Section 8:
Vulnerability Lifecycle and Lab 3

CB: Edan, Matt, Phillip and Karman

Administrivia

Upcoming due dates:

- December 8th, 11:59pm: Lab 3 Due
- December 13th, 11:59pm: Final Project Due

Lab 3: Getting started

```
→ ~/Downloads 130 $ rsync -av lab3.tar.gz philipmg@attu.cs.washington.edu:~
philipmg@attu.cs.washington.edu's password:
sending incremental file list
lab3.tar.gz

sent 2,738,716 bytes  received 35 bytes  608,611.33 bytes/sec
total size is 2,737,939  speedup is 1.00
```

```
philipmg@attu2:~$ ls lab3*
lab3.tar.gz
philipmg@attu2:~$ tar -xzvf lab3.tar.gz
lab3_data/
lab3_data/README.md
lab3_data/target/
lab3_data/target/tinyserv/
lab3_data/target/tinyserv/files/
```

```
rsync -av lab3.tar.gz philipmg@attu.cs.washington.edu:~
ssh philipmg@attu.cs.washington.edu
tar -xzvf lab3.tar.gz
```

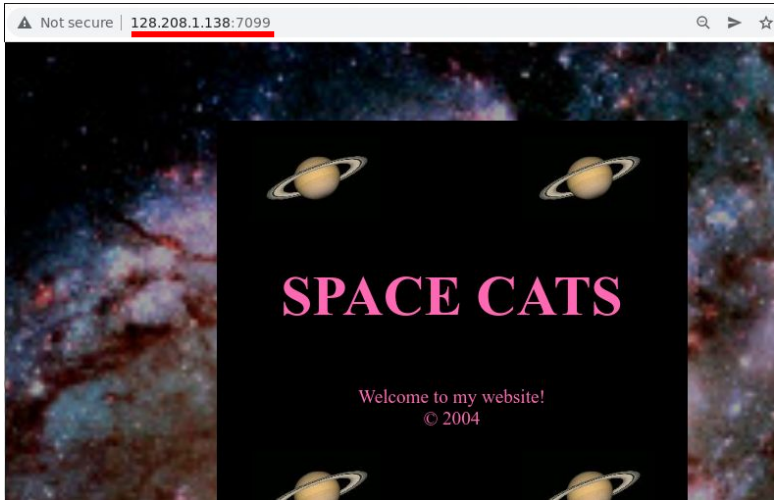
```
philipmg@attu2:~$ tree lab3_data
lab3_data
├── README.md
├── sploit
│   ├── hmac.c
│   ├── hmac.h
│   ├── Makefile
│   ├── nonsploit.c
│   ├── sha1.c
│   ├── sha1.h
│   ├── sploit1.c
│   └── sploit2.c
└── target
    ├── README.md
    └── tinyserv
        ├── 400.html
        ├── 404.html
        ├── files
        │   ├── index.html
        │   ├── pictures
        │   │   ├── athena.png
        │   │   ├── background.jpg
        │   │   ├── boof.png
        │   │   ├── cheetoh.png
        │   │   ├── lil.png
        │   │   ├── rhea.png
        │   │   └── saturn.gif
        │   └── testfile.html
        ├── hmac.c
        ├── hmac.h
        ├── login.html
        ├── main.c
        ├── Makefile
        ├── sha1.c
        ├── sha1.h
        └── strnstr.c

5 directories, 29 files
```

```
philipmg@attu2:~$ cd lab3_data/target/
philipmg@attu2:~/lab3_data/target$ cat README.md
# tinyserv
A simple multi-threaded HTTP server.

# Building
run `make` in the tinyserv directory

This won't work on codered, which is too old!

# Usage
run `./tinyserv 127.0.0.1 $PORT .` after building where $PORT is
your group number plus 7000

You can then talk to it via your favorite web browser, e.g.
http://127.0.0.1:$PORT/files/index.html

or

http://127.0.0.1:$PORT/login.html
philipmg@attu2:~/lab3_data/target$ cd tinyserv/
philipmg@attu2:~/lab3_data/target/tinyserv$ make
gcc main.c -g -o tinyserv -lpthread
```

```
cd lab3_data/target/
cat README.md
cd tinyserv/
make
```

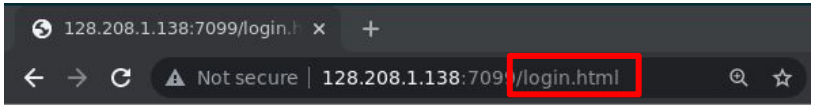
```
philipmg@attu2:~/lab3_data/target/tinyserv$ ifconfig | head -2
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 128.208.1.138  netmask 255.255.255.0  broadcast 128.208.1.255
philipmg@attu2:~/lab3_data/target/tinyserv$ ./tinyserv 128.208.1.138 7099 files/

------------------------------
Admin password follows, keep it secret:
QwQWVmzk4FVyqjYX3RUT
------------------------------
    Listening on port 7099
       EXIT WITH CTRL+C
------------------------------
```

