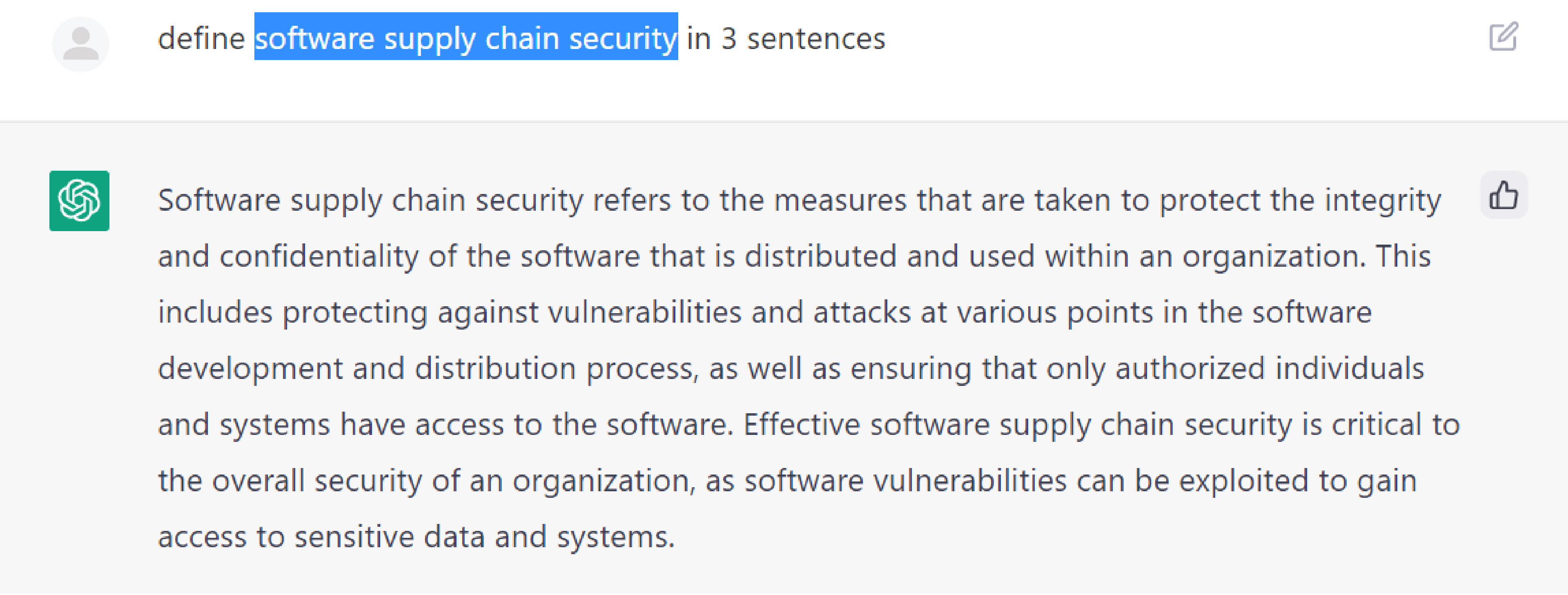
Grant Thornton

Software Supply Chain Risks

Trends and Insights

January 2023

Software supply chain security defined



define software supply chain security in 3 sentences

Software supply chain security refers to the measures that are taken to protect the integrity and confidentiality of the software that is distributed and used within an organization. This includes protecting against vulnerabilities and attacks at various points in the software development and distribution process, as well as ensuring that only authorized individuals and systems have access to the software. Effective software supply chain security is critical to the overall security of an organization, as software vulnerabilities can be exploited to gain access to sensitive data and systems.

In the news…

“Despite being discontinued in 2005, the Boa web server continues to be implemented by different vendors across a variety of IoT devices and popular software development kits (SDKs).”

“Thousands of smartphone applications in Apple and Google's online stores contain computer code developed by Pushwoosh, which presents itself as based in the United States, but is actually Russian”

“The official software repository for the Python language, Python Package Index (PyPI), has been targeted in a complex supply chain attack that appears to have successfully poisoned at least two legitimate projects”

“An unknown threat actor has created a malicious Python package that appears to be a software development kit (SDK) for a well-known security client from SentinelOne”

Select Statistics

Free and Open Source Software (FOSS) constitutes ***70-90%*** *of any given piece of modern software solutions.*

--Linux Foundation, 2022

92% *of Open-Source Software (OSS) contain outdated or vulnerable code. 650% YoY increase in OSS attacks.*

-Tech.co, 2022

1,300 *malicious packages found in popular Node Package Manager (npm) JavaScript package manager*

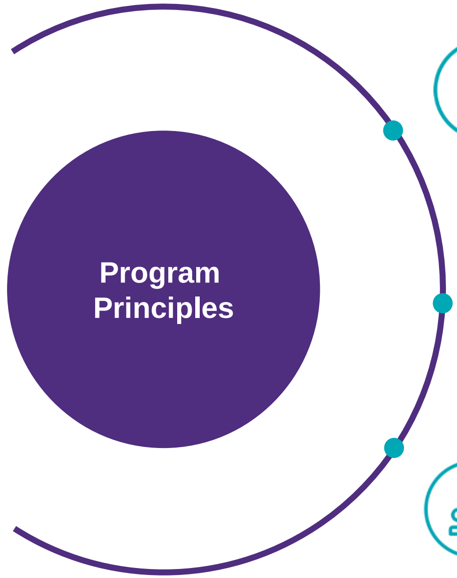
-Securityweek.com, 2022

Vulnerabilities in third party products or services that result in a security breach cost an average of ***$4.55M****.*

