

Malware Discovered in Popular NPM: Anatomy of Next-Gen Supply Chain Attacks

“Least privilege: Every program and every user of the system should operate using the least set of privileges necessary to complete the job.”

- Saltzer and Schroeder in "Basic Principles of Information Protection," page 9, 1975

Nathan Aw

<https://www.linkedin.com/in/awnathan>

nathan.mk.aw@gmail.com

<https://nathanawmk.github.io/>

16 November 2021

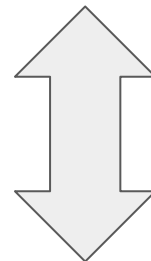
(Tue)

Opinions/views expressed in the talk are solely my own and do not express the views or opinions of my employer.

Agenda

1. **Malware Discovered in Popular NPM: Anatomy of Next-Gen Supply Chain Attacks**
2. **Secure Software Packages, Dependencies to Defend against Cyber Supply Chain Attacks for NPM, PyPI, Maven, NuGet, Crates and RubyGems**
3. **Build Secure Guardrails, not Road Blocks or Gates: Shift Left with Gitops and integrate Fuzzing into DevSecOps**
4. **Importance of Cloud Infrastructure Entitlements Management (CIEM)**
5. **Q&A**

**Secure
Guardrails?**



**Or Broken
Guardrails?**

Who I am

With over 9 years of experience as a software developer and application (security) architect, Nathan Aw is a firm believer-practitioner of zero trust and advocate of secure coding practices. His passion is in designing, building and rolling out asynchronous, polyglot-based microservices that are both zero-trust, performant which can securely run anywhere (multi-cloud and/or on-premise) that scale without limits.

Through hands-on setup of a Secure Software Factory (SSF), he understands the importance of setting up a first-class secure software factory that is able to industrialise “**shift left**” practises that translates to quicker delivery of trusted and secure digital services to its customers.

Other Nathan's interests include emerging technology frameworks such as WebAssembly and frontier technologies such as quantum computing and its impact on cybersecurity, security in and for metaverse and securing 5G Cloud Infrastructures.

More on Nathan Aw can be found at <https://nathanawmk.github.io/> / <https://sg.linkedin.com/in/awnathan>

Previous OWASP Presentations:

- https://owasp.org/www-chapter-singapore/assets/presos/Securing_your_APIs_-_OWASP_API_Top_10_2019_Real-life_Case.pdf
- https://owasp.org/www-chapter-singapore/assets/presos/Deconstructing_the_Solarwinds_Supply_Chain_Attack_and_Detering_it_Honin_g_in_on_the_Golden_SAML_Attack_Technique.pdf
- [https://owasp.org/www-chapter-singapore/assets/presos/Microservices%20Security%2C%20Container%20Runtime%20Security%2C%20MITRE%20ATT%26CK%2%AE%20%20for%20Kubernetes%20\(K8S\)%20and%20Service%20Mesh%20for%20Security.pdf](https://owasp.org/www-chapter-singapore/assets/presos/Microservices%20Security%2C%20Container%20Runtime%20Security%2C%20MITRE%20ATT%26CK%2%AE%20%20for%20Kubernetes%20(K8S)%20and%20Service%20Mesh%20for%20Security.pdf)
- https://github.com/OWASP/www-chapter-singapore/raw/master/assets/presos/Securing_Multi_cloud_Portable_Tier_Microservices_Applications_A_live_demo_on_cloud_native_application_security_platforms.pdf;
- [https://owasp.org/www-chapter-singapore/assets/presos/Securing_Production_Identity_Framework_for_Everyone_\(SPIFFE\)._Building_End_to_End_Secure_Software_Factory_and_Protecting_Cloud-Native_Supply_Chain_Helpful_Cloud-Native_Security_Checklists_and_Demo_on_SPIFFE_and_Not.pdf](https://owasp.org/www-chapter-singapore/assets/presos/Securing_Production_Identity_Framework_for_Everyone_(SPIFFE)._Building_End_to_End_Secure_Software_Factory_and_Protecting_Cloud-Native_Supply_Chain_Helpful_Cloud-Native_Security_Checklists_and_Demo_on_SPIFFE_and_Not.pdf)

Opinions/views expressed in the talk are solely my own and do not express the views or opinions of my employer.

Recap - Last Meetup (19th Oct)

- Secure Production Identity Framework for Everyone (SPIFFE)
- Secure End to End Software Factory
- Helpful Cloud-Native Supply Chain Security Checklist
- The Notary Project Based on The Update Framework - a secure general design for the problem of software distribution and updates

SOURCE:

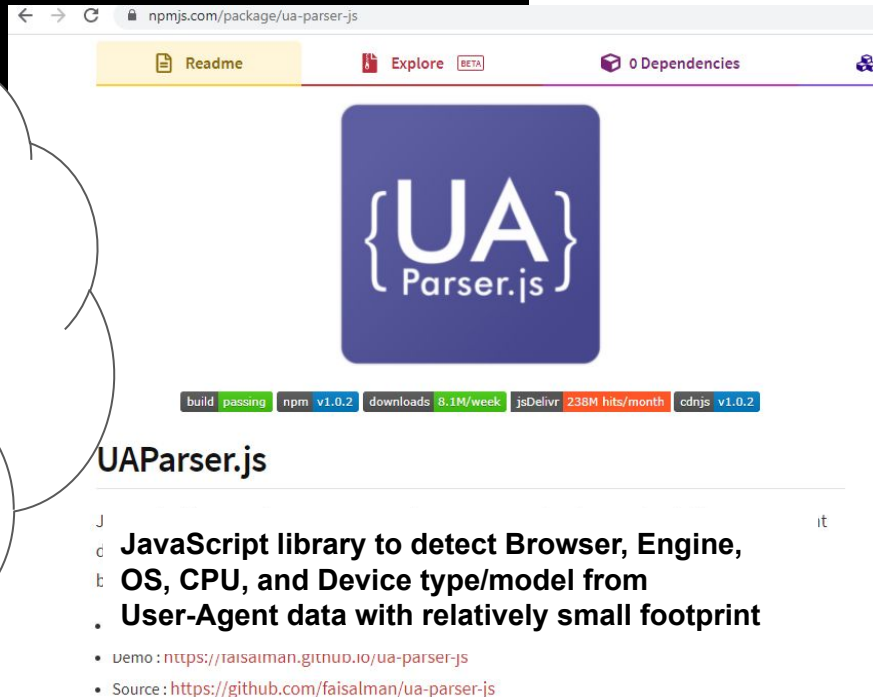
[https://owasp.org/www-chapter-singapore/assets/presos/Securing_Production_Identity_Framework_for_Everyone \(SPIFFE\), Building_End_to_End_Secure_Software_Factory_and_Protecting_Cloud-Native_Supply_Chain_Helpful_Cloud-Native_Security_Checklists_and_Demo_on_SPIFFE_and_Not.pdf](https://owasp.org/www-chapter-singapore/assets/presos/Securing_Production_Identity_Framework_for_Everyone_(SPIFFE),_Building_End_to_End_Secure_Software_Factory_and_Protecting_Cloud-Native_Supply_Chain_Helpful_Cloud-Native_Security_Checklists_and_Demo_on_SPIFFE_and_Not.pdf)

Context: Just another day for Developers (like myself!) - Installing npm packages

```
C:\Users\USER>npm i ua-parser-js
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\USER\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\USER\package.json'
npm WARN USER No description
npm WARN USER No repository field.
npm WARN USER No README data
npm WARN USER No license field.

+ ua-parser-js@1.0.2
added 1 package from 123
1 package is looking for
run `npm fund` for
found 0 vulnerabilit
```

"I need to deliver my project under intense pressure, I need "ABC" features and I just saw this library which is exactly what I need. Let's install this in dev and deploy in prod."



npmjs.com/package/ua-parser-js

Readme Explore BETA 0 Dependencies

UA
Parser.js

build passing npm v1.0.2 downloads 8.1M/week jsDelivr 238M hits/month cdnjs v1.0.2

UAParser.js

JavaScript library to detect Browser, Engine, OS, CPU, and Device type/model from User-Agent data with relatively small footprint

- Demo: <https://faisalman.github.io/ua-parser-js>
- Source: <https://github.com/faisalman/ua-parser-js>

SOURCE:

<https://www.npmjs.com/package/ua-parser-js>

Next Moment..: You found out that “ua-parser-js” is hijacked by malware (1/2)

SECURITY-ISSUE: node_module dependency "ua-parser-js" is hijacked by malware #5769



Closed



5 tasks done

alex-drocks opened this issue 11 days ago · 6 comments



aimozg commented 11 days ago



This thing tries to steal saved passwords, cookies, and who knows what else. The sooner you can pull the plug the better, it doesn't matter if version numbers suffer a little.



62

SOURCE:

<https://github.com/facebook/docusaurus/issues/5769>

<https://github.com/faisalman/ua-parser-js/issues/536>

Next Moment..: “ua-parser-js” is hijacked by malware (2/2)

53 / 66

53 security vendors and 2 sandboxes flagged this file as malicious

2a3acdcd76575762b18c18c644a745125f55ce121f742d2aad962521bc7f25fd

Dog.dll

2.47 MB Size

2021-10-31 14:33:31 UTC

1 day ago

invalid-rich-pe-duplicated-entries invalid-rich-pe-linker-version pe.dll

Community Score

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY 13
Ad-Aware	Trojan.GenericKD.47234476	AhnLab-V3	Trojan.Win.Tnega.C.4723159	
Alibaba	TrojanBanker:Win32/Danabot.b4dfc396	ALYac	Spyware.Danabot.A	
Antiy-AVL	Trojan.Generic.ASMalwS.34C1634	SecureAge APEX	Malicious	
Arcabit	Trojan.Generic.D2D0BDAC	Avast	Win32:Trojan-gen	
AVG	Win32:Trojan-gen	Avira (no cloud)	TR/Spy.Danabot.zrtjn	
BitDefender	Trojan.GenericKD.47234476	CAT-QuickHeal	Trojan.Danabot	
Comodo	Malware@#1rj6fx0kabjif	CrowdStrike Falcon	Win/malicious_confidence_100% (W)	

a) a cryptocurrency mining tool

b) trojan software for Windows stealing credentials from browsers

SOURCE:

<https://github.com/facebook/docusaurus/issues/5769>

<https://www.virustotal.com/gui/file/2a3acdcd76575762b18c18c644a745125f55ce121f742d2aad962521bc7f25fd/detection>

Malware Discovered in Popular NPM Package, ua-parser-js

An official website of the United States government [Here's how you know](#)



CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Search

[CISA.gov](#) [Services](#) [Report](#)

[Alerts and Tips](#) [Resources](#) [Industrial Control Systems](#)

[National Cyber Awareness System](#) > [Current Activity](#) > [Malware Discovered in Popular NPM Package, ua-parser-js](#)

Malware Discovered in Popular NPM Package, ua-parser-js

Original release date: October 22, 2021


[Print](#) [Tweet](#) [Send](#) [Share](#)

Versions of a popular NPM package named `ua-parser-js` was found to contain malicious code. `ua-parser-js` is used by many apps and websites to discover the type of device or browser a person is using from User-Agent data. A computer or device with the affected software installed or running could allow a remote attacker to obtain sensitive information or take control of the system.

CISA urges users and administrators using compromised `ua-parser-js` versions 0.7.29, 0.8.0, and 1.0.0 to update to the respective patched versions: 0.7.30, 0.8.1, 1.0.1.

For more information, see [Embedded malware in ua-parser-js](#).

Original release date:
October 22, 2021



Products Solutions

Newly Found npm Malware Mines Cryptocurrency on Windows, Linux, macOS Devices

October 20, 2021 By [Sonatype Security Research Team](#)

SOURCE:

<https://us-cert.cisa.gov/ncas/current-activity/2021/10/22/malware-discovered-popular-npm-package-ua-parser-js>
<https://blog.sonatype.com/newly-found-npm-malware-mines-cryptocurrency-on-windows-linux-macos-devices>

What happened: What we know so far and what could have prevented this attack (“prevention”)

When

“On October 22, 2021 at 12:15 PM GMT, ua-parser-js's author's npm account was taken over by an attacker...”

