Nmap 으로 호스트 발견하기!!

By: Mark Wolfgang
moonpie@moonpie.org

November 2002

Table of Contents

1. 소개

외부 penetration tests(PT)를 수행하는 컴퓨터 보안 기술자로서, 매우 넓은 IP 주소 영역을 가진 단체를 평가할 때마다 다람쥐 쳇바퀴 도는 듯한 도전으로 보일 수 있을 것이다. multiple class B, 약간 적은 class C, 마지막으로 시간의 제약이란 것에 직면하게 되었을 때 처음 해야 하는 것은 무엇인지? 유명한 스캐너로 그 주소 영역을 모두 스캔할려고 Go 버튼을 누른 다음, 완료될때까지 기다리고만 있으면 결과가 정확하길 바라는지? 그 스캐너가 찾은 모든 접근 가능한 호스트를 어떻게 믿을 수 있는지? 심지어 살아있는 호스트를 발견하기 위해 사용되는 스캐너의 동작 방식은 알기나 하는지?

이 문서는 위 질문에 답하려 할 것이고, 포트 스캐닝이나 취약성 평가를 정밀하게 수행하기 전에 접근할 수 있는 호스트 를 발견하기 위한, 그러한 기술들에 대하여 (매우 기술적인 면을) 설명할 것이다.

참고: 그 영역의 가능한 모든 IP 주소내 65535개의 TCP와 UDP 포트를 스캔하지 않는한, 몇몇 사람들은 penetration tester(PT)는 절대적으로 부족하지 않다고 알고 있을 것이다. 나는 철저하게는 동의하는 반면에, 그러한 사치스런 스캔을 한다면 시간 낭비라는 것을 말하고 싶다. 정보 보호에 관한 근원적인 주제는 균형 맞추기와 찬반 평가이다. 무조건 완벽하게 하려고 시간을 고려하지 않는다면, 당신은 모든 IP 주소를 무턱대고 계속 스캔하려 할 것이다. 하지만, 균형은 완벽히 하려는 것과 시간에 맞춰 끝내는 계획사이에 세워지는 것이며, 아마도 이것을 읽은 당신은 계획을 좀더 완벽하게 만들어 주는 정확하며 효율적인 스캔에대한 몇가지 기술을 배울 것이다.

1.1 호스트 발견이 무엇이냐??

내가 penetration test(PT)의 현재 현상을 설명할때 사용할 용어인 호스트 발견의 뜻은 네트워크를 통해 접근할 수 있는 호스트를 결정하기 위한 시도이다. 뚜렷하게 기술된 파이어월 ruleset 때문에, 매번 그 뒤에 있는 호스트s의 숫자를 정확하게 알기란 쉬운 일은 아니다. 근래 class C를 스캔해야할 동료의 심정은 매우 비싼 상용 스캐너를 돌리고 싶어 한다는 것이었고, 결국 그 비싼 스캐너를 돌려봤지만 호스트 하나밖에 못찾았다. 이 문서에 그려진 기법을 사용하여, 나는 하나 뿐 아니라 17개 호스트s까지 찾을 수 있었다. 상용 스캐너는 호스트 발견에 대한 옵션이 매우 부족했으며, 발견 방식을 명확하게 조율하기 위한 구성이 허술했다. 이것은 nmap과 호스트 발견에 관한 문서이기 때문에. nmap이 어떻게 호스트를 발견하는지 알려줄 것이다. 그리고 penetration test(PT)나 취약성 평가 측면에서 발견을 향상 시키기위해한 nmap 옵션을 어떻게 사용해야 하는지 알게 될 것이다.

