SHA-512, etc)		3	SCVS	9
	The details of an integrated third-party component should be reported in an SBOM for the product developed		3	ESF	26
	Checksums of all components are accessible and delivered out-of-band whenever those components are packaged or distributed		2	SCVS	11
	Checksums of all first-party and third-party components are documented for every build		1	SCVS	11
Downstream verification of provenance includes validating the authenticity of the provenance			2	SLSA	42
Build platform implements strong controls to prevent secret material used to sign the provenance from being accessible to the user-defined build steps.			3	SLSA	43
Ensure pipeline steps sign the SBOM produced			3	CIS	40
	SBOM has been signed by publisher, supplier, or certifying authority	Implement processes and technologies to validate the integrity of resources all the way from development to production. When a resource is generated, the process will include signing that resource using an external resource signing infrastructure	3	SCVS, OW	9, 33
	SBOM signature verification exists		3	SCVS	9
	SBOM signature verification is performed		3	SCVS	9
Ensure all artifacts are signed by the build pipeline itself			3	CIS	47
Conduct code signing on a physically isolated network segment			3	ESF	33
Gerenciamento de aplicativos de terceiros integrados à plataforma de build					

Establish vetting procedures to ensure 3rd parties granted access to resources anywhere across the engineering ecosystem are approved prior to being granted access to the environment, and that the level of permission they are granted is aligned with the principle of least privilege		3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Method of integration. Make sure all methods of integration for each system are covered (including marketplace apps, plugins, OAuth applications, programmatic access tokens, etc.).		3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Level of permission granted to the 3rd party		3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Level of permission actually in use by the 3rd party		3	OW	32
Ensure each 3rd party is limited and scoped to the specific resources it requires access to and remove unused and/or redundant permissions		3	OW	32
Periodically review all 3rd parties integrated and remove those no longer in use		3	OW	32
3rd parties which are integrated as part of the Build process should run inside a scoped context with limited access to secrets and code, and with strict ingress and egress filters		3	OW	32
Auditoria do ambiente				
Ensure the build environment is logged	Log all access to the build pipeline Record the Build Environment Application build pipeline maintains a verifiable audit log of all build job	3	CIS, ESF, CNCF, SCVS	29, 29, 30, 10
Identify and build an inventory of all the systems in use within the organization, containing every instance of these systems (specifically relevant for self-managed systems e.g. Jenkins)		3	OW	36
Once all relevant systems are identified, the next step is ensuring that all relevant logs are enabled, as this is not the default state in the different systems. Visibility should be optimized around both human access as well as programmatic access through all the various measures it is allowed. It is important to place an equal level of emphasis on identifying all relevant audit log sources, as well as the applicative log sources		3	OW	36
Shipping logs to a centralized location (e.g. SIEM), to support aggregation and correlation of logs between different systems for detection and investigation		3	OW	36
Creating alerts to detect anomalies and potential malicious activity, both in each system on its own and anomalies in the code shipping process, which involves multiple systems and requires deeper knowledge in the internal build and deployments processes		3	OW	36
Ensure resource consumption of build workers is monitored	Deploy monitoring tools to software factory do detect malicious behaviour	2	CIS, CNCF	35, 34
Ensure changes to pipeline files are tracked and reviewed		3	CIS	36

CONTROLES - DEPLOY E DISTRIBUIÇÃO	EQUIVALÊNCIAS	NÍVEL DE MATURIDADE	FONTE	
			FRAMEWORK	PÁGINA
Estruturação do ambiente				
Secure the repository according to information security management standards or guidelines to ensure its integrity		1	ESF	36
Ensure access to production environment is limited		1	CIS	59
Ensure network access to the systems is aligned with the principle of least access		1	OW	28
Ensure webhooks of the package registry are secured		1	CIS	52
Ensure that transport layer security of the distribution system is configured properly to ensure the confidentiality, integrity and authenticity of information to be transfered		1	ESF	36
Package repository enforces use of TLS for all interactions		1	SCVS	12
Package manager validates TLS certificate chain to repository and fails securely when validation fails		1	SCVS	12
Use The Update Framework		3	CNCF	38
Gerenciamento de vulnerabilidades				
Manage new vulnerabilities associated with the package manager		2	ESF	36
Maintain an inventory of systems and versions in use, including mapping of a designated owner for each system		2	OW	28
Continuously check for known vulnerabilities in these components		2	OW	28
If a security patch is available, update the vulnerable component. If not, consider removing the component / system, or reduce the potential impact of		2	OW	28
Gerenciamento de usuários				
User account management		1	LAD	7
Ensure default passwords are not used		1	CIS	59
