





Welcome to

11. Defensive: Dependencies

Security in Web Development Elective, KEA

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse  

Slides are available as PDF, kramse@Github

11-defensive-dependencies-security-in-web.tex in the repo security-courses

Goals for today



Today's goals:

- Finding dependencies and issues related to these
- Exam project talks

Photo by Thomas Galler on Unsplash

Plan for today



Subjects, defending against:

- Dependencies
- Scanning for dependencies
- Virtual environments
- Reporting security issues

Exercises

- Creating a security.txt file
- Check NPM dependencies in JuiceShop
- Checking KEA Web Apps

Time schedule



- 1) Chapter 27. Securing Third-Party Dependencies 45min
- 2) Working with dependencies in JuiceShop 45min
- 3) KEA Web Apps 45min
- 4) Talk about exam project 45min

As always we will not follow this to the minute, but be flexible

Reading Summary



Web Application Security, Andrew Hoffman, 2020, ISBN: 9781492053118

Part III. Defense, chapter 27

27. Securing Third-Party Dependencies

How did you like the book?

What was the good and the bad? Offensive first and then defensive, did it work for you?

Did you expect more coding in this class?

A lot of security is about processes, operations and running things *securely*

27. Securing Third-Party Dependencies



Because Part III is all about defensive techniques to stifle hackers, this chapter is all about protecting your application from vulnerabilities that could arise when integrating with third-party dependencies.

Source: *Web Application Security*, Andrew Hoffman, 2020, ISBN: 9781492053118

Evaluating Dependency Trees



Third-party dependencies, their dependencies, and the dependencies of those dependencies (etc., etc.) make up what is known as a dependency tree (see Figure 27-1). Using the `npm ls` command in an npm-powered project, you can list an entire dependency tree out for evaluation. This command is powerful for seeing how many dependencies your application actually has, because you may not consider the subdependencies on a regular basis.

Source: *Web Application Security*, Andrew Hoffman, 2020, ISBN: 9781492053118

- Book mentions manual code reviews ... not scalable!
- Running code audit programs is better
- Should we try with a project the size of JuiceShop? `npm ls | wc -l` gave 144 output lines!
- Many python programs have `requirements.txt`, see <https://pip.pypa.io/en/stable/reference/requirements-file-format/>

Dependency problems



In an ideal world, each component in an application that relied on JQuery to function (like the preceding example) would rely on the same version of JQuery. But in the real world, that is rarely the case.

...

A real-world dependency tree often looks like the following:

Primary Application v1.6 → JQuery 3.4.0

Primary Application v1.6 → SPA Framework v1.3.2 → JQuery v2.2.1

Primary Application v1.6 → UI Component Library v4.5.0 → JQuery v2.2.1

- Again multiple programming languages have methods to isolate applications, and parts of applications
- Python has multiple, one example virtual environment <https://docs.python.org/3/tutorial/venv.html>
- Ruby has RVM <https://rvm.io/>
- Docker containers and microservices also benefit from isolation



RVM lets you deploy **each project with its own completely self-contained and dedicated environment**, from the specific version of ruby, all the way down to the precise set of required gems to run your application. Having a precise set of gems also avoids the issue of version conflicts between projects, which can cause difficult-to-trace errors and hours of hair loss. With RVM, NO OTHER GEMS than those required are installed. This makes working with multiple complex applications, where each has a long list of gem dependencies, much more efficient. **RVM lets you easily test gem upgrades**, by switching to a new clean set of gems to test with, while leaving your original set intact. It is flexible enough to even let you maintain a set of gems per environment, or per development branch, or even per individual developer's taste!

Source: <https://rvm.io/>

- I use a Ruby based content management system, so have used RVM for years, Jekyll <https://jekyllrb.com/>

Automated Evaluation of Dependencies



The easiest way to begin finding vulnerabilities in a dependency tree is to compare your application's dependency tree against a well-known CVE database. These databases host lists and reproductions of vulnerabilities found in well-known OSS packages and third-party packages that are often integrated in first-party applications. You can download a third-party scanner (like Snyk), or write a bit of script to convert your dependency tree into a list and then compare it against a remote CVE database. In the npm world, you can begin this process with a command like: `npm list -- depth=[depth]` .

You can compare your findings against a number of databases, but for longevity's sake you may want to start with the NIST as it is funded by the US government and likely to stick around for a long time.

- NOT an easy task, there are multiple complex applications doing this
- The prices for those tools should indicate how complex this is

Scan for vulnerabilities in Docker



Looking to speed up your development cycles? Quickly detect and learn how to remediate CVEs in your images by running `docker scan IMAGE_NAME`. Check out [How to scan images](#) for details.

Vulnerability scanning for Docker local images allows developers and development teams to review the security state of the container images and take actions to fix issues identified during the scan, resulting in more secure deployments. **Docker Scan runs on Snyk engine**, providing users with visibility into the security posture of their local Dockerfiles and local images.

Source: <https://docs.docker.com/engine/scan/>

- Multiple other tools exist, and I would like to NOT name them
- Searching for "scan docker for vulnerabilities" will show lots of links, YMMV
- We already discussed some of the things Github scans for in repositories hosted there



aka Use Security Principles

- Separation of Concerns

One way to mitigate this risk is to run the third-party integration on its own server (ideally maintained by your organization).