--IBM, 2022

© 2022 Grant Thornton LLP | All rights reserved | U.S. member firm of Grant Thornton International Ltd

Designing a Successful Software Supply Chain Security Program



Developer-centric

- Developers are empowered to make remediation & prioritization decisions
- Rather than gating production, security is socialized into the DevOps process and culture
- Engineers are enabled with tooling that that fits with their existing workflows

Everything as Code

- Where possible, repeatable steps and decisions are automated to keep pace with production and assure a “secure enough” product
- Security testing is automated as part of the CI/CD pipeline
- Security and risk policies are addressed programmatically through code

Dependency-aware

- Dependency checks are performed during all stages of the pipeline
- Visibility into which dependencies are actually used in the product
- Admission controls are established to ensure insecure dependencies are timely identified, prioritized, and remediated

Grant Thornton

Software Supply Chain Security Framework

Governance & Risk Management

- Roles & Responsibilities
- Decision Trees
- Policies & Security Requirements
- Threat Modeling
- Risk Assessment & Treatment
- Reporting

Pillars

Source Code Management

- Source code inventory
- Version control

Deployment Controls

- Policy automation
- Security testing
- Binary authorization

Pedigree & Provenance

- Origin traceability
- Verification
- Tamper Resistance

Dependency Management

- Component analysis
- Vulnerability management

Incident Management

- Triage, response, recovery
- Reporting & disclosure

Foundation

Standards & Frameworks

- NIST's Software Secure Development Framework (SP 800-218)
- Supply Chain Levels for Software Artifacts (SLSA) Framework
- DevSecOps Maturity Model (DSOMM)
- OWASP Top 10 CI/CD Risks

Grant Thornton

© 2022 Grant Thornton LLP | All rights reserved | U.S. member firm of Grant Thornton International Ltd

Journey Toward Secure Software Supply Chain

To establish a software supply chain security program, organizations should assess existing risks, and extend security controls at different stages of the software development lifecycle.

0 Assess risk posture of your dependencies and engineering environment

1 Extend AppSec/DevSecOps internal controls to cover CI/CD security and 3rd party dependencies

2 Improve quality of OSS dependencies to prevent dependency chain abuse

3 Maintain continuous inventory of systems. Harden and check for known vulnerabilities in these components.

4 Establish pipeline flow control mechanisms to ensure code and artifacts cannot be shipped without verification

5 Implement processes and technologies to validate the integrity of resources (e.g. code signing)

6 Continuously map all collaborators and ensure their identities are aligned with the principle of least privilege

7 Implement governance controls around 3rd party services with permissions to the engineering environment

8 Implement mitigations against Poisoned Pipeline Execution attack vector across SCM and CI systems

9 Maintain continuous visibility into the use of dependencies and security state of the engineering ecosystem

Learning → **Emerging** → **Functional**

Grant Thornton

Software Supply Chain Security Framework

Governance & Risk Management

- Roles & Responsibilities
- Decision Trees
- Policies & Security Requirements
- Threat Modeling
- Risk Assessment & Treatment
- Reporting

Pillars

Source Code Management

- Source code inventory
- Version control

Deployment Controls

- Policy automation
- Security testing
- Binary authorization

Pedigree & Provenance

- Origin traceability
- Verification
- Tamper Resistance

Dependency Management

- Component analysis
- Vulnerability management

Incident Management

- Triage, response, recovery
- Reporting & disclosure

Foundation

Standards & Frameworks

- NIST's Software Secure Development Framework (SP 800-218)
- Supply Chain Levels for Software Artifacts (SLSA) Framework
- DevSecOps Maturity Model (DSOMM)
- OWASP Top 10 CI/CD Risks

Grant Thornton

© 2022 Grant Thornton LLP | All rights reserved | U.S. member firm of Grant Thornton International Ltd

“Shift Left” Reimagined

Relationship between security and engineering teams is not always without friction: developers are spending increasing amounts of their time dealing with the proliferation of security tools, triaging vulnerability reports, and working through long backlogs.

Controlling Outputs
How AppSec is done today

- Security teams want code tested earlier in the process, but those tests are not easily actionable which slows down production
- As the result, security is frequently not fully assessed until the product is already built

Improving Inputs
Risk management for today’s “assembled” software

- With over 80% of software consisting of OSS, orgs should prioritize selecting less risky OSS assets
- Reusable code vetted / tested prior to use results in shorter triage-fix cycles, with some testing gates even eliminated

Grant Thornton

Dependency Management

Software dependency risk management is the process of identifying, assessing, and mitigating the risks associated with the use of third-party software libraries or other dependencies in an organization's software development process.

Visibility

Transient Dependencies

Understand which indirect, or transient, dependencies are included in your software builds

Dependency Quality

Quantify the quality of dependencies in your software and associated risks. Identify unused, unmaintained, and outdated dependencies

Reachability

Use call graphs to understand which functions called by your code contain vulnerabilities

Control

Select Better Dependencies

Oversight through governance policies on OSS selection. Quantify risk based on leading risk indicators, not just known vulnerabilities.

Prioritize Vulnerabilities

Reduced number of highs and criticals. Revised vulnerability triage, analysis, and remediation workflows.

Prepare for the Inevitable

Incorporate continuously updated visibility into dependencies into incident response workflows to prevent another log4j fumble.

