Proste narzędzie wykrywające usługi open relay i podatnej na UDP amplification wraz systemem powiadomień

Mateusz Porada 259449

Spis treści

1. Cele projektu

Celem niniejszego projektu jest napisanie i przedstawienie działania programu sprawdzającego czy dany serwer jest serwerem open relay oraz czy usługę przy podanym adresie IP można wykorzystać do ataku UDP amplification. W tym celu stworzono programy w języku Python oraz zaprezentowano jego działanie bazując na maszynach wirtualnych.

2. Definicje

a. Open relay

W protokole SMTP istnieje możliwość konfiguracji serwera mail jako serwera relay. Serwery relay będą przekazywały przesyłane maile dalej do docelowych serwerów poczty elektronicznej. Jednak taka konfiguracja serwera może powodować wiele problemów. Jeśli przekazywanie maili zostanie zezwolone dla zbyt dużej liczby użytkowników np. zezwalając na ruch ze wszystkich adresów IP, serwer taki będzie tak zwanym serwerem open relay. Potencjalny atakujący będzie w stanie wysłać maile np. ze spamem przez open relay, a ten z kolei przekaże spam do serwerów użytkowników. Mail taki może nie zostać wykryty przez działające filtry spamu, ponieważ serwer docelowy może traktować serwer open relay jako zaufany.

b. UDP Amplification

UDP amplification jest atakiem denail-of-service bazującym na tym, że protokół UDP jest protokołem bezpołączeniowym. Oznacza to, że nie wymagane jest wcześniejsze ustanowienie sesji przed rozpoczęciem wysyłania ruchu. Skutkuje to, iż pakiety nie mają żadnej gwarancji na dostarczenie do usługi docelowej, a sama usługa będzie odpowiadać na żądania bazując jedynie na danych adresowych odebranych pakietów, nie oczekując na żadną odpowiedź. Stwarza to możliwość wysyłania zalania ofiary odpowiedziami usług UDP. Atakujący wysyłając specjalnie spreparowane pakiety z IP źródłowym urządzenia ofiary do usługi, natomiast ta będzie odpowiadać na żądania, ale na podstawiony adres. Wysyłając wystarczająco dużo pakietów do usługi ta może zalać ofiarę swoimi odpowiedziami powodując utratę możliwości komunikacji przez zajęcie całej przepustowości. Istotnym elementem tego ataku jest wielkość przesyłanych danych od atakującego i odpowiedzi na nie przez serwery usług. Część zapytań np. w DNS może mieć duże mniejsze rozmiary niż pakiety będące odpowiedzią na te zapytania. Potencjalnie dla atakującego skutkuje to dużo mniejszym wygenerowanym ruchem niż ruch który zacznie otrzymywać ofiara.

3. Skrypt open relay

a. Opis środowiska

Do przedstawienia sposobu działania programu stworzono następujące maszyny wirtualne:

I. Ubuntu 22.04 – działająca jako serwer open relay
II. Kali Linux – działająca jako atakujący/urządzanie sprawdzające czy serwer jest open relay

Serwerem poczty wykorzystywanym w projekcie jest open sourcowy serwer Postfix. Aby sprawić, aby dany serwer stał się serwerem open relay należy w pliku konfiguracyjnym serwera /etc/postfix/main.cf zmienić opcje:

relayhost = 0.0.0.0/0
mynetworks = 0.0.0.0/0

Obie maszyny wirutalne zostały uruchomione dla konfiguracji sieci Sieć NAT w podsieci 10.0.2.0/24. Adresy maszyn to:

i. Serwer Postfix – 10.0.2.15
ii. Kali - 10.0.2.4

Aby zapewnić, że działające maszyny rzeczywiście mają ze sobą połączenie oraz, że serwer Postfix jest open relay przeprowadzono test narzędziem nmap używając skryptu smtp-open-relay.nse. Wynik działania tego programu to:

```
┌──(kali㉿kali)-[~]
└─$ nmap --script smtp-open-relay.nse -p 25,465,587 10.0.2.15
Starting Nmap 7.92 ( https://nmap.org ) at 2023-01-26 13:14 EST
Nmap scan report for 10.0.2.15
Host is up (0.00087s latency).

PORT    STATE  SERVICE
25/tcp  open   smtp
|_smtp-open-relay: Server is an open relay (16/16 tests)
465/tcp closed smtps
587/tcp closed submission

Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
```

b. Program

Program openrelay.py został stworzony w Pythonie jako skrypt który próbuje połączyć się z serwerem o podanym adresie a następnie wysłać maila posługując się nim jako relay. Program posiada CLI z następującymi opcjami:

```
usage: openrelay.py [-h] -a A [-p P] [--from_addr FROM_ADDR]
                    [--to_addr TO_ADDR] [--msg MSG]

Program pozwala na wykrywanie czy serwer SMTP jest open relay

optional arguments:
  -h, --help            show this help message and exit
  -a A                  Adres serwera
  -p P                  Port serwera SMTP
  --from_addr FROM_ADDR
                        Adres źródłowy maila użyty w teście
  --to_addr TO_ADDR     Adres docelowy maila użyty w teście
  --msg MSG             Wiadmość wysłana w testowym mailu
```