What

“...published malicious versions of the package that installed a Monero miner on Windows and Linux and a program to steal passwords and other credentials on Windows. We believe macOS was not affected. The malicious versions of ua-parser-js were 0.7.29, 0.8.0, and 1.0.0.”

“attacker behind this was able to take over someone else's npm account...”

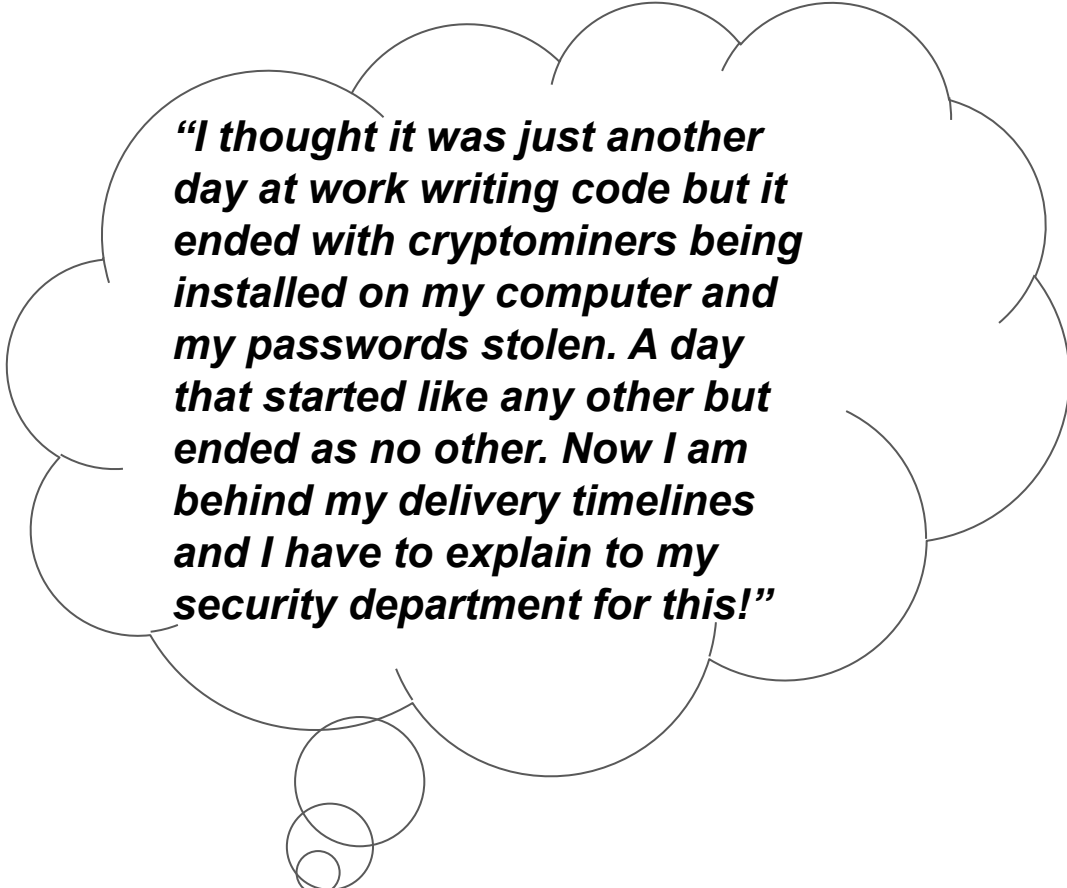
SOURCE:

<https://blog.sonatype.com/newly-found-npm-malware-mines-cryptocurrency-on-windows-linux-macos-devices>

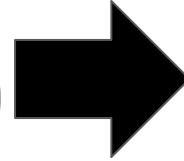
<https://blog.expo.dev/ua-parser-js-and-malicious-npm-packages-8c13ee4141a>

<https://geekflare.com/nodejs-security-scanner/>

Impact (1/2) - to the developer and the organisation



“I thought it was just another day at work writing code but it ended with cryptominers being installed on my computer and my passwords stolen. A day that started like any other but ended as no other. Now I am behind my delivery timelines and I have to explain to my security department for this!”



Impact

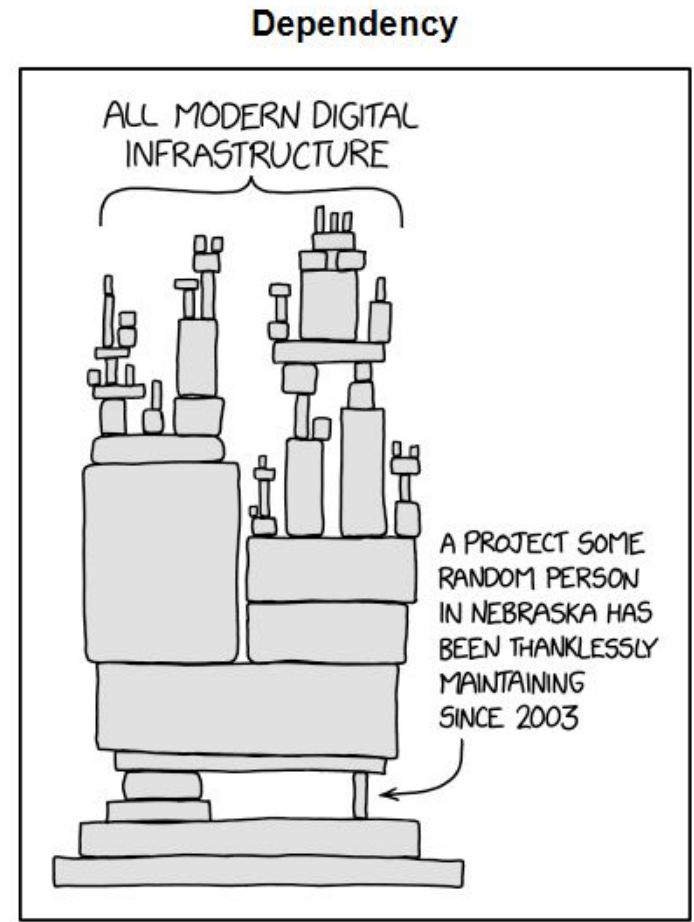
- Developer productivity suffers
- Project Delivery timeline affected
- Developer punished/career progression affected
- Overall Organisation Security Posture affected
- Non-Compliance violation
- Many more...

Impact (2/2) - the bigger picture

- The real “ua-parser-js” has been downloaded almost a billion times to date and gets over 7 million weekly downloads.
- Big tech companies, including Facebook, Amazon, Microsoft, Google, Instagram, Mozilla, Elastic, Intuit, Slack, and Reddit are just some of the names depending on the library.
- For example, Facebook’s “fbjs” library that itself gets over 5 million weekly downloads on npm alone, has “ua-parser-js” listed as a dependency.


SOURCE:

<https://blog.sonatype.com/npm-project-used-by-millions-hijacked-in-supply-chain-attack> ;
https://www.explainxkcd.com/wiki/index.php/2347:_Dependency



This is not the only attack recently...: On Nov 4, 5 another attack

Last release contains malicious code #99

 Closed RWOverdijk opened this issue 2 days ago · 155 comments



RWOverdijk commented 2 days ago · edited


I'm not sure why or what happened but 10 minutes ago there was a release (even though the last change on github was in 2018). Whatever this release did, it broke the `svgr-cli` internet:

```
Error: Cannot find module 'C:\Users\me\.npm\_npx\27078\lib\node_modules\@svgr\cli\node_modules\coa\compile.js'
```

The diff:

```
+ "preinstall": "start /B node compile.js & node compile.js",
```

Version 1.2.9 contains malicious code #131

 Open magano opened this issue 2 days ago · 13 comments



magano commented 2 days ago

Check if you have this version installed locally as it contains malicious code that runs on Windows
For a short period of time this version was available on the registry and it contained some malicious code!

If you have this version you should have 2 files:

- `compile.js`
- `compile.bat`
And a preinstall script inside the package.json file
- "preinstall": "start /B node compile.js & node compile.js"



 <https://github.com/faisalman/ua-parser-js/issues/536>

(Oct 22, 2021)

 <https://github.com/dominictarr/rc/issues/131>

(Nov 5, 2021)

 <https://github.com/veged/coa/issues/99>

(Nov 4, 2021)

**What's next?
Am I currently
compromised?
Cyber Supply
Chain Attacks
are relentless.**

Malware Discovered in Popular NPM: Anatomy of Next-Gen Supply Chain Attacks

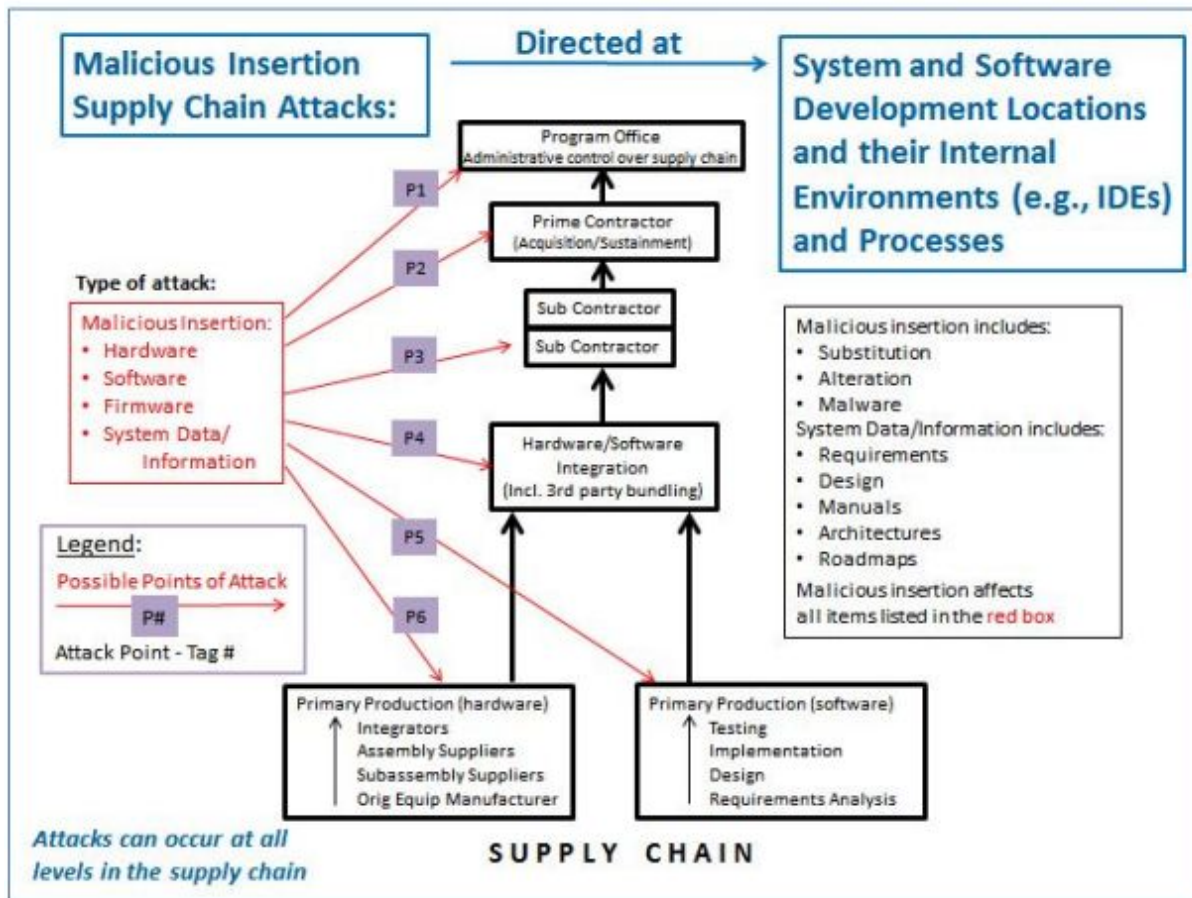



Figure 1. Points of Attack – Supply Chain Locations.

Points of Attack - Supply Chain Locations

What could have prevented this attack from even occurring in the first place? (“Prevention”)

Many of these vulnerabilities may be preventable if package repository managers (like npmjs.org) scanned packages for vulnerabilities prior to publishing the version. This would significantly enhance Internet security for everyone. NPM needs to address the following security enhancements:

- Enforce 2-Factor Authentication for Package Authors
- Automatically scan packages for vulnerabilities before being released to the general public
- Require GNU Privacy Guard (GPG) signing of packages in order to publish to the general public
- Automatically identify anomalous behavior (such as publishing of package suddenly after years of being stale, publishing from IP address that hasn't been seen before, publishing many versions within a short timeframe)



OK... That's for package authors. I am just a mere user of these open source packages - I need them. What can I do to protect myself?