GrantThornton

© 2022 Grant Thornton LLP | All rights reserved | U.S. member firm of Grant Thornton International Ltd

ENDOR LABS

Open Source Supply Chain Risks & Safeguards

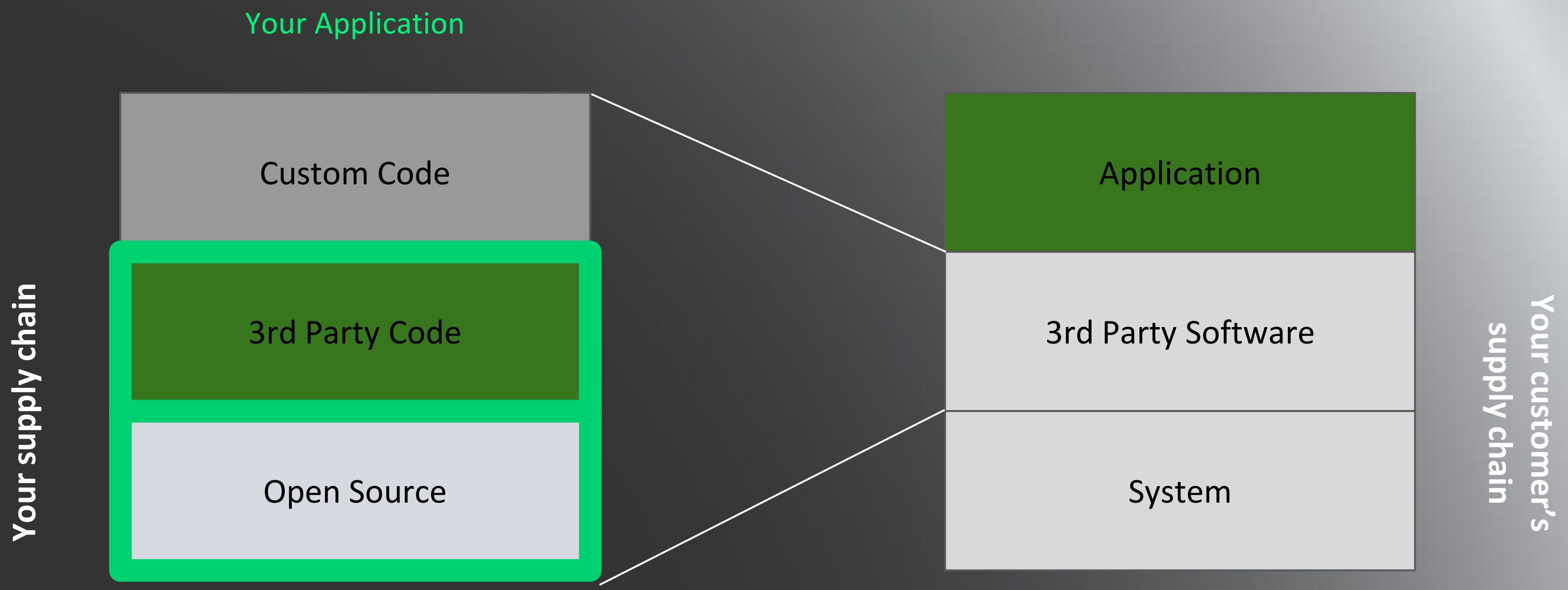
What constitutes a software supply chain attacK?

01

According to ChatGPT…

An open source software supply chain attack is a type of cyber attack in which an attacker infiltrates the supply chain of an open source software project and injects malicious code into the project's codebase. This can occur at any point in the development or distribution process. When users download and install the compromised version of the software, the malicious code is executed on their systems, potentially giving the attacker unauthorized access to the user's device or network.

Open source software is widely used because it is typically free and the source code is openly available for anyone to review, modify, and distribute. This can make it an attractive target for attackers, as they can potentially gain access to a large number of users by compromising a popular open source project. To protect against supply chain attacks, it is important for open source software developers and users to regularly update their software and be vigilant about checking for and installing security patches.



Pramod Gosavi: Do you need "Supply Chain Security" or SBOM?

Software Supply Chain Vulnerabilities vs. Software Supply Chain Attacks

SoK: Taxonomy of Attacks on Open-Source Software Supply Chains

P. Ladisa, H. Plate, M. Martinez and O. Barais, "SoK: Taxonomy of Attacks on Open-Source Software Supply Chains," in 2023 2023 IEEE Symposium on Security and Privacy

https://riskexplorer.endorlabs.com

ENDOR LABS

Open Source Supply Chain Attack Tree

The attack tree focuses on open-source based software development practices, which involve the consumption of numerous open-source components throughout the entire development lifecycle. In this context, the attacker's top-level goal is to place malicious code in open-source artifacts such that it is executed in the context of downstream projects, e.g., during its development or runtime.

- Develop distinct malicious package from scratch
- Create name confusion with legitimate package
- Subvert legitimate package

