Handin Gearup
Goals

Learn about OS privileges and security by breaking an autograder on a multi-user Linux system (think: dept. machines)

• Learn how it works, and how to break it!
• After that, write vulnerability reports about each vulnerability

Throwback to the pre-Gradescope era…
Like Flag, but with a twist

You have all the source code! => Different form of thinking/analysis vs. Flag!

https://github.com/brown-csci1660/handin-source
The assignment

CS1660: find and write up four (4) distinct vulnerabilities
CS1620/CS2660: find and write up five (5) distinct vulnerabilities

<table>
<thead>
<tr>
<th>Vulnerability Category</th>
<th>Category ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exfiltrated Process Information</td>
<td>exfil-pi</td>
</tr>
<tr>
<td>Path Sanitation Bypass</td>
<td>path-byp</td>
</tr>
<tr>
<td>Symlink Traversal</td>
<td>symlinkt</td>
</tr>
<tr>
<td>Unsanitized Environment Variables</td>
<td>env-vars</td>
</tr>
<tr>
<td>Misconfigured Blocklists / Safelists</td>
<td>listconf</td>
</tr>
<tr>
<td>TOCTOU (Race Condition)</td>
<td>racecond</td>
</tr>
<tr>
<td>Misconfigured File / Directory Permissions</td>
<td>permconf</td>
</tr>
<tr>
<td>Escaping chroot or Sandbox</td>
<td>breakout</td>
</tr>
</tbody>
</table>
The mechanics

Like Flag, this project has another container

- Instead of starting a site, gives you a shell as a student in cs666

- You get to “complete” an assignment that you submit for autograding
  => But you shouldn’t actually do the assignment…
Counting vulnerabilities

...now depends on two parameters:

- Which file contains the issue (based on the source code)
- The vulnerability category (Appendix B in the handout)

\((\text{file}, \text{category})\)
Counting vulnerabilities

...now depends on two parameters:

- Which file contains the issue (based on the source code)
- The vulnerability category (Appendix B in the handout)

You cannot count multiple vulnerabilities with the same (file, category) tuple:

- (autograder.sh, racecond)  
- (autograder.sh, permconf)  
- (autograder.sh, racecond)
- (handin.go, racecond)

Not OK
The mechanics

Same general model as Flag:

- Info about vulns => Wiki: https://brown-csci1660.github.io/handin-wiki

- How to start /set up/reset the container => Setup Guide: https://hackmd.io/@cs1660/handin-setup-guide

Also in the setup guide
- How to run the autograder
- Super helpful hints and tips for specific exploits (skim now, come back for them later)
How you’ll work on the project

• Handin container: gives you a shell on a Blue University system as user alice

• You can access the CS666 filesystem, which contains the autograder and other useful stuff
To start: poke around the filesystem…
=> What can you find? What might all these files do?
The autograder: cs666_handin

... must be run by students, but act with TA privileges (to use solutions, record grades)

How? setgid!

```
alice@e1100aeda57e:~$ ls -la /course/cs666/bin
total 6748
drwxrwsr-x 2 ta cs-666ta 4096 Mar 8 23:13 .
drwxrwsr-x 1 ta cs-666ta 4096 Mar 8 23:13 ..
-rwxrwsr-x 1 ta cs-666ta 2629538 Mar 8 23:13 cs666_handin
-rwxrwsr-x 1 ta cs-666ta 2411624 Mar 8 23:13 report
```
The autograder: cs666_handin

... must be run by students, but act with TA privileges (to use solutions, record grades)
How? setgid!

```
alice@e1100aede57e:~$ ls -la /course/cs666/bin
总和 6748
drwxrwsr-x 2 ta cs-666ta 4096 Mar 8 23:13 .
drwxrwsr-x 1 ta cs-666ta 4096 Mar 8 23:13 ..
-rwxrwsr-x 1 ta cs-666ta 2629538 Mar 8 23:13 cs666_handin
... .
-rwxrwsr-x 1 ta cs-666ta 2411624 Mar 8 23:13 report
```

No matter who runs cs666_handin, process with permissions of cs666-ta group
Idea: make autograder do something unintended => do actions as TA!
The assignment: Ivy

• You can submit one assignment: Ivy

• You have a stencil: on handin, gets combined with template code, compiled, and executed
  => Some restrictions on what your code can do...

Autograder general steps:
1. Creates tar archive of your submission
2. Unpacks archive and replaces template code with TA version (eg. main.go)
3. Compiles and executes your code!
The assignment: Ivy

• You can submit one assignment: Ivy

• You have a stencil: on hand in, gets combined with template code, compiled, and executed

  => Some restrictions on what your code can do...

  => DO NOT complete Ivy again. Instead try to break the autograder:
  How could you get full points without doing the assignment?
  Could you do anything worse? (Hint: yes.)
Demo: autograder
Some new languages

Autograder code is in Bash (shell scripting) and Go (a memory-safe systems language)

Haven’t seen Go before? Uncomfortable with bash? Don’t worry!

• Like PHP, we don’t expect you to know it ahead of time
• See setup guide, course website for some resources, lots more online

=> Like with Flag, feel free to use online resources to help you!
How to get started

Don’t panic. You can do this.

1. Poke around the filesystem, try to submit to the autograder

2. Try and follow some details of what the scripts are doing
   ⇒ Which scripts are invoked when? What is their purpose?
How to get started

Don’t panic. You can do this.

1. Poke around the filesystem, try to submit to the autograder

2. Try and follow some details of what the scripts are doing
   ⇒ Which scripts are invoked when? What is their purpose?

3. Based on class and wiki, what parts of the framework might you be able to control?
Categorizing Impact

When you find a vulnerability, what can you do with it?
Identify the **severity** in your vulnerability report:

<table>
<thead>
<tr>
<th>Severity Classification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary Code Execution</td>
<td>Execute arbitrary code as the TA group.</td>
</tr>
<tr>
<td>Data Modification</td>
<td>Change existing data that you should not be allowed to modify.</td>
</tr>
<tr>
<td>Data Exfiltration</td>
<td>Gain access to data that you should not have access to.</td>
</tr>
<tr>
<td>Data Theft</td>
<td>Trick the infrastructure into believing that somebody else’s data is your own (for example, use another student’s handin as your own). Unlike Data Exfiltration, this does not require you to have access to the data yourself.</td>
</tr>
<tr>
<td>Metadata Exfiltration</td>
<td>Get access to metadata that you should not have access to. Metadata includes whether or not other students have handed in, the names (but not contents) of files in restricted parts of the file tree (under /course/cs666), etc.</td>
</tr>
</tbody>
</table>
cs666_whoami: Demonstrating arbitrary code exec

Normal operation:
alice@e1100aede57e:~$ cs666_whoami
uid: 1000
euid: 1000
gid: 1000
egid: 1000

gid: 1000

When running with elevated privileges....
alice@e1100aede57e:~$ <do some exploit>
uid: 1000
euid: 1000
gid: 1000
egid: 1101

Note: your terminal output doesn't need to look pretty, so long as we can see that the output indicates you're running with elevated permissions.
Important container terminology

• Container **image** ("image"): read-only package with files/settings for how the container runs

• Container **instance** ("container"): created when container started, read-write