

Practical Open Source Security

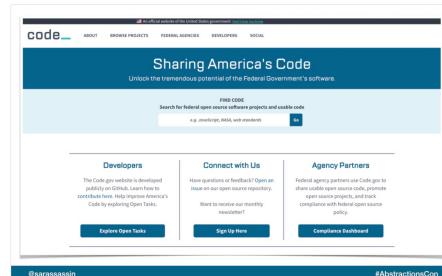
Sara Cope

Practical Open Source Security

Sara Cope (she/her)
code.gov

@sarassassin

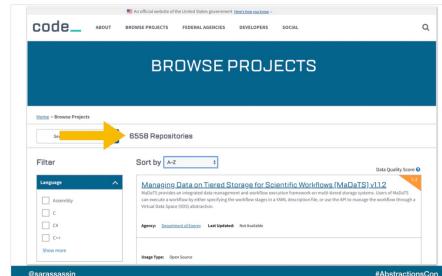
#AbstractionsCon



The screenshot shows the homepage of code.gov. At the top, there's a dark header with the code.gov logo and navigation links for ABOUT, BROWSE PROJECTS, FEDERAL AGENCIES, DEVELOPERS, and SOCIAL. Below the header, a large teal banner reads "Sharing America's Code" with the subtitle "Unlock the tremendous potential of the Federal Government's software". A search bar is present with placeholder text "Search for federal open source projects and code" and a dropdown menu showing "a.js, javascript, node, and modules". Below the banner, there are three main sections: "Developers" (with a sub-note about GitHub), "Connect with Us" (with a note about contributing to an open issue on GitHub), and "Agency Partners" (with a note about sharing code). Buttons for "Explore Open Tasks", "Sign Up Here", and "Compliance Dashboard" are also visible.

@sarassassin

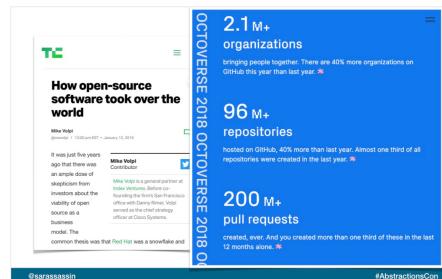
#AbstractionsCon



The screenshot shows the "BROWSE PROJECTS" page on code.gov. It features a large blue header with the text "BROWSE PROJECTS" and a count of "6558 Repositories". Below the header, there's a search bar and a "Sort by" dropdown set to "A-Z". A yellow arrow points to the repository count. To the right, there's a "Data Quality Score" button. The main content area lists repositories with columns for "Name", "Language", "Last Updated", and "Data Quality Score". A filter sidebar on the left allows users to search by language (Assembly, C, C++, C#, etc.) and type (Open Source). Buttons for "Explore Open Tasks" and "Sign Up Here" are at the bottom.

@sarassassin

#AbstractionsCon



The screenshot is from a TechCrunch article by Mike Hiltz. The title is "How open-source software took over the world". The article discusses the growth of GitHub, noting that it was just five years ago that there were only a few hundred repositories. It highlights the rise of organizations like the Linux Foundation and the increasing number of pull requests. A sidebar on the right shows statistics: "2.1 M+ organizations bringing people together", "96 M+ repositories hosted on GitHub", and "200 M+ pull requests created, ever. And you created more than one third of these in the last 12 months alone." The date of the article is January 12, 2018.

@sarassassin

#AbstractionsCon



The screenshot shows a line graph titled "Module Counts" with the URL "http://www.modulecounts.com/". The Y-axis represents the number of packages, ranging from 0 to 1,200,000. The X-axis represents years from 2012 to 2019. The data shows a sharp increase starting around 2015, reaching approximately 1,000,000 packages by 2019. Below the graph, the text "So are package managers like npm" is displayed.

@sarassassin

#AbstractionsCon

I work on an open source project that's about other open source projects and consumes open source projects so I'm basically in open source and advocating for open source all day long. It's amazing and I love it.

When code.gov was first launch in 2016 it had under 50 projects.

Today there are more than 6 thousand.

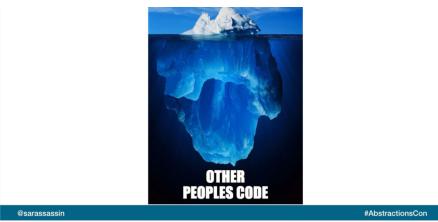
Popularity of open source isn't just skyrocketing in the federal government.