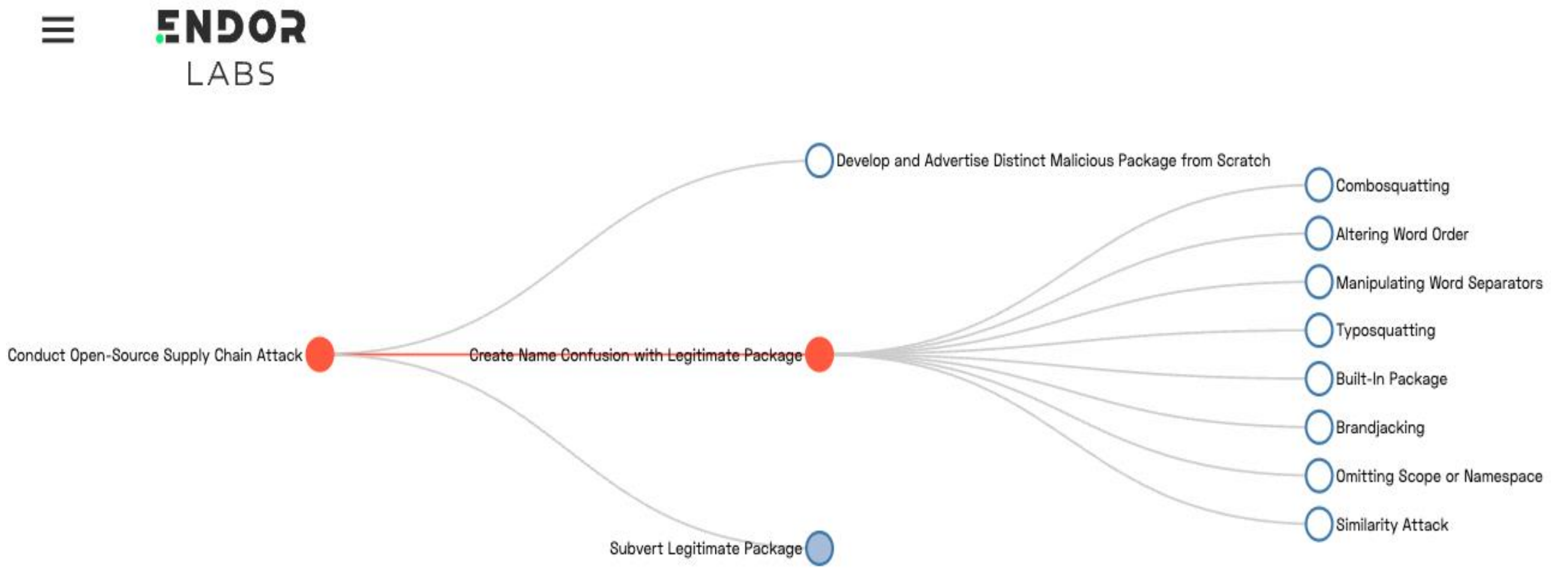
DEVELOP AND ADVERTISE DISTINCT MALICIOUS PACKAGE FROM SCRATCH

This attack vector covers the creation of a new, seemingly legitimate and useful open-source project with the intention to use it for spreading malicious code, either from the beginning or at a later point in time. Besides creating the project and developing useful functionality, the attacker is required to advertise the project in order to attract victims.

CREATE NAME CONFUSION WITH LEGITIMATE PACKAGE

The general idea behind name confusion is that attackers craft new component names that resemble names of legitimate open-source or system components, suggest trustworthy authors or play with common naming patterns in different languages or ecosystems.

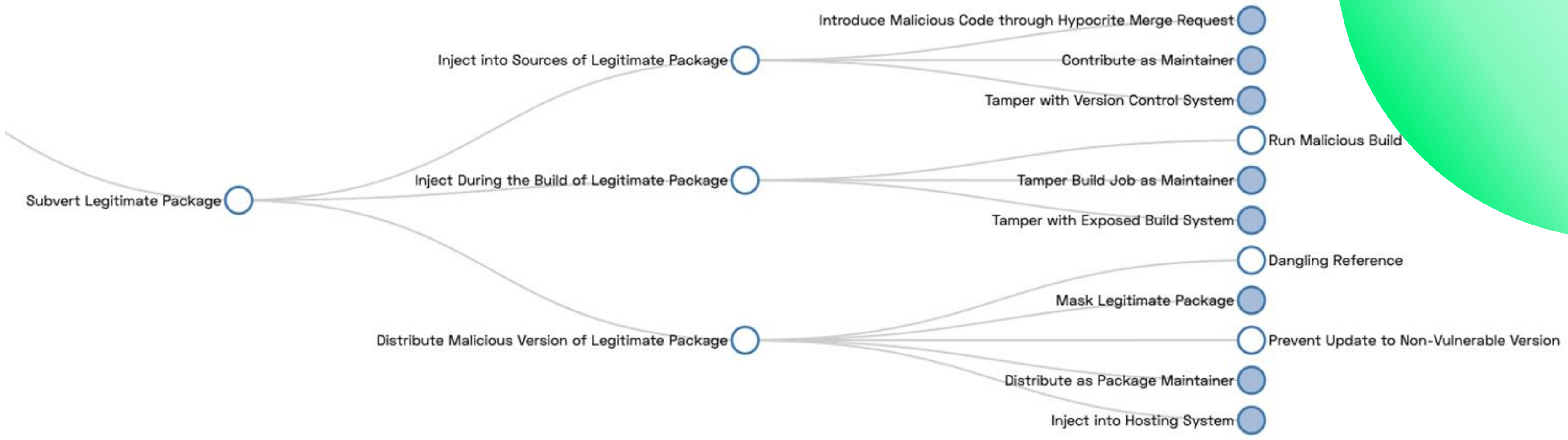
CREATE NAME CONFUSION WITH LEGITIMATE PACKAGE


ENDOR
LABS
Conduct Open-Source Supply Chain Attack
Develop and Advertise Distinct Malicious Package from Scratch
Create Name Confusion with Legitimate Package
Subvert Legitimate Package
Combosquatting
Altering Word Order
Manipulating Word Separators
Typosquatting
Built-In Package
Brandjacking
Omitting Scope or Namespace
Similarity Attack

