NASA HW12 - 金哲安(B12902118)

1. Linux 大小事

References

- B12902116 (林靖昀)
- B12902066 (宋和峻)
- https://eitca.org/cybersecurity/eitc-is-lsa-linux-system-administration/basic-linux-sysadmin-tasks/user-account-management/examination-review-user-account-management/how-are-passwords-stored-and-managed-in-linux/#:~:text=To summarize%2C passwords in Linux,the integrity of user credentials.
- https://stackoverflow.com/questions/71839786/in-the-case-of-a-normal-user-with-no-access-to-etc-shadow-file-how-can-the-pass
- https://bromiley.medium.com/torvalds-tuesday-sample-post-bde3d5c6d05e
- https://www.digitalocean.com/community/tutorials/how-fail2ban-works-to-protect-services-on-a-linux-server
- https://linux.vbird.org/linux_server/rocky9/0180firewall.php

(a)

The user passwords are stored in the `/etc/shadow` file, which is only readable by the root user. The passwords are first salted and then hashed, to prevent them from being cracked by rainbow tables.

(b)

The `/usr/bin/passwd` file is owned by the root user and has the SETUID bit set. Normal users executing `/usr/bin/passwd` will have their process' effective user ID change to 0 (root) and so can change the contents of `/etc/shadow`.

(c)

The log is in `/var/log/auth.log`. This file stores sudo command usages, root sessions, and ssh activities.

(d)

1. Use Fail2ban which monitors the log file for signs of brute-force attacks. When a pattern is detected, it temporarily bans the offending IP
2. Use iptables to limit ssh attempts to 6 per minute per IP.

2. 畫中有話

References

- B12902116 (林靖昀)
- B12902066 (宋和峻)

(a)

In `hide.py`, the data is turned into a byte string and stored in the pixels of the image. Each character is 1 byte, which is 8 bits. Each pixel has 3 channels (RGB), so 3 pixels have 9 channels. Every 3 pixels are used to store 1 character, with 1 channel storing 1 bit of the character. If a bit is 1, the least significant bit of the channel will be set to 1. Otherwise, it will be set to 0. Only 8 channels out of the 9 will be used for this encoding. It will take a total of $3 \times \text{characters}$ pixels to store the whole data.

(b)

Execute `code/2b.py`

Flag: `HW12{S4KiCh4n_sakiCHAN_S4k1ChaN}`

I first read the image and then read all the pixels. Then for every 3 pixels (9 channels), I take the first 8 channels and extract their least significant bit and put together to form a byte, which represents a decrypted character. I keep decrypting until I reach a null byte and then print out the decrypted string that I got.

3. Alya Judge

References

- B12902116 (林靖昀)
- B12902066 (宋和峻)
- https://zh.wikipedia.org/zh-tw/Cookie
- https://stackoverflow.com/questions/77340063/flask-session-cookie-tampering

(a)

Flag: `HW12{r3MeM8eR_To_s3t_S7R0Ng_PAS5W0rds}`

Use Burp Suite to open up a browser and connect to Alya Judge. Turn on intercept and try to visit http://140.112.91.4:45510/submissions/../accounts/accounts. The packet captured shows that the request has been resolved to GET /accounts/accounts HTTP/1.1, so send the packet to repeater instead.

Change the header to GET /submissions/../accounts/accounts HTTP/1.1 and send it out. From the response, fysty's password can be seen as 40c3d69c8a012e181bd63d215d61a1df44e8fe7c182da6d24f26b0fae5348010, so paste it to https://crackstation.net/ and crack it to get the password: mortis00

Finally, login as fysty and see his submission at the bottom that contains the flag.

(b)

Flag: `HW12{e5x5Vw2qC}`

Submitting `HW12{` gives a score of 33. I manually tried `HW12{?` where `?` is in `[0-9A-Za-z_]` and found out that `HW12{e` increased the score. And so I kept on trying all the letters one by one until I get the entire flag.

(c)

Install flask-unsign

```
pip install flask-unsign
```

On chrome browser, login as fysty and open up developer tools (press F12). Go to Applications > Cookies > http://140.112.91.4:45510. There is a cookie whose name is session. Copy the value and decode it.

```
flask-unsign --decode --cookie "eyJ1c2VybmFtZSI6ImZ5c3R5In0.aDrYbg.0oHZU9Xc7nEkvxsT8I9ESHoIQGM"
```

It gives

```
{'username': 'fysty'}
```

Create a new cookie with the flask secret key.

```
flask-unsign --sign --cookie '{"username":"admin"}' --secret 'A_super_SecUrE_$eCR37_keY'
```

It gives

```
eyJ1c2VybmFtZSI6ImFkbWluIn0.aDrX5w.n6mqq0CiVb7OjiXUXK2LIVLmqqE
```

Go back to the browser and double click on the value of the cookie then replace it. Finally on the webpage, click My Submissions to see admin's submission. The flag is in the submission at the bottom.
Flag: `HW12{i_l1KE_a15CR3am_MoRE_7H4n_Co0Ki3s}`

4. Introduction to gnireenignE esreveR

(a)

Execute `code/4a.c`
Flag: `HW12{hW0_8UT_WiTH_r3V3Rse_eN91NE3rinG}`

`chal.exe` used the OTP key `nAs4202S` (although shorter than the plaintext) to encrypt the flag like what HW11 problem 6(b) did and store it in a char array called pattern. When the code is executed, it compares if the argument string is the same as decrypted plaintext during runtime. If it is, then it prints `Congratulations! You found the flag!`, otherwise, it prints `Haha! wrong >:)!!!!!!`.