출발하기에 앞서, 밑에 잇는 nmap의 기본적인 명령을 분석 하겠다:

```
nmap -sS -O 172.26.1.0/29
```

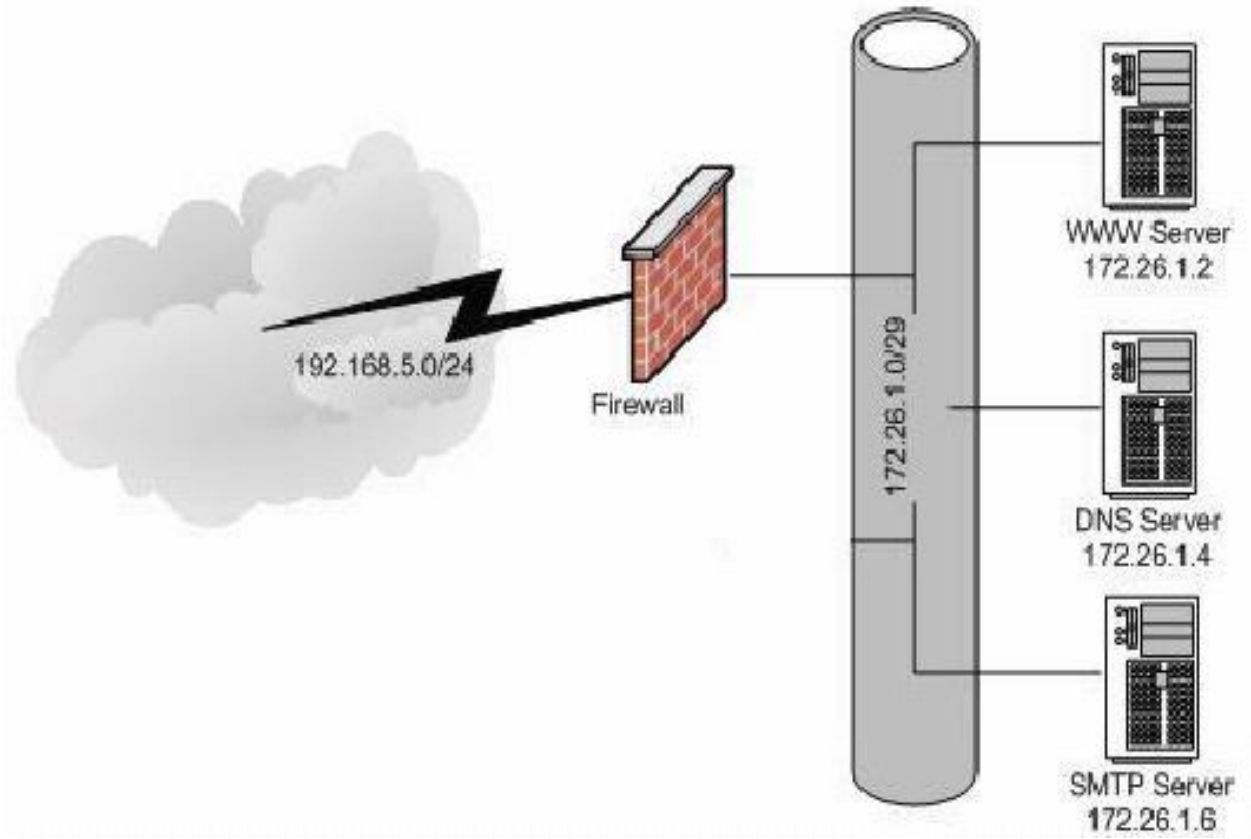
위 nmap 명령이 가지고 있는 3가지 다른 작업:

1) 호스트 발견
2) 포트 스캐닝
3) OS fingerprinting

이 문서는 호스트 발견에 촛점을 맞추고 있기 때문에, 나머지 두개는 제외 하고 첫번째 작업에 대해서 깊히 알아볼 것이다.

참고: 스캔에서 호스트 발견을 제외 할 수도 있고(-P0 옵션 사용으로), 그렇게 된다면 nmap은 곧바로 포트 스캔 작업으로 바로 바뀔 것이다.

파이어월 뒤에 숨어 있는 호스트를 찾는 최선의 방식을 논하기 위해 서로 다른 ruleset을 가진 파이어월의 DMZ환경을 이용할 것이다. DMZ 구조는 아래 그림에서 보여주고 있다.



이 그림은 안쪽으로 흐르는 트래픽을 필터링하는 파이어월이 가진 전형적인 DMZ의 모습을 보여준다. 이런 상황에서 그 룰을 읽게 쉽게한 pseudo-rulesets을 사용할 것이다. 사실 그 룰의 문법은 PF와 영어가 뒤섞여 있기 때문에, 보기에는 쉽다. - 그렇지만 단지 읽기 위해 필요할 뿐이다. nmap은 version 3.00을 사용할 것이다.