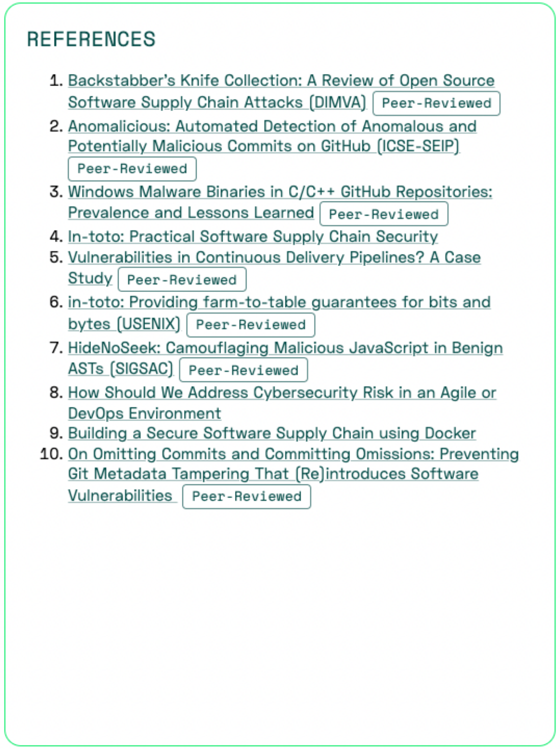
SUBVERT LEGITIMATE PACKAGE

This attack vector covers all attacks aiming to corrupt an existing, legitimate project, which requires compromising one or more of its numerous stakeholders or resources, e.g. its source code repository, build system or distribution infrastructure.

SUBVERT LEGITIMATE PACKAGE

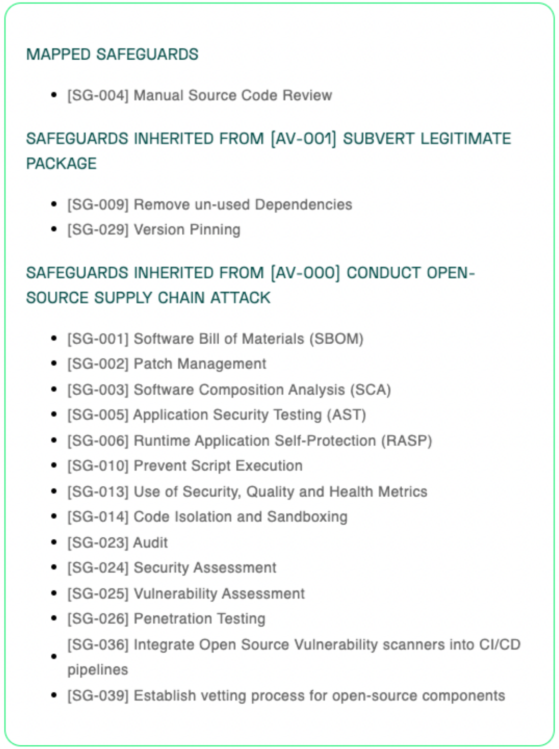

Subvert Legitimate Package
Inject into Sources of Legitimate Package
Introduce Malicious Code through Hypocrite Merge Request
Contribute as Maintainer
Tamper with Version Control System
Inject During the Build of Legitimate Package
Run Malicious Build
Tamper Build Job as Maintainer
Tamper with Exposed Build System
Distribute Malicious Version of Legitimate Package
Dangling Reference
Mask Legitimate Package
Prevent Update to Non-Vulnerable Version
Distribute as Package Maintainer
Inject into Hosting System

SUBVERT LEGITIMATE PACKAGE



REFERENCES

1. Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks (DIMVA) `Peer-Reviewed`
2. Anomalicious: Automated Detection of Anomalous and Potentially Malicious Commits on GitHub (ICSE-SEIP) `Peer-Reviewed`
3. Windows Malware Binaries in C/C++ GitHub Repositories: Prevalence and Lessons Learned `Peer-Reviewed`
4. In-toto: Practical Software Supply Chain Security
5. Vulnerabilities in Continuous Delivery Pipelines? A Case Study `Peer-Reviewed`
6. in-toto: Providing farm-to-table guarantees for bits and bytes (USENIX) `Peer-Reviewed`
7. HideNoSeek: Camouflaging Malicious JavaScript in Benign ASTs (SIGSAC) `Peer-Reviewed`
8. How Should We Address Cybersecurity Risk in an Agile or DevOps Environment
9. Building a Secure Software Supply Chain using Docker
10. On Omitting Commits and Committing Omissions: Preventing Git Metadata Tampering That (Re)introduces Software Vulnerabilities `Peer-Reviewed`



MAPPED SAFEGUARDS

- [SG-004] Manual Source Code Review

SAFEGUARDS INHERITED FROM [AV-001] SUBVERT LEGITIMATE PACKAGE

- [SG-009] Remove un-used Dependencies
- [SG-029] Version Pinning

SAFEGUARDS INHERITED FROM [AV-000] CONDUCT OPEN-SOURCE SUPPLY CHAIN ATTACK

- [SG-001] Software Bill of Materials (SBOM)
- [SG-002] Patch Management
- [SG-003] Software Composition Analysis (SCA)
- [SG-005] Application Security Testing (AST)
- [SG-006] Runtime Application Self-Protection (RASP)
- [SG-010] Prevent Script Execution
- [SG-013] Use of Security, Quality and Health Metrics
- [SG-014] Code Isolation and Sandboxing
- [SG-023] Audit
- [SG-024] Security Assessment
- [SG-025] Vulnerability Assessment
- [SG-026] Penetration Testing
- [SG-036] Integrate Open Source Vulnerability scanners into CI/CD pipelines
- [SG-039] Establish vetting process for open-source components

