Comparação de abordagens de proteção à cadeia de suprimento de software

Article *in* RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação · July 2024

DOI: 10.5281/zenodo.10119798

5 authors, including:

Joao Gondim
University of Brasília
55 PUBLICATIONS **428** CITATIONS
SEE PROFILE

Rafael Rabelo Nunes
University of Brasília
50 PUBLICATIONS **87** CITATIONS
SEE PROFILE

risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Iberian Journal of Information Systems and Technologies

Recebido/Submission: 14/02/2024
Aceitação/Acceptance: 04/05/2024

Comparação de abordagens de proteção à cadeia de suprimento de *software*

Carlos Eduardo Miranda Zottmann[1], João José Costa Gondim[1],
Thiago Melo Stuckert do Amaral[1], Edvan Gomes da Silva[1], Rafael Rabelo Nunes[1]

carlos.zottmann@gmail.com; gondim@unb.br; thiago.melo.stuckert@gmail.com; edvan402@gmail.com; rafaelrabelo@unb.br.

[1] Departamento de Engenharia Elétrica – Universidade de Brasília (UnB), Brasília – DF – Brasil

Pages: 657-675

Resumo: A cadeia de suprimentos de *software* é um vetor de ataque em crescimento para incidentes cibernéticos complexos. A proteção contra ataques dessa natureza ainda é um problema em aberto para a comunidade internacional. Existem diversas abordagens definindo controles de segurança para mitigar riscos nas etapas da cadeia de suprimentos, porém não existe uma visão holística sobre essas abordagens. Esse estudo realiza um levantamento de taxonomias relativas ao tema e uma análise das abordagens de maneira a conseguir destacar as características e proporcionar uma melhor visibilidade da aplicabilidade dos controles propostos. Por fim, são providas comparações entre as abordagens com o intuito de fornecer insumos para organizações que desejam robustecer sua cadeia de suprimentos de *software*.

Palavras-chave: cadeia de suprimento de software; riscos cibernéticos; ataques; controles de segurança

Comparison of Software Supply Chain Protection Approaches

Abstract: The software supply chain is a hot spot for complex cyber incidents. Protection against such attacks is still an open problem for the international community. There are several approaches defining security controls to mitigate risks in the supply chain stages, however there is no holistic view of these approaches. This study carries out a survey of taxonomies on the subject and an analysis of the approaches to highlight the characteristics and provide better visibility of the applicability of the proposed controls. Finally, comparisons between approaches are provided with the aim of providing input for organizations wishing to strengthen their software supply chain.

Keywords: software supply chain; cyber risks; attacks; security controls

1. Introdução

A cadeia de suprimento de *software* é cada vez mais explorada como vetor de ataque em incidentes cibernéticos (Lella et al., 2021), sendo um dos grandes desafios da

comunidade internacional destacado pelo diretor executivo da agência norte-americana de cibersegurança, David Koh, no relatório "*Global Cybersecurity Outlook* 2022" do Fórum Econômico Mundial (Pipikaite et al., 2022).

Se há algum tempo ataques à cadeia de suprimento de *software* exploravam vulnerabilidades descobertas em produtos já existentes, atualmente esses ataques têm buscado inserir código malicioso em um produto durante as etapas de sua construção, de forma que esse código (e suas vulnerabilidades associadas) sejam distribuídos aos consumidores de forma em princípio legítima (Enduring Security Framework, 2022b).

Como exemplos notórios desse tipo de incidente podemos citar o que ocorreu com a empresa norte-americana Solar Winds (Peisert et al., 2021) e a vulnerabilidade da biblioteca de código aberto Log4J (Wetter & Ringland, 2021). Considerando essa tendência, diversas organizações públicas e privadas publicaram *frameworks*, guias, processos e ferramentas com medidas para se proteger desse tipo de ameaça. Essas ferramentas e boas práticas devem ser incorporadas em diferentes pontos do ciclo de desenvolvimento de fornecedores e consumidores de software. Entretanto, ainda não há consenso sobre quais são as melhores abordagens para se mitigar o risco de exploração desse tipo de vulnerabilidade.

Dessa forma, o objetivo deste artigo é comparar diversas dessas abordagens de forma a oferecer uma visão global dos desafios e oportunidades para enfrentar essa ameaça. Para isso, o artigo está organizado desta maneira: além desta introdução, há um referencial teórico onde encontram-se as definições sobre ataques à cadeia de suprimentos e taxonomias relativas ao assunto (Seção 2), seguida da identificação dos *frameworks*, guias, processos e ferramentas voltados à proteção da cadeia de suprimento de *software* (Seção 3), da relação de ferramentas e procedimentos que podem auxiliar na implementação das boas práticas preconizadas pelos frameworks detalhados na Seção 3, e, por fim, a comparação das abordagens (Seção 4).

2. Referencial Teórico

De forma a melhor delimitarmos o escopo deste artigo, faz-se necessário definir o que são ataques à cadeia de suprimento 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 (CNCF, 2021).

Ataques à cadeia de suprimento de software podem ocorrer de algumas formas distintas, e de taxonomias de ataque podem auxiliar na compreensão tanto do cenário de ameaças à cadeia de suprimento de *software* quanto nas abordagens de proteção. Identificou-se, na literatura, propostas de taxonomias que versam sobre ambos os assuntos.

O trabalho "*Taxonomy of Attacks on Open-Source Software Supply Chains*" (Ladisa et al., 2022) propõe uma taxonomia detalhada de ataques à cadeia de suprimento de *software*, a partir de uma revisão sistemática da literatura, validada por pesquisas

realizadas junto a especialistas no domínio da segurança e a desenvolvedores de *software*, esses últimos por serem grandes consumidores desse tipo de produto.