지금 192.168.5.0/24 네트웍의 호스트를 스캔할 것이며, 우리의 IP 주소는 192.168.5.20 이다.

별다른 점이 없다면, 모두 발견하기위한 nmap 스캔 명령은 아래와 같다:

```
nmap -sP 172.26.1.0/29
```

-sP 옵션은 실행되는 것만 발견하며, nmap의 기본 스캔 발견 방식과 동일하다.

2. nmap의 기본 방식 즐겨보기 :)

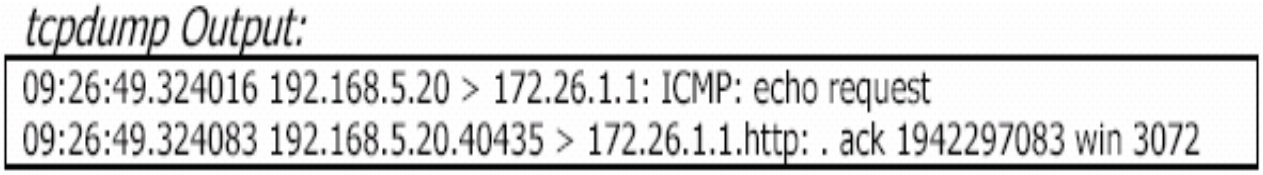
nmap의 행위를 수정하는 것이 어떻게 그리고 왜 필요한지 알아 보기 전에, 그것의 기본적 행위에 대한 이해와, 호스트 발견에 어떻게 부적절한가에 대한 이해에 충실해야 한다.

네트웍이나 호스트에 대해 포트 스캔 (nmap -sS target)이나 핑 sweep(nmap -sP target)이 실행될때, nmap은 그 스캔 범위에 안에 모든 목표를 향해서 ICMP echo request 패킷과 TCP pings을 일제히 보낸다. TCP ping은 목표 호스트의 80번 포트로 향하는 ACK flag가 설정된 TCP 패킷 이라 할 수 있다. 접근 가능한 호스트에서의 희망하는 답변은 그 호스트가 살아 있음을 증명하는 RST flag를 설정한 TCP 패킷과 ICMP echo reply중 하나이다. 대답이 없다면 그 호스트는 죽었거나, 파이어월에 의해 보호된 것이다. 그러므로 80 포트로 접근할 순 없다.

목표에 대한 포트 스캔이 그 호스트에 도달 했는지 아닌지 nmap이 결정할 뿐이다. nmap의 기본 발견 방식은 현재 사실을 중심으로 동작할 뿐이며, 접근 가능한 호스트를 결정하는 것은 틀릴 수도 있다. 불행히도 nmap은 매우 정직하여 우리의 명령으로 인해 발생된 결과로 상황을 보고할 뿐이다.

아래는 딱 한 호스트만 발견하기 위한 가장 기본적인 nmap 발견 명령이다:

```
Nmap -sP 172.26.1.1
```



위 그림으로 ICMP ping 과 TCP ping이 전달된다는 걸 알 수 있다.

2.1 상황 1 - Filtering 안하는 파이어월

처음 상황은 필터랑 안하는 파이어월에서 발견 과정의 진행 방식을 설명할 것이다. (예를 들자면, 그 파이어월은 라우터로써 아주 간단하게 동작한다.:))

명령:

```
nmap -sP 172.26.1.0/29
```

파이어월 Ruleset:

어디부터 어디로 오든 가든 다 통과 시킨다.

