Cryptography and Network Security HW3 Report

B07902116 資工三 陳富春

1 IoT

1. Get the login username and password:
   I unzip the `IoT VM.ova` and get `IoT-disk001.vmdk`. Then I mount it and find `/var/www/html/cns-iot-geeks/info_leaks`, there are some usernames and their SHA-1 hashed passwords. The wanted login username and password is `txsupport` and `support`.
   I try to get the `leak_infos` in another way. When the IoT machine is on, I use `nmap -p 80 192.168.1.0/24` and find that 192.168.1.110 is on. Then I connect to 192.168.1.110:80, and use `cht` and `chtcgpon` to login the website. Then I click the `"CNS IoT Geeks 2021"` in the bottom. After that, I append `info_leaks` in the address bar and get the file.

2. Get the flag:
   There is a file `Total_Rickall_steg.jpg` in the machine. Then I use the website `https://futureboy.us/stegano/decinput.html` to decode the steg in the jpg file.
   Flag: CNS{OW4sp_IOT_7oP_10_i5_imPor7ANT}

Discuss with r09922016 劉厚辰

2 Randomness Casino

a) Because the server gives all other contributions to me first, I can arbitrarily pick my contribution as long as the result generated from these contributions is bigger than 0.5.
Flag: CNS{Th1s_1s_the_f1nal_contr1but1on_attack}

b) This time the server asks me to provide my commitment first, so I create two different contributions that share the same MD5 hash values. Then I can submit the contribution that causes my desired result. Because it is possible that the two contributions cannot cause good result, I play 200 times to have a higher chance of earning enough money.
Flag: CNS{Be_careful_0f_HASH_C0llisi0n}

c) This time the server use a modified MD5. So I create two files (128 bytes) that have `CNS3` in the beginning, and my two possible contributions are the latter 125 bytes. Besides, MD5 is Merkle–Damgård construction, appending `hw3` to two colliding files does not make any difference.
Thus, when the server uses its modified MD5 to check the commitment, the server cannot distinguish my two different contributions because `CNS3 + contribution + hw3` share the same hash value.
Flag: CNS{MD5_is_NOT_colli5ion_re5i5tant}

I use hashclash to create the MD5 collision files `collision1.bin` and `collision2.bin` that have `CNS3` in the beginning for part b and c.
Reference: `https://github.com/cr-marcstevens/hashclash`

3 VDF

a) I keep the values of x^{2^i} when computing $y = x^{2^T}$ to accelerate the computation for the proof π. I finish the computation in about 50 seconds.
Flag: CNS{YA!U_have_made_WeSoLowSki's_proooooooof!}

b) Part 1:
A bad group is chosen, so I compute the group order G and construct a list that keeps all group elements. Then I can reduce the computation of $y = x^{2^T}$ to $y = x^{(2^T \ mod \ G)}$.
To find the proof π, I compute $x^{2^T mod \ L}$ and iterate the group element g to check if it satisfies $y = g^{(L \ mod \ G)} x^{(2^T \ mod \ L)}$. If the g satisfies the condition, then the g is the proof π.

Flag:
CNS{S=uentiality_depENDs_on_group_of_unkNOWn_order}

Part 2:
For the same T and x, I generate $y' = x^{(2^T \ mod \ G)+1}$ and find the corresponding proof $\pi'$to finish the challenge.

Flag:
CNS{SoUNdneSS_also_depeNDSS_on_group_of_UNknown_order}

4 DDoS

1. Botnet
   I use `crontab -l` to find if there is any process periodically running up and find `/bin/watchdog`. Then I stop it to finish this part.

2. Backdoor
   I use command `ps -x` and find that there are two suspicious processes with PID 13 and 30.

```
cns@e1858957d643:~$ ps -x
   PID TTY      STAT   TIME COMMAND
    13 ?        Sl     0:00 node /usr/bin/forever --sourceDir /var/blog index.js
    23 ?        S      0:01 dnsmasq
    30 ?        Sl     0:18 /usr/bin/node /var/blog/index.js
114033 pts/1    Ss     0:00 bash
114100 pts/1    R+     0:00 ps -x
```

   So I kill these two processes and remove the backdoor.

3. DNS Amp

   Task 1:
   The DNS server config file is `/etc/dnsmasq.conf`. To query the server, I add `listen-address=127.0.0.1` and use the command `dig host2.cns.tw @127.0.0.1 -p 5353 any` to send a DNS request.
   Then I receive 8 responses, including A/AAAA, TXT and MX type records, and the message size is 357 bytes. The DNS request size is about 83 bytes, so the amplication rate is about 4.3.

   Task 2:
   I comment all records except A/AAAA type records in `dnsmasq.conf` so that the DNS server only replies A/AAAA responses.

4. Challenge D
   In `Service` function, if the input is not a number, `scanf("%d", &choice)` will cause infinite loop. So I change it to `scanf("%s", buf)` where `buf` is a char array. Then if `strlen(buf) == 1`, `choice = buf[0] - '0'`. This can avoid non-digit input causing infinite loop.

5. Challenge E
   In `Calculate` function, the for loop use `int i` to iterate while the variable `stop` is `short int`. If user lets `stop` be -1, the for loop will iterate too many times and cause long delay.
   I patch the function by changing the `int i` to `short int i` and changing `int now` to `_Bool now`. This can avoid the extra calculation in the for loop.

6. Challenge F
   Task 1:
   reDOS is DoS against regular expression. Attacker create some malicious input that make the server need some time to check if the input match the pattern of regular expression.

   Bonus:

There is reDoS vulnerability in `origin.c`. The `([a-zA-Z0-9\-]+\.)+` and `([a-zA-Z0-9\-]?[a-zA-Z]+)+` parts in the regex pattern will produce many possible paths for input such as `aaa...aaaa!` and the server will spend lots of time on checking it.
To defense, pattern like `(a+)+` that produce many possible paths should be avoided.

5 Smart Contract

a)

1. CallMeFirst
   I call the `hw3.callMeFirst(ID)` to finish the challenge.
2. BribeMe
   I save 1 ether to my account when deploying the contract and then call `hw3.bribeMe().value(1 ether)(ID)` to finish the challenge.
3. GuessRandomNumber
   `https://kovan.etherscan.io/block/24804001`
   Using the parent hash and timestamp from the block record can generate the correct random number.
4. Reentry
   I let my fallback function call `hw3.reentry()` again. Then the value of c3Flag is 2 and the challenge is finished.
5. Bonus
   Because the `flashloaning` is stored in `uint8` type, it will overflow at 256. So I reentry `hw3.flashloan(10000)` until the `flashloaning` overflow and become zero.
   After that, I can call `hw3.bonus_verify(ID)` to finish the challenge and return the flashloan I borrow. And I can get one CNS Token as reward.

b) When a function executes in the halfway and has some intermediate states, it calls other function and the called function enter the caller function with intermediate states. In this situation, a reentry occurs and it is possible to launch reentry attack.
This can prevented by not calling other function when the states are not finalized or using reentrancy guards to avoid reentrant call.