O trabalho foi ampliado em uma segunda publicação, "*Journey to the Center of Software Supply Chain Attacks*" (Ladisa et al., 2023). Enquanto o primeiro trabalho havia identificado 107 vetores únicos de ataque, a quantidade chegou a 117 no segundo. Em ambos, foram identificados 33 controles de segurança aplicáveis para a prevenção desses vetores.

Os 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* (SAP SE, 2023), de forma que possa ser dinamicamente consultada por qualquer interessado.

O primeiro nível da árvore de ataques já nos traz os principais tipos de ataques conhecidos contra cadeias de suprimento de *software*: Desenvolver e Anunciar Novos Pacotes Maliciosos, Criar Confusão de Nomes contra Pacotes Legítimos, e Subverter Pacotes Legítimos.

Adicionalmente, relaciona 33 controles de segurança aos 117 vetores de ataque, indicando 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).

Por fim, o trabalho cita um artigo (Ohm et al., 2020) que relaciona os principais objetivos de ataques à cadeia de suprimento de *software*: ativação de Shell reverso, instalação de *droppers*, exfiltração de dados, negação de serviço e ganho financeiro.

Um cotejamento entre a taxonomia proposta por esses trabalhos e as taxonomias sugeridas pelos *frameworks* identificados nos permite consolidar quatro pilares ou ambientes principais referentes à cadeia de suprimento de software: fontes, dependências, ambientes e processos de compilação, e ambientes e processos de distribuição de software.

Em outra abordagem, o artigo "*What is Software Supply Chain Security?*" (Melara & Bowman, 2022) propõe uma taxonomia quanto aos objetivos das soluções de segurança para a cadeia de suprimentos de software, dividida em dois níveis. Por simplicidade, citamos os três objetivos do primeiro nível, a saber: Objetivos para estabelecimento de confiança; Objetivos para Ferramentas resilientes; e Objetivos para processos resilientes.

As duas taxonomias aqui apresentadas nos permitem melhor compreender o cenário de segurança relativo à software supply chain, classificando os tipos de ataques hoje conhecidos, e os objetivos dos *frameworks* e soluções de segurança disponíveis.

As iniciativas que visam a proteção da cadeia de suprimento de software contra os tipos de ataques identificados pelas taxonomias aqui mencionadas podem ser divididas em *frameworks* de boas práticas em desenvolvimento, que propõem padrões a serem observados para contribuir com a segurança da cadeia de suprimento de software, e em ferramentas e procedimentos que podem ser utilizados para a implementação desses padrões.

3. *Frameworks* de boas práticas em desenvolvimento

Neste tópico apresentaremos os *frameworks* que identificamos na literatura como sendo voltados à implementação de boas práticas em desenvolvimento seguro referentes à cadeia de suprimento de software, resumindo suas principais características, de forma a facilitar a tarefa de comparação. Esses *frameworks* são estruturados em torno de um ou mais dos quatro pilares da cadeia de suprimento de software que citamos na seção anterior.

Não serão incluídos aqui, em função da restrição de escopo, documentos referentes à cadeia de suprimentos cibernéticas como um todo, tais como o *Cybersecurity Supply Chain Risk Management Practices* do NIST (Boyens et al., 2022).

Os *frameworks* serão avaliados em função do público-alvo ao qual são destinados, bem como em função da possibilidade de automação da verificação de suas automações e da existência de modelos de maturidade que possam guiar sua adoção paulatina por parte das organizações interessadas.

3.1. Secure Software Development Framework (SSDF)

O *Secure Software Development Framework* (SSDF) (Souppaya et al., 2022), é um modelo que traz recomendações de segurança de alto nível a serem integradas ao ciclo de desenvolvimento de *software* (SDLC). Tais recomendações também podem ser utilizadas para a definição de requisitos de segurança junto a fornecedores, e aquisição de software aderentes às práticas recomendadas.

O SSDF possui como conceitos principais: as organizações devem garantir que pessoas, processos e tecnologia estejam preparadas para o desenvolvimento seguro de software; Devem proteger todos os componentes de seus software contra modificações e acessos não autorizados; Devem produzir software seguro, contendo o mínimo possível de vulnerabilidades em suas versões disponibilizadas; Devem identificar vulnerabilidades residuais e responder a elas de forma apropriada, assim como prevenir ocorrências similares no futuro.

O modelo não especifica como cada prática deve ser implementada, mais sim os resultados esperados a partir de cada uma delas. Com isso, recomenda que as organizações que o adotem consultem suas referências para detalhamento sobre as implementações.

O SSDF recomenda uma série de práticas associadas a cada um de seus conceitos principais. Cada prática é descrita em termos de sua definição, tarefas que devem ser executadas, exemplos de implementação e referências. Em sua versão atual, traz uma relação total de 19 práticas subdivididas em um total de 42 tarefas.

3.2. Secure Supply Chain Consumption Framework (S2C2F)

O S2C2F tem como objetivo definir um processo de garantia de segurança e redução de riscos para o consumo, pelo trabalho de desenvolvimento, de dependências de Software de Código Aberto, tais como NuGet e NPM, incluindo código fonte, pacotes de idiomas, módulos, componentes, contêiners, bibliotecas ou binários.

Por meio de uma abordagem de redução de riscos baseada em ameaças, os objetivos específicos do S2C2F são:

- Prover um programa robusto de governança de uso de Software de Código Aberto;
- Diminuir o Tempo Médio de Correção (MTTR) de vulnerabilidades conhecidas em *Software* de Código Aberto;
- Prevenir o consumo de pacotes de código aberto maliciosos ou que tenham sido comprometidos.