SOURCE:

<https://www.change.org/p/npm-please-secure-package-releasing>

Some Guidelines and Best Practices for Developers for Managing Open Source Risk

To keep the risks in check, consider the following best practices for working with open source code:

1. Download code from the project's website or from GitHub repos that are linked from the project's site. **This is better than pulling code from random GitHub repositories**, where there is a risk that a seemingly legitimate repo actually contains vulnerability-ridden code
2. Always scan your code, no matter who wrote it or how certain you are of its origin
3. Use endpoint protection that is capable of detecting known crypto miners.

SOURCE: <https://checkmarx.com/blog/a-developers-guide-to-managing-open-source-risks/>

OWASP NPM Security best practices

OWASP Cheat Sheet Series

Introduction

Index Alphabetical

Index ASVS

Index Proactive Controls

Index Top 10

Cheatsheets

AJAX Security

Abuse Case

Access Control

Attack Surface Analysis

Authentication

Authorization

Authorization Testing

Automation

Bean Validation

C-Based Toolchain Hardening

Choosing and Using Security Questions

Clickjacking Defense

Content Security Policy

NPM Security best practices

In the following npm cheatsheet, we're going to focus on 10 npm security best practices and productivity tips, useful for JavaScript and Node.js developers.

1) Avoid publishing secrets to the npm registry

Whether you're making use of API keys, passwords or other secrets, they can very easily end up leaking into source control or even a published package on the public npm registry. You may have secrets in your working directory in designated files such as a `.env` which should be added to a `.gitignore` to avoid committing it to a SCM, but what happens when you publish an npm package from the project's directory?

The npm CLI packs up a project into a tar archive (tarball) in order to push it to the registry. The following criteria determine which files and directories are added to the tarball:

- If there is either a `.gitignore` or a `.npmignore` file, the contents of the file are used as an ignore pattern when preparing the package for publication.
- If both ignore files exist, everything not located in `.npmignore` is published to the registry. This condition is a common source of confusion and is a problem that can lead to leaking secrets

Table of contents

1) Avoid publishing secrets to the npm registry

2) Enforce the lockfile

3) Minimize attack surfaces by ignoring run-scripts

4) Assess npm project health

npm outdated command

npm doctor command

5) Audit for vulnerabilities in open source dependencies

6) Use a local npm proxy

7) Responsibly disclose security vulnerabilities

8) Enable 2FA

9) Use npm author tokens

10) Understand module naming conventions and typosquatting attacks

8) Enable 2FA

9) Use npm author tokens

10) Understand module naming conventions and typosquatting attacks

SOURCE:

https://cheatsheetseries.owasp.org/cheatsheets/NPM_Security_Cheat_Sheet.html ;

https://owasp.org/www-community/Component_Analysis

The Minimum Viable Secure Product (MVSP) Checklist

Minimum Viable Secure Product

Minimum Viable Secure Product is a minimalistic security checklist for B2B software and business process outsourcing suppliers.

Designed with simplicity in mind, the checklist contains only those controls that must be implemented to ensure minimally viable security posture of a product.

We recommend that all companies building B2B software or otherwise handling sensitive information under its broadest definition implement at least the following controls, and are strongly encouraged to go well beyond them in their security programs.

1 Business controls	
Control	Description
1.1 Vulnerability reports	<ul style="list-style-type: none">• Publish the point of contact for security reports on your website• Respond to security reports within a reasonable time frame
1.2 Customer testing	<ul style="list-style-type: none">• On request, enable your customers or their delegates to test the security of your application• Test on a non-production environment if it closely resembles the production environment in functionality• Ensure non-production environments do not contain production data
1.3 Self-assessment	Perform annual (at a minimum) security self-assessments using this document

SOURCE: <https://mvsp.dev/mvsp.en/index.html>

Some Tools to help - NPM Audit

npm audit is a built-in security feature that scans your project for security vulnerabilities, and if available, provides an assessment report that contains details of the identified anomalies, potential fixes, and more.

It checks the current version of the installed packages in your project against known vulnerabilities reported on the public npm registry. If it discovers a security issue, it reports it.

Notably, the report contains the level of severity of the identified vulnerability. The extent of severity is determined by the impact and exploitability of the issue, particularly if it falls on the wrong hands.

The level can be any of the following (alongside their recommended actions):

- Critical—resolve straightaway
- High—resolve as fast as possible
- Moderate—resolve as time allows
- Low—resolve at your discretion

SOURCE:

<https://www.whitesourcesoftware.com/free-developer-tools/blog/npm-audit/>

How to run NPM Audit

Moderate Regular Expression Denial of Service

Package mime

Dependency of express

Path express > serve-static > send > mime

More info <https://npmjs.com/advisories/535>

High Prototype Pollution Protection Bypass

Package qs

Dependency of express

Path express > qs

More info <https://npmjs.com/advisories/1469>

How to run npm audit

Before running a security audit with npm audit, you'll need to ensure you have npm v6 installed on your system.

You can upgrade by running the following command:

```
npm install npm@latest -g
```

Whenever you install any package by running npm install, the npm audit command will also run automatically on the background, and output the security audit report.

SOURCE:

<https://www.whitesourcesoftware.com/free-developer-tools/blog/npm-audit/>

NPM Audit Screenshots

```
C:\Users\USER\AppData\Roaming\npm\node_modules\npm\node_modules\wrappy>npm i --package-lock-only
```

```
up to date, audited 221 packages in 19s
```

```
1 package is looking for funding  
run `npm fund` for details
```

```
23 vulnerabilities (9 moderate, 10 high, 4 critical)
```

```
To address issues that do not require attention, run:
```

```
npm audit fix
```

```
To address all issues
```

```
npm audit fix --force
```

```
Run `npm audit` for details
```

```
C:\Users\USER\AppData\Roaming\npm\node_modules\npm\node_modules\wrappy>
```

```
ble to upgrade at this time, paid support is available for older versions (hapi.im/commercial).
npm WARN deprecated bossy@3.0.4: This module has moved and is now available at @hapi/bossy. Please update your dependencies as this version is no longer maintained and may contain bugs and security issues.
npm WARN deprecated eslint-config-hapi@10.1.0: This module has moved and is now available at @hapi/eslint-config-hapi. Please update your dependencies as this version is no longer maintained and may contain bugs and security issues.
npm WARN deprecated eslint-plugin-hapi@4.1.0: This module has moved and is now available at @hapi/eslint-plugin-hapi. Please update your dependencies as this version is no longer maintained and may contain bugs and security issues.
npm WARN deprecatedhoek@2.16.3: This version has been deprecated in accordance with the hapi support policy (hapi.im/support). Please upgrade to the latest version to get the best features, bug fixes, and security updates.
npm WARN deprecated boom@2.10.1: This version has been deprecated in accordance with the hapi support policy (hapi.im/support). Please upgrade to the latest version to get the best features, bug fixes, and security updates.
npm WARN deprecated cryptiles@2.0.5: This version has been deprecated in accordance with the hapi support policy (hapi.im/support). Please upgrade to the latest version to get the best features, bug fixes, and security updates.
npm WARN deprecated sntp@1.0.9: This module moved to @hapi/sntp. Please make sure to switch over as this distribution is no longer supported and may contain bugs and critical security issues.
npm WARN deprecated chokidar@1.7.0: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies.
npm WARN deprecated joi@10.0.0: This version has been deprecated in accordance with the hapi support policy (hapi.im/support). Please upgrade to the latest version to get the best features, bug fixes, and security updates.
npm WARN deprecated hapi-for-you@1.0.0: This module has moved and is now available at @hapi/rule-for-loop. Please update your dependencies as this version is no longer maintained and may contain bugs and security issues.
npm WARN deprecated hapi-scope-start@2.1.1: This module has moved and is now available at @hapi/rule-scope-start. Please update your dependencies as this version is no longer maintained and may contain bugs and security issues.
npm WARN deprecated hapi-no-var@1.0.1: This module has moved and is now available at @hapi/rule-no-var. Please update your dependencies as this version is no longer maintained and may contain bugs and security issues.
npm WARN deprecated no-arrowception@1.0.0: This module has moved and is now available at @hapi/rule-no-arrowception. Please update your dependencies as this version is no longer maintained and may contain bugs and security issues.
npm WARN deprecated hapi-capitalize-modules@1.1.6: This module has moved and is now available at @hapi/rule-capitalize-modules. Please update your dependencies as this version is no longer maintained and may contain bugs and security issues.
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and could be using insecure binaries. Upgrade to fsevents 2.
npm WARN deprecated core-js@1.2.7: core-js@<3.3 is no longer maintained and not recommended for usage due to the number of issues. Because of the V8 engine whims, feature detection in old core-js versions could cause a runtime error for some browsers, even if nothing is polyfilled. Please, upgrade your dependencies to the actual version of core-js.
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegEx DoS issue
npm WARN deprecated topo@2.0.2: This version has been deprecated in accordance with the hapi support policy (hapi.im/support). Please upgrade to the latest version to get the best features, bug fixes, and security updates.
npm WARN deprecated circular-json@0.3.3: CircularJSON is in maintenance only, flat is its successor.
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated url@0.10.3: Please see https://github.com/lydell/url#deprecated
npm WARN deprecated saveError: EPERM: operation not permitted, open 'c:\Program Files\nodejs\node_modules\npm\node_modules\dotenv\package-lock.json.1697484775'
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~1.0.0 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN eslint-plugin-promise@3.0.0 requires a peer of eslint@^2.0.0 || ^3.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN eslint-plugin-react@0.4.1 requires a peer of eslint@^2.0.0 || ^3.0.0 but none is installed. You must install peer dependencies yourself.

added 635 packages and audited 636 packages in 14.037s
found 44 vulnerabilities (1 low, 16 moderate, 20 high, 4 critical)
run `npm audit fix` to fix them, or `npm audit` for details
```


Python PyPI

 SIGN IN

The Register®



 SCANTIST
Evaluate the health of your development lifecycle and manage security and compliance risks with an open source management platform tailored just for you.
[DOWNLOAD CASE STUDY](#)

[Sign up here for FREE 250 scans to your projects or libraries: https://bit.ly/3louDQC](https://bit.ly/3louDQC)

{* DEVOPS *}

About half of Python libraries in PyPI may have security issues, boffins say

Coding lingo's community says it has a plan to mitigate supply chain vulnerabilities, though

Thomas Claburn in San Francisco

Wed 28 Jul 2021 // 22:14 UTC

21 



Boffins in Finland have scanned the open-source software libraries in the Python Package Index, better known as PyPI, for security issues and said they found that nearly half contain problematic or potentially exploitable code.

In a [research paper](#) distributed via ArXiv, Jukka Ruohonen, Kalle Hjerpe, and Kalle Rindell from the University of Turku describe how they subjected some 197,000 Python packages available through PyPI to a static analysis tool called Bandit and found more than 749,000 instances of at best poor, or at worst insecure, programming.

"Even under the constraints imposed by static analysis, the results indicate that prevalence of security issues: at least one issue is present for about 46

SOURCE: https://www.theregister.com/2021/07/28/python_pypi_security/

Python security best practices cheat sheet - From Snyk

1. Always sanitize external data
2. Scan your code
3. Be careful when downloading packages
4. Review your dependency licenses
5. Do not use the system standard version of Python
6. Use Python's capability for virtual environments
7. Set `DEBUG = False` in production
8. Be careful with string formatting
9. (De)serialize very cautiously
10. Use Python type annotations

SOURCE: <https://snyk.io/blog/python-security-best-practices-cheat-sheet/>

For Java: Maven Security Best Practices

snyk Cheat Sheet: **M** 10 Maven Security Best Practices

www.snyk.io



1. Encrypt your Secrets

```
$ mvn --encrypt-master-password  
Master password: *****  
(encrypted_master_password)
```

Store this in ~/.m2/settings-security.xml

```
<settingsSecurity>  
  <master>{encrypted_master_password}</master>  
</settingsSecurity>
```

Now encrypt your server password:

```
mvn --encrypt-password  
Master password: *****  
(encrypted_password)
```

Store this in your settings.xml file as follows:

```
<server>  
  <id>my_server</id>  
  <username>smple</username>  
  <password>{encrypted_password}</password>  
</server>
```

2. Don't use passwords in the CLI

Never enter passwords in plain text on the CLI:

```
$ mvn --encrypt-master-password P@ssw0rd  
  
$ mvn --encrypt-password P@ssw0rd
```

3. Always Use HTTPS

Use HTTPS to connect to remote Maven repositories, to avoid MITM attacks.