Everyone seems to either want to publish an open source project, be a maintainer or contribute to an open source project.

Which is great because that creates even more software that's available for everyone else to consume.

And we can see that in the way that package managers like NPM are growing as well. And as a result, we're adding more and more dependencies to our applications.

So we see apps shifting to use more and more dependencies so much that amount of code we're writing at times can be very small compared to the amount of code we're consuming.



@sarassassin

#AbstractionsCon

Using someone else's code isn't just set it and forget, that code at any time could potentially contain vulnerabilities.



@sarassassin

#AbstractionsCon

So who's responsibility is it to be sure the dependencies in your project stay secure?

caption: group of people sitting around a table playing the "nose goes" (not it) game.

This was me until the government shutdown last year. With 2 front-end devs on our team, I worked more on the design UI side while the over dev worked more on the "back of the front" stack like JavaScript, components and the like. But during the shutdown, both of the other 2 developers left our team. So I went from being on an engineering team to BEING the engineering team. I had to figure out a lot of things really quickly and security has been one of those topics. And that's really what I'm going to talk to you about today, all these security things we should be doing.



@sarassassin

#AbstractionsCon

"As a developer consuming open source dependencies, it's my responsibility to make sure my dependencies are secure."

If you're using open source (and you most certainly are), you need to take on the responsibility of keeping it secure. You must set up the tools and processes that will help you stay safe, and raise awareness of this risk throughout your organization.



SARASSASSIN

#AbstractionsCon

The 2017 Equifax breach serves as a very painful lesson of what could happen if you don't adopt a security mindset and take action to keep your application dependencies secure.

Equifax had a massive data breach that exposed the personal information of millions of people. It was completely preventable. This breach was caused by an open source package dependency which had an identified vulnerability that went unpatched for 2 months.

<https://www.synopsys.com/blogs/software-security/equifax-apache-struts-vulnerability-cve-2017-5638/>



@sarassassin

#AbstractionsCon

So I want to go over specific things you could be doing for the applications that you develop which use open source dependencies.



@sarassassin

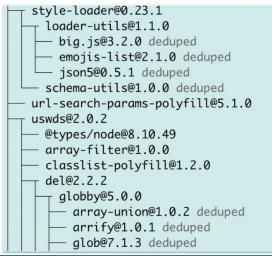
#AbstractionsCon

Figure out the dependencies you have and what packages those depend on.

Not everyone comes in to a project at the beginning and knows exactly what's being used. So take the time to go through the dependencies list to see what's there.

Look at your dependency graph

`npm ls`



@sarassassin

#AbstractionsCon

Just stop and look at your dependencies. Familiarize and refresh yourself with what's there. That way when you hear about a new vulnerability you might have a better reaction time when you know that your app is using certain packages.

If you're an npm user, you can execute `npm ls` to display a dependency tree.

`npm outdated`

Package	Current	Wanted	Latest
karma-phantomjs-launcher	MISSING	1.0.4	1.0.4
@angular/animations	4.4.7	4.4.7	8.2.3
@angular/common	4.4.7	4.4.7	8.2.3
@angular/compiler	4.4.7	4.4.7	8.2.3
@angular/core	4.4.7	4.4.7	8.2.3
@angular/forms	4.4.7	4.4.7	8.2.3
@angular/http	4.4.7	4.4.7	7.2.15

@sarassassin

#AbstractionsCon

Use `npm outdated` to check the registry to see if any installed packages are currently outdated

The more outdated a package is, the more likely it is to itself have vulnerable dependencies.

So npm outdated can tell you exactly how far behind you are in dependency versions.

It's better to patch these up on your own terms instead of waiting for a vulnerability to be disclosed that you're forced to immediately act on.

dependencies
Edit your life frequently and ruthlessly. It's your masterpiece after all.
NATHAN W. MORRIS

Scan for unused dependencies

<https://www.npmjs.com/package/depcheck>

<https://www.npmjs.com/package/npm-check>

@sarassassin

#AbstractionsCon

There are tools available to help with finding unused dependencies, here are a few.

<https://www.npmjs.com/package/depcheck>

<https://www.npmjs.com/package/npm-check>

Set up monitoring mechanisms to stay aware of new vulnerabilities as they are discovered so you can keep track of security announcements affecting those dependencies and any versions released.