Wynik działania programu testując ustawiony wcześniej serwer Postfix to:

```
┌──(kali㉿kali)-[~/openrelay]
└─$ python3 openrelay.py -a 10.0.2.15 -p 25
Server is open relay
```

W skrypcie dodano również możliwość zdefiniowania innych parametrów tj.: adres docelowy i źródłowy mail oraz treść widomości.:

```
┌──(kali㉿kali)-[~/openrelay]
└─$ python3 openrelay.py -a 10.0.2.15 -p 25 --msg "moja wiadomosc" --from_add
r mail@mail.com --to_addr 259449@student.pwr.edu.pl
Server is open relay
```

Skrypt został również przetestowany na zdalnych innych serwerach, lecz we wszystkich przypadkach nie był w stanie utworzyć połączenia.

```
┌──(kali㉿kali)-[~/openrelay]
└─$ python3 openrelay.py -a pwr.edu.pl -p 25
Conetction on this port is not possible, port is closed or filtered
```

Jak widać w wyżej przedstawionych wynikach testu skrypt jest w stanie poprawnie wykrywać serwery open relay poczty, jednak ze względu na duże ryzyko związane z taką konfiguracją serwera SMTP w większości przypadków dostęp do nich będzie ograniczony przez administratorów dzięki, firewallom oraz odpowiednie parametry daemona poczty.

4. Skrypt UDP Amplification

a. Opis środowiska

Ze względu na naturę ataku UDP amplifcation, możliwe jest wykorzystanie wielu usług do zalania ofiary odpowiedziami. W samym skrypcie możliwe jest użycie trzech z nich tj.: DNS, NTP oraz TFTP. W celach testowych użyto znane adresy IP, na których świadczone są te usługi i tylko dla TFTP wymagane było utworzenie własnego serwera. Te adresy to:

i. 8.8.8.8, 9.9.9.9, 208.67.222.222 – serwer DNS
ii. 0.pl.pool.ntp.org, 2.pl.pool.ntp.org, 0.europe.pool.ntp.org, 1.europe.pool.ntp.org – serwer NTP

Do przeprowadzania testów dla protokołu TFTP uruchomiono serwer na taj samej maszynie na której uruchomiono wcześniej serwer Postfix. Wykorzystanym daemonem usługi był tftpd-hpa skonfigurowany zgodnie z plikiem /etc/default/tftpd-hpa Do testów również użyto maszyny wirtualnej Kali Linux zawierającą się w tej samej sieci. Adresacja tych maszyn pozostała taka sama. Wcześniej wspomniana konfiguracja TFTP używała parametrów z następującymi wartościami:

```
  GNU nano 6.2                    /etc/default/tftpd-hpa
#/etc/default/tftpd-hpa
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/tftp"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure --create"
```

Komunikacja z tak ustawionym serwerem była możliwa co zostało sprawdzone dzięki następującym poleceniom:

```
┌──(kali㉿kali)-[~/openrelay]
└─$ tftp 10.0.2.15
tftp> status
Connected to 10.0.2.15.
Mode: netascii Verbose: off Tracing: off Literal: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
```

```
┌──(kali㉿kali)-[~]
└─$ mkdir tftp

┌──(kali㉿kali)-[~]
└─$ cd tftp

┌──(kali㉿kali)-[~/tftp]
└─$ tftp 10.0.2.15
tftp> get file.txt
tftp> quit

┌──(kali㉿kali)-[~/tftp]
└─$ ls
file.txt

┌──(kali㉿kali)-[~/tftp]
└─$ head file.txt
viurmstcupwrgpnvngpcbyqvaraollxblllbaznaezhqfjicgxcdquduwnaxvfvcaqucedsodrbly
osfosjjzfblobmgmomupydvmdtszoauexbcdswkaukxkpsxlofldclzwikzhsoxtozxlvxopuvvcu
tfxfxtrzfildfjwzvyhbayaqqoqrxdoqoiyoapvujhkywvxmbopovshdoftlieghakcjjlzrodoxs
vofujrgoemntcomztpnrgdhpuakuqovfaafzxpsdlajsljshuyiuardakrdqymlvidbhfcquegutk
iqnqkqinmgaagnuewcpgsozogzfitplqfpaplmljqloeawtyeuesfamwdwcdhhzfswnmtxherwiiy
dsujdgkjgttkkkuiuamyphjzipfqzkpxxpsotdpsqjwftlyirjwkmwmjmuiggnzfxdrrhpsngfidi
```