tcpdump Output:

```
08:59:58.840249 192.168.5.20 > 172.26.1.0: ICMP: echo request
08:59:58.840667 192.168.5.20.60923 > 172.26.1.0.http: . ack 2990889584 win 3072
08:59:58.840726 192.168.5.20 > 172.26.1.1: ICMP: echo request
08:59:58.840764 192.168.5.20.60923 > 172.26.1.1.http: . ack 1015938099 win 3072
08:59:58.840801 192.168.5.20 > 172.26.1.2: ICMP: echo request
08:59:58.840838 192.168.5.20.60923 > 172.26.1.2.http: . ack 1228729075 win 3072
08:59:58.840876 192.168.5.20 > 172.26.1.3: ICMP: echo request
08:59:58.840914 192.168.5.20.60923 > 172.26.1.3.http: . ack 1769982015 win 3072
08:59:58.840952 192.168.5.20 > 172.26.1.4: ICMP: echo request
08:59:58.840989 192.168.5.20.60923 > 172.26.1.4.http: . ack 1859940754 win 3072
08:59:58.841027 192.168.5.20 > 172.26.1.5: ICMP: echo request
08:59:58.841064 192.168.5.20.60923 > 172.26.1.5.http: . ack 1596045207 win 3072
08:59:58.841103 192.168.5.20 > 172.26.1.6: ICMP: echo request
08:59:58.841140 192.168.5.20.60923 > 172.26.1.6.http: . ack 550856434 win 3072
08:59:58.841178 192.168.5.20 > 172.26.1.7: ICMP: echo request
08:59:58.841215 192.168.5.20.60923 > 172.26.1.7.http: . ack 2476756145 win 3072
08:59:58.841886 172.26.1.2 > 192.168.5.20: ICMP: echo reply
08:59:58.842149 172.26.1.4 > 192.168.5.20: ICMP: echo reply
08:59:58.842377 172.26.1.2.http > 192.168.5.20.60923: R
1228729075:1228729075(0) win 0 (DF)
08:59:58.842699 192.168.5.5 > 192.168.5.20: ICMP: echo reply
08:59:58.842905 172.26.1.4.http > 192.168.5.20.60923: R
1859940754:1859940754(0) win 0 (DF)
08:59:58.843263 172.26.1.6 > 192.168.5.20: ICMP: echo reply
08:59:58.843487 172.26.1.6.http > 192.168.5.20.60923: R 550856434:550856434(0)
win 0 (DF)
```

결과:
호스트를 전부 찾았다 :)

tcpdump의 출력물로 (위에 보이죠:)) nmap이 어떻게 작업을 하는지 확실히 알 수 있다. ICMP와 TCP 패킷을 172.26.1.0/29 영역 내 모든 호스트에 보내고 있다. 그리고 답장이 오길 기다린다. 접근 가능한 호스트에서 보내온 답장은 노란 색으로 칠해놨다. (친절하구만..)

2.2 상황 2 - 기본적인 규칙을 가진 파이어월

이번 상황은 일반적인 ruleset을 가진 곳에서 기본 발견 방식이 어떻게 작동되는지 설명할 것이다.

명령:

```
nmap -sP 172.26.1.0/29
```

파이어월 Ruleset:
어디부터 어디로 오든 가든 80, 53, 25번 포트를 향하는 TCP 패킷만 통과된다.
그 외에껀 다 버린다.

tcpdump Output:

```
09:12:21.505016 192.168.5.20 > 172.26.1.0: ICMP: echo request
09:12:21.505125 192.168.5.20.60212 > 172.26.1.0.http: . ack 3755150488 win 3072
09:12:21.505166 192.168.5.20 > 172.26.1.1: ICMP: echo request
09:12:21.505204 192.168.5.20.60212 > 172.26.1.1.http: . ack 4073218537 win 3072
09:12:21.505242 192.168.5.20 > 172.26.1.2: ICMP: echo request
09:12:21.505280 192.168.5.20.60212 > 172.26.1.2.http: . ack 3464075465 win 3072
09:12:21.505318 192.168.5.20 > 172.26.1.3: ICMP: echo request
09:12:21.505355 192.168.5.20.60212 > 172.26.1.3.http: . ack 962650084 win 3072
09:12:21.505393 192.168.5.20 > 172.26.1.4: ICMP: echo request
09:12:21.505430 192.168.5.20.60212 > 172.26.1.4.http: . ack 337683576 win 3072
09:12:21.505468 192.168.5.20 > 172.26.1.5: ICMP: echo request
09:12:21.505505 192.168.5.20.60212 > 172.26.1.5.http: . ack 1839298263 win 3072
09:12:21.505544 192.168.5.20 > 172.26.1.6: ICMP: echo request
09:12:21.505581 192.168.5.20.60212 > 172.26.1.6.http: . ack 1701634905 win 3072
09:12:21.505619 192.168.5.20 > 172.26.1.7: ICMP: echo request
09:12:21.505656 192.168.5.20.60212 > 172.26.1.7.http: . ack 4287112447 win 3072
09:12:21.506577 172.26.1.2.http > 192.168.5.20.60212: R 3464075465:3464075465(0) win 0 (DF)
09:12:21.506830 172.26.1.6.http > 192.168.5.20.60212: R 1701634905:1701634905(0) win 0 (DF)
09:12:21.507104 172.26.1.4.http > 192.168.5.20.60212: R 337683576:337683576(0) win 0 (DF)
```