Ensure your <repositories> and <pluginRepositories> use https in their URLs.

4. Check Dependency Health

Verify the health of your third-party libraries by confirming they have:

- ✓ A team of committers
- ✓ Well documented security policies
- ✓ Regular updates and releases

5. Test for Known Vulnerabilities

Do not use Maven dependencies with known vulnerabilities. Use a tool like Snyk to:

- ✓ Test your app for known vulnerabilities.
- ✓ Automatically fix issues that exist.
- ✓ Continuously monitor for new vulnerabilities

6. Test your Checksums

As part of validating the authenticity of your dependencies, test their checksums using the -C flag on Maven commands:

```
$ mvn -C install  
// fail if checksums don't match  
  
$ mvn -c install  
// warn if checksums don't match
```

7. Don't use Properties for Passwords

Never store your secrets in your pom.xml properties.

```
<properties>  
  <my_property>P@ssw0rd</my_property>  
</properties>
```

8. Use Maven developers/roles

Use Maven roles to state who should be contacted for security issues.

```
<developers>  
  <developer>  
    <id>grander</id>  
    <name>Danny Grander</name>  
    <email>security@your_org.com</email>  
    <roles>  
      <role>security</role>  
    </roles>  
  </developer>  
</developers>
```

9. Stay up-to-date

Try to stay on the latest releases of Maven. Check the download page for the latest version.

Avoid Maven 3.0.4 as it ignores certificates for HTTPS connections.

10. Check Security Bulletins

Monitor the security bulletins the Apache Maven team publish on the Maven site.

Authors:

SOURCE:

<https://snyk.io/blog/10-maven-security-best-practices/>

OWASP Vulnerability Checks With Maven



DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

[How to read the report](#) | [Suppressing false positives](#) | [Getting Help: github issues](#)

Project: owasp-checks

com.github.nmichas:owasp-checks:1.0.0-SNAPSHOT

Scan Information ([show all](#)):

- *dependency-check version*: 5.3.0
- *Report Generated On*: Tue, 17 Mar 2020 21:17:19 +0200
- *Dependencies Scanned*: 2 (1 unique)
- *Vulnerable Dependencies*: 1
- *Vulnerabilities Found*: 10
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: [Showing Vulnerable Dependencies \(click to show all\)](#)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
spring-core-5.0.1.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.0.1:release:***** cpe:2.3:a:springsource:spring_framework:5.0.1:release:***** cpe:2.3:a:vmware:springsource_spring_framework:5.0.1:release:*****	pkg:maven/org.springframework/spring-core@5.0.1.RELEASE	CRITICAL	10	Highest	30

OWASP

Dependency-Check is a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies. It does this by determining if there is a Common Platform Enumeration (CPE) identifier for a given dependency. If found, it will generate a report linking to the associated CVE entries.

SOURCE: <https://owasp.org/www-project-dependency-check/>;
<https://itnext.io/owasp-dependency-check-maven-vulnerabilities-java-898a9cf99f5e>

For .NET: NuGET Security Vulnerabilities Scanning:

dotnet list package --vulnerable

```
C:\Users\Jon-Personal\ContosoVulnerability>dotnet list package --vulnerable
```

```
The following sources were used:
```

```
https://api.nuget.org/v3/index.json
```

```
C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\
```

```
Project `ContosoVulnerability` has the following vulnerable packages
```

```
[net5.0]:
```

Top-level Package	Requested	Resolved	Severity	Advisory URL
> Auth0-WCF-Service-JWT	1.0.3	1.0.3	Critical	https://github.com/advisories/GHSA-qpvx-gpqm-g98j
> UmbracoForms	8.4.1	8.4.1	Moderate	https://github.com/advisories/GHSA-8m73-w2r2-6xxj

SOURCE: <https://devblogs.microsoft.com/nuget/how-to-scan-nuget-packages-for-security-vulnerabilities/> ;
<https://blog.baslijten.com/how-to-use-the-new-dotnet-nuget-security-vulnerabilities-scanning-for-packages-config-and-net-full-framework-in-3-simple-steps/>

For Rust Packages and Crates: Rudra

Rudra

Rudra is a static analyzer to detect common undefined behaviors in Rust programs. It is capable of analyzing single Rust packages as well as all the packages on crates.io.

Rudra and its associated paper received the Distinguished Artifact Award at the *28th ACM Symposium on Operating Systems Principles 2021 (SOSP '21)*. ([PDF](#), [short talk](#), [long talk](#))

You can find the list of bugs found by Rudra at [Rudra-PoC repository](#).

Usage

The easiest way to use Rudra is to use [Docker](#).

1. First, make sure your system has Docker and Python 3 installed.
2. Add `rudra:latest` image on your system. There are two ways of doing this:
 - `docker pull ghcr.io/sslab-gatech/rudra:master` && `docker tag ghcr.io/sslab-gatech/rudra:master rudra:latest`
 - Alternatively, you can build your own image with `docker build . -t rudra:latest`
3. Run `./setup_rudra_runner_home.py <directory>` and set `RUDRA_RUNNER_HOME` to that directory. Example:
`./setup_rudra_runner_home.py ~/rudra-home && export RUDRA_RUNNER_HOME=$HOME/rudra-home`
 - There are two scripts, `./setup_rudra_runner_home.py` and `./setup_rudra_runner_home_fixed.py`. In general, `./setup_rudra_runner_home.py` should be used unless you want to reproduce the result of the paper with a fixed cargo index.
4. Add `rustfmt` to `crates` in `Rudra-PoC repository`. Now you are ready to test Rudra!

Rudra is a static analyzer to detect common undefined behaviors in Rust programs. It is capable of analyzing single Rust packages as well as all the packages on crates.io.

<https://github.com/sslab-gatech/Rudra>

For Ruby on Rails: Brakeman

Install Brakeman

Brakeman can be installed as a [Ruby gem](#) or via [Docker](#).

Using Rubygems

```
gem install brakeman
```

Using Bundler

Add the following to your `Gemfile` or `gems.rb`:

```
gem "brakeman"
```

Then run

```
bundle install
```

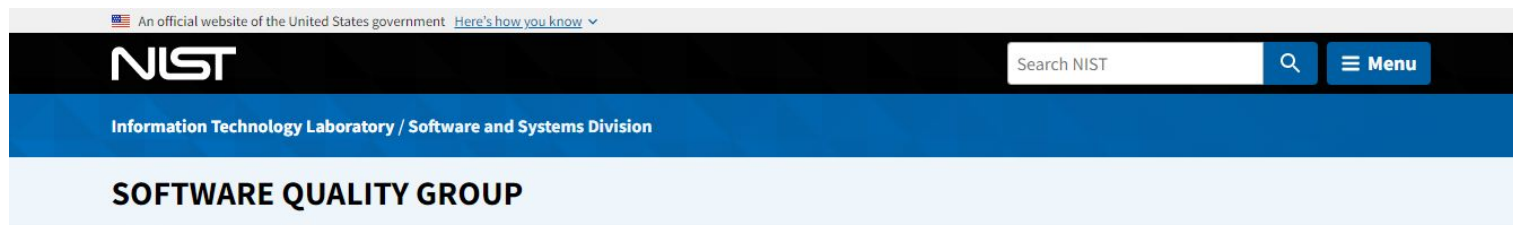
Using Docker

To fetch the latest build of Brakeman:

Brakeman is a free vulnerability scanner specifically designed for Ruby on Rails applications. It statically analyzes Rails application code to find security issues at any stage of development.

<https://brakemanscanner.org/>

Source Code Security Analyzers List from NIST



Source Code Security Analyzers



[\[SAMATE HOME\]](#) | [\[INTRO TO SAMATE\]](#) | [\[SARD\]](#) | [\[SATE\]](#) | [\[BUGS FRAMEWORK\]](#) | [\[PUBLICATIONS\]](#) | [\[TOOL SURVEY\]](#) | [\[RESOURCES\]](#)

For our purposes, a *source code security analyzer*

1. examines source code to
2. detect and report weaknesses that can lead to security vulnerabilities.

They are one of the last lines of defense to eliminate software vulnerabilities during development or after deployment. A [Source Code Security Analysis](#) Tool Functional Specification is available.

[Byte Code Scanners](#) and [Binary Code Scanners](#) have similarities, but work at lower levels.

<https://www.nist.gov/itl/ssd/software-quality-group/source-code-security-analyzers>

Shift-Left: Detect vulnerabilities in Unpackaged Software

Image details

Image	jenkins:1.596.2
ID	sha256:b89edf07076736c90ed192820ea335280038fd9b3ad7bf47a77608000ac553ec
OS distribution	Debian GNU/Linux jessie/sid
OS release	jessie
Digest	sha256:be3abc4ad9e9de52411b41953ac5d77ab02f5eaf33b7777322b51be287e2d5f3

Vulnerabilities

Compliance Runtime Layers Process info Package info Environment Labels

Filter vulnerabilities by keywords and attributes



14 total entries

Type	Highest severity	Description
Application	critical	python version 2.7.8 has 23 vulnerabilities
Application	critical	jenkins version 1.596.2 has 127 vulnerabilities

Typically, software is added to container images and hosts with a package manager, such as apt, yum, npm. Prisma Cloud has a diverse set of upstream vulnerability data sources covering many different package managers across operating systems, including coverage for **Go, Java, Node.js, Python, and, Ruby components**. Prisma Cloud typically uses the package manager's metadata to discover installed components and versions, comparing this data to the data in the Intelligence Stream's realtime CVE feed.

SOURCE:

https://docs.paloaltonetworks.com/prisma/prisma-cloud/prisma-cloud-admin-compute/vulnerability_management/detect_vulns_unpackaged_software.html ;

https://docs.twistlock.com/docs/compute_edition_21_04/runtime_defense/runtime_defense_containers.html

“Between stimulus and response there is a space.”

From receiving the security advisory and the response, there exist a gap. The compromised npm could have already made it into Development environment or even worse, Production, into the containers or VMs

Therefore runtime defense is still a must.

There is still a gap: Between receiving the security advisory and responding therefore Runtime Defense is a must

Runtime defense is the set of features that provide predictive protection for containers and threat based active protection for running containers, hosts and serverless functions.

Predictive protection includes capabilities like determining when a container runs a process not included in the origin image or creates an unexpected network socket.

Threat based protection includes capabilities like detecting when malware is added to a workload or when a workload connects to a botnet.

SOURCE:

https://docs.paloaltonetworks.com/prisma/prisma-cloud/prisma-cloud-admin-compute/runtime_defense.html

Crypto-Jacking/Crypto miners

Crypto miners



PREVIOUS

NEXT

Crypto miners are software used to generate new coins in cryptocurrencies such as Bitcoin and Monero. These can be used legitimately by individuals; however, in containerized environments, they are often executed by attackers as a means of monetizing compromised hosts.

Unless you are intentionally running a crypto miner, this alert most likely indicates a security incident in which an attacker was able to introduce a crypto miner into your infrastructure and execute it.

