



سایین اعلا گلبرگ سپهرآرا

پاییز ۱۴۰۲

مقدمه

در این پروژه هدف این است که یک دستگاه قربانی پیدا بکنیم و پس اتصال به دستگاه قربانی فایل bash را اجرا کنیم که اطلاعاتی از سیستم قربانی بدهد. در این مرحله پروژه به چند فاز تقسیم که در فاز اول درباره attacker است و در فاز دوم درباره نحوه اتصال و ساخت شبکه با استفاده از docker است گفته شده است

Attacker

در قسمت attacker فایل bash.sh وجود دارد که در ابتدا بررسی میکند با IP address مربوطه کدام subnet ها و ip ها reachable هستند و از بین این ها کدام port ها باز هستند که در فایل outputfile ذخیره میشوند (برای پیدا کردن port های باز از nmap استفاده شد) در فایل merge.py هم دو فایل username و password.csv را که لیستی از یوزرنیم و پسورد های ما است را خوانده و تمامی حالات ممکن و قابل قبول را در فایل username-pass.csv میریزد این برنامه یک اسکریپت Bash است که برای تست ورود به سیستم های مختلف با استفاده از یک فایل CSV حاوی اطلاعات ورود (نام کاربری و رمز عبور) و یک فایل CSV حاوی آدرس های IP استفاده می شود. هدف اصلی این برنامه تست ورود به سیستم با استفاده از ابزار sshpass است. در ابتدا، متغیرهایی برای مسیر فایل CSV حاوی اطلاعات ورود (password.csv)، (outputfile) CSVIP(IPs)، سپس، با دستور echo، سربرگ CSV به فایل خروجی اضافه می شود. سپس با استفاده از دستور tail، خط اول (سربرگ) فایل IPs را صرف نظر می کنیم. در حلقه اول، while آدرس های IP را خوانده و در متغیر ip ذخیره می کنیم. درون حلقه دوم، while نام کاربری و رمز عبور را خوانده و در متغیرهای username و password ذخیره می کنیم. سپس، نام کاربری و رمز عبور را به فایل خروجی اضافه کرده و با استفاده از ابزار sshpass و دستور ssh، تلاش برای ورود به سیستم با اطلاعات ورود فعلی انجام می دهیم. اگر ورود موفقیت آمیز بود، وضعیت ورود را به "Success" تغییر می دهیم و نتیجه ورود را به فایل خروجی اضافه می کنیم. در صورت ورود موفقیت آمیز، حلقه while بیرونی نیز قطع می شود. در صورتی که هیچ ورود موفقیت آمیزی در حلقه while داخلی یافت نشود، برنامه به حلقه while بیرونی برمی گردد و اجرای برنامه خاتمه می یابد.

```
admin,11111
172.22.0.1,admin,11111,Failed
admin,1234567
172.22.0.1,admin,1234567,Failed
admin,dragon
172.22.0.1,admin,dragon,Failed
admin,ginger
172.22.0.1,admin,ginger,Failed
administrador,moon
172.22.0.1,administrador,moon,Failed
autodiscover,123456
172.22.0.1,autodiscover,123456,Failed
autodiscover,password
172.22.0.1,autodiscover,password,Failed
autodiscover,12345678
172.22.0.1,autodiscover,12345678,Failed
autodiscover,qwerty
172.22.0.1,autodiscover,qwerty,Failed
autodiscover,123456789
172.22.0.1,autodiscover,123456789,Failed
autodiscover,12345
172.22.0.1,autodiscover,12345,Failed
root,root
172.22.0.2,root,root,Success
(reverse-i-search)~*~
```

Victim

در نسخه قربانی ابتدا نیاز داریم یک شبکه داکر بسازیم
این برنامه یک اسکریپت Bash است که برای محیط تست و شبیه‌سازی یک شبکه داخلی با استفاده از Docker استفاده می‌شود. در طول اجرای برنامه، شبکه داخلی ایجاد می‌شود و سرویس‌های مختلفی روی سرورهای هدف اجرا می‌شوند. همچنین یک سرور وب و یک ماشین حمله نیز ایجاد می‌شوند.
برنامه در ابتدا با استفاده از دستور `echo` خطوط جداکننده برای بخش‌های مختلف برنامه چاپ می‌کند.

