Week 5, Lecture 2

COMP6[48]43

Hamish Cox

Mar 19th, 2025

Lecture 1 Recap

- Midterm.
- Midterm walkthroughs (omitted from recording, go to your tutes).
- LFD/LFI.

WTF are we doing today?

- Very high level overview of a bunch of vulnerabilities that follow similar themes.
- As you've seen in previous lectures, you shouldn't trust user-controlled input!
- Keep data (user input) and control (your code/query/commands/etc.) separate!

LFD/Path Traversal and LFI Recap

- Local File Disclosure: being able to read an arbitrary file, typically by exploiting...
- Path Traversal: injecting content into a path (e.g. `"pages/" + request.args.get("page")`).
  - Can also allow you to perhaps write to an arbitrary file if the injection point is for a write call.
- Local File Inclusion: being able to run a file on the filesystem (or even better, your own file/content) as code. Usually PHP: `include("pages/" + $_GET["page"])` (this is where the name “inclusion” comes from).
  - If you can execute PHP code for example, you could write some PHP code into logs or an uploaded file and then ‘include’ that file so that it is executed.

Aside: Useful files to read

- `/etc/passwd` or `/etc/hosts` - usually nothing too interesting, but you are more or less guaranteed to be able to read it, so a good test for if you have LFD/path traversal.
- `/proc/self` - this directory (if readable) contains a bunch of useful information about the current process.
  - The 'proc' filesystem is generated on the fly by Linux.
  - `/proc/self/environ` has environment variables for the current process.
  - `/proc/self/cmdline` has the command that was executed to run the current process.

Aside: Useful files to read

- Source code, webserver configuration, etc, but you likely have to guess filenames (e.g. `/etc/apache2/apache2.conf`) or find a way to list them.
- In CTFs: `/flag`, `/password`, `/flag.txt`, `/password.txt`
- Keep in mind that any and all of this (or `../`) can be behind a WAF.

Server Side Template Injection (SSTI)

We've seen Jinja2 a couple times in lectures before, let's play around with it a bit.

Other Libraries

Other libraries don’t even bother providing a DSL and let you run whatever you want (without arcane language-specific wizardry) in regular code, e.g. ERB and Slim from Ruby, PHP itself, ASP Razor.

Others will be ‘sandboxed’ and be limited - they aren’t always secure, e.g. Handlebars, Nunjucks, Lodash (all JS).

Server-Side Request Forgery (SSRF)

Fancy way to say request injection.

This is effectively what HaaS is, but we didn’t let you change the destination, only the content.

Server-Side Request Forgery (SSRF)

Fancy way to say request injection.

This is effectively what HaaS is, but we didn’t let you change the destination, only the content.

Server-Side Request Forgery (SSRF)

Fancy way to say request injection.

```
requests.post(
  "https://important-api.com" + request.args.get("path"),
  headers={"Authorization": "..."}
)
```

Is this a good idea?

This is effectively what HaaS is, but we didn't let you change the destination, only the content.

Server-Side Request Forgery (SSRF)

Fancy way to say request injection.

```
requests.post(
  "https://important-api.com/" + request.args.get("path"),
  headers={"Authorization": "..."}
)
```

What about this?

This is effectively what HaaS is, but we didn't let you change the destination, only the content.