به نام خدا

گزارش پروژه امنیت سیستم‌های کامپیوتری

گروه ۱۷

آرمان حیدری

پارسا عیسی زاده

غزل زمانی نژاد

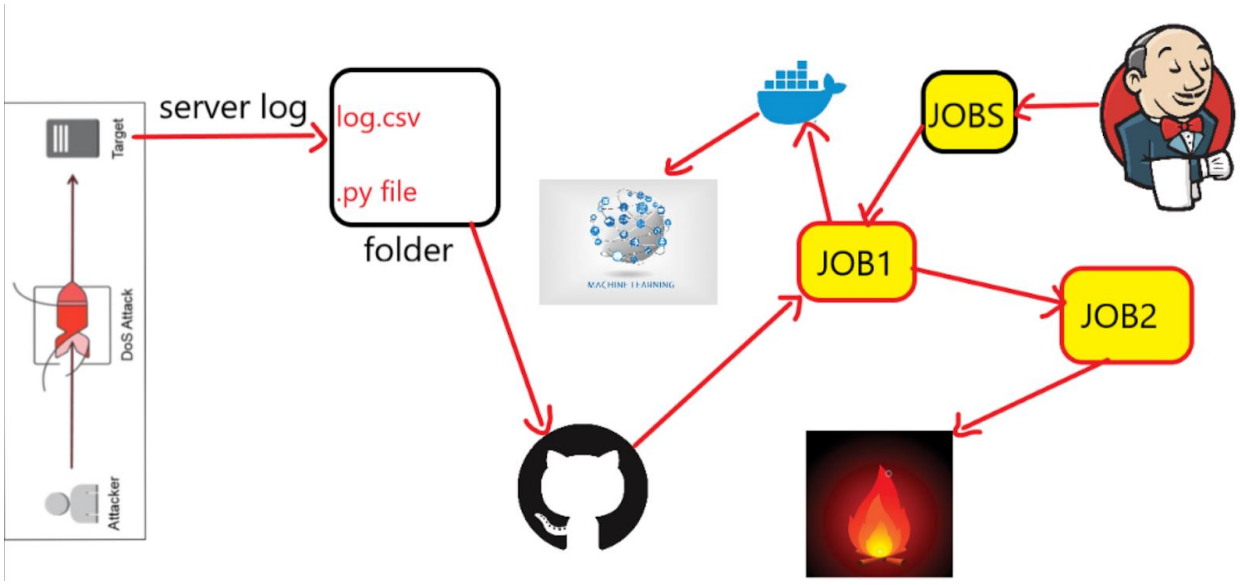
بخش اول

حمله dos به حمله ای می گویند که در آن با تعداد ریکوئست های بیش از حد منابع سخت افزاری دچار مشکل می شوند و سایت از خدمتگزاری باز می ایستد .

در این پروژه هدف ما این بود که حمله dos را تشخیص داده و سپس برای آن حمله یک رویکرد پیاده سازی کنیم .

رویکرد ما بلاک کردن ip حمله کننده بود . از آنجایی که dos attack را باید تشخیص دهیم و ddos نیست ، درخواست های با تعداد بالا تنها از طریق یک ip address صورت می‌پذیرد در نتیجه میخواهیم آن ip address را بلاک کنیم .

روال کلی ای که در ابتدا در نظر داشتیم به شکل زیر بود:



به این گونه که از http server ای که داریم یک log تهیه میکنیم که در آن ip address ها و زمان ریکوئستشان را ثبت میکنیم . سپس آن را روی git می گذاریم و هر زمان که آن فایل تغییری کرد یک پایپلاین که شامل چندین job است توسط jenkins اجرا می شود .

به دو دلیل از این راه صرف نظر کردیم :

1. سرعت پایین آپلود کردن روی git

2. امنیت پایین git (به دلیل این که هر بار بین ما و git ارتباطی صورت میگیرد که امکان حمله را بالا میبرد)

در نتیجه تصمیم بر آن شد که همچنان log بگیریم ، هر 30 ثانیه یک بار ip های malware را تشخیص دهیم و در یک فایل بریزیم .

سپس به زمان هر ریکوئست آن فایل را چک کنیم و اگر ip ای که ریکوئست را زده malware بود برایش خدمت رسانی نکنیم و بلاکش کنیم .

فایلی که در آن http server را به اجرا گذاشتیم به شکل زیر است :

```
if __name__ == "__main__":
    webServer = HTTPServer((hostName, serverPort), MyHandler)
    print("Server started http://%s:%s" % (hostName, serverPort))

    try:
        webServer.serve_forever()
    except KeyboardInterrupt:
        pass

    webServer.server_close()

    print("Server stopped.")
```

که در آن از پکیج http پایتون استفاده کردیم .