O S2C2F é construído em torno de três conceitos principais, a saber:

- Controlar as formas de obtenção de todos os artefatos. Os desenvolvedores tipicamente fazem uso de uma variedade de formas de obtenção das dependências que utilizam: clones a partir do Git, *download* via wget, copiar e colar código fonte, *check-in* de binários no repositório central da organização, consumo direto a partir de gerenciadores públicos de pacotes, reempacotamento de software de código aberto em arquivos .zip, curl, apt-get, sub-módulos git, e outros. Faz-se necessário padronizar o 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.
- Aprimoramento contínuo do processo. O S2C2F foi organizado em um modelo de maturidade, de forma a auxiliar as organizações na priorização dos requerimentos que devam ser implementados. Adicionalmente, coloca ênfase na compreensão de novas ameaças, e requer avaliação regular dos controles implementados, de forma a garantir a segurança contínua em função de avanços na tecnologia ou do surgimento de novas ameaças.
- Escala. Considerando as questões envolvidas na escala de consumo de *software* de código aberto por uma organização, de forma a evitar a sobrecarga de trabalho que seria necessária para a implantação e manutenção de estruturas centralizadas, o S2C2F prevê ferramentas para 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.

É composto por oito práticas de segurança: Internalize os artefatos; Verifique as vulnerabilidades; Inventarie; Atualize; Audite; Garanta a segurança; Recompile a partir dos fontes; e Corrija e Comunique ao mantenedor.

O S2C2F propõe um modelo de maturidade de quatro níveis, de forma a permitir que as organizações realizem progressos incrementais a partir de seu estágio de segurança atual até o estágio 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;

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

Em 2021, um grupo técnico da CNCF criou um guia de boas práticas para a proteção da cadeia de suprimentos (Cloud Native Computing Foundation, 2021). Esse guia aborda controles para a proteção do código-fonte, das dependências, do processo de compilação, dos artefatos de software e do processo de entrega de software.

O guia possui quatro princípios para a proteção da cadeia de suprimentos:

- O primeiro é que a cadeia deve ser confiável. Essa confiança é adquirida por meio de verificações de atestados emitidos por primitivas criptográficas.
- O segundo ponto primordial é a automação. Esse fator diminui o risco de intervenção humana durante o processo de desenvolvimento.
- O terceiro fator é o controle do ambiente de compilação. Os usuários e ferramentas devem ter permissões restritas nesse ambiente.
- O quarto princípio está relacionado com uma robusta gestão de identidade. Todas as entidades envolvidas devem ter suas identidades verificadas mutuamente utilizando uma autenticação forte.

Define ainda cinco estágios para a segurança da cadeia de suprimento de software: Segurança do código-fonte; Segurança dos componentes; Segurança dos processos de compilação (*build pipelines*); Segurança dos artefatos; Segurança das implantações.

Na proteção do código fonte são enfatizados controles sobre a verificação das assinaturas do código submetidos para a plataforma de versionamento. Desta forma, é garantida a autoria do código utilizado na solução de software. Além de aspectos de controle do ambiente de desenvolvimento como estabelecimento de papéis e responsabilidades dos membros da equipe.

Outro ponto importante é o estabelecimento de uma autenticação segura com múltiplos fatores. Assim como a verificação das bibliotecas de terceiros utilizadas no desenvolvimento. Por fim, podemos citar a proteção da esteira de desenvolvimento garantido a segurança dos processos de compilação e entrega dos artefatos produzidos.

3.4. *Supply-chain Levels for Software Artifacts* (SLSA)

O SLSA (Open Source Security Foundation (OpenSSF), 2023) foi desenhado de maneira a apresentar provas verificáveis de que a cadeia de suprimentos foi protegida. É utilizado pelo Google internamente desde 2013 e é obrigatório para todos os projetos que são lançados em produção.

Define três perspectivas que devem ser analisadas para a proteção da cadeia de suprimentos: “código”, “compilação” e “dependências”. O código deve ser protegido contra modificações não autorizadas. A compilação deve robusta para garantir que seja reproduzível. E as dependências devem ser checadas.

É estruturado em termos de trilhas e seus respectivos níveis. No momento o modelo definiu níveis apenas para a trilha de compilação (*build*), porém já prevê as trilhas de código fonte e de plataformas de compilação (*build platforms*). Em cada trilha

são definidos níveis, em ordem crescente, que expressam a qualidade das medidas de segurança implementadas.

Para a trilha de compilação (build) a versão atual especifica três níveis, 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). Quando não há qualquer implementação de segurança em uma determinada trilha, é utilizada a expressão SLSA 0.

3.5. *Center of Internet Security* (CIS) Software supply chain security guide

Em 2022, o CIS lançou um guia para proteção da cadeia de suprimentos de software (Center for Internet Security (CIS), 2022), propondo controles relacionados ao código-fonte, às dependências, à esteira de desenvolvimento, à implantação e aos artefatos. Tem como conceito principal impulsionar padrões que estejam surgindo no mercado, tais como a SLSA e a The Update Framework, provendo recomendações de segurança referentes às plataformas suportadas. Contém recomendações de segurança aplicáveis a cinco categorias principais: código fonte, esteiras de CI/CD, dependências, artefatos e implantação.