Pay attention to what's happening



2. Scan regularly

Next you want to regularly scan for vulnerabilities so you know what's going on with your dependencies and when to patch.

Add security audits to your code review

A security audit could exist as part of a code review where peers ensure that secure code best practices are followed, or by running different variations of security audits such as static or dynamic application security testing. Whether manual or automatic audits, they are all a vital part of detecting and reducing vulnerabilities in your application, and should be executed as regularly and early in the development phase as possible in order to reduce risks of exposure and data breaches at a later stage

Command Line Interface
Source Code Manager
Browser-based
Continuous Integration

There are several approaches you can take to scan for vulnerabilities and I would recommend a combination of all of these.

```
npm audit
```

High	Arbitrary File Overwrite
Package	tar
Patched in	<code>>=2.2.2 <3.0.0 >=4.4.2</code>
Dependency of	node-sass
Path	node-sass > node-gyp > tar
More info	https://npmjs.com/advisories/803

Found 185 vulnerabilities (21 low, 37 moderate, 127 high) in 11618 scanned packages
Run `npm audit fix` to fix 91 of them.
60 vulnerabilities require semver-major dependency updates.
34 vulnerabilities require manual review. See the full report for details.

CLI:

The output varies by tool, but most will list either the known vulnerabilities or vulnerable paths, and provide information for each of them. This typically includes a title, description, severity score, and more.

It even provides `npm audit fix` to actually go ahead and upgrade those dependencies to patched releases where possible.

```
snyk CLI
```

X High severity vulnerability found in lodash
Description: Prototype Pollution
Info: <https://snyk.io/vuln/SNYK-JS-Lodash-73638>
Introduced through: favicons-webpack-plugin@0.0.9 > favicons@4.8.6 > cheerio@0.19.0 > lodash@3.10.0
Fixed in: 4.17.11

Organization: snyrscope
Package manager: npm
Target file: package-lock.json
Open source: no
Project path: /Users/snyrscope/Documents/Sites/code-gov-repos/code-gov-front-end
Local Snyk policy: found

Tested 1037 dependencies for known vulnerabilities, found 9 vulnerabilities, 167 vulnerable paths.
Run `snyk wizard` to address these issues.

We use another tool called Snyk at the command line which can act pretty similar to `npm audit` and also provides an upgrade wizard.

Use a source code manager tool (github, bitbucket, gitlab) to test for vulnerabilities

Using an SCM integration offers a few advantages over the CLI:

- Easier to test or browse multiple issues at once
- The web interface is naturally richer than the terminal, often allowing better usability
- It simplifies continuous testing and fixing, as we'll see later on

Search or jump to... Pull requests Issues Marketplace Explore

Code Issues Pull requests Projects Wiki Insights Settings

Alerts Advisors Policy

You are in the (temp) of automated security fixes. If you have any questions, give us feedback.

Learn more about automated security fixes

Security Alerts Off: Automated security fixes ▾ Dismiss all ▾

3 Open 0 Closed Sort ▾

lodash Jul 10, 2019 by GitHub package-lock.json critical severity

lodash.template Jul 10, 2019 by GitHub package-lock.json critical severity

Github actually has a security tab on their interface now where you can see details of any detected vulnerabilities.

lodash
[Open](#) GitHub issued this alert on Jul 10
 Remediation
 Upgrade lodash to version 4.17.13 or later. For example:

```
dependencies {
  ...
  "lodash": "4.17.13"
  ...
}
```

 Always verify the validity and compatibility of suggestions with your codebase.
Details
 CVE-2019-16744
 Vulnerable versions: 4.17.13
 Affected versions of lodash are vulnerable to Prototype Pollution. This occurs when an object's prototype is altered by the developer before it is passed into adding or modifying properties of Object.prototype using a constructor pattern.

Bump nokogiri from 1.10.3 to 1.10.4 #19
 Merged jeremyzarler merged 1 commit into master from dependabot/bundler/nokogiri-1.10.4 10 hours ago
 Conversation 0 Commits 1 Checks 0 Files changed 1
 dependabot bot commented on behalf of github 18 hours ago
 Bumps nokogiri from 1.10.3 to 1.10.4.
 ▶ Release notes
 ▶ Changelog
 ▶ Commits
 Compatibility 96%
 Dependabot will resolve any conflicts with this PR as long as you don't alter it yourself. You can also trigger a rebase manually by commenting @dependabot: rebase.