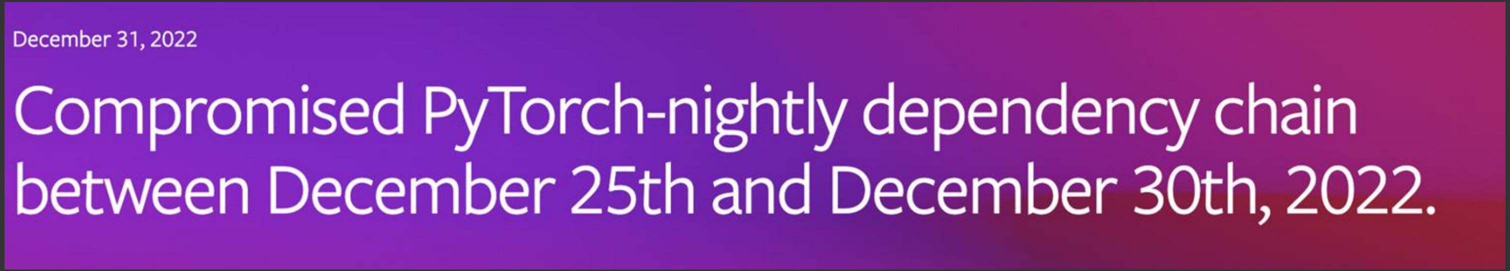
Recent Incident: SentinelSneak

Malicious PyPI module poses as security software development kit

A malicious Python file found on the PyPI repository adds backdoor and data exfiltration features to what appears to be a legitimate SDK client from SentinelOne.

Recent Incident: PyTorch

Next-gen supply chain attack in an ML package



December 31, 2022

Compromised PyTorch-nightly dependency chain between December 25th and December 30th, 2022.

17k forks

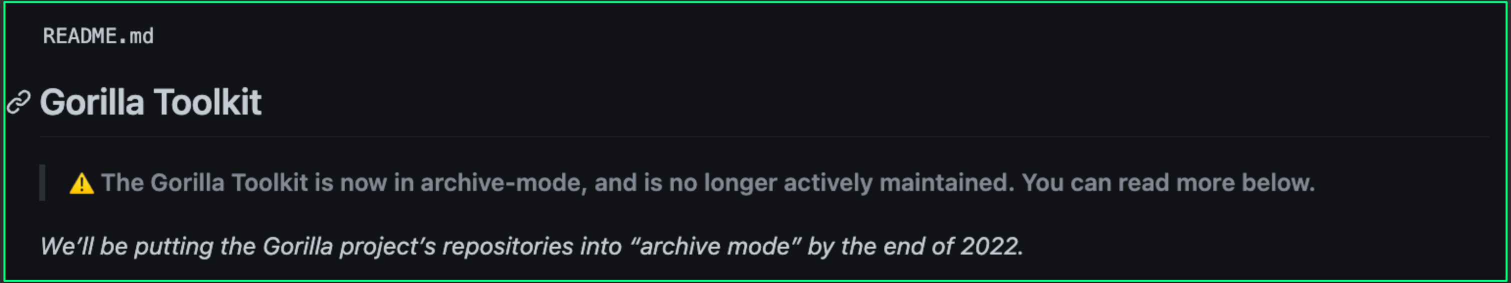
Used in over 187K repositories

61k GitHub stars

Popular ML framework by Meta

Recent Incident: Gorilla

Risk is not always captured as a CVE



README.md

Gorilla Toolkit

⚠️ The Gorilla Toolkit is now in archive-mode, and is no longer actively maintained. You can read more below.

We'll be putting the Gorilla project's repositories into "archive mode" by the end of 2022.