Don’t use 7099: Use a port number that ends with your group number! Otherwise you will conflict with others




```
ifconfig | head -2
./tinyserv 128.208.1.138 7099 files/
```

Lab 3: The admin interface

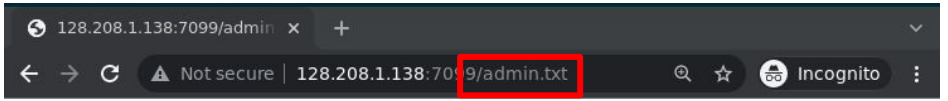


admin

••••••••••••••••••••

Log in

Password was printed to console (see previous slide) QwQWVmz...




```
GET / HTTP/1.1
GET /pictures/saturn.gif HTTP/1.1
GET /pictures/cheetoh.png HTTP/1.1
GET /pictures/lil.png HTTP/1.1
GET /pictures/athena.png HTTP/1.1
GET /pictures/boof.png HTTP/1.1
GET /pictures/background.jpg HTTP/1.1
GET /favicon.ico HTTP/1.1
GET /pictures/saturn.gif HTTP/1.1
GET /pictures/cheetoh.png HTTP/1.1
GET /pictures/athena.png HTTP/1.1
GET /pictures/lil.png HTTP/1.1
GET /pictures/boof.png HTTP/1.1
GET /pictures/background.jpg HTTP/1.1
GET /login.html HTTP/1.1
GET /login.html HTTP/1.1
GET /login.html?user=admin&password=QwQWVmzk4FVyqjYX3RUT
HTTP/1.1
GET /admin.txt HTTP/1.1
```

Lab 3: The sploits

Exploit 1 is incomplete, clearly. The copy you've recovered appears to be missing the payload (no shellcode!) but it certainly looks exploitable. You'll need to at least identify the crash.

Exploit 2 doesn't seem to do anything bad to the server, no crash, no changes to any files. But it looks like someone has found a way to read the admin logs without logging in!

Lab 3: Using the sploits

Terminal 1: run tinyserv

```
philipmg@attu2:~/lab3_data/target/tinyserv$ ifconfig | head -2
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 128.208.1.138  netmask 255.255.255.0  broadcast 128.208.1.255
philipmg@attu2:~/lab3_data/target/tinyserv$ ./tinyserv 128.208.1.138 7099 files/

-----------------------------
Admin password follows, keep it secret:
QwQWVmzk4FVyqjYX3RUT
-----------------------------
    Listening on port 7099
       EXIT WITH CTRL+C
-----------------------------
```

Don't use 7099: Use a port number that ends with your group number! Otherwise you will conflict with others

```
cd lab3_data/sploit/
make
./sploit2 128.208.1.138 7099
```

Terminal 2: run a sploit

```
philipmg@attu2:~$ cd lab3_data/sploit/
philipmg@attu2:~/lab3_data/sploit$ make
gcc -g -o sploit1 sploit1.c
gcc -g -o sploit2 sploit2.c
gcc -g -o nonsploit nonsploit.c
philipmg@attu2:~/lab3_data/sploit$ ./sploit2 128.208.1.138 7099
###Start server reply with secret admin.txt
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 613
Connection: close

GET / HTTP/1.1
GET /pictures/saturn.gif HTTP/1.1
GET /pictures/cheetoh.png HTTP/1.1
GET /pictures/lil.png HTTP/1.1
GET /pictures/athena.png HTTP/1.1
GET /pictures/boof.png HTTP/1.1
GET /pictures/background.jpg HTTP/1.1
GET /favicon.ico HTTP/1.1
GET /pictures/saturn.gif HTTP/1.1
GET /pictures/cheetoh.png HTTP/1.1
GET /pictures/athena.png HTTP/1.1
GET /pictures/lil.png HTTP/1.1
GET /pictures/boof.png HTTP/1.1
GET /pictures/background.jpg HTTP/1.1
GET /login.html HTTP/1.1
GET /login.html HTTP/1.1
GET /login.html?user=admin&password=QwQWVmzk4FVyqjYX3RUT HTTP/1.1
GET /admin.txt HTTP/1.1
GET /admin.txt HTTP/1.1
###End secret admin.txt
```

Lab 3: Tips

- Your job: write two Root Cause Analyses, one for each sploit, using our template
- Check that you understand broadly what tinyserv does. Run it and use it a little bit
- The source code `sploit1.c` and `sploit2.c` are provided. Read them!
- You can run tinyserv with gdb: this is especially handy for debugging a crash. E.g.:
  - `$ gdb tinyserv`
  - `(gdb) run 127.0.0.1 7099 files`
- You'll need to identify (among other things) a *bug class*. Look at OWASP Top 10 for some example bug classes
- tinyserv is very vulnerable: what other vulnerabilities can you find?