b. Opis programu

Programem realizującym skanowanie w poszukiwaniu usług podatnych na UDP amplification jest scanner.py. Należy zwrócić uwagę iż korzystanie z tego programu wymaga podwyższenie uprawnień w systemie np. przez użycie sudo. Sam skaner symuluje atak w dużo mniejszej skali niż rzeczywiste ataki, Wynika to z tego, że wszystkie odpowiedzi od usługi będą przychodzić na adres IP testującego. Zbyt dużo wygenerowanego ruch spowodowałoby brak możliwości komunikacji przez system. Domyślnie program wysyła zapytania do usług

czekając na ich odpowiedz 30 razy. Wartość ta może zostać zmieniona przez użytkownika dzięki odpowiedniemu użyciu opcji -n w poleceniu. Ważnym elementem symulowanego ataku jest fakt, że odpowiedź na dany pakiet może być dużo większa niż pakiet z zapytaniem. Uwzględniono to również w programie scanner.py, gdzie liczona jest wartość Amplification ratio która jest stosunkiem długości pakietu odebranego do pakietu wysłanego. Przez sam skrypt głownie sprawdzane są następujące aspekty: czy serwer usługi zablokuje podejrzany ruch generowany w bardzo krótkim czasie z jednego powtarzającego się adresu oraz czy generowanie odpowiedzi na przychodzące pakiety nie będzie zbyt wymagające obliczeniowo dla serwera. Pakiety wysyłane do serwerów usług generowane są dzięki modułowi Scapy w Pythonie. Sam program możemy używać z następującymi opcjami:

```
┌──(kali㉿kali)-[~/openrelay]
└─$ sudo python3 scanner.py -h
usage: scanner.py [-h] [-d] [-n] [-t] -a A [-c C] [--file FILE] [--domain DOMAIN]

Program pozwala na wykrywanie, czy serwer usługi jest podatny na atak UDP Amplification

optional arguments:
  -h, --help       show this help message and exit
  -d               Sprawdzenie serwera DNS
  -n               Sprawdzenie dla serwera NTP
  -t               Sprawdzenie dla serwera TFTP
  -a A             Adres serwera lub domena
  -c C             Ilość pakietów do wysłania
  --file FILE      Nazwa pliku dla serwera TFTP
  --domain DOMAIN  Nazwa domeny dla zapytania do serwera DNS
```

Pierwszym przetestowanym protokołem był protokół DNS. Dla każdego adresu testowane są różne typy query: A, AAAA, MX, ANY oraz CNAME. Dzięki temu użytkownik jest wstanie wskazać query o największym Amplification ratio. Ponadto w programie możliwe jest ustawienie różnych domen, których będą dotyczyły zapytania przez opcje –domain. Domyślnie sprawdzana jest domena google.com. Wyniki testów na publicznych serwerach DNS są następujące:

```
┌──(kali㉿kali)-[~/openrelay]
└─$ sudo python3 scanner.py -a 8.8.8.8 -d
Host 8.8.8.8 jest podatny na atak dla query=ANY
Amplification ratio:  1.8928571428571428
Host 8.8.8.8 jest podatny na atak dla query=A
Amplification ratio:  1.2857142857142858
Host 8.8.8.8 jest podatny na atak dla query=AAAA
Amplification ratio:  1.5
Host 8.8.8.8 jest podatny na atak dla query=CNAME
Amplification ratio:  1.8928571428571428
Host 8.8.8.8 jest podatny na atak dla query=MX
Amplification ratio:  1.375

┌──(kali㉿kali)-[~/openrelay]
└─$ sudo python3 scanner.py -a 208.67.222.222  -d -c 20
Host 208.67.222.222 jest podatny na atak dla query=ANY
Amplification ratio:  1.8928571428571428
Host 208.67.222.222 jest podatny na atak dla query=A
Amplification ratio:  1.2857142857142858
Host 208.67.222.222 jest podatny na atak dla query=AAAA
Amplification ratio:  1.5
Host 208.67.222.222 jest podatny na atak dla query=CNAME
Amplification ratio:  1.8928571428571428
Host 208.67.222.222 jest podatny na atak dla query=MX
Amplification ratio:  1.375

┌──(kali㉿kali)-[~/openrelay]
└─$ sudo python3 scanner.py -a 9.9.9.9 -d --domain pwr.edu.pl
Host 9.9.9.9 jest podatny na atak dla query=ANY
Amplification ratio:  1.875
Host 9.9.9.9 jest podatny na atak dla query=A
Amplification ratio:  1.2857142857142858
Host 9.9.9.9 jest podatny na atak dla query=AAAA
Amplification ratio:  1.875
Host 9.9.9.9 jest podatny na atak dla query=CNAME
Amplification ratio:  1.875
Host 9.9.9.9 jest podatny na atak dla query=MX
Amplification ratio:  3.5357142857142856
```