결과:
호스트를 전부 찾았다:)

이번 파이어월 ruleset은 깨물어 주고 싶을 정도로 서툴지만, 보기드문 경우는 아니다. 이건 stateful하지 않고, 또 TCP 핑도 아무 문제없이 통과 시킨다. 그렇지만 끝에 보이는 cleanup(그 외에껀 다 버린다.) 때문에 ICMP 패킷이 통과하진 못한다.

2.3 상황 3 - 특별한 규칙을 가진 파이어월

이번 상황은 조금 더 특별한 ruleset에서는 어떻게 기본 발견 방법이 동작하는지 설명한다.

명령:

```
nmap -sP 172.26.1.0/29
```

파이어월 Ruleset:

172.26.1.2으로 가는 tcp 패킷은 80번 포트로만 통과할 수 있다.
172.26.1.4으로 가는 tcp 패킷은 53번 포트로만 통과할 수 있다.
172.26.1.6으로 가는 tcp 패킷은 25번 포트로만 통과할 수 있다.
그 외에껀 다 버린다.

tcpdump Output:

```
08:05:07.733225 192.168.5.20 > 172.26.1.0: ICMP: echo request
08:05:07.733334 192.168.5.20.44273 > 172.26.1.0.http: . ack 1621467562 win 2048
08:05:07.733375 192.168.5.20 > 172.26.1.1: ICMP: echo request
08:05:07.733412 192.168.5.20.44273 > 172.26.1.1.http: . ack 1213996683 win 2048
08:05:07.733450 192.168.5.20 > 172.26.1.2: ICMP: echo request
08:05:07.733487 192.168.5.20.44273 > 172.26.1.2.http: . ack 3921129299 win 2048
08:05:07.733525 192.168.5.20 > 172.26.1.3: ICMP: echo request
08:05:07.733561 192.168.5.20.44273 > 172.26.1.3.http: . ack 193217699 win 2048
08:05:07.733598 192.168.5.20 > 172.26.1.4: ICMP: echo request
08:05:07.733635 192.168.5.20.44273 > 172.26.1.4.http: . ack 3130918107 win 2048
08:05:07.733672 192.168.5.20 > 172.26.1.5: ICMP: echo request
08:05:07.733709 192.168.5.20.44273 > 172.26.1.5.http: . ack 638273071 win 2048
08:05:07.733746 192.168.5.20 > 172.26.1.6: ICMP: echo request
08:05:07.733783 192.168.5.20.44273 > 172.26.1.6.http: . ack 4151673259 win 2048
08:05:07.733820 192.168.5.20 > 172.26.1.7: ICMP: echo request
08:05:07.733856 192.168.5.20.44273 > 172.26.1.7.http: . ack 228721671 win 2048
08:05:07.734611 172.26.1.2.http > 192.168.5.20.44273: R 3921129299:3921129299(0) win 0 (DF)
```

결과:
호스트 딱 하나밖에 못 찾았다.
지금 필터링 규칙은 특별하기에, nmap 기본 발견 방법으론 충분하지 않다.
도착지의 WWW 서버로 향하는 TCP 핑만 그 파이어월을 통과했을 뿐이다.

2.4 상황 4 - 특별한 규칙을 가진 Stateful 파이어월

이번 상황에서 파이어월은 stateful inspection하게 동작한다. 파이어월 ruleset 또한 매우 특별하다.