Vulnerability Lifecycle

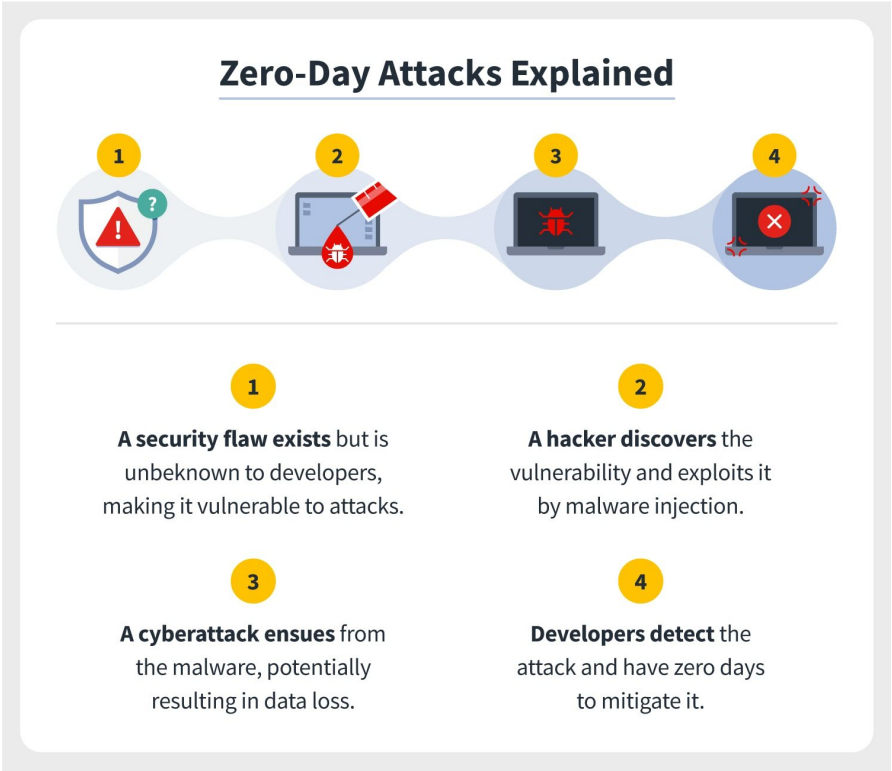
Overview: Options

- Detection in the Wild
  - RCA
- Responsible Disclosure
  - Bug Bounties
- Gray Markets

In the Wild

Zero-Day Exploits

- Zero-Day Attack timeline
- Once a patch is written, the exploit is no longer zero-day
- Also, if a vulnerability is responsibly disclosed and not being exploited in the wild, it is not a zero-day vulnerability



How are zero-day exploits created?

- STEP 1: Attack Surface Analysis
  - An adversary will study part of the system that they have legitimate access to
- STEP 2: Fuzz Testing
  - You test the edge cases of the system and feed unexpected or random values and monitor the behavior of the system
- STEP 3: Development
  - After a vulnerability has been identified, it needs to be implemented into the target system
  - It is also important to hide exploit code in case of discovery, there are two protection techniques:
    - Metamorphic
    - Polymorphic
- STEP 4: Delivery:
  - Deliver the malware to the target system, which can be done through the network automatically or user interaction

How do we detect them, then?

- Statistics-based detection
  - Relies on data about previously detected exploits
- Signature-based detection
  - Relies on existing databases of exploit signatures, but this doesn't work on zero-day attacks
- Behavior-based detection
  - Relies on looking for how the exploit interacts with the target system and focusing on interactions with existing software rather than code itself

RCA (Root Cause Analysis)

- The objective of RCA is to find the root cause of a problem and eliminate it for good
- Analysis steps
  - Define event
  - Find causes
  - Finding the root cause - asking why
  - Find solutions
  - Take action
  - Verify solution effectiveness

<UWE-484-01>: <Description/Title>

Authors:

The Basics

Disclosure Date: 11/24/2021
Product: tinyserv
Reporter(s): CSE 484 Staff

The Code

Exploit sample: see sploit{1,2}.c
Did you have access to the exploit sample when doing the analysis? yes

The Vulnerability

Bug class (1p):
Vulnerability details (3p):
Thoughts on how this vuln might have been found (fuzzing, code auditing, variant analysis, etc.) (1p):

The Exploit

(The terms exploit primitive, exploit strategy, exploit technique, and exploit flow are defined here.)
Exploit primitive (1p):
Exploit strategy (or strategies) (2p):

The Next Steps

Proposed patch plan (2p):

What are potential detection methods for similar 0-day vulnerabilities? (bonus 1p):

Responsible Disclosures

Bug Bounties

- Created by companies
- Bounties typically range from $100-50000 depending on the severity of the bug.
- Used to counteract gray markets

3rd party vs 1st party

hackerone

Bug Bounty Incentives

Not always just money.