سپس با استفاده از دستور `network docker create` یک شبکه با نام `testnet` ایجاد می‌شود. بعد از آن، با استفاده از دستورات `docker run` سه سرور هدف با نام‌های `ser1`، `ser2` و `ser3` ایجاد می‌شوند و به شبکه `testnet` متصل می‌شوند. این سرورها روی تصویری با نام `target` اجرا می‌شوند و در حالت پیش فرض با کاربر `root` اجرا می‌شوند.

سپس با استفاده از دستورات `exec`، `docker`، سرویس‌های `FTP`، `SSH` و `crond` روی هر سرور هدف فعال می‌شوند.

در مرحله بعد، یک سرور وب با نام web ایجاد می‌شود که در بستر شبکه testnet قرار دارد و پورت ۸۰۰۰ را به پورت ۸۰۰۰ سیستم میزبان متصل می‌کند.

در نهایت، یک ماشین حمله با نام attacker ایجاد می‌شود که به شبکه testnet متصل می‌شود و در حالت تعامل اجرا می‌شود.

[illegible]

این برنامه یک فایل Dockerfile است که برای ساخت یک تصویر Docker استفاده می‌شود. این تصویر بر اساس توزیع Alpine استوار است و برخی از بسته‌ها و تنظیمات مورد نیاز را نصب و پیکربندی می‌کند. در ابتدا، از تصویر پایه Alpine:latest استفاده می‌شود. این تصویر حاوی جدیدترین نسخه از توزیع Linux Alpine است.

سپس، با استفاده از دستور **USER** کاربر ریشه (root) تعیین می‌شود که برای اجرای دستورات بعدی استفاده می‌شود.

در دستور بعدی، با استفاده از دستور RUN و دستور passwd رمز عبور برای کاربر root به "root" تنظیم می‌شود.

سپیس، با استفاده از دستور RUN بسته‌های مورد نیاز را نصب می‌کنیم. این شامل برزورسانی بسته‌ها (apk، update)، نصب بسته‌های (openrc، openssh-server، openssh، busybox-extras، nmap، curl، sshpass و bash است.

استفاده از دستور echo تنظیمات SSH را در فایل `(PermitRootLogin)(PasswordAuthentication).rc-etc/ssh/sshd` `statustouch/run/openrc/softlevel`.

در دستور بعدی، بسته `vsftpd` نیز نصب می‌شود.

سپس، با استفاده از دستور `rc-update` سرویس `vsftpd` به عنوان سرویس پیش فرض برای اجرا تنظیم می‌شود.

در دستور بعدی، با استفاده از دستور `COPY` تمام فایل‌های موجود در مسیر فعلی (فایل‌های سورس) را به مسیر `/root/` درون تصویر `Docker` منتقل می‌کنیم.

در نهایت، با استفاده از دستور `CMD` فرآیند اصلی تعیین می‌شود که در اینجا یک `Shell Bash` است و در داخل تصویر `Docker` اجرا خواهد شد.

برنامه فوق یک فایل `Dockerfile` است که برای ساخت تصویر `Docker` استفاده می‌شود. این تصویر شامل سیستم عامل `Alpine` است که به عنوان پایه استفاده می‌شود و برخی از سرویس‌ها و ابزارهای دیگر نیز در آن نصب می‌شوند.

در ابتدا، کاربر ریشه (`root`) تعریف می‌شود و سپس سه پورت (`۲۲`، `۱۲۳۴` و `۵۶۷۸`) با استفاده از دستور `EXPOSE` اعلام می‌شوند. این پورت‌ها برای ارتباط با سرویس‌های دیگر استفاده خواهند شد.