- Secure Package Management

One way of mitigating risk from third-party packages installed this way is to individually audit specific versions of the dependency, then “lock” the semantic version to the audited version number.

I recommend using everything you know, be skeptical, be vigilant, only include what is really needed

I often recommend making mirrors of all dependencies, so I can reinstall from scratch without using an internet connection! If a package is removed in the official repositories, it can be real bad



How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript

Code pulled from NPM – which everyone was using

UPDATED Programmers were left staring at broken builds and failed installations on Tuesday after someone toppled the Jenga tower of JavaScript.

A couple of hours ago, **Azer Koçulu unpublished more than 250 of his modules from NPM**, which is a popular package manager used by JavaScript projects to install dependencies.

Koçulu yanked his source code because, we're told, one of the modules was called Kik and that apparently attracted the attention of lawyers representing the instant-messaging app of the same name.

...

Unfortunately, one of those dependencies was left-pad. The code is below. It pads out the lefthand-side of strings with zeroes or spaces. And **thousands of projects including Node and Babel relied on it.**

Source: https://www.theregister.com/2016/03/23/npm_left_pad_chaos/

Using modules, sure – but sometimes don't



```
module.exports = leftpad;

function leftpad (str, len, ch) {
  str = String(str);

  var i = -1;

  if (!ch && ch !== 0) ch = ' ';

  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }

  return str;
}
```

Source: left-pad module from NPM

If you depend on a simple module from a random stranger, and it gets removed or taken over ...

Reporting Security Issues – security.txt



Summary “When security risks in web services are discovered by independent security researchers who understand the severity of the risk, they often lack the channels to disclose them properly. As a result, security issues may be left unreported. security.txt defines a standard to help organizations define the process for security researchers to disclose security vulnerabilities securely.”

security.txt files have been implemented by Google, Facebook, GitHub, the UK government, and many other organisations. In addition, the UK's Ministry of Justice, the Cybersecurity and Infrastructure Security Agency (US), the French government, the Italian government, and the Australian Cyber Security Centre endorse the use of security.txt files.

Source: <https://securitytxt.org/>

- In the old days we always had abuse@domain.tld webmaster@domain.tld postmaster@domain.tld - still highly recommended!

E-mail best current practice



MAILBOX	AREA	USAGE
-----	-----	-----
ABUSE	Customer Relations	Inappropriate public behaviour
NOC	Network Operations	Network infrastructure
SECURITY	Network Security	Security bulletins or queries
...		
MAILBOX	SERVICE	SPECIFICATIONS
-----	-----	-----
POSTMASTER	SMTP	[RFC821] , [RFC822]
HOSTMASTER	DNS	[RFC1033-RFC1035]
USENET	NNTP	[RFC977]
NEWS	NNTP	Synonym for USENET
WEBMASTER	HTTP	[RFC 2068]
WWW	HTTP	Synonym for WEBMASTER
UUCP	UUCP	[RFC976]
FTP	FTP	[RFC959]

Source: *RFC-2142 Mailbox Names for Common Services, Roles and Functions* D. Crocker. May 1997

Exercise



Now lets do the exercise

Security.txt 15 min

which is number **50** in the exercise PDF.

Exercise



Now lets do the exercise

Check NPM dependencies in JuiceShop 20 min

which is number **51** in the exercise PDF.

Summary Dependencies



Today's web applications often have thousands, if not more, of individual dependencies required for application functionality to operate as normal. Ensuring the security of each script in each dependency is a massive undertaking. As such it should be assumed that any third-party integration comes with at least some amount of expected risk (in exchange for reduced development time).

However, while this risk cannot be eliminated, it can be mitigated in a number of ways.

...

To conclude, third-party dependencies always present risk, but careful integration with some thought behind it can mitigate a lot of the upfront risk your application would otherwise be exposed to.

Source: *Web Application Security*, Andrew Hoffman, 2020, ISBN: 9781492053118

Part III Summary and Conclusion



You have completed Web Application Security. Ideally you have learned a lot about securing and exploiting web applications that you can take elsewhere and put to good use. There is still much more to learn. To become a web application security expert, you will need to be exposed to many more topics, technologies, and scenarios. This book isn't a comprehensive glossary of web application security lessons; instead, the topics were specifically chosen based on a few criteria.

You also should have some background on the history of software security and the evolution of hacking. This has been foundational in the lead-up to web application recon, offensive techniques, and defensive mitigations.

- Congratulations!

KEA Wants to know



- Which KEA application is the worst, security wise?

Exercise



Now lets do the exercise

Quick Survery of KEA Web Apps 45min

which is number **52** in the exercise PDF.

Workshop next time!



We will do another kind of teaching/working.

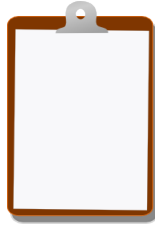
Meet in the same room, but move tables – see if we can get a large common table

I will be available in class.

- You will be working on the exercises, and
- We can discuss which you may skip
- I will let you know which ones I think are essential
- We will discuss subjects in Web Application Security
- You will have a chance to give constructive criticism on the book, slides, exercises, teaching, ...

There will be cookies! (Store bought christmas cookies like brunkager, pebernødder etc.)

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools