دانشکده مهندسی کامپیوتر

پروژه ایریدیوم

امنیت سیستم‌های کامپیوتری

مدرس: دکتر ابوالفضل دیانت

محمدحسین عباسپور، فرزان رحمانی

شماره دانشجویی: ۹۹۵۲۱۴۳۳، ۹۹۵۲۱۲۷۱

نیم سال دوم
سال تحصیلی ۱۴۰۳-۱۴۰۲

۱ مقدمه

در سال ۲۰۱۶ بدافزاری به نام Mirai دست به آلوده کردن هزاران دستگاه با ارسال درخواست DNS به سرور و مختل کردن آن، باعث از بین رفتن دسترسی بسیاری از وبسایت‌های معروف جهان شد. در این پروژه قصد داریم که یک شبیه‌ساز از این بدافزار بوجود بیاوریم.

۲ توضیح پروژه

در این پروژه ما باید شبکه را اسکن کنیم و به دنبال پورت‌های باز شبکه بگردیم و در آنهایی که برای سرویس ssh میباشند، با آزمایش چندین رمز به این سیستم‌ها نفوذ کرده و بدافزاری را روی آنها بارگذاری کنیم تا اطلاعات مهم سیستم را جمع‌آوری کند. برای نفوذ از چند رمز معروف مانند admin استفاده کردیم و اگر رمز آن سیستم جزو این پسورهای معروف نباشد نمیتوانیم وارد سیستم شویم. دقیقا مشابه روش Mirai.

۳ ساختار پروژه

برای پیاده‌سازی این پروژه از docker استفاده کردیم. در ابتدا به کمک داکر یک شبکه داکر ایجاد کردیم و داخل آن تعداد container ساختیم. در ابتدا باید برای هر سرویس یک image بسازیم:

- attacker-image
- target-server-image
- web-server-image

سپس بعد از اینکه تمام این سرویس‌ها را بالا آوردیم، حمله را انجام میدهیم.

۱.۳ attacker-image

وظیفه این ماشین همانطور که از اسمش پیداست برای حمله به هدف میباشد. این ماشین بعد از بالا آمدن با استفاده از اسکریپت scan.sh اقدام به اسکن پورت‌های باز و شناسایی آنهایی که سرویس ssh بر روی آنها اجرا میشود میکند. این اسکریپت در یک محدوده از IP ها، تمام هاست‌های فعال را بررسی میکند که آیا پورت مدنظر را دارد یا نه. سپس آنها را ذخیره میکند.

در گام بعدی با استفاده از اسکریپت hack.sh شروع به حمله به تک تک پورت‌های ذخیره شده‌ی هاست‌ها میکند. برای حمله نیاز به تعدادی رمز داریم تا آنها را بر روی قربانی تست کنیم. این رمزها داخل فایل user_password.csv قرار دارند: پس از تست کردن این رمزها، در صورتی که ارتباط با قربانی برقرار شد، بدافزار

1	farzan	farzan
2	admin	1234
3	abbas	abbas
4	admin	4321
5	1234	1234
6	hi	bye
7	mmd	mmd
8	bro	bro
9	hello	hello
10	root	root
11	salam	salam
12	jesus	christ
13	mininet	mininet
14	admin	admin
15	superuser	superuser

از وب‌سرور بر روی سرور قربانی آپلود میشود.