+10k weekly clones on each package

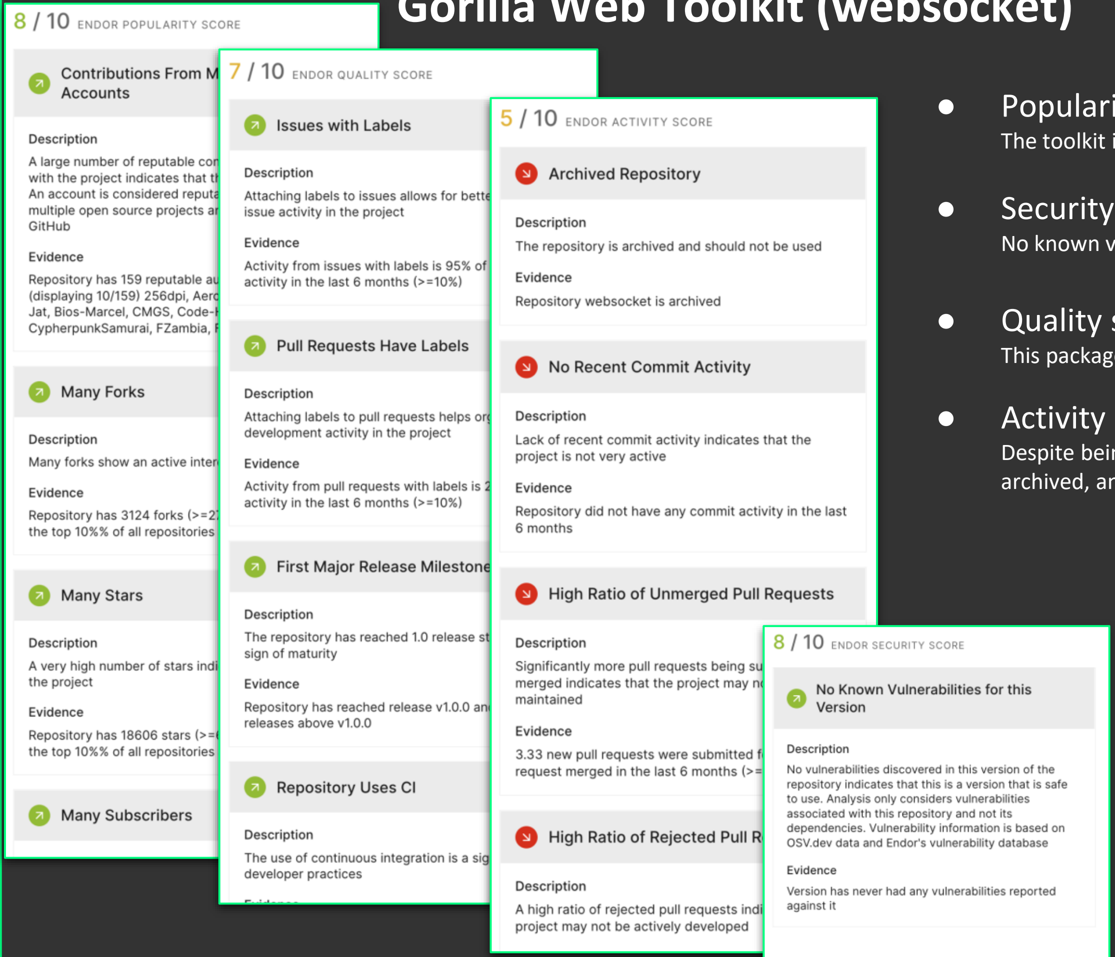
Used in over 90K repositories

18k GitHub stars

Most popular HTTP service for Go

Governance starts with selection

Gorilla Web Toolkit (websocket)



8 / 10 ENDOR POPULARITY SCORE

Contributions From M
Accounts

Description
A large number of reputable con
with the project indicates that th
An account is considered reputa
multiple open source projects ar
GitHub

Evidence
Repository has 159 reputable au
(displaying 10/159) 256dpi, Aero
Jat, Bios-Marcel, CMGS, Code-H
CypherpunkSamurai, FZambia, F

Many Forks

Description
Many forks show an active inter

Evidence
Repository has 3124 forks (>=2
the top 10%% of all repositories

Many Stars

Description
A very high number of stars indi
the project

Evidence
Repository has 18606 stars (>=
the top 10%% of all repositories

Many Subscribers

7 / 10 ENDOR QUALITY SCORE

Issues with Labels

Description
Attaching labels to issues allows for bette
issue activity in the project

Evidence
Activity from issues with labels is 95% of
activity in the last 6 months (>=10%)

Pull Requests Have Labels

Description
Attaching labels to pull requests helps org
development activity in the project

Evidence
Activity from pull requests with labels is 2
activity in the last 6 months (>=10%)

First Major Release Milestone

Description
The repository has reached 1.0 release st
sign of maturity

Evidence
Repository has reached release v1.0.0 an
releases above v1.0.0

Repository Uses CI

Description
The use of continuous integration is a sig
developer practices

5 / 10 ENDOR ACTIVITY SCORE

Archived Repository

Description
The repository is archived and should not be used

Evidence
Repository websocket is archived

No Recent Commit Activity

Description
Lack of recent commit activity indicates that the
project is not very active

Evidence
Repository did not have any commit activity in the last
6 months

High Ratio of Unmerged Pull Requests

Description
Significantly more pull requests being su
merged indicates that the project may no
maintained

Evidence
3.33 new pull requests were submitted f
request merged in the last 6 months (>=

High Ratio of Rejected Pull R

Description
A high ratio of rejected pull requests indi
project may not be actively developed

8 / 10 ENDOR SECURITY SCORE

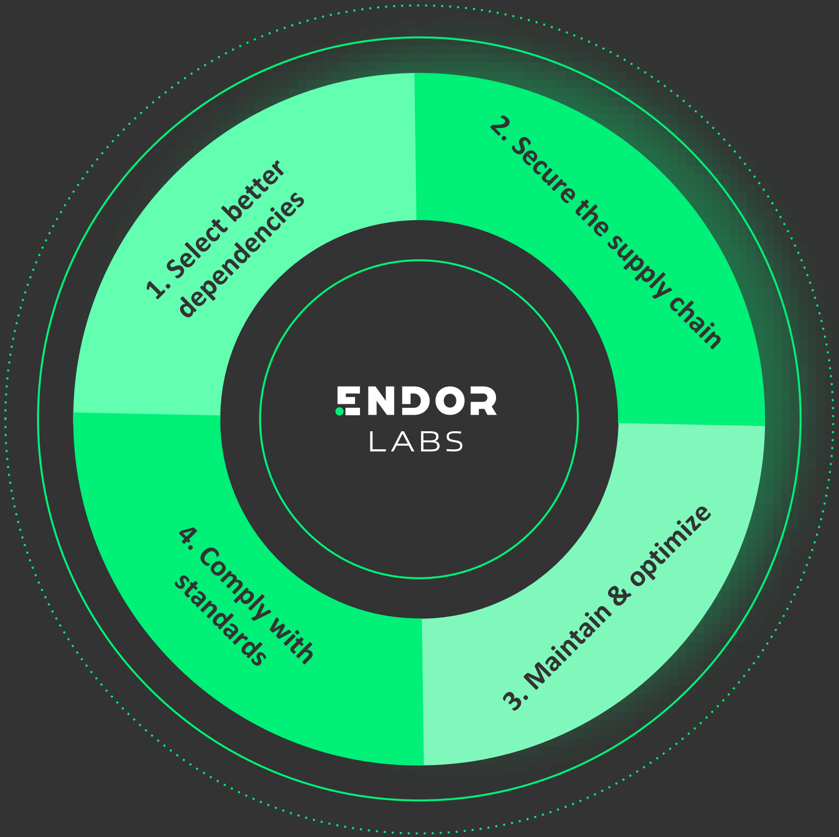
No Known Vulnerabilities for this
Version