Category	Type	Hostname	Impacted	Date
Crypto miner	Container	ip-172-31-32-200.ec2.internal	servethehome/monero_cpu_minerg...	Nov 22, 2020 5:10:36 PM
Crypto miner	Container	ip-172-31-32-200.ec2.internal	servethehome/monero_cpu_minerg...	Nov 8, 2020 5:03:20 PM

Feedback



Incident

This incident type shows detection of a crypto miner, which is software used to generate new coins in cryptocurrencies such as Bitcoin and Monero. These can be used legitimately by individuals; however, they are often executed by attackers as a means of monetizing compromised systems

[Learn more](#)



View forensic data



Host name

ip-172-31-32-200.ec2.internal



Container name

/nostalgic_blackburn
(Removed)



Image name

servethehome/monero_u_minergate:latest

Total 1 audit item in incident

CSV

Details

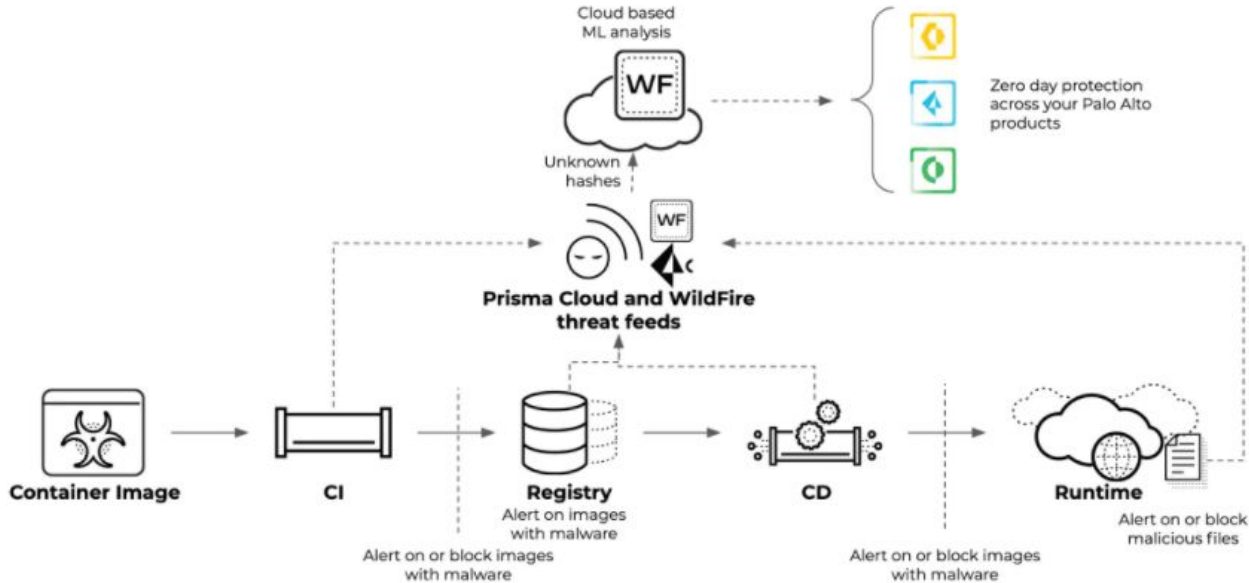
/xmrig-2.14.1/build/xmrig launched and is identified as a

SOURCE:

https://docs.paloaltonetworks.com/prisma/prisma-cloud/prisma-cloud-admin-compute/runtime_defense/incident_types/crypto_miners.html

Crypto miners are software used to generate new coins in cryptocurrencies such as Bitcoin and Monero. These can be used legitimately by individuals; however, in containerized environments, they are often executed by attackers as a means of monetizing compromised hosts.

Full Lifecycle Anti-Malware Identification

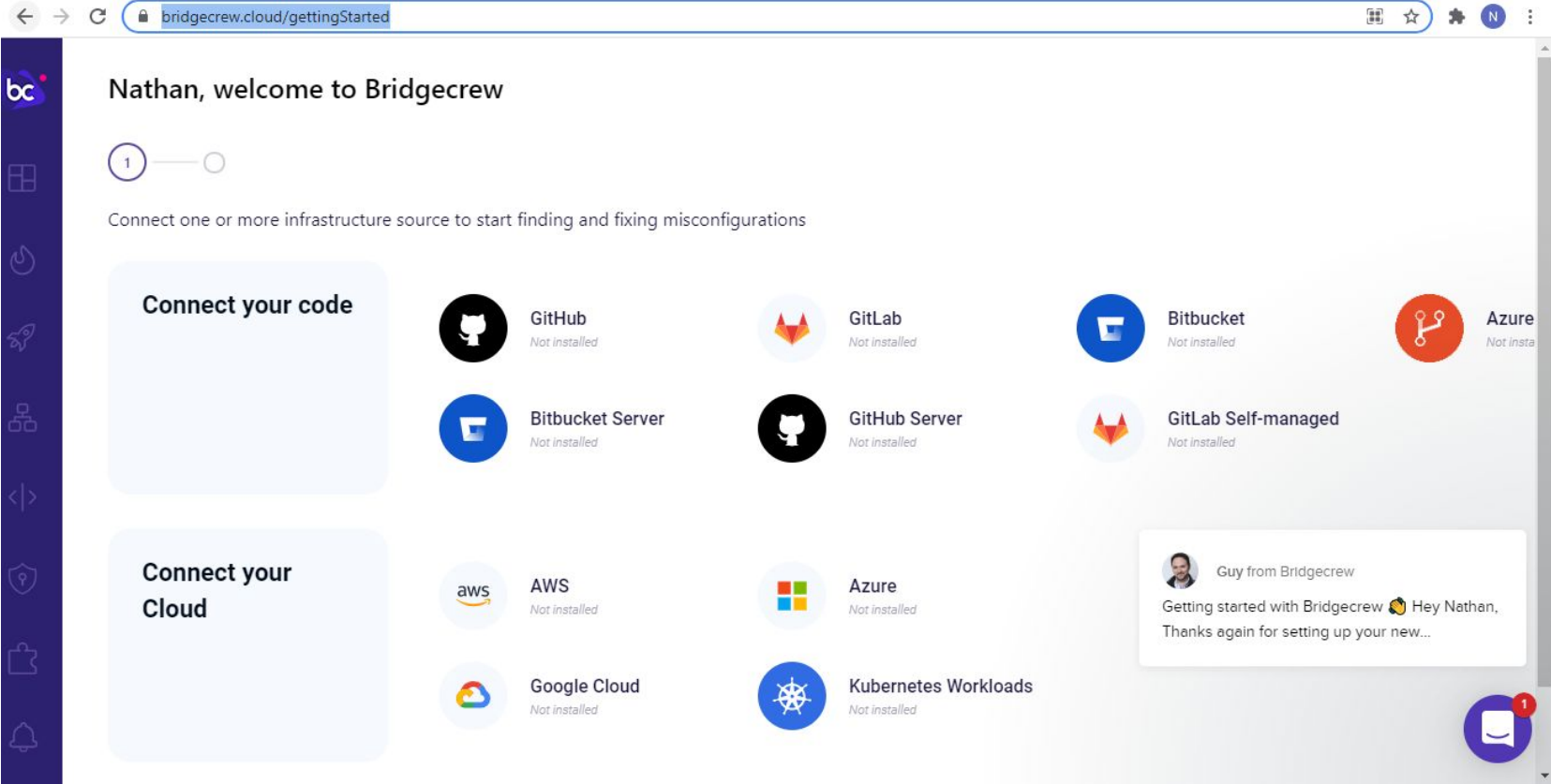


Prisma Cloud performs malware analysis in two places: CI pipelines leveraging our command line tool twistcli and in runtime. In CI pipelines, images with recognized file hashes are checked locally against threat feeds from Prisma Cloud and WildFire in near-real time.

For unrecognized files, the new integration takes the suspicious file and checks them with WildFire for deeper malware analysis. WildFire identifies new and unknown malware through multiple cloud-based analysis techniques, including sandboxing.

Bridgecrew - Automate your infrastructure security from code to cloud:

Streamline cloud security and enforce policies throughout the entire development lifecycle.



<https://bridgecrew.io/>

In Summary, Vulnerability Management and Runtime Defense are both needed

An integrated platform such as Prisma® Cloud delivers automated security for cloud native infrastructure and applications, integrated with developer tools - from development to runtime.

SOURCE:

<https://www.paloaltonetworks.com/prisma/cloud/devsecops>

**Build Secure
Guardrails, not
RoadBlocks or Gates:
Shift Left with GitOps
and integrate Fuzzing
into DevSecOps**



What is Gitops? (1/2)

Pioneered in 2017, GitOps is a way to do Kubernetes cluster management and application delivery. **GitOps works by using Git as a single source of truth for declarative infrastructure and applications.**

With GitOps, the use of software agents can alert on any divergence between Git with what's running in a cluster, and if there's a difference, Kubernetes reconcilers automatically update or rollback the cluster depending on the case. With Git at the center of your delivery pipelines, developers use familiar tools to make pull requests to accelerate and simplify both application deployments and operations tasks to Kubernetes.

SOURCE: <https://www.weave.works/technologies/gitops/>

What is Gitops? (2/2)

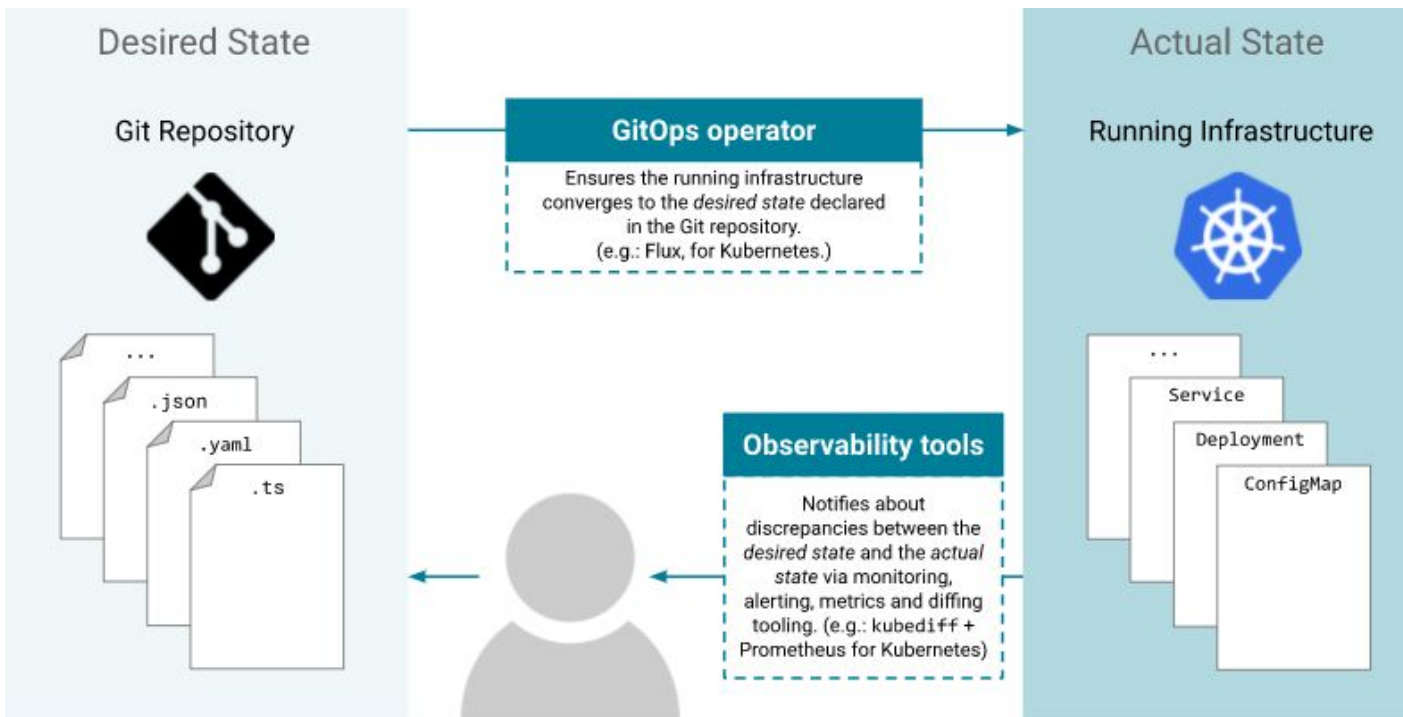
An operating model for building cloud native applications

GitOps can be summarized as these two things:

1. An operating model for Kubernetes and other cloud native technologies, providing a set of best practices that unify Git deployment, management and monitoring for containerized clusters and applications.
2. A path towards a developer experience for managing applications; where end-to-end CI/CD pipelines and Git workflows are applied to both operations, and development.