Vulnerability detection in the browser

Use webhint to improve your website
 webhint is a linting tool that will help you with your site's accessibility, speed, security and more, by checking your code for best practices and common errors. Use the online scanner or the CLI to start checking your site for errors.

https://webhint.io/scanner

Lighthouse
 Contents Get started Run Lighthouse in Chrome DevTools Load and run the Node command line tool Run Lighthouse as a Chrome Extension
 Lighthouse is an open source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, and more.
 You can run Lighthouse in Chrome DevTools, from the command line, or as a Node module. You give Lighthouse a URL to audit, it runs a series of audits on the page, and then it generates a report file with the page's URL. From there, use the failing audits as indicators on how to improve the page. Each audit has a reference doc explaining why the audit is important, as well as how to fix it.

3. Educate yourself

They've also added automated security fixes which you can turn on to have their dependabot submit PR's to update packages that have identified vulnerabilities.

More recently, though, Google and Microsoft integrated detection of vulnerable JS libraries into their web development tools, Lighthouse and Webhint. Lighthouse is also embedded in Chrome's and Opera's browser dev tools, further simplifying and raising the visibility of this test. I encourage you to try these tools out.

As you're seeing vulnerabilities in packages you'll hopefully want to educate yourself a bit on what exactly the issue is.

A good first place to start is to read the CVE for the vulnerability.

DetailsCVE-2019-10744 [🔗](#)

Vulnerable versions: 4.17.13

Patched version: 4.17.13

critical severity

Affected versions of lodash are vulnerable to Prototype Pollution.

The function `defaultsDeep` could be tricked into adding or modifying properties of `Object.prototype` using a constructor payload.

Common Vulnerabilities and Exposures

CVE-2019-15316cve.mitre.org

A CVE is created with a unique number when a specific vulnerability is found in a package.

cve.mitre.org

MITRE is a government-funded organization that puts out standards to be used by the information security community.



To follow new CVE announcements, you can checkout @CVEnew on Twitter.

CVE-2016-0020 - In Microsoft Internet Explorer 11 on Windows through 2016-09-20 has weak folder permissions, leading to privilege escalation. Its NT AUTHORITY\SYSTEM account has write access to the %TEMP% folder. SetGroupSecurity to leverage a TOCTOU race condition... cwe.mitre.org

Common Weakness Enumeration

cwe.mitre.org

@sarassassin #AbstractionsCon

CWE CATEGORY: Web Problems

Name	Type ID	Name
HasMember	79	Insurer Neutralization of Data Define Web Page Generation (Cross-site Scripting)
HasMember	352	Cross-Site Request Forgery (CSRF)
HasMember	444	Incorrect Interpretation of HTTP Requests (HTTP Request Smuggling)
HasMember	450	Incorrect Interpretation of XML Requests (XML Request Smuggling)
HasMember	614	Sensitive Cookies in HTTPS Session Without Secure Attribute
HasMember	646	Reliance on File Name or Extension of External Suggested File Type
HasMember	776	Insurer Restriction of Resource Entry References in DTD (DTD Entity Expansion)
HasMember	779	Insurer Restriction of Resource Reference in DTD (DTD Entity Expansion)
HasMember	827	Insurer Control of Document Type Definition
HasMember	1004	Sensitive Cookie Without Transport Layer Protection
HasMember	1021	Insurer Restriction of Restricted URL Leaves or Frames
HasMember	1022	Use of Non-Link-to-External-Script with Unknown-User Access

@sarassassin #AbstractionsCon

CVE - Common Vulnerabilities and Exposures
specific issues with specific products

CWE - Common Weakness Enumeration
general security problems that could affect any product

<https://danielmiessler.com/blog/mitre-quick-reference/>

@sarassassin #AbstractionsCon



4. Create a security policy

@sarassassin #AbstractionsCon

Vulnerability disclosure policy

This policy describes what systems and types of research are covered under this policy, how to send us vulnerability reports, and how long we let security researchers to wait before publicly disclosing vulnerabilities.

Guidelines

We require that you:

- Make every effort to avoid privacy violations, degradation of user experience, disruption to production systems, and destruction or manipulation of data.