Clear submission guidelines

Underpaying or downplaying severity

Safety

Computer Fraud and Abuse Act (Hacking == Illegal)

Ignoring Vulnerabilities

Many companies choose to ignore vulnerabilities or take long processing times to fix a bug.

These types of behaviors incentives researchers to explore alternatives to bug bounty programs.

Disclosure of three 0-day iOS vulnerabilities and critique of Apple Security Bounty program

Common paradigm: Time-limited disclosures

Gray Markets

Exploit Brokers

Various government agencies

zerodium

Brokers buy zero-days...

NORTHROP GRUMMAN

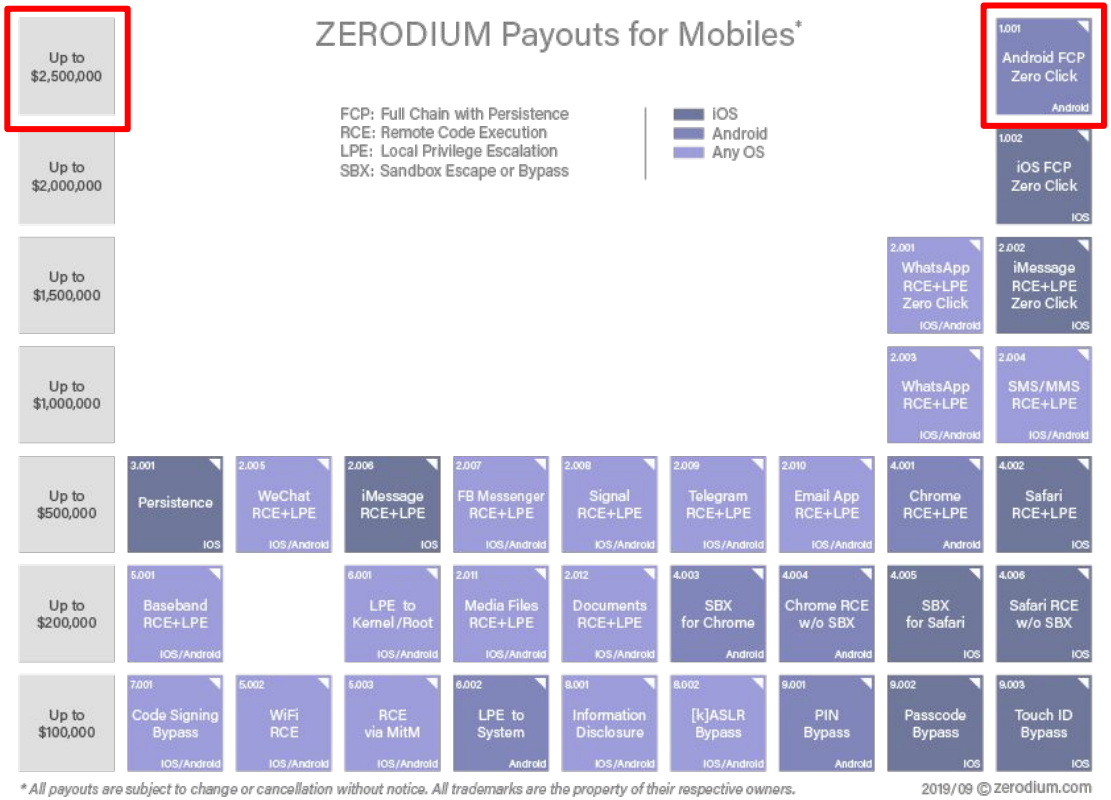
Private defense companies

and sell them to

???

Who knows what else?

Gray market bounties



Google's maximum bug bounty payout for android is just $1 million

Why are these governments and private companies willing to pay so much more for the exploits?

Gray Markets: How is this legal?

- Exploit sales have some first-amendment protection in the USA
- Exploits have some legitimate uses
  - You're selling knowledge of how to infiltrate a computer system
  - but you're not necessarily going to do something illegal with it
- Governments buy the exploits, so largely haven't cracked down

Ethics Question

Imagine that you are on the security team for Tesla and you received 100 bug disclosures among 3 security engineers. How would you handle this situation?