Na proteção do código, são elencadas uma série de medidas como gestão de mudanças, gerenciamento de repositórios, controle de acesso dos *commits,* controle de fornecedores. Já em relação aos controles da esteira de desenvolvimento podemos destacar a proteção do ambiente de compilação, dos agentes envolvidos, das etapas e a integridade do *pipeline*. É aconselhado que sejam conduzidas verificações nos pacotes de dependências de terceiros. Também são sugeridas configurações seguras para o ambiente de implantação e medidas de rastreabilidade de origem e verificação dos acessos aos artefatos entregues.

3.6. *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 cadeia de segurança de software. Nesse contexto, produziu guias voltados a diferentes partes interessadas na cadeia de suprimento de *software*, a saber:

- Securing the Software Supply Chain for Customers(Enduring Security Framework, 2022a);
- Securing the Software Supply Chain for Developers(Enduring Security Framework, 2022b);
- Securing the Software Supply Chain for Suppliers(Enduring Security Framework, 2022c).

Esses documentos trazem cenários de ameaças e ações de mitigação recomendadas para cada etapa das atividades ligadas à cadeia de suprimento 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 derivados dos *frameworks* NIST SSDF e SLSA.

3.7. OWASP Software Component Verification Standard (SCVS)

O SVCS se anuncia como sendo um esforço da comunidade, em busca de definição de um *framework* para identificar atividades, controles e melhores práticas que contribuam com o incremento da segurança da cadeia de suprimento de *software*. Defende que a cadeia de suprimento de *software* envolve tecnologia, pessoas, processos, instituições e outras variáveis. É desenhado para ser incrementado em etapas, permitindo que as organizações implantem controles de forma incremental.

É organizado em torno de seis famílias de controles, cuja verificação pode ser automatizada: Inventário, Lista de componentes de software (*Software bill of materials* – SBOM), Ambiente de compilação, Gerenciamento de Pacotes, Análise de Componentes, e Origem e linhagem. Pode ser utilizado para a avaliação da capacidade interna de uma organização no tocante à segurança de seu processo de desenvolvimento com relação à cadeia de suprimento de software, como também para a avaliação de fornecedores, com base em sua documentação e metadados dos projetos, o que inclui a divulgação de SBOMs por parte desses.

A Tabela 1 oferece um resumo das características dos *frameworks* apresentados.

Framework	Público-alvo	Responsável	Tipo	Modelo de maturidade	Automa-tizável	Correlações
SSDF	Clientes, Desenvolvedores, Fornecedores	NIST	Framework	Não	Não	BSIMM, OWASP SAMM, NIST SP 800-53 e NIST SP 800-161
S2C2F	Desenvolvedores	Microsoft	Framework	Sim	Não	SCITT; CIS Software Supply Chain Security Guide, NIST SSDF,; NIST SP 800-161r1; OWASP SCVS; CNCF Software Supply Chain Best Practices; SLSA

Framework	Público-alvo	Responsável	Tipo	Modelo de maturidade	Automa-tizável	Correlações
Software Supply Chain Best Practices	Desenvolvedores	CNCF	Guia	Não (**)	Não	Dependency Track, Sigstore, SLSA, OSSF Open Source Project Criticality Score, OSSF Scorecard.
SLSA	Clientes Desenvolvedores Fornecedores	Google	Guia	Sim	Sim	In-toto
CIS Software supply chain security guide	Desenvolvedores	Center for Internet Security (com Aqua Security, Microsoft, Paypal, Red Hat, CyberArk, Axonius e Sysdig)	Guia	Não	Não	SLSA; The Update Framework
OWASP SCVS	Clientes, Desenvolvedores Fornecedores	OWASP	Framework	Não (*)	Sim	NIST 800-161 Organizations
ESF – Securing the Software Supply Chain	Clientes, Desenvolvedores Fornecedores	Enduring Security Framework	Guia	Não	Não	NIST SSDF e SLSA

(*) Porém provê três níveis de verificação, que podem ser utilizados para a certificação no SCVS.
(**) Entretanto provê um modelo de avaliação.

Tabela 1 – Resumo dos *frameworks*

4. Ferramentas e procedimentos

Além dos *frameworks* apresentados na seção anterior, identificamos também ferramentas e procedimentos desenvolvidos para prover funcionalidades voltadas à segurança da cadeia de suprimento de software.

Assim como no caso dos *frameworks,* serão avaliados de acordo com seu público-alvo e, adicionalmente, com relação às funcionalidades que possuem, classificadas entre atestação de nível de segurança, atestação do nível de conformidade com o processo, e comprovação de integriadade.

4.1. in-toto

In-toto (The Linux Foundation, 2023a) é um sistema para assegurar a forma como um software é desenvolvido, construído, testado e empacotado, e atesta a integridade

e possibilidade de verificação de todas as ações executadas durante as etapas de desenvolvimento, compilação, testes e implantação de software.

Tem como objetivo prover integridade, autenticidade e auditabilidade à cadeia de suprimento de software como um todo, de forma que todas as etapas na cadeia de suprimento sejam bem definidas, que tenham explícitas as partes envolvidas em cada etapa, e que sejam executadas de acordo com os requisitos especificados pelo responsável pelo produto.