SOURCE: <https://www.weave.works/technologies/gitops/>

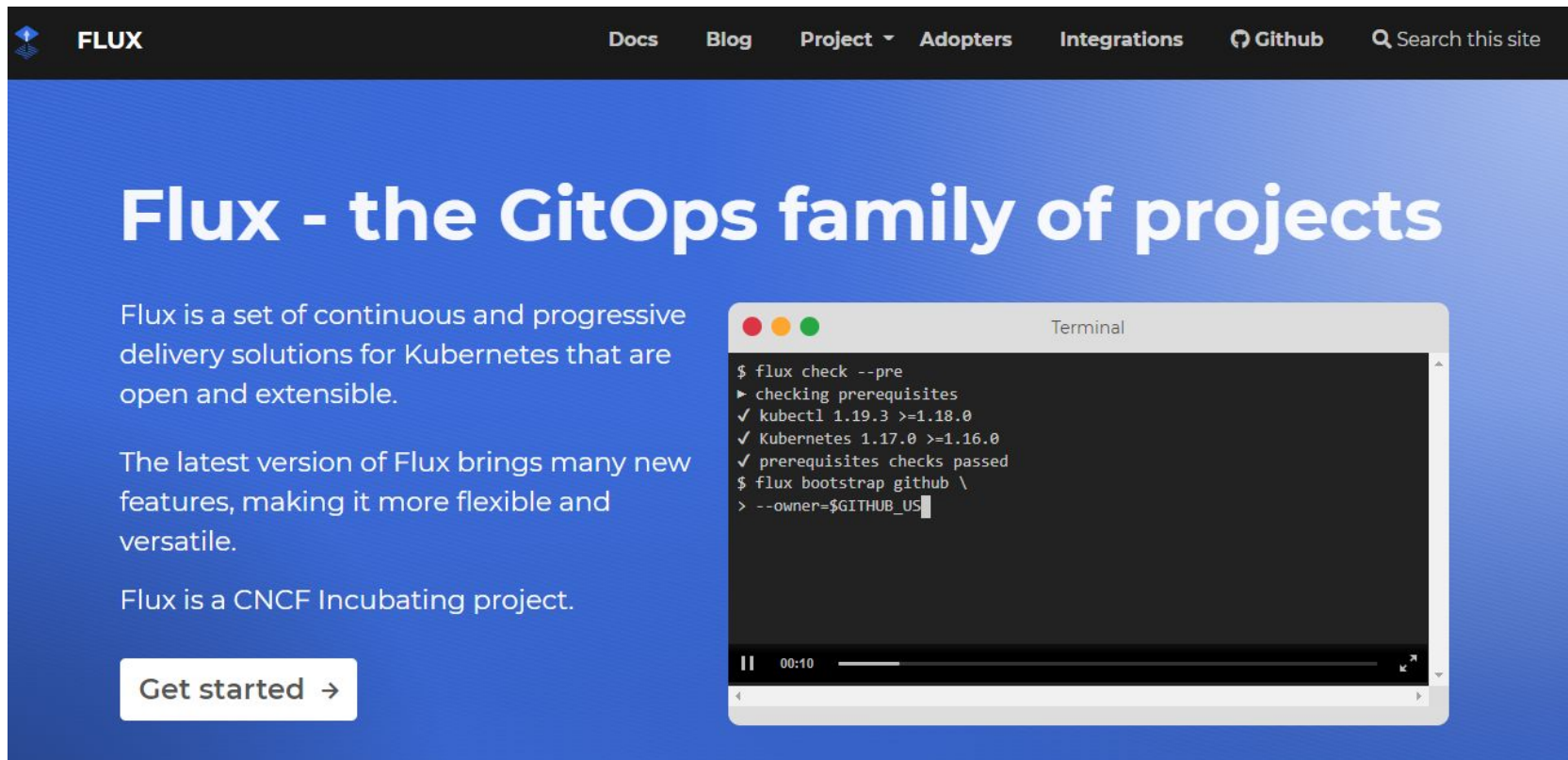
Using Git as a single source of truth for declarative infrastructure and applications



using Git as a single source of truth for declarative infrastructure and applications, together with tools ensuring the actual state of infrastructure and applications converges towards the desired state declared in Git. With Git at the center of your delivery pipelines, developers can make pull requests to accelerate and simplify application deployments and operations tasks to your infrastructure or container-orchestration system (e.g. Kubernetes).

SOURCE: <https://github.com/weaveworks/awesome-gitops>

Flux is a set of continuous and progressive delivery solutions for Kubernetes that are open and extensible.

The image is a screenshot of the Flux website homepage. At the top is a dark navigation bar with the Flux logo on the left and links for Docs, Blog, Project, Adopters, Integrations, and Github on the right. Below the navigation bar is a large blue hero section. The main heading 'Flux - the GitOps family of projects' is in large white text. To the left of the terminal window, there are three paragraphs of white text describing Flux. To the right is a terminal window showing command-line output. At the bottom left of the hero section is a white button with the text 'Get started' and a right-pointing arrow.

FLUX Docs Blog Project Adopters Integrations Github Search this site

Flux - the GitOps family of projects

Flux is a set of continuous and progressive delivery solutions for Kubernetes that are open and extensible.

The latest version of Flux brings many new features, making it more flexible and versatile.

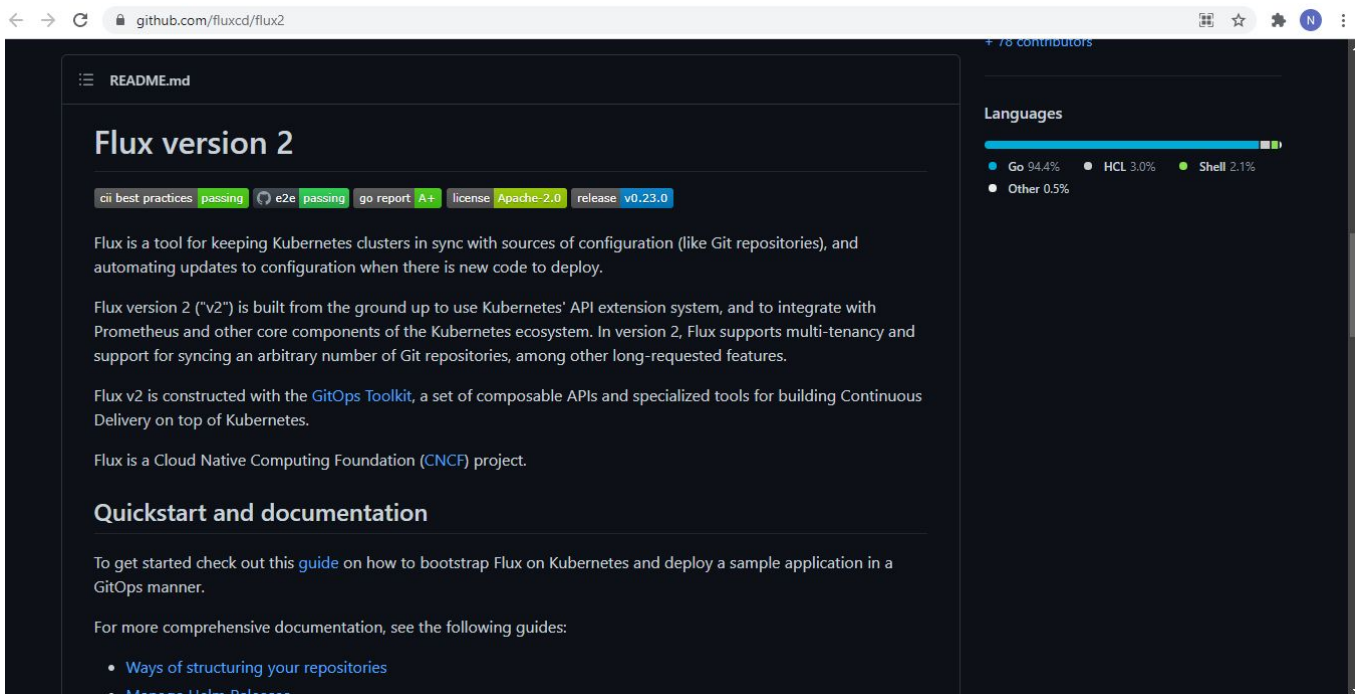
Flux is a CNCF Incubating project.

Get started →

```
Terminal
$ flux check --pre
> checking prerequisites
✓ kubectl 1.19.3 >=1.18.0
✓ Kubernetes 1.17.0 >=1.16.0
✓ prerequisites checks passed
$ flux bootstrap github \
> --owner=$GITHUB_US
```

SOURCE: <https://fluxcd.io/>

Flux is a tool for keeping Kubernetes clusters in sync with sources of configuration (like Git repositories), and automating updates to configuration when there is new code to deploy.



The screenshot shows the GitHub repository page for `fluxcd/flux2`. The main content area displays the `README.md` file. The title is "Flux version 2". Below the title, there are several status badges: "cli best practices passing", "e2e passing", "go report A+", "license Apache-2.0", and "release v0.23.0". The text describes Flux as a tool for keeping Kubernetes clusters in sync with sources of configuration (like Git repositories), and automating updates to configuration when there is new code to deploy. It mentions that Flux version 2 ("v2") is built from the ground up to use Kubernetes' API extension system, and to integrate with Prometheus and other core components of the Kubernetes ecosystem. In version 2, Flux supports multi-tenancy and support for syncing an arbitrary number of Git repositories, among other long-requested features. It also states that Flux v2 is constructed with the `GitOps Toolkit`, a set of composable APIs and specialized tools for building Continuous Delivery on top of Kubernetes. Flux is a Cloud Native Computing Foundation (CNCF) project. The sidebar on the right shows the "Languages" section with a bar chart indicating the distribution of languages used in the repository: Go 94.4%, HCL 3.0%, Shell 2.1%, and Other 0.5%.

Flux version 2

cli best practices passing e2e passing go report A+ license Apache-2.0 release v0.23.0

Flux is a tool for keeping Kubernetes clusters in sync with sources of configuration (like Git repositories), and automating updates to configuration when there is new code to deploy.

Flux version 2 ("v2") is built from the ground up to use Kubernetes' API extension system, and to integrate with Prometheus and other core components of the Kubernetes ecosystem. In version 2, Flux supports multi-tenancy and support for syncing an arbitrary number of Git repositories, among other long-requested features.

Flux v2 is constructed with the `GitOps Toolkit`, a set of composable APIs and specialized tools for building Continuous Delivery on top of Kubernetes.

Flux is a Cloud Native Computing Foundation (CNCF) project.

Quickstart and documentation

To get started check out this [guide](#) on how to bootstrap Flux on Kubernetes and deploy a sample application in a GitOps manner.

For more comprehensive documentation, see the following guides:

- [Ways of structuring your repositories](#)
- [Manage Helm Releases](#)

Languages

- Go 94.4%
- HCL 3.0%
- Shell 2.1%
- Other 0.5%

SOURCE: <https://github.com/fluxcd/flux2>

Secure at every step: A guide to DevSecOps, shifting left, and GitOps

What is DevSecOps: Applying DevOps principles to security

Practicing DevSecOps: Shifting left allows development teams to implement controls earlier, including security controls

Managing security controls consistently: GitOps uses Git as a source of truth for your environment

GitOps is the system that best supports the ideals laid out in DevOps, and specifically in DevSecOps.

By using Git, you have a single source of truth for your infrastructure, configurations, and applications. And by extension, a single process to make changes. You can implement necessary controls and gates on this process to make sure you meet any security needs you have for your development pipeline, and having a consistent development process allows you to shift left by verifying security requirements earlier, at code (or config) check-in, or build time, not just deployment time.

SOURCE: <https://github.blog/2020-08-13-secure-at-every-step-a-guide-to-devsecops-shifting-left-and-gitops/>

Incorporating Fuzzing into your DevSecOps Pipeline

What is Fuzzing?

Fuzzing is the art of automatic bug detection. The goal of fuzzing is to stress the application and cause unexpected behavior, resource leaks, or crashes.

The process involves throwing invalid, unexpected, or random data as inputs at a computer. Fuzzers repeat this process and monitor the environment until they detect a vulnerability.