Ensure factor authorization to certify certain artifacts is limited		1	CIS	48
Determine an acceptable period for disabling/removing stale accounts and disable/remove any identity which has surpassed the predetermined period of inactivity		1	OW	9
Prevent employees from using their personal email addresses, or any address which belongs to a domain not owned and managed by the organization, against the SCM, CI, or any other CI/CD platform		1	OW	10
Continuously monitor for non-domain addresses across the different systems and remove non-compliant users		1	OW	10
Refrain from allowing users to self-register to systems, and grant permission on an as-needed basis		1	OW	10
Ensure user management of the package registry is not local	Avoid creating local user accounts. Instead, create and manage identities using a centralized organization component (IdP)	1	CIS, OW	50, 10
Whenever local user accounts are in use, ensure that accounts which no longer require access are disabled/removed and that security policies around all existing accounts match the organization's policies		1	OW	10
Ensure anonymous access to artifacts is revoked		1	CIS	50
Autenticação dos usuários				
Package repository requires strong authentication		2	SCVS	11
Ensure user access to the package registry utilizes Multi-Factor Authentication (MFA)	Secure authentication (e.g., Multi-Factor Authentication (MFA), password recycle, session timeout, token protection)	2	CIS, LAD	49, 7
Package repository supports multi-factor authentication component publishing		2	SCVS	11
Package repository components have been published with multi-factor authentication		3	SCVS	11
Avoid sharing the same set of credentials across multiple contexts. This increases the complexity of achieving the principle of least privilege as well as having a negative effect on accountability		2	OW	25
Prefer using temporary credentials over static credentials. In case static credentials need to be in use - establish a procedure to periodically rotate all static credentials and detect stale credentials		2	OW	25
Configure usage of credentials to be limited to predetermined conditions (like scoping to a specific source IP or identity) to ensure that even in case of compromise, exfiltrated credentials cannot be used outside your environment		2	OW	25
Permissões de usuários				
Limit which artefacts any given party is authorized do certify		3	CNCF	35
Build in a system for rotating and revoking private keys		3	CNCF	36
Ensure number of permitted users who may upload new artifacts is minimized		2	CIS	49
Conduct a continuous analysis and mapping of all identities across all systems within the engineering ecosystem. For each identity, map the identity provider, level of permissions granted and level of permissions actually used. Ensure all methods of programmatic access are covered within the analysis		2	OW	9
Remove permissions not necessary for the ongoing work of each identity across the different systems in the environment		2	OW	9
Continuously map all external collaborators and ensure their identities are aligned with the principle of least privilege		2	OW	10
Whenever possible, grant permissions with a predetermined expiry date - for both human and programmatic accounts - and disable their account once the work is done		2	OW	10

Refrain from granting base permissions in a system to all users, and to large groups where user accounts are automatically assigned to		2	OW	10
Avoid using shared accounts. Create dedicated accounts for each specific context, and grant the exact set of permissions required for the context in question		2	OW	10
Establish procedures to continuously map credentials found across the different systems in the engineering ecosystem - from code to deployment. Ensure each set of credentials follows the principle of least privilege and has been granted the exact set of permission needed by the service using it		2	OW	25
Segurança dos artefatos				
Ensure deployment configuration files are separated from source code		2	CIS	54
Ensure scanners are in place to identify and prevent sensitive data in deployment configuration	Verify that secrets are removed from any type of artifact, such as from layers of container images, binaries, or Helm charts	2	CIS, OW	55, 26
Ensure access to deployment configurations are limited to specific members		2	CIS	56
Ensure scanners are in place to secure Infrastructure as Code (IaC) instructions		2	CIS	56
Ensure deployment configuration manifests are verified		2	CIS	57
Ensure deployment configuration manifests are pinned to a specific, verified version		2	CIS	57
Ensure deployments are automated		2	CIS	58
Package manager documents package installation in machine-readable form		1	SCVS	12
Ensure the deployment environment is reproducible		3	CIS	58
Encrypt artefacts before distribution & ensure only authorized platforms have decryption capabilities		3	CNCF	36
Ensure artifacts are encrypted before distribution		3	CIS	47
Ensure only authorized platforms have decryption capabilities of artifacts		3	CIS	48
Ensure clients can perform Verification of Artefacts and associated metadata		3	CNCF	37
Usage of tools for signing and verification of code and artifacts provide a way to prevent unverified software from being delivered down the pipeline. Examples for such projects are in-toto, SLSA, and Sigstore, all part of the Linux Foundation		3	OW	34
The package including the correct metadata should be signed by a cryptographically-secure signature algorithm before being uploaded to the repository	Code signing	3	ESF, LAD, OW	35, 7, 33