Seus principais conceitos são os seguintes:

- Integridade e autenticação do produto final
  - O produto recebido pelo cliente foi produzido pela equipe correta
- Auditabilidade e conformidade do processo
  - O produto recebido pelo cliente seguiu a especificação definida pelo responsável pelo projeto
- Rastreabilidade e atestação
  - As condições sob as quais cada etapa da cadeia de suprimento foi executada podem ser identificadas, assim como os materiais utilizados e os produtos resultantes
- Autenticação das etapas
  - Os responsáveis pela execução das diferentes etapas na cadeia de suprimento provêem evidência da execução utilizando um identificador que não pode ser forjado
- Separação de tarefas e privilégios
  - As diferentes etapas que compõem a cadeia de suprimentos podem ser atribuídas a diferentes responsáveis

O in-toto não preconiza boas práticas de desenvolvimento seguro. É, na verdade, uma especificação de solução de atestação de que as boas práticas definidas pelo responsável pelo produto tenham sido efetivamente seguidas em toda a cadeia de suprimentos.

Atualmente já há produtos disponíveis no mercado que implementam a especificação do in-toto, tais como a ferramenta Witness (Testifysec, 2023), desenvolvida pela empresa Testifysec, e integrável com as soluções e padrões GitLab, GitHub, AWS, GCP, Maven e JWT.

4.2. Sigstore

Sigstore (The Linux Foundation, 2023b) é um projeto de *software* de código aberto que tem por objetivo incrementar a segurança da cadeia de suprimento de *software*. É um conjunto de *framework* e ferramentas que possibilitam que desenvolvedores e clientes assinem de forma segura e verifiquem artefatos de *software* e arquivos de versão, imagens de contêiner, arquivos binários, lista de componentes de *software* (SBOMs) e outros tipos de documento.

Se propõe a resolver as fragilidades do processo tradicional de assinatura digital de artefatos. Assim, ao invés de se basear no modelo de assinaturas baseado em pares de chave pública e privada, onde há desafios relativos ao gerenciamento das chaves

envolvidas, e da revogação dessas chaves quando necessário, adota um modelo baseado em identidade do assinador.

Nesse modelo é utilizado o conceito de chaves efêmeras, utilizadas para um único evento de assinatura, eliminando a necessidade de gerenciamento de chaves. Os eventos de assinatura são gravados em um log transparente público, resistente a modificações não autorizadas, de forma a possibilitar a auditoria dos eventos de assinatura.

O processo se inicia a partir de um software cliente do Sigstore, como o Cosign. É realizada uma requisição de um certificado de assinatura de software à autoridade certificadora que compõe o Sigstore. A requisição é efetuada por meio do fornecimento de um token de identidade vinculado ao OpenID Connect. A autoridade certificadora é implementada por meio do software Fulcio.

A autoridade certificadora emite um certificado efêmero associado ao token de identidade fornecido. Ao assinar um artefato, um resumo (*digest*) desse artefato, a assinatura e o certificado são gravados em um repositório de log transparente persistente, denominado Rekor. Dessa forma, os eventos de assinatura podem ser auditados.

Para verificar um artefato, o cliente do Sigstore deve utilizar a chave pública contida no certificado, verificar se a identidade associada ao certificado coincide com a identidade esperada, verifica a assinatura do certificado a partir da autoridade certificadora do Sigstore, e por fim verifica o registro do evento de assinatura no Rekor. Esse procedimento garante tanto a origem do artefato quanto sua integridade.

4.3. Compilação determinística.

Compilação Determinística (Software Freedom Conservancy, 2023) é um projeto que tem como objetivo oferecer meios para que qualquer pessoa possa verificar que nenhuma falha foi introduzida no processo de construção de um produto de software, reproduzindo byte por byte seus pacotes binários a partir de uma determinada fonte.

Uma construção (compilação) é determinística (ou reproduzível) quando, a partir de uma determinada fonte, ambiente de compilação e instruções de compilação, qualquer um possa criar cópias idênticas de todos os artefatos especificados. Essa propriedade normalmente é obtida por meio do cálculo de *hashes* utilizando algoritmos seguros. Os atributos relevantes desses componentes são definidos pelos autores ou distribuidores.

Para o projeto, os seguintes conceitos são relevantes:

- Código fonte: normalmente é um "*checkout*" de uma versão específica a partir de um sistema de controle de versões, ou a partir de um arquivo de códigos fonte
- Artefatos: incluem executáveis, pacotes de distribuição, ou imagens do sistema de arquivos. Normalmente não incluem logs de compilação ou saídas auxiliares
- Autores ou distribuidores: são partes (indivíduos ou organizações) que queiram demonstrar a integridade de um conjunto de artefatos. Normalmente incluem autores de software ou mantenedores de distribuições.
- Para que se obtenha compilações determinísticas é importante que os ambientes de compilação original e aquele onde esteja sendo realizada uma segunda compilação para a verificação de integridade sejam o mais semelhantes possível. O projeto relaciona dezesseis variações em variáveis de ambiente que

podem influenciar negativamente na obtenção de compilações determinísticas: Metadados contidos em arquivamentos, Informações sobre a arquitetura dos sistemas, Identificador do Build, Caminho da compilação, "Timestamp" da compilação, Codificação dos arquivos, Ordenação do sistema de arquivos, Permissões nos arquivos, Informações de localização (locale), Dependências de pacotes, Aleatoriedade, Referências a endereços de memória, Codificação de Snippets, Nome DNS do sistema, Memória não inicializada, Informações do usuário.

4.4. Trusted Attestation and Compliance for Open Source (TACOS)

TACOS (Tidelift, 2023) é um *framework* que tem como objetivo avaliar as práticas de desenvolvimento de projetos de código aberto a partir de um conjunto de padrões de desenvolvimento definidos pelo Secure Software Development Framework (SSDF) especificado pelo NIST.

Define uma especificação interpretável por software que fornecedores de software podem utilizar como parte de sua documentação de auto atestação de conformidade com os requisitos definidos pelo memorando OMB M-22-18, emitido pela presidência dos EUA, referentes ao incremento da segurança da Cadeia de Suprimento de Software.