سپس handler را به ازای هر ریکوئست تعریف میکنیم :

```python
class MyHandler(BaseHTTPRequestHandler):
    def setup(self) -> None:
        return super().setup()

    def do_GET(self):
        ip_addr = str(self.client_address[0])
        row = [ip_addr, datetime.now()]

        malware_ips = get_malware_ips()
        if ip_addr in malware_ips :
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()

            self.wfile.write(bytes("<html><head><title>https://pythonbasics.org</title></head>", "utf-8"))
            self.wfile.write(bytes("<p>Request: %s</p>" % self.path, "utf-8"))
            self.wfile.write(bytes("<body>", "utf-8"))
            self.wfile.write(bytes("<p>You are dos attack</p>", "utf-8"))
            self.wfile.write(bytes("</body></html>", "utf-8"))

        else :
            df = pd.read_csv('request_logger.csv')
            df.loc[df.shape[0]] = row
            df.to_csv('request_logger.csv', index=False)

            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()

            self.wfile.write(bytes("<html><head><title>https://pythonbasics.org</title></head>", "utf-8"))
```

می‌بینیم که اگر ip جزو ip های بدکار نباشد، صرفا ریکوئست آن را در فایل request_logger میریزد. همچنین زمان ثبت آن را هم نگه میداریم که اگر یک ip خیلی سریع ریکوئست های زیادی میزد آن را شناسایی کنیم. نمونه ای از محتوای این فایل به این صورت است:

server.py × request_logger.csv × dos_detec

```
IP,time
127.0.0.1,2022-07-03 23:18:06.811287
127.0.0.1,2022-07-03 23:18:07.175286
127.0.0.1,2022-07-03 23:18:10.602286
127.0.0.1,2022-07-03 23:18:10.899049
127.0.0.1,2022-07-03 23:18:13.055181
127.0.0.1,2022-07-03 23:18:13.148188
127.0.0.1,2022-07-03 23:18:17.531581
127.0.0.1,2022-07-03 23:18:17.828576
127.0.0.1,2022-07-03 23:18:19.181577
127.0.0.1,2022-07-03 23:18:19.275580
127.0.0.1,2022-07-03 23:18:19.757581
127.0.0.1,2022-07-03 23:18:20.063584
127.0.0.1,2022-07-03 23:18:20.637584
127.0.0.1,2022-07-03 23:18:20.745578
127.0.0.1,2022-07-03 23:19:23.949354
127.0.0.1,2022-07-03 23:19:24.043337
127.0.0.1,2022-07-03 23:19:24.781694
127.0.0.1,2022-07-03 23:19:25.090603
```

که چون فقط با دستگاه خودم به آن با فاصله زمانی چندین ثانیه ریکوئست زده ام به این صورت شده است.

lp_addr را به دست می آوریم ، ابتدا توسط متد get_malware_ips (که جلو تر توضیح داده خواهد شد) از یک فایل ip های malware را میخوانیم . سپس اگر این ip_addr در آن فایل نبود ریسپانس عادی سایت را نشان می دهیم و خدمت رسانی میکنیم .

اما اگر آدرس در آن فایل بود به آن یک پیغام "You are dos attack" نشان می دهیم .

- نکته : در واقعیت نباید به dos attack ریسپانس 200 باز گردانیم ولی در اینجا برای نشان دادن تشخیص ریسپانس 200 ارسال میکنیم .

متد get_malware_ips به شکل زیر است :

```
def get_malware_ips ():
    try :
        file1 = open('/home/snapp/Documents/Uni /CS/project/ddos-detection/malware_ips.txt', 'r')
        Lines = file1.readlines()
        malware_ips = set()

        for line in Lines:
            for ip_address in line.strip().split(','):
                malware_ips.add(ip_address)

        return malware_ips
    except :
        return []
```

در اینجا صرفا می آییم خط به خط یک فایل را میخوانیم و ip ها را درون یک set میریزیم و آن را باز میگردانیم .

- از set استفاده کردیم زیرا برای صرفه جویی در حافظه و سرعت ، duplicate data نداشته باشیم .

از این متد در هنگام ذخیره سازی ip address ها در فایل هم استفاده میکنیم .

یک متد داریم که هر چند ثانیه یک بار نیاز است که فراخوانی شود ، متد dos_detect .

این متد روی log های ما میگردد و ip address هایی که در یک بازه زمانی کوتاه ثبت شده اند و تعدادشان از یک treshold ای بیشتر است را به عنوان malware ip تشخیص میدهد و در فایل میریزد .

```python
def dos_detect(requests_threshold=15):
    requests_df = pd.read_csv('/home/snapp/Documents/Uni /CS/project/ddos-detection/request_logger.csv')
    all_ips = list(requests_df['IP'].unique())
    malware_ips = []
    for ip in all_ips:
        ip_df = requests_df[requests_df['IP'] == ip]
        if ip_df.shape[0] >= requests_threshold:
            malware_ips.append(ip)

    file_object = open('/home/snapp/Documents/Uni /CS/project/ddos-detection/malware_ips.txt', 'a')

    something_added = False
    for malware_ip in malware_ips:
        malware_ips_till_now = get_malware_ips()

        # guard detections
        if malware_ip in malware_ips_till_now :
            continue
        if malware_ip == '':
            continue

        something_added = True
        file_object.write(malware_ip)
        file_object.write(',')

    if something_added:
        file_object.write('\n')
```

```python
    file_object.close()
    return malware_ips
```

در این جا نیز باز متد get_malware_ips فراخوانی می شود تا اگر داده ای قبلا malware تشخیص داده شده است دوباره روی فایل ذخیره نشود .