	Implement processes and technologies to validate the integrity of resources all the way from development to production. When a resource is generated, the process will include signing that resource using an external resource signing infrastructure	3		
Ensure all signed artifacts are validated upon uploading the package registry	Package repository requires code signing to publish packages to production repositories	2	CIS, SCVS	51, 12
Ensure all versions of an existing artifact have their signatures validated		1	CIS	51
Package manager verifies the integrity of packages when they are retrieved from remote repository		1	SCVS	12
Package manager verifies the integrity of packages when they are retrieved from file system		1	SCVS	12
Alternatively, the package can include cryptographic hashes		1	ESF	35
Use Binary software composition analysis tools to investigate what exactly is included in the final deliveries		1	ESF	34
Ensure artifacts contain information about their origin	Package repository contents are congruent to an authoritative point of origin for open source components	1	CIS, SCVS	52, 11
Compare the binary analysis output and the other artifacts from the build process		3	ESF	34
Include the SBOMs within the final packages, with all primary (top level) components and their transitive dependencies listed		3	ESF	35
Binary components are retrieved from a package repository		1	SCVS	11
Use a container registry that supports OCI image-spec images		3	CNCF	36
Dependency resolution rules		2	LAD	7
Ensure private artifacts are not allowed to be pulled from external registries		2	CIS	53
Establish internal repository mirrors and reference one private feed, not multiple		2	LAD	7
Package repository supports security incident reporting		2	SCVS	11
Package repository automates security incident reporting		3	SCVS	12
Package repository notifies publishers of security issues		2	SCVS	12
Package repository notifies users of security issues		3	SCVS	12
Ensure clients can verify the "freshness" of files		2	CNCF	37
Package repository provides a verifiable way of correlating component versions to specific source codes in version control		2	SCVS	12

Package repository requires and/or performs static code analysis prior to publishing a component and makes results available for others to consume		3	SCVS	12
Package manager does not execute component code		1	SCVS	12
Scoped packages		3	LAD	7
Gerenciamento de aplicativos de terceiros				
Establish vetting procedures to ensure 3rd parties granted access to resources anywhere across the engineering ecosystem are approved prior to being granted access to the environment, and that the level of permission they are granted is aligned with the principle of least privilege		3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Method of integration. Make sure all methods of integration for each system are covered (including marketplace apps, plugins, OAuth applications, programmatic access tokens, etc.).		3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Level of permission granted to the 3rd party		3	OW	31
Introduce controls and procedures to maintain continuous visibility over all 3rd parties integrated to CI/CD systems, including: Level of permission actually in use by the 3rd party		3	OW	32
Ensure each 3rd party is limited and scoped to the specific resources it requires access to and remove unused and/or redundant permissions		3	OW	32
Periodically review all 3rd parties integrated and remove those no longer in use		3	OW	32
Auditoria do ambiente				
Ensure changes in deployment configuration are tracked		3	CIS	55
Detect and prevent drifts and inconsistencies between code running in production and its CI/CD origin, and modify any resource that contains a drift		3	OW	6
Establish a process to periodically review all system configurations for any setting that can have an effect on the security posture of the system, and ensure all settings are optimal		3	OW	28
Measures aimed at detecting configuration drifts (e.g. resources in cloud environments which aren't managed using a signed IAC template), potentially indicative of resources that were deployed by an untrusted source or process		3	OW	34
Ensure changes in package registry configuration are audited		3	CIS	51
Identify and build an inventory of all the systems in use within the organization, containing every instance of these systems (specifically relevant for self-managed systems e.g. Jenkins)		3	OW	36
Once all relevant systems are identified, the next step is ensuring that all relevant logs are enabled, as this is not the default state in the different systems. Visibility should be optimized around both human access as well as programmatic access through all the various measures it is allowed. It is important to place an equal level of emphasis on identifying all relevant audit log sources, as well as the applicative log sources		3	OW	36
Shipping logs to a centralized location (e.g. SIEM), to support aggregation and correlation of logs between different systems for detection and investigation		3	OW	36
Creating alerts to detect anomalies and potential malicious activity, both in each system on its own and anomalies in the code shipping process, which involves multiple systems and requires deeper knowledge in the internal build and deployments processes		3	OW	36
Package repository provides auditability when components are updated		1	SCVS	12