O conceito principal do *framework* é o estabelecimento de parcerias com mantenedores de software de código aberto para a implementação e atestação de que seus produtos sigam um conjunto definido de padrões de desenvolvimento seguro, incluindo os previstos pelo NIST SSDF.

Os mantenedores parceiros são remunerados pela Tidelift, de forma que o acesso às informações sobre a atestação de projetos às práticas de segurança definidas é sujeito ao pagamento de uma assinatura.

As categorias de práticas de desenvolvimento seguro preconizadas pelo NIST SSDF, e, consequentemente, objetos de atestação pelo *framework* TACOS, são:

- Preparar a organização: garantir que as pessoas, processos e tecnologias da organização estejam preparados para executar o desenvolvimento seguro de software
- Proteger o software: as organizações devem proteger todos os componentes de seu software contra modificações e acessos não autorizados
- Produzir software seguro: as organizações devem produzir software com um mínimo de vulnerabilidades
- Responder a vulnerabilidades: as organizações devem identificar vulnerabilidades residuais em seus software, e devem possuir processos e responsabilidades para responder apropriadamente

A partir dessas boas práticas, o TACOS emite uma atestação, que é um arquivo digitalmente assinado, representando estrutura de dados simples que contém os metadados de atestação e informações de atestação referentes às práticas de desenvolvimento seguro relativas aos pacotes de software que componham o produto.

4.5. OpenSSF Scorecard

O OpenSSF Scorecard (OpenSSF, 2023) tem como principal objetivo identificar a postura de segurança de um projeto de *software* e avaliar os riscos que suas dependências introduzem.

Tem como público-alvo os mantenedores de projetos de software, que podem utilizar suas funcionalidades para incrementar o nível de segurança de seus produtos, e, assim, atrair um maior número de usuários. Organizações também podem se beneficiar do projeto, incluindo-o em sua esteira de integração contínua para validar a postura de segurança das dependências utilizadas. E, por fim, os clientes finais, que podem consultar as informações publicadas pelo Scorecard para avaliar a postura de segurança dos projetos que publiquem sua avaliação pelo Scorecard.

Essas informações são resultado de verificações que o Scorecard executa sobre mais de um milhão de projetos de software de código aberto. Atualmente esses testes são executados apenas sobre projetos publicados na plataforma GitHub. A depender da opção dos mantenedores dos projetos, a nota alcançada pode ser exibida na página do projeto, por meio dos "badges" da plataforma.

O Scorecard verifica, de forma automatizada, vulnerabilidades que afetem diferentes partes da cadeia de suprimento de software, incluindo código fonte, compilação, dependências, testes e aspectos de manutenção do projeto.

Cada verificação gera uma avaliação numérica em uma escala de 0 a 10, e um nível de risco associado. O nível de risco, baseado na facilidade de exploração da falha encontrada, adiciona um peso à avaliação numérica, e o conjunto das verificações produz uma avaliação numérica agregada, final, que então exprime a postura de segurança do projeto.

Adicionalmente, provê recomendações para a mitigação dos problemas encontrados, de forma a incrementar a segurança do processo de desenvolvimento.

As verificações levam em consideração melhores práticas de segurança e padrões da indústria.

Na versão atual a ferramenta realiza 18 verificações, subdivididas em 3 temas: práticas de segurança holísticas, avaliação de riscos das fontes, e avaliação de riscos do processo de compilação, conforme detalhamento a seguir:

- Práticas de segurança holísticas: Vulnerabilidades, Ferramenta de atualização de dependências, Se o projeto é manutenido, Política de segurança, Licença, Melhores Práticas de Integração Contínua, Testes de integração contínua, *Fuzzing* e Análise estática de código fonte (SAST).
- Avaliação de riscos das fontes: Artefatos Binários, Proteção de "Branch", *Workflow* perigoso, Revisão de código e Contribuidores.
- Avaliação de riscos da compilação: Dependências fixas, Permissões dos Tokens, Empacotamento e Versões Assinadas.

4.5. Open Worldwide Application Security Project (OWASP) CI/CD Top 10 Security Risks

A OWASP elencou os dez maiores riscos associados a esteiras de desenvolvimento (OWASP, 2022), com o intuito de oferecer para a comunidade um guia de como se proteger dos mesmos.

Os riscos apontados são:

- Controle insuficiente dos mecanismos de fluxo;
- Gerenciamento inadequado de identidade e acesso;
- Abuso na cadeia de dependências;
- Envenenamento do pipeline de execução;
- Controle insuficiente do acesso no pipeline;
- Higienização insuficiente das credenciais;
- Configuração insegura do sistema;
- Uso descontrolado de plugins de terceiros;
- Validação inapropriada da integridade dos artefatos;
- Visibilidade e logs insuficientes.

A garantia de segurança da esteira de CI/CD contribui sobremaneira para a segurança da cadeia de suprimento de software envolvida em um determinado projeto, na medida em que mitiga riscos de inserção não autorizada de componentes no processo de compilação.

4.6. Ferramentas de rastreamento de dependências

Por fim, trazemos uma relação de ferramentas que se dedicam ao rastreamento de dependências utilizadas nos diversos projetos de software desenvolvidos por uma determinada organização. Por meio desse tipo de ferramentas é possível a obtenção de um inventário das dependências utilizadas em tais projetos (tanto diretas quanto transitivas), e, assim, facilitar a tarefa de identificação, por exemplo, de projetos afetados por uma determinada dependência sobre a qual tenham sido identificadas vulnerabilidades ou incidentes de segurança. Exemplos desse tipo de ferramentas são os seguintes: Dependency track (OWASP, 2023) e Dependabot (Github, 2023) .