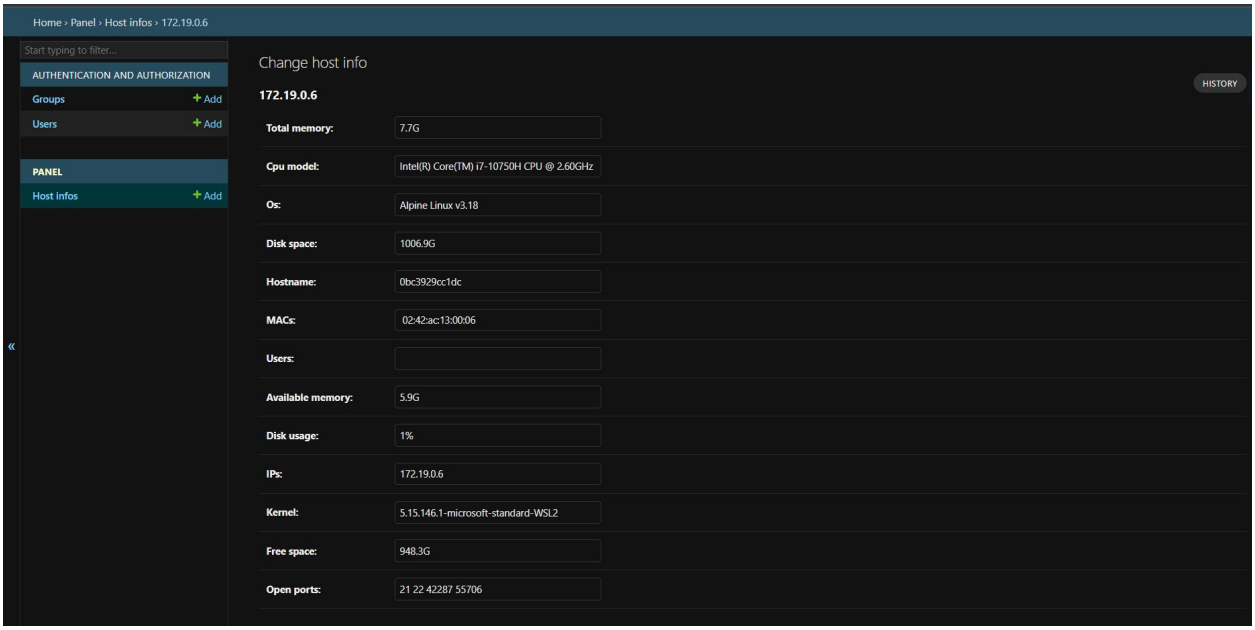
۲.۳ target-server-image

این سرورها، سرورهای قربانی هستند که بدافزار بر روی آنها قرار میگیرد. در اینجا ۵ عدد سرور را با استفاده از داکر بالا می‌آوریم.

۳.۳ web-server-image

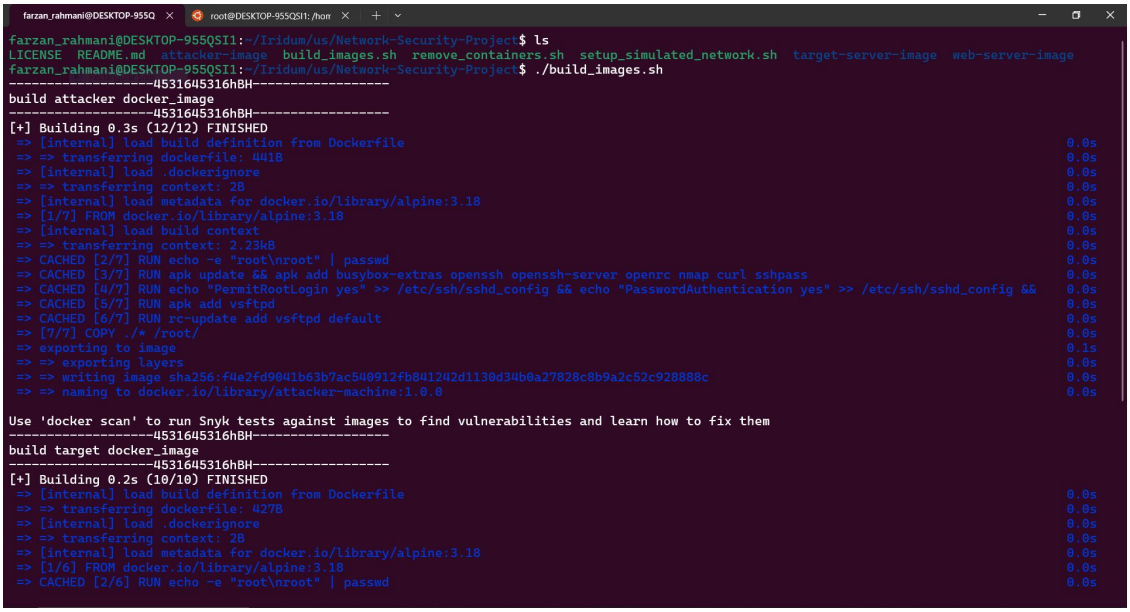
این وب سرور ساده را با استفاده از django ایجاد کردیم. وظیفه این ماشین دانلود بدافزار، ارسال اطلاعات قربانی، ذخیره اطلاعات در یک دیتابیس و یک رابط کاربری برای مشاهده اطلاعات میباشد. در اینجا چون خیلی پروژه سنگینی نبود از دیتابیس پیش‌فرض django یعنی sqlite استفاده کردیم. وظایفی که ذکر شد توسط اسکریپتی به نام infogather.sh انجام میشود.

نمونه‌ای از اطلاعات ذخیره‌شده در دیتابیس در شکل زیر قابل مشاهده است:



۴ اجرای پروژه

برای اجرای پروژه در ابتدا باید تمام image ها را ایجاد کنیم. برای این کار از اسکریپت build_images.sh استفاده میکنیم.