@sarassassin #AbstractionsCon

Mitre also keeps a list of common issues they see, or CWE's. So you can visit this database to get a better idea of specific types of security vulnerabilities.

cwe.mitre.org

There's a whole informational database about different types of vulnerabilities which is interesting and fun to dive-in to.

Just to recap these:

CWE stands for Common Weakness Enumeration, and has to do with the vulnerability—not the instance within a product or system.

CVE stands for Common Vulnerabilities and Exposures, and has to do with the specific instance within a product or system—not the underlying flaw.

Here's a great blog post that recaps these and more:
<https://danielmiessler.com/blog/mitre-quick-reference/>

Establish a policy and process to quickly roll out a security fix release of your software product once supporting frameworks or libraries needs to be updated for security reasons. Best is to think in terms of hours or a few days, not weeks or months. Most breaches we become aware of are caused by failure to update software components that are known to be vulnerable for months or even years.

On code.gov, we've added a SECURITY.md file which give info on several areas. Here are some ideas for your security policy:

How quickly should we be releasing security fixes

How can someone let us know if we have a security vulnerability

Who to contact for more information

Continuously tracking your application's dependencies for vulnerabilities and efficiently addressing them is no simple feat. When you can, it's great to automate a lot of these things to remove some of the security burden.

5. Automate as



much as possible



@sarassassin

#AbstractionsCon

Scan on commit

<https://18f.gsa.gov/2017/09/26/automated-scanning-for-sensitive-information/>

@sarassassin

#AbstractionsCon

Often when developing open source software, and especially software that relies on outside services, you'll find that you have to manage sensitive information. While there are a large number of things that can be considered sensitive, open source developers often deal with sensitive items such as API tokens, passwords, and private keys that are required for the system to function. One thing you can do is to scan your code on commit to check for these types of items.

Here's a blog post which details the tool we use and includes a quick setup script: <https://18f.gsa.gov/2017/09/26/automated-scanning-for-sensitive-information/>

Test locally

You can add these CLI scanning applications to your tests so you can run them locally as part of your tests along with linting, accessibility testing, unit testing and anything else you might want to verify before making a pull request.

Add PR checks



	All checks have passed	3 successful checks
Hide all checks		
	circleci: build	Your tests passed on CircleCI!
	codeclimate	All good!
	security/snyk - package.json (Code.gov)	No manifest changes detected

@sarassassin

#AbstractionsCon

We've also added checks to our open source project using web hooks so that all pull requests are scanned for new vulnerabilities.

Add continuous monitoring

To continuously avoid known vulnerabilities in your dependencies, integrate a scanning tool into your build system.

@sarassassin

#AbstractionsCon

@sarassassin

#AbstractionsCon

@sarassassin

#AbstractionsCon

Snyk is one of the tools that we use to continuously monitor our projects. It will give you an immediate notification of any security vulnerabilities as well as send you a weekly email summary of new vulnerabilities.

@sarassassin

#AbstractionsCon

It actually gives you a dashboard to you can see all your projects in one place which is really helpful.

Searched repositories

Search View last import log Add project

GitHub | @code.gov/style

GSA/code-gov-front-end

@sarassassin

#AbstractionsCon

WhiteSource

Shift Left Your Open Source Management

According to studies, the cost of fixing security issues and uncovered bugs grows as the later they are found in processes. This is what shifting left is all about - uncovering as many issues as possible as early as possible in the software development process, to decrease the cost of fixing them and to improve the quality of your code.

Late Detection of Vulnerabilities & Bugs is EXPENSIVE

Stage	Cost
Requirements	\$1
Development	\$10
Testing	\$100
Deployment	\$1000
Post-mortem	\$10000

@sarassassin

#AbstractionsCon

WhiteSource is another example of a monitoring tool you can use and it's actually what GitHub uses to do their security monitoring.

Recap:

- Know what you're using
- Scan regularly
- Educate yourself
- Create a security policy
- Automate as much as possible

@sarassassin

#AbstractionsCon

Just to recap these things:

Remember this is a
community effort

@sarassassin

#AbstractionsCon

And finally please remember that Open source is created by the community, and we as participants in that should work to keep it secure.

Please try to be a responsible member of both this ecosystem and your organization and accept ownership for controlling the risk from the libraries you use.

I hope you all stay secure!

Thank you!

<http://bit.ly/abstractions-saracope>

@sarassassin

@sarassassin

#AbstractionsCon