A Tabela 2 consolida as principais características das ferramentas e procedimentos apresentados.

Ferramentas	***Público-alvo***	***Organização Responsável***	***Atestação de nível de segurança***	***Atestação de conformidade com processo***	***Comprovação de integridade***	***Correlações***
in-toto	*Fornecedores e clientes*	*Thie Linux Foundation*	-	*Sim*	-	*TACOS*
Sigstore	*Fornecedores e clientes*	*OpenSSF (Google, Red Hat, Chainguard, GitHub e Universidade de Purdue)*	-	-	*Sim*	*Não há*

Ferramentas	*Público-alvo*	*Organização Responsável*	*Atestação de nível de segurança*	*Atestação de conformidade com processo*	*Comprovação de integridade*	*Correlações*
Compilação determinística	*Fornecedores e clientes*	*projeto Reproducible Builds sob a organização Software Freedom Conservancy*	-	-	*Sim*	
TACOS	*Fornecedores e clientes*	*Tidelift*	*Sim*	-	-	*NIST SSDF, OpenSSF Scorecard, CIS Software Supply Chain Security Guide e SLSA*
OpenSSF Scorecard	*Desenvolvedores, Fornecedores e Clientes*	*OpenSSF (Cisco, Datto, Endor Labs, Google e outros)*	*Sim*	-	-	*TACOS*
OWASP Top 10 CI/CD security risks	*Desenvolvedores*	*OWASP*	-	-	-	
Dependency Track	*Desenvolvedores*	*OWASP*	-	-	-	
Dependabot	*Desenvolvedores*	*GitHub*	-	--	-	
Pyrsia	*Desenvolvedores*	*CD Foundation (Jfrog, Docker, DeplyHub, Huawei, FutureWei, Oracle*	-	-	*Sim*	

Tabela 2 – Ferramentas e procedimentos

5. Comparação das abordagens

Com relação aos *frameworks* relacionados no item 3, observa-se que alguns se classificam efetivamente como *frameworks,* enquanto os demais se classificam como guias. Os que se classificam como *frameworks* (SSDF, S2C2F e OWASP SCVS) possuem escopo mais abrangente, definindo boas práticas de forma mais ampla, voltadas aos seus respectivos públicos-alvo. O SLSA, apesar de se classificar como um guia (conjunto de recomendações), em nosso entendimento se assemelha mais a um *framework,* uma vez que também se propõe a cobrir um escopo mais abrangente, cobrindo três pilares importantes da cadeia de suprimento de software (fontes, dependências e compilação).

Já os que se classificam como guias são mais direcionados a determinados ambientes ou práticas, e possuem caráter mais técnico, provendo recomendações de configurações de ambientes, e assuntos semelhantes.

Faz-se importante notar ainda que apenas o SLSA e o OWASP SCVS são automatizáveis no todo ou em parte, significando que todos os demais *frameworks* ou guias requerem atividades manuais pelos responsáveis por suas implementações em suas respectivas organizações.

Com relação às ferramentas e procedimentos, observa-se que cobrem um espectro de casos de uso distintos.

Enquanto o in-toto tem como objetivo oferecer evidências de que um determinado projeto foi realizado observando as etapas e responsáveis originalmente estabelecidos, soluções como o TACOS e o OpenSSF Scorecard procuram ofecercer uma avaliação da postura de segurança efetiva de um determinado projeto. São soluções que se alinham ao objetivo de estabelecimento de confiança citado no referencial teórico.

Soluções como Sigstore, Compilação determinística e Pyrsia objetivam garantir a integridade do projeto, fornecendo garantias aos seus consumidores de que estão utilizando os projetos originais, sem que tenham sido infectados por código espúrio, e, assim, também se aliam ao objetivo de estabelecimento de confiança.

Por fim, os procedimentos recomendados pelo OWASP Top 10 CI/CD Risks visam ofercer recomendações de configurações seguras para a esteira de integração e entrega contínuas, e produtos como o Dependency Track e o Dependabot visam dotar os desenvolvedores de informações a respeito do uso de componentes de terceiros em seus projetos (dependências), e da disponibilização de atualizações desses componentes, possibilitando que os projetos que as utilizam também sejam adequadamente atualizados. Assim, fortalecendo a segurança dos ambientes de desenvolvimento, e contribuindo para a segurança dos produtos finais, contribuem para os objetivos para ferramentas resilientes e para processos resilientes.

6. Conclusões

Como podemos observar, existe uma grande existe uma diversidade de abordagens para proteção da cadeia de suprimentos de *software*.

Tais abordagens vêm sendo elaboradas e desenvolvidas a partir da experiência adquirida no enfrentamento de casos reais de incidentes que afetaram a cadeia de suprimento de software. Por exemplo, no caso da exploração da empresa Solar Winds, os agentes maliciosos conseguiram comprometer a esteira de desenvolvimento de software da companhia, portanto o código malicioso inserido era versionado como código autêntico e em uma abordagem utilizando blockchain seria considerado como código legítimo. Algumas das soluções aqui apontadas já oferecem proteção contra as vulnerabilidades exploradas nesse ataque.

O presente trabalho traz um detalhamento dessas abordagens e uma comparação entre as mesmas, de forma tal que organizações que estejam em busca de boas práticas e recomendações para o incremento da segurança da cadeia de suprimento de software tenham disponíveis um panorama geral dos *frameworks,* ferramentas e procedimentos aceitos pela comunidade, além de indicações do escopo e finalidade de cada um,

permitindo assim que possam selecionar aqueles que forem mais adequados às suas necessidades.