سپس با استفاده از دستور `RUN` دستورات لازم برای نصب بسته‌ها و تنظیمات سیستم اجرا می‌شوند. این شامل بروزرسانی بسته‌ها (`apk update`)، نصب بسته‌های `busybox-extras`، `busybox-openrc`، `openssh-server`، `openssh`، `curl` و `openrc` است. همچنین تنظیماتی برای `SSH` اعمال می‌شود تا اجازه ورود ریشه (`PermitRootLogin`) و احراز هویت با رمز عبور (`PasswordAuthen`) (`PasswdAuthen`) (`Authentication`) را فعال کند. در نهایت، سرویس‌ها و سطح نرم‌افزاری راه‌اندازی می‌شوند.

دستورات بعدی شامل نصب بسته `vsftpd` و اضافه کردن آن به لیست سرویس‌های راه‌اندازی به صورت پیش فرض است (`rc-update add vsftpd default`).

در نهایت، با استفاده از دستور `CMD` پردازش `/bin/bash` به عنوان فرآیند اصلی برنامه تعریف می‌شود که به عنوان `Shell` پیش فرض در داخل تصویر `Docker` اجرا خواهد شد. محتوا کامل خروجی در فایل `result.txt` نمایش داده می‌شود.

```
9ebcd59adc:/attacker# ls
bash1.sh  bash2-connectToDevices.sh  bash2.sh  login_results.csv  output.csv  user-pass.csv
```

در قسمت `attacker` هم یک برنامه تحت وب توسط زده شده. `django` زده شده است که اطلاعات فرد در در برنامه `script.sh` این برنامه یک اسکریپت `Shell` است که اطلاعات سیستم را جمع آوری کرده و در یک فایل `JSON` ذخیره می‌کند. سپس این فایل `JSON` را به یک آدرس وب ارسال می‌کند.

در ابتدا، مسیر فایل خروجی و نام فایل `JSON` تعیین می‌شود و یک فایل جدید باز می‌شود. سپس با استفاده از دستورات `echo` و `grep` نوع و نسخه لینوکس را از فایل `/etc/os-release` استخراج می‌کند و در فایل `JSON` ذخیره می‌کند.

در مرحله بعد، با استفاده از دستور `cat`، اطلاعات پردازنده را از فایل `/proc/cpuinfo` استخراج می‌کند و در فایل `JSON` ذخیره می‌کند.

سپس، با استفاده از دستور `date`، تاریخ و زمان سیستم را جمع آوری می‌کند و در فایل `JSON` ذخیره می‌کند.

در بخش بعدی، با استفاده از دستور `df`، اطلاعات فضای مصرف شده در دیسک‌ها را جمع آوری می‌کند و در فایل `JSON` ذخیره می‌کند.

در آخر، با استفاده از دستور `curl`، فایل `JSON` را به یک آدرس وب ارسال می‌کند. آدرس وب ورودی به عنوان پارامتر اول به اسکریپت داده می‌شود و فایل `JSON` به عنوان بدنه درخواست `POST` با استفاده از پوشه `'app/post-info'` در آدرس وب ارسال می‌شود. برای ران کردن `ssh` از این دستور استفاده شده است:

```
root@Golbang:/mnt/f/Term7/security/Iridium/Iridium# cd Victim-1
root@Golbang:/mnt/f/Term7/security/Iridium/Iridium/Victim-1# docker exec -it eager_wright rc-service sshd start
* Caching service dependencies...
Service 'acpid' needs non existent service 'dev'
Service 'hwdm' needs non existent service 'dev'
Service 'machinid' needs non existent service 'dev'
Service 'watchdog' needs non existent service 'dev'
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/blkio/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/cpu/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/cpuset/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/devices/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/freezer/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/hugetlb/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/memory/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/misc/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/net_cls/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/net_prio/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/perf_event/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/pids/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/rdma/tasks: Read-only file system
/lib/rc/sh/openrc-run.sh: line 108: can't create /sys/fs/cgroup/systemd/tasks: Read-only file system
ssh-keygen: generating new host keys: RSA ECDSA ED25519
* Starting sshd ...
root@Golbang:/mnt/f/Term7/security/Iridium/Iridium/Victim-1#
```

دیتا بیس :