در مرحله بعد باید شبکه داکر را بوجود بیاوریم. برای این کار از اسکریپت setup_sim.sh استفاده میکنیم. این اسکریپت شبکه داکر را ایجاد میکند سپس image های:

- attacker-server-image
- target-server-image
- web-server-image

را اجرا میکند.

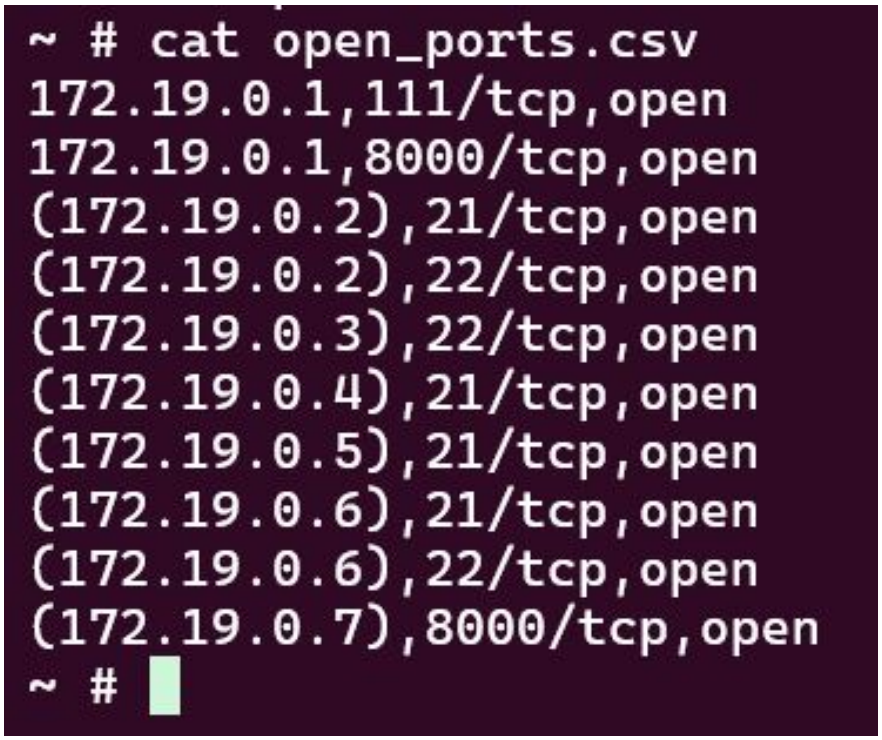
```
farzan_rahmani@DESKTOP-955Q  root@DESKTOP-955QSI1: /hom
 => [internal] load build context                                                                    0.0s
 => => transferring context: 162.27kB                                                                0.0s
 => CACHED [2/5] WORKDIR /usr/src/app                                                                0.0s
 => [3/5] COPY ./requirements.txt /usr/src/app                                                       0.0s
 => [4/5] RUN pip install -r requirements.txt                                                       51.8s
 => [5/5] COPY . /usr/src/app                                                                        0.1s
 => exporting to image                                                                               1.4s
 => => exporting layers                                                                              1.4s
 => => writing image sha256:c4ec20f6706b6654b1cfff7de3854884d122cbdaff6d4466287a8c3c93571f80         0.0s
 => => naming to docker.io/library/web-server:1.0.0                                                  0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
farzan_rahmani@DESKTOP-955QSI1:~/Iridum/us/Network-Security-Project$ ls
LICENSE  README.md  attacker-image  build_images.sh  remove_containers.sh  setup_simulated_network.sh  target-server-image  web-server-image
farzan_rahmani@DESKTOP-955QSI1:~/Iridum/us/Network-Security-Project$ ./setup_simulated_network.sh
------------------5231452314hBH------------------
Create docker_network
------------------5231452314hBH------------------
baaacc3ff5a520af61be5bee29b4e1f92a81cb9fd409ad7efad26ddf3a62fb1e
------------------5231452314hBH------------------
Create target servers
------------------5231452314hBH------------------
d63965c512fee06a1fc7442b418ea2404099d0a7d2fe57e34d14a660f14574d7
e8147124303c860b8773d984eef6b4c4a3198ac463d6e1663107b3e0705a19c9
623411d91e3317l46e30b270e433ba6cece21f20a0a3aec67290e36669444dfd50
ae6041425acec2b4f3560103b83e7c171eedcd88deded0c5d8d02cc983941e9f
0bc3929cc1dc5d27598c7c91c759ee8a45bbcb2d87d33c4855ad838d738d2439
------------------5231452314hBH----------------------------------------
Start ssh and ftp services on target servers
------------------5231452314hBH----------------------------------------
-----------------------------------
Create web server
-----------------------------------
e8cb62a3667a328c67d15bc750551530c9a73152cb9b5f6479a9aefb86e0cf58
-----------------------------------
Create attack machine
-----------------------------------
/ #
```