Como trabalhos futuros podemos elencar uma análise mais aprimorada dos custos de implementação das práticas e controles propostos pelas abordagens analisadas. Por meio dessa análise de custos, um gestor de segurança da informação poderia realizar uma análise de custo-benefício da implementação dos controles e dessa forma desenhar um melhor *roadmap* de adoção priorizando controles mais eficientes.

Outro ponto importante que pode ser abordado é a promoção da transparência entre fornecedor e consumidor. É interessante que seja estimulado um comportamento de cooperação entre as organizações de forma a agilizar a resposta aos incidentes cibernéticos. Dessa forma, próximos estudos podem elaborar quais são os mecanismos para que essa colaboração se fortaleça e propicie a proteção da cadeia de suprimentos de *software*.

Referências

Boyens, J., Smith, A., Bartol, N., Winkler, K., Holbrook, A., & Fallon, M. (2022). *NIST - Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*. https://doi.org/10.6028/NIST.SP.800-161r1

Center for Internet Security (CIS). (2022). *Software Supply Chain Security Guide*. https://www.cisecurity.org/terms-of-use-for-non-member-cis-products/

Cloud Native Computing Foundation. (2021). *Software Supply Chain Best Practices*. https://project.linuxfoundation.org/hubfs/CNCF_SSCP_v1.pdf

Enduring Security Framework. (2022a). *Securing the Software Supply Chain - Recommended Practices Guide for Customers*. https://media.defense.gov/2022/Nov/17/2003116445/-1/-1/0/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_CUSTOMER.PDF

Enduring Security Framework. (2022b). *Securing the Software Supply Chain - Recommended Practices Guide for Developers*. https://media.defense.gov/2022/Sep/01/2003068942/-1/-1/0/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_DEVELOPERS.PDF

Enduring Security Framework. (2022c). *Securing the Software Supply Chain - Recommended Practices Guide for Suppliers*. https://media.defense.gov/2022/Oct/31/2003105368/-1/-1/0/SECURING_THE_SOFTWARE_SUPPLY_CHAIN_SUPPLIERS.PDF

Github. (2023). *Dependabot · Automated dependency updates built into GitHub*. https://github.com/dependabot

Ladisa, P., Plate, H., Martinez, M., & Barais, O. (2022). *Taxonomy of Attacks on Open-Source Software Supply Chains*. https://sap.github.io/risk-explorer-for-software-supply-chains/

Ladisa, P., Ponta, S. E., Sabetta, A., Martinez, M., & Barais, O. (2023). *Journey to the Center of Software Supply Chain Attacks*. https://doi.org/10.1109/XXX.0000.0000000

Lella, I., Theocharidou, M., Tsekmezoglou, E., Malatras, A., Garcia, S., & Valeros, V. (2021). *ENISA THREAT LANDSCAPE FOR SUPPLY CHAIN ATTACKS*. https://doi.org/10.2824/168593

Melara, M. S., & Bowman, M. (2022). *What is Software Supply Chain Security?* https://arxiv.org/pdf/2209.04006.pdf

Ohm, M., Plate, H., Sykosch, A., & Meier, M. (2020). *Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks*. https://arxiv.org/pdf/2005.09535.pdf

Open Source Security Foundation (OpenSSF). (2023). *SLSA • Supply-chain Levels for Software Artifacts*. SLSA.dev. https://slsa.dev/

OpenSSF. (2023). *OpenSSF Scorecard*. https://securityscorecards.dev/

OWASP. (2022). *OWASP Top 10 CI/CD Security Risks*. https://github.com/OWASP/www-project-top-10-ci-cd-security-risks/blob/main/index.md

OWASP. (2023). *Dependency-Track | Software Bill of Materials (SBOM) Analysis*. https://dependencytrack.org/

Peisert, S., Schneier, B., Okhravi, H., Massacci, F., Benzel, T., Landwehr, C., Mannan, M., Mirkovic, J., Prakash, A., & Michael, J. B. (2021). Perspectives on the SolarWinds Incident. Em *IEEE Security and Privacy* (Vol. 19, Número 2, p. 7–13). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/MSEC.2021.3051235

Pipikaite, A., Bueermann, G., Joshi, A., Jurgens, J., Bissell, K., Aguirre, C., Browder, T., & Pruitt, J. (2022). *Global Cybersecurity Outlook 2022*. https://www3.weforum.org/docs/WEF_Global_Cybersecurity_Outlook_2022.pdf

SAP SE. (2023). *Risk Explorer for Software Supply Chains*. https://sap.github.io/risk-explorer-for-software-supply-chains/#/attacktree

Software Freedom Conservancy. (2023). *Reproducible Builds*. https://reproducible-builds.org/

Souppaya, M., Scarfone, K., & Dodson, D. (2022). *Secure Software Development Framework (SSDF) version 1.1*. https://doi.org/10.6028/NIST.SP.800-218

Testifysec. (2023). *Witness - pluggable framework for software supply chain risk management*. https://github.com/testifysec/witness

The Linux Foundation. (2023a). *in-toto | A framework to secure the integrity of software supply chains*. https://in-toto.io/

The Linux Foundation. (2023b). *Sigstore*. https://www.sigstore.dev/

Tidelift. (2023). *TACOS Framework*. https://github.com/tacosframework

Wetter, J., & Ringland, N. (2021). *Understanding the Impact of Apache Log4j Vulnerability*. Google Online Security Blog. https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html