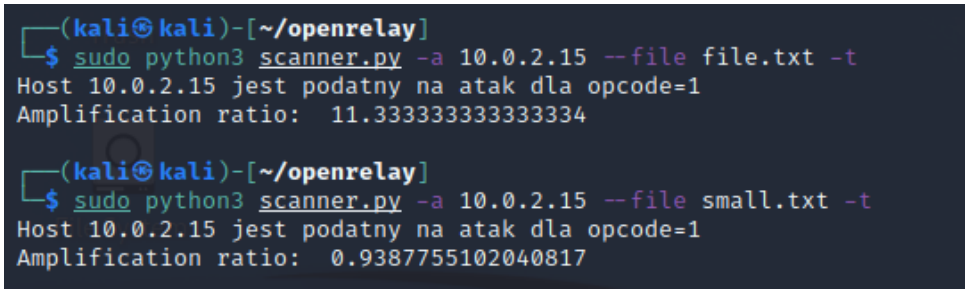
Następnym sprawdzonym protokołem jest protokół synchronizacji czasu NTP. Dla niego testy sprawdzają pakiety z mode=6, czyli pakiety żądające konfiguracji serwera oraz mode = 3 żądające czasu. Pakiety z żądaniem konfiguracji serwera byłyby znacznie skuteczniejsze w potencjalnym ataku jednak dla czterech sprawdzonych publicznych serwerów żaden z nich nie odpowiadał na takie żądania.

```
┌──(kali㉿kali)-[~/openrelay]
└─$ sudo python3 scanner.py -a 0.pl.pool.ntp.org -n
[sudo] password for kali:
Host 0.pl.pool.ntp.org nie jest podatny na atak dla mode=6
Host 0.pl.pool.ntp.org jest podatny na atak dla mode=3
Amplification ratio:  1.0

┌──(kali㉿kali)-[~/openrelay]
└─$ sudo python3 scanner.py -a 1.europe.pool.ntp.org -n
Host 1.europe.pool.ntp.org nie jest podatny na atak dla mode=6
Host 1.europe.pool.ntp.org nie jest podatny na atak dla mode=3

┌──(kali㉿kali)-[~/openrelay]
└─$ sudo python3 scanner.py -a 0.europe.pool.ntp.org -n
Host 0.europe.pool.ntp.org nie jest podatny na atak dla mode=6
Host 0.europe.pool.ntp.org nie jest podatny na atak dla mode=3

┌──(kali㉿kali)-[~/openrelay]
└─$ sudo python3 scanner.py -a 2.pl.pool.ntp.org -n -c 20
Host 2.pl.pool.ntp.org nie jest podatny na atak dla mode=6
Host 2.pl.pool.ntp.org jest podatny na atak dla mode=3
Amplification ratio:  1.0
```

Ostatnim protokołem sprawdzanym w scanner.py jest protokół przesyłania plików TFTP. W skrypcie możliwe jest jedynie żądanie odczytania pliku umieszczonego na serwerze. Jest to najskuteczniejsza metoda i przy bardzo dużym rozmiarze pliku oraz wystarczającej liczbie zapytań możemy szybko zapełnić całą przepustowość ofiary. Opcode w takim pakiecie TFTP jest równy 1. W programie występuje przymus podania nazwy pliku przez opcje –file gdyż tylko plik istniejący na serwerze jesteśmy w stanie wykorzystać do testów jak i ataku. Wszystkie pliki wykorzystywane w testach będą przesyłane w trybie natascii tj.: w standardowym kodzie ASCII. Testy dla serwera TFTP przeprowadzono dla dwóch plików o różnych rozmiarach:

```
mailserver@mailserver:/tftp$ ls -l
razem 104
-rw-rw-rw- 1 tftp tftp 100000 sty 27 17:00 file.txt
-rw-r--r-- 1 root root      6 sty 27 17:44 small.txt
mailserver@mailserver:/tftp$
```

Wyniki tych testów prezentują się następująco:



Jak widać wykorzystanie małych plików przy tego typy atakach na serwer TFTP nie ma sensu ze względu na amplification ratio poniżej 1. Natomiast duże pliki stanowią bardzo dobry materiał do przeprowadzenia ataku.