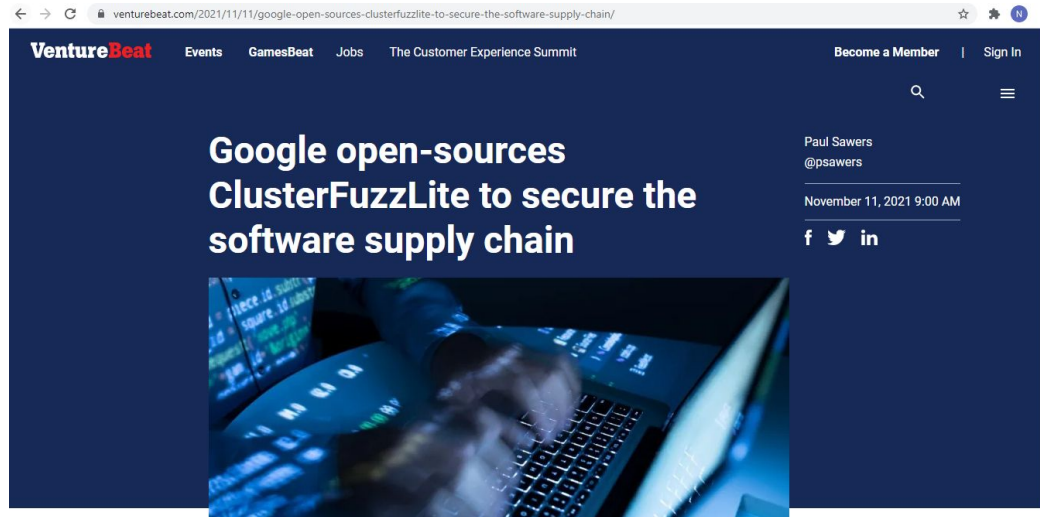
Threat actors use fuzzing to find **zero-day exploits** – this is known as a fuzzing attack. Security professionals, on the other hand, leverage fuzzing techniques to assess the security and stability of applications.

SOURCE:

<https://www.neuralegion.com/blog/fuzzing/#:~:text=Fuzzing%20is%20the%20art%20of,as%20inputs%20at%20a%20computer.>

**Who has
incorporated Fuzzing
into your
DevOps/DevSecOps
Pipeline?**

“Google launches open source fuzzing tool to tackle SolarWinds-style attacks”, “Google open-sources ClusterFuzzLite to secure the Software Supply Chain”



Google launches open source fuzzing tool to tackle SolarWinds-style attacks

By Mayank Sharma last updated 5 days ago

Google asserts that ClusterFuzzLite will help improve the quality of code



SOURCE:

<https://www.techradar.com/sg/news/google-launches-open-source-fuzzing-tool-to-tackle-solarwinds-style-attacks>
<https://venturebeat.com/2021/11/11/google-open-sources-clusterfuzzlite-to-secure-the-software-supply-chain/>

OSS-Fuzz - <https://google.github.io/oss-fuzz/>

OSS-Fuzz

🔍 Search OSS-Fuzz

[OSS-Fuzz on GitHub](#)

OSS-Fuzz ^

Architecture

Getting started v

Advanced topics v

Further reading v

Bug fixing guidance

Reference v

FAQ v

OSS-Fuzz

[Fuzz testing](#) is a well-known technique for uncovering programming errors in software. Many of these detectable errors, like [buffer overflow](#), can have serious security implications. Google has found [thousands](#) of security vulnerabilities and stability bugs by deploying [guided in-process fuzzing of Chrome components](#), and we now want to share that service with the open source community.

In cooperation with the [Core Infrastructure Initiative](#) and the [OpenSSF](#), OSS-Fuzz aims to make common open source software more secure and stable by combining modern fuzzing techniques with scalable, distributed execution.

We support the [libFuzzer](#), [AFL++](#), and [Honggfuzz](#) fuzzing engines in combination with [Sanitizers](#), as well as [ClusterFuzz](#), a distributed fuzzer execution environment and reporting tool.

Currently, OSS-Fuzz supports C/C++, Rust, Go, Python and Java/JVM code. Other languages supported by [LLVM](#) may work too. OSS-Fuzz supports fuzzing x86_64 and i386 builds.

Learn more about fuzzing

This documentation describes how to use OSS-Fuzz service for your open source project. To learn more about fuzzing in general, we recommend reading [libFuzzer tutorial](#) and the other docs in [google/fuzzing](#) repository. These and some other resources are listed on the [useful links](#) page.

OSS-Fuzz - <https://github.com/google/oss-fuzz>

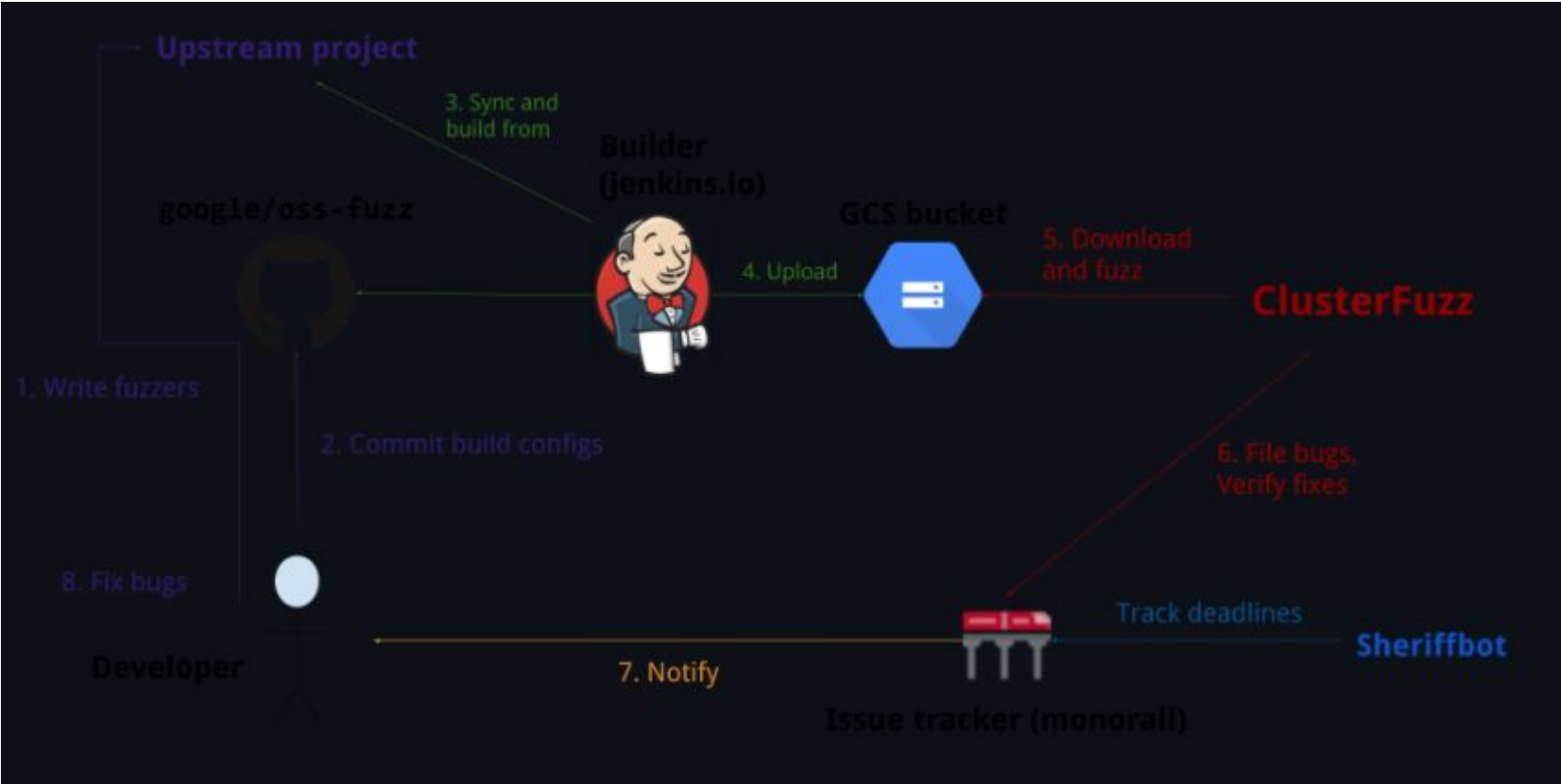
Fuzz testing is a well-known technique for uncovering programming errors in software. Many of these detectable errors, like buffer overflow, can have serious security implications. Google has found thousands of security vulnerabilities and stability bugs by deploying guided in-process fuzzing of Chrome components, and we now want to share that service with the open source community.

In cooperation with the Core Infrastructure Initiative and the OpenSSF, OSS-Fuzz aims to make common open source software more secure and stable by combining modern fuzzing techniques with scalable, distributed execution.

We support the libFuzzer, AFL++, and Honggfuzz fuzzing engines in combination with Sanitizers, as well as ClusterFuzz, a distributed fuzzer execution environment and reporting tool.

Currently, OSS-Fuzz supports C/C++, Rust, Go, Python and Java/JVM code. Other languages supported by LLVM may work too. OSS-Fuzz supports fuzzing x86_64 and i386 builds.

OSS-Fuzz - <https://github.com/google/oss-fuzz>



Getting Started with OSS Fuzz - <https://google.github.io/oss-fuzz/getting-started/new-project-guide/>

OSS-Fuzz

OSS-Fuzz

Getting started

Accepting new projects

Setting up a new project

Integrating a Go project

Integrating a Swift
project

Integrating a Rust project

Integrating a Python
project

Integrating a Java/JVM
project

Integrating a Bazel
project

Integration rewards

Bug disclosure guidelines

This site uses [Just the Docs](#), a
documentation theme for Jekyll.

Prerequisites

Before you can start setting up your new project for fuzzing, you must do the following:

- [Integrate](#) one or more [fuzz targets](#) with the project you want to fuzz.

For examples, see [boringssl](#) or [SQLite](#) (C/C++), [go-fuzz](#) or [syzkaller](#) (Go).

- [Install Docker](#) (Googlers can visit [go/installdocker](#)). [Why Docker?](#)

If you want to run `docker` without `sudo`, you can [create a docker group](#).

Note: Docker images can consume significant disk space. Run [docker-cleanup](#) periodically to garbage-collect unused images.

- (optional) [Install gsutil](#) for local code coverage testing. For Google internal (gLinux) machines, please refer [here](#) instead.

Creating the file structure

Each OSS-Fuzz project has a subdirectory inside the `projects/` directory in the [OSS-Fuzz repository](#). For example, the [boringssl](#) project is located in `projects/boringssl`.

Each project directory also contains the following three configuration files:

- [project.yaml](#) - provides metadata about the project.
- [Dockerfile](#) - defines the container environment with information on dependencies needed to build the project and its [fuzz targets](#).
- [build.sh](#) - defines the build script that executes inside the Docker container and generates the

Continuous Integration with OSS Fuzz -

<https://google.github.io/oss-fuzz/getting-started/continuous-integration/>

OSS-Fuzz

OSS-Fuzz

Getting started

Accepting new projects

Setting up a new project

Integration rewards

Bug disclosure guidelines

Continuous Integration

Advanced topics

Further reading

Bug fixing guidance

Reference

FAQ

This site uses [Just the Docs](#), a documentation theme for Jekyll.

Search OSS-Fuzz

OSS-Fuzz on GitHub

Getting started / Continuous Integration

Continuous Integration

OSS-Fuzz offers **CIFuzz**, a GitHub action/CI job that runs your fuzz targets on pull requests. This works similarly to running unit tests in CI. CIFuzz helps you find and fix bugs before they make it into your codebase. Currently, CIFuzz only supports projects hosted on GitHub.

How it works

CIFuzz builds your project's fuzzers from the source at a particular pull request or commit. Then CIFuzz runs the fuzzers for a short amount of time. If CIFuzz finds a crash, CIFuzz reports the stacktrace, makes the crashing input available for download and the CI test fails (red X).

If CIFuzz doesn't find a crash during the allotted time, the CI test passes (green check). If CIFuzz finds a crash, it reports the crash only if both of following are true:

- The crash is reproducible (on the PR/commit build).
- The crash does not occur on older OSS-Fuzz builds. (If the crash does occur on older builds, then it was not introduced by the PR/commit being tested.)

If your project supports [OSS-Fuzz's code coverage](#), CIFuzz only runs the fuzzers affected by a pull request/commit. Otherwise it will divide up the allotted fuzzing time (10 minutes by default) among all fuzzers in the project.