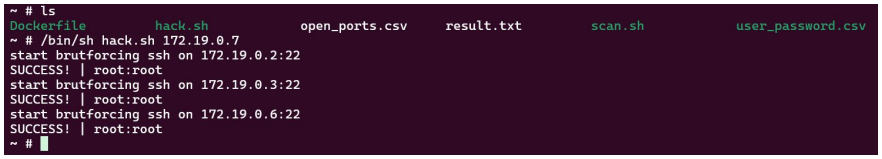
ابتدا با استفاده از دستور docker network inspect آدرس شبکه داکر را پیدا میکنیم:

```
farzan_rahmani@DESKTOP-955QS  root@DESKTOP-955QSI1: /hom
root@DESKTOP-955QSI1:/home/farzan_rahmani/Iridum# docker network inspect simulated_network
[
    {
        "Name": "simulated_network",
        "Id": "baaacc3ff5a520af61be5bee29b4e1f92a81cb9fd409ad7efad26ddf3a62fb1e",
        "Created": "2024-05-02T12:54:22.055087203Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "0bc3929cc1dc5d27598c7c91c759ee8a45bbcb2d87d33c4855ad838d738d2439": {
                "Name": "ser5",
                "EndpointID": "46cfbdee9b3e0d19ce71be550c9f557b7bf6750ac766df4d657d73751e7cf8b0",
                "MacAddress": "02:42:ac:13:00:06",
                "IPv4Address": "172.19.0.6/16",
                "IPv6Address": ""
            },
            "623411d91e3317l46e30b270e433ba6cece21f20a0a3aec67290e36669444dfd50": {
                "Name": "ser3",
                "EndpointID": "8dfc81f018d5b309e7f7a94496b3485e3629120babcdf001df34348972022a0",
                "MacAddress": "02:42:ac:13:00:04",
                "IPv4Address": "172.19.0.4/16",
                "IPv6Address": ""
            },
            "ae6041425acec2b4f3560103b83e7c171eedcd88deded0c5d8d02cc983941e9f": {
                "Name": "ser4",
                "EndpointID": "71bf113a82ccd469c42eef643e6b9650195199018542f275fdebd5a60cc98b45",
                "MacAddress": "02:42:ac:13:00:05",
                "IPv4Address": "172.19.0.5/16",
                "IPv6Address": ""
            },
            "d63965c512fee06a1fc7442b418ea2404099d0a7d2fe57e34d14a660f14574d7": {
                "Name": "ser1",
                "EndpointID": "237d1cbeb1250b3307998ccdd01e11e029c9139f40e8cd08ae4221714d6d741a",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            },
            "e8147124303c860b8773d984eef6b4c4a3198ac463d6e1663107b3e0705a19c9": {
                "Name": "ser2",
```

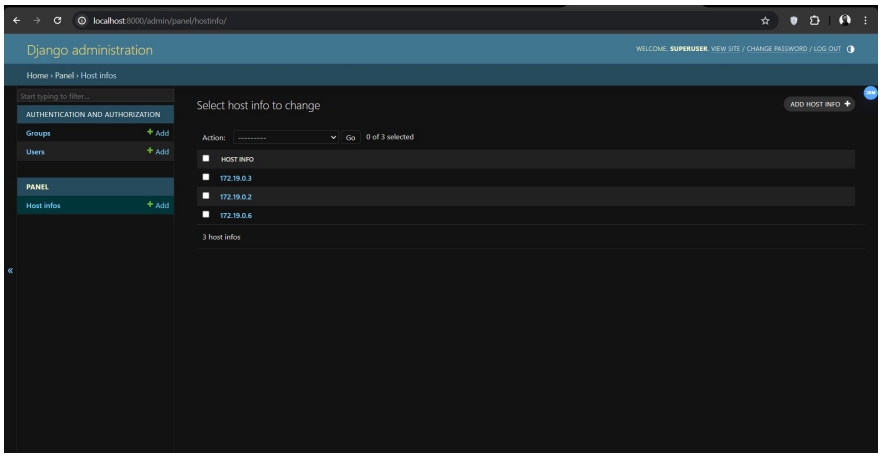
سپس اسکریپت scan.sh اجرا میکنیم محدوده مدنظر برای اسکن را به آن میدهیم. پس از اتمام، پورت‌های باز داخل فایل open_ports.csv ذخیره میشوند:



در مرحله بعد اسکریپت hack.sh را اجرا میکنیم:



اطلاعات ارساشده توسط سرورهای قربانی در دیتابیس ذخیره می‌شوند:



در آخر برای متوقف کردن و حذف شبکه داکر و container ها از اسکریپت remove_containers.sh استفاده میکنیم.