این تسک باید هر چند ثانیه یکبار فراخوانی شود در نتیجه یک periodic task است .

برای نوشتن تسک های پریودیک ابتدا سراغ ابزار celery رفتیم . در این ابزار از یک rabbitmq broker استفاده می شود که تسک ها روی آن queue می شوند و سپس به ترتیب اجرا می‌شوند .

اما ما میخواستیم که ارتباطمان با دنیای بیرون حتی الامکان قطع شود و به یک broker ای request نفرستیم . در نتیجه رفتیم سراغ cron job .

```
snapp@snapp-ThinkPad-E14:~/Documents/Uni /CS/project/celerity-practice$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

* * * * * /bin/python3 "/home/snapp/Documents/Uni /CS/project/ddos-detection/dos_detect.py"
```

Cronjob یک نرم افزاریست که در سرور ها bash command هایی را در فاصله های زمانی مشخص اجرا میکنند .

cron job یک دستور لینوکس است که برای برنامه ریزی وظایفی که در آینده اجرا می شوند استفاده می شود. این معمولاً برای برنامه‌ریزی کاری که به صورت دوره‌ای اجرا می‌شود استفاده می‌شود - به عنوان مثال، برای ارسال اعلان هر روز صبح.

اغلب، اسکریپت هایی که به عنوان cron job نامیده می شوند، فایل ها یا پایگاه های داده را تغییر می دهند. با این حال، آنها می توانند کارهای دیگری را نیز انجام دهند که داده های روی سرور را تغییر نمی دهند، مانند ارسال اعلان های ایمیل.

پس به کمک این job در هر بازه زمانی متد دوباره فراخوانی می شود .

- دلیل اینکه همه آدرس فایل ها در برنامه absolute بود استفاده از cronjob بود .

اما در cronjob برای زیر یک دقیقه زمانبندی ای نداریم . در نتیجه ناچار شدیم از طریق زیر اقدام کنیم:

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

* * * * * /bin/python3 "/home/snapp/Documents/Uni /CS/project/ddos-detection/dos_detect.py"

* * * * * (sleep 10 ; bin/python3  "/home/snapp/Documents/Uni /CS/project/ddos-detection/dos_detect.py")

* * * * * (sleep 20 ; bin/python3  "/home/snapp/Documents/Uni /CS/project/ddos-detection/dos_detect.py")

* * * * * (sleep 30 ; bin/python3  "/home/snapp/Documents/Uni /CS/project/ddos-detection/dos_detect.py")

* * * * * (sleep 40 ; bin/python3  "/home/snapp/Documents/Uni /CS/project/ddos-detection/dos_detect.py")

* * * * * (sleep 50 ; bin/python3  "/home/snapp/Documents/Uni /CS/project/ddos-detection/dos_detect.py")
```

در این صورت تنها ip ای malware تشخیص داده می شود که در ده ثانیه بیش از ۱۵ ریکوئست بزند .

```python
def dos_detect(requests_threshold=15):
    requests_df = pd.read_csv('request_logger.csv')
    all_ips = list(requests_df['IP'].unique())
    malware_ips = []
    for ip in all_ips:
        ip_df = requests_df[requests_df['IP'] == ip]
        if ip_df.shape[0] >= requests_threshold:
            malware_ips.append(ip)
    file_object = open('malware_ips.txt', 'a')
    something_added = False
    for malware_ip in malware_ips:
        malware_ips_till_now = get_malware_ips()

        # guard detections
        if malware_ip in malware_ips_till_now:
            continue
        if malware_ip == '':
            continue

        something_added = True
        file_object.write(malware_ip)
        file_object.write(',')
    if something_added:
        file_object.write('\n')

    file_object.close()
    return malware_ips
```

- این عدد ۱۵ قابل تغییر است .
- درواقع اگر یک ip خاصی بیش از 15 ریکوئست در یک بازه 10 ثانیه ای زده باشد، آن را بدکار تشخیص میدهد.

منابع:

- Introduction to Celery — Celery 5.2.7 documentation
- https://1829034.medium.com/detecting-ddos-attack-with-clustering-47fe491a56a5
- https://www.seobility.net/en/wiki/CronJob
- https://www.hivelocity.net/kb/what-is-cron-job/