Description
No vulnerabilities discovered in this version of the
repository indicates that this is a version that is safe
to use. Analysis only considers vulnerabilities
associated with this repository and not its
dependencies. Vulnerability information is based on
OSV.dev data and Endor's vulnerability database

Evidence
Version has never had any vulnerabilities reported
against it

- Popularity score - High
  The toolkit includes 9 packages, each with over 10K unique weekly clones
- Security score - High
  No known vulnerabilities in the latest release
- Quality score - Medium
  This package uses best practices and is well maintained
- Activity score - Low
  Despite being one of the post popular Golang projects, the toolkit has been archived, and now poses an operational and security risk

Dependency Lifecycle Management builds robust software, secures supply chains, and meets emerging compliance needs



Dependency Diagnostic

ENDOR LABS

Grant Thornton's Dependency Diagnostic, powered by Endor Labs, enables our clients understand their use of dependencies in software, quantify uncovered risks, and qualify those risks for product teams and executive leadership.

Diagnostic Outcomes

1 **Mitigate Supply Chain Threats**
Map your dependencies; identify where vulnerable packages are used, and what applications depend on them.

2 **Assess Dependency Quality**
Evaluate dependencies based on security, quality, popularity, and maintainer activity

3 **Prioritize Vulnerabilities**
Eliminate false positive vulnerabilities for unreachable & test dependencies.

GrantThornton

© 2022 Grant Thornton LLP | All rights reserved | U.S. member firm of Grant Thornton International Ltd

Dependency Diagnostic

Grant Thornton's Dependency Diagnostic, powered by Endor Labs, enables our clients understand their use of dependencies in software.

1 **Mapping**

Understand the dependency graph for new and existing dependencies

2 **Analysis**

Answer questions about where, how, and by whom dependencies are being used

3 **Prioritization**

Of the 1,000s vulnerabilities in your code, understand which are reachable and exploitable

4 **Reporting**

Qualify risks and required changes to product teams and executive leadership

5 **Planning**

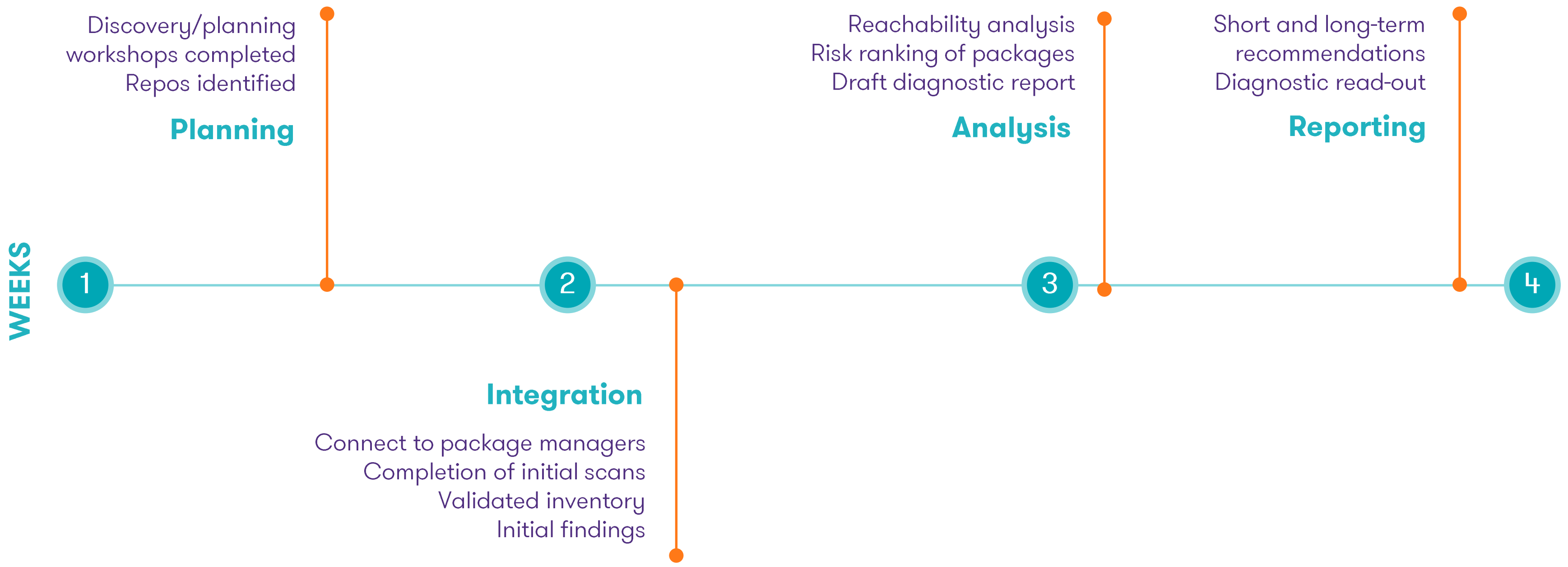
Describe activities required to address discovered risks across governance, people, process, and tech

Grant Thornton

© 2022 Grant Thornton LLP | All rights reserved | U.S. member firm of Grant Thornton International Ltd

Dependency Diagnostic

Duration of the diagnostic is four (4) weeks. Activities and key milestones are shown below.



© 2022 Grant Thornton LLP | All rights reserved | U.S. member firm of Grant Thornton International Ltd

Grant Thornton

Max Kovalsky
max.Kovalsky@us.gt.com
646-354-0463

Nic LaBuz
nic@endor.ai
216-496-0246

Thank you!