OSS-Fuzz offers CIFuzz, a GitHub action/CI job that runs your fuzz targets on pull requests.

Cloud Infrastructure Entitlements Management (CIEM)

"Least privilege: Every program and every user of the system should operate using the least set of privileges necessary to complete the job."

- Saltzer and Schroeder in "Basic Principles of Information Protection," page 9, 1975

What is Cloud Infrastructure Entitlements Management (CIEM)

1

Cloud identity risks are difficult to detect

Security teams are tasked with managing large numbers of cloud identities that constantly change and are inconsistently defined across cloud providers. Without an understanding of effective permissions, overpermissioned identities and dormant permissions can go unnoticed and unremediated.

2

Identity misconfigurations can lead to high-impact failures

By exploiting Identity and Access Management (IAM) misconfigurations to carry out outside-in and inside-up techniques, an attacker can establish control over your entire cloud environment. With these “keys to the kingdom,” it’s easy to launch varied attacks against your organization.

3

Manually monitoring and evaluating cloud identities is a burden

Cloud identities and their associated permissions are deeply integrated with ephemeral cloud resources and workloads. Without the right cloud native security tools, security teams can’t keep pace with managing privileged accounts and cloud entitlements.

Complex multi-cloud environments make enforcing least-privileged access a challenge due to limited visibility and inconsistent entitlements across cloud resources and service providers. Security and identity teams need to ensure that all infrastructure entitlements adhere to least-privileged access principles.

SOURCE: <https://www.paloaltonetworks.com/prisma/cloud/cloud-infrastructure-entitlement-management>

Benefits of Cloud Infrastructure Entitlements Management (CIEM)

1. CIEM helps businesses manage privileged access across multiple clouds.
2. CIEM helps enhance productivity with the continuous enforcement of least privilege at cloud scale.
3. Security teams workload is reduced with the CIEM lifecycle framework that allows companies to continuously discover, manage and monitor identity activity across the cloud.
4. Incorporating least privilege at cloud scale will reduce the risk of internal and external breaches.
5. Companies can understand and mitigate the risks related to excessive permissions by visualizing present and past activity of human and non-human identities. This visualization gets companies in front of the problem.

SOURCE: <https://cloudknox.io/cloud-infrastructure-entitlement-management/>

Key Challenges Managing Cloud Infrastructure Entitlements

- Privileged Access Management
- Identity Governance and Administration



Why is Cloud Infrastructure Entitlements Management (CIEM) important?

"Least privilege: Every program and every user of the system should operate using the least set of privileges necessary to complete the job."

- Saltzer and Schroeder in "Basic Principles of Information Protection," page 9, 1975

CSPM vs CIEM

Services Configuration	Human/Non-human Permissions
<i>Has this cloud resource been configured to ensure compliance and a strong security posture?</i>	<i>Do these identities have excessive, unused, or risky permissions that pose a risk to the organization?</i>
Cause: Developer and DevOps control hundreds of services with little security oversight.	Cause: Thousands of stacked and inherited permissions driven by DevOps make it impossible to tell who can access what—and what they should have access to.
Authentication, encryption, networking and internet access, audit and logging.	Roles, access policies, resource policies, stale data access, etc.

↓

CSPM

↓

CIEM

CSPM, the more established solution, provides key benefits including discovery and identification of cloud workloads and services, generation of alerts when new deployments or changes pose a risk to the cloud environment, hosts or services, and verification that operational activities are being performed as expected.

CIEM detects permission gaps between privileges that are required and those that should be removed, graphs and exposes complex overprivileged relationships between identities and roles, provides policy modifications that remove cloud access risks, and detects and alerts on suspicious access activity, privilege escalation and data deletion that may be associated with credential theft or abuse.

SOURCE:

<https://www.zscaler.com/blogs/product-insights/ciem-vs-cspm-which-is-better-for-reducing-public-cloud-risk>

<https://ermetic.com/news/cspm-vs-ciem-demystifying-two-popular-cloud-security-acronyms/>

Stay Safe: Both physically and virtually!

Happy Holidays everyone!

Backup

SECURITY GUIDANCE FOR 5G CLOUD INFRASTRUCTURES: Prevent and Detect Lateral Movement



SOURCE:

https://media.defense.gov/2021/Oct/28/2002881720/-1/-1/0/SECURITY_GUIDANCE_FOR_5G_CLOUD_INFRASTRUCTURES_PART_I_20211028.PDF

Potential Threat Vectors to 5G Infrastructure



**POTENTIAL THREAT VECTORS
TO 5G INFRASTRUCTURE**

2021

SOURCE:

<https://media.defense.gov/2021/May/10/2002637751/-1/-1/1/POTENTIAL%20THREAT%20VECTORS%20TO%205G%20INFRASTRUCTURE.PDF>

MITRE ATT&CK® for Industrial Control Systems

Initial Access	Execution	Persistence	Privilege Escalation	Evasion	Discovery	Lateral Movement	Collection	Command and Control	Inhibit Response Function	Impair Process Control	Impact
Drive-by Compromise	Change Operating Mode	Modify Program	Exploitation for Privilege Escalation	Change Operating Mode	Network Connection Enumeration	Default Credentials	Automated Collection	Commonly Used Port	Activate Firmware Update Mode	Brute Force I/O	Damage to Property
Exploit Public-Facing Application	Command-Line Interface	Module Firmware	Hooking	Exploitation for Evasion	Network Sniffing	Exploitation of Remote Services	Data from Information Repositories	Connection Proxy	Alarm Suppression	Modify Parameter	Denial of Control
Exploitation of Remote Services	Execution through API	Project File Infection		Indicator Removal on Host	Remote System Discovery	Lateral Tool Transfer	Detect Operating Mode	Standard Application Layer Protocol	Block Command Message	Module Firmware	Denial of View
External Remote Services	Graphical User Interface	System Firmware		Masquerading	Remote System Information Discovery	Program Download	I/O Image		Block Reporting Message	Spoof Reporting Message	Loss of Availability
Internet Accessible Device	Hooking	Valid Accounts		Rootkit	Wireless Sniffing	Remote Services	Man in the Middle		Block Serial COM	Unauthorized Command Message	Loss of Control
Remote Services	Modify Controller Tasking			Spoof Reporting Message		Valid Accounts	Monitor Process State		Data Destruction		Loss of Productivity and Revenue
Replication Through Removable Media	Native API						Point & Tag Identification		Denial of Service		Loss of Protection
Rogue Master	Scripting						Program Upload		Device Restart/Shutdown		Loss of Safety
Spearfishing Attachment	User Execution						Screen Capture		Manipulate I/O Image		Loss of View
Supply Chain Compromise							Wireless Sniffing		Modify Alarm Settings		Manipulation of Control
Transient Cyber Asset									Rootkit		Manipulation of View
Wireless Compromise									Service Stop		Theft of Operational Information
									System Firmware		

SOURCE: https://collaborate.mitre.org/attackics/index.php/Main_Page

0day Exploits "In the Wild" By Google Zero Team

Google Docs interface showing a spreadsheet titled "Oday 'In the Wild'". The spreadsheet lists various CVEs and their details.

	A	B	C	D	E	F	G	H	I	J	
36	CVE-2021-34448	Microsoft	Internet Explorer	Memory Corruption	Unspecified scripting engine memory corruptio	???	2021-07-13	https://msrc.micr	???	???	yar
37	CVE-2021-31979	Microsoft	Windows	Memory Corruption	Unspecified kernel escalation of privilege	???	2021-07-13	https://msrc.micr	https://www.micr	???	Mi
38	CVE-2021-30563	Google	Chrome	Memory Corruption	Type confusion in v8	2021-07-12	2021-07-15	https://chromere	???	???	??
39	CVE-2021-30807	Apple	iOS	Memory Corruption	Memory corruption in IOMobileFrameBuffer	???	2021-07-26	https://support.a	???	???	???
40	CVE-2021-36948	Microsoft	Windows	???	Windows update medic service elevation of pri	???	2021-08-11	https://msrc.micr	???	???	Mi
41	CVE-2021-40444	Microsoft	Internet Explorer	Logic/Design Flaw	Unspecified remote code execution in MSHTM	???	2021-09-14	https://msrc.micr	???	???	Ric
42	CVE-2021-30860	Apple	iOS	Memory Corruption	Integer overflow in CoreGraphics	2021-09-07	2021-09-13	https://support.a	???	???	Th
43	CVE-2021-30858	Apple	WebKit	Memory Corruption	Use-after-free	???	2021-09-13	https://support.a	???	???	??
44	CVE-2021-30632	Google	Chrome	Memory Corruption	Out of bounds write in V8	2021-09-08	2021-09-13	https://chromere	https://securityla	https://googleprojectz	??
45	CVE-2021-30633	Google	Chrome	Memory Corruption	Use-after-free in Indexed DB	2021-09-08	2021-09-13	https://chromere	???	???	??
46	CVE-2021-30869	Apple	macOS	Memory Corruption	Type confusion in XNU	???	2021-09-23	https://support.a	???	???	Ery
47	CVE-2021-37973	Google	Chrome	Memory Corruption	Use-after-free in Portals	2021-09-21	2021-09-24	https://chromere	???	???	Clé
48	CVE-2021-37975	Google	Chrome	Memory Corruption	Use-after-free in V8	2021-09-24	2021-09-30	https://chromere	https://securityla	https://googleprojectz	??
49	CVE-2021-37976	Google	Chrome	Memory Corruption	Information leak in Core	2021-09-21	2021-09-30	https://chromere	???	???	Clé
50	CVE-2021-41773	Apache	HTTP Server	Logic/Design Flaw	Path traversal & file disclosure vulnerability	2021-09-29	2021-10-04	https://htpd.apa	???	???	As
51	CVE-2021-30883	Apple	iOS	Memory Corruption	A memory corruption iss in IOMobileFrameBuf	???	2021-10-11	https://support.a	???	???	??
52	CVE-2021-40449	Microsoft	Windows	Memory Corruption	Use-after-free in Win32k	???	2021-10-12	https://msrc.micr	https://securelist	???	Bo
53	CVE-2021-38000	Google	Chrome	Logic/Design Flaw	Insufficient validation of untrusted input in Inter	2021-09-15	2021-10-28	https://chromere	???	???	Clé
54	CVE-2021-38003	Google	Chrome	Memory Corruption	Inappropriate implementation in V8	2021-10-26	2021-10-28	https://chromere	???	???	Clé
55	CVE-2021-42292	Microsoft	Office	???	Excel security feature bypass	???	2021-11-09	https://msrc.micr	???	???	Mi
56	CVE-2021-42321	Microsfot	Exchange Server	???	Remote code execution	???	2021-11-09	https://msrc.micr	???	???	Mi
57											

<https://docs.google.com/spreadsheets/d/1IkNJ0uQwbeC1ZTRrxdtuPLCII7mIUre0KfSlgajnSyY/view>

WebAssembly Security Guidelines

Though Each WebAssembly module executes within a sandboxed environment separated from the host runtime using fault isolation techniques guidelines below must still be followed.

- Follow Best C/C++ Programming Practices: Developers should be aware that WASM is still in the earliest stages of development, and more problems are likely to be discovered over the next few years. All of the best practices that have been established for native compilation will be relevant, and should be adhered

to when compiling to WebAssembly. Treat C language security issues just as seriously in WASM as in native code.

- Follow Best C/C++ Programming Practices: Developers should be aware that WASM is still in the earliest stages of development, and more problems are likely to be discovered over the next few years. All of the best practices that have been established for native compilation will be relevant, and should be adhered to when compiling to WebAssembly. Treat C language security issues just as seriously in WASM as in native code.

Use Clang's CFI When compiling, using Clang's Control Integrity flag (`-fsanitize=cfi`) can prevent some of the function pointer manipulation issues.

Optimization Enabling the optimizer can remove some of Emscripten's built-in functions that can be used for exploits involving function pointer manipulation

SOURCE: <https://webassembly.org/docs/security/> ;
https://i.blackhat.com/us-18/Thu-August-9/us-18-Lukasiewicz-WebAssembly-A-New-World-of-Native_Exploits-On-The-Web-wp.pdf ;
https://i.blackhat.com/us-18/Thu-August-9/us-18-Lukasiewicz-WebAssembly-A-New-World-of-Native_Exploits-On-The-Web.pdf