명령:

```
nmap -sP 172.26.1.0/29
```

파이어월 Ruleset:

state를 지키면서 172.26.1.2의 80번 포트로 가는 tcp 패킷은 통과
state를 지키면서 172.26.1.4의 53번 포트로 가는 tcp 패킷은 통과
state를 지키면서 172.26.1.6의 25번 포트로 가는 tcp 패킷은 통과
그 외에껀 다 지운다.

tcpdump Output:

```
08:46:23.548456 192.168.5.20.44390 > 172.26.1.2.http: . ack 3476163011 win 2048
08:46:23.548468 192.168.5.20 > 172.26.1.3: ICMP: echo request
08:46:23.548501 192.168.5.20.44390 > 172.26.1.3.http: . ack 1149703540 win 2048
08:46:23.548559 192.168.5.20 > 172.26.1.4: ICMP: echo request
08:46:23.548596 192.168.5.20.44390 > 172.26.1.4.http: . ack 1314586500 win 2048
08:46:23.548635 192.168.5.20 > 172.26.1.5: ICMP: echo request
08:46:23.548673 192.168.5.20.44390 > 172.26.1.5.http: . ack 2068473993 win 2048
08:46:23.548712 192.168.5.20 > 172.26.1.6: ICMP: echo request
08:46:23.548749 192.168.5.20.44390 > 172.26.1.6.http: . ack 2732407633 win 2048
08:46:23.548789 192.168.5.20 > 172.26.1.7: ICMP: echo request
08:46:23.548825 192.168.5.20.44390 > 172.26.1.7.http: . ack 2518875875 win 2048
```

결과:
호스트 하나도 못찾았다. ㅠ.-

위 결과에서 우리는 TCP 핑, ICMP 패킷 모두 통과 못했다는 걸 알 수 있다. "cleanup" 규칙(그 외에껀 다 버린다.)이 ICMP를 버렸고, TCP 핑은 파이어월이 버렸다. 왜냐하면 이전에 성립된 연결이 아니기 때문이다.

이런 경우의 파이어월에서 nmap의 기본 발견 방식이 충분하지 못한 이유를 지금 정확히 해둬야 한다.

3. nmap의 발견 옵션 이해하기

바로 전에 nmap의 기본 발견 옵션이 4개중 2개의 상황에서 충분하지 못하단 걸 보여줬다. nmap으로 호스트를 발견하는 것에 확신을 갖기 위해, 몇 가지 발전된 옵션의 사용법을 배워야만 한다.

3.1 Customizing TCP Pings

기본적인 TCP 핑은 80번 포트(WWW)로 보내며 ACK flag를 설정한 TCP 패킷형태이다. 위에서 전달된 패킷은, stateful 파이어이 전에 성립된 연결의 한 부분인지 아닌지 확인하기 위해 state 테이블에서 찾아볼 것이다. 이 패킷은 바로 악당이라고 알려지며, 즉시 폐기되므로 호스트 발견을 예방할 수 있다. stateful inspection하게 작동하며 특별한 ruleset을 가진 파이어월은 오직 인증된 호스트로 TCP 핑을 통과 시킬 것이다. 그러므로 customize TCP 핑이 필요하다.

Nmap에서 TCP 핑내부의 대부분 옵션을 customize할 수 있다. 첫째로, ACK flag대신 SYN flag를 설정할 수 있다. 이건 -PS 옵션으로 아주 쉽게 할 수 있다. 완전한 명령은 아래와 같다:

```
nmap -sP -PS 172.26.1.2
```

tcpdump Output:

```
10:48:13.656653 192.168.5.20.50992 > 172.26.1.2.http: S 3312451587:3312451587(0) win 2048
```

위 명령에서도 기본 포트는 80번 이지만, SYN flag가 설정되어 있기에 (그것을 규칙으로 금지하지 않는다면) stateful 파이어월을 통과 할 것이다. -PS옵션에 목적지 포트를 추가하여, 발견을 더욱 customize할 수 있다. customized nmap 발견 명령은 적어도 아래와 비슷할 것이다:

nmap -sP -PS25 172.26.1.2

tcpdump Output:

```
10:49:50.436438 192.168.5.20.63376 > 172.26.1.2.smtp: S 948961283:948961283(0) win 4096
```

이 명령은 기본 80번 포트 대신 25번 포트로 (SYN flag가 설정된) TCP 핑을 보낸다.

또 다른 옵션 중 하나는 customizing TCP 핑의 특정 송신 포트를 설정할때 고려해 보아야 한다. 이 옵션은 반드시 항상 동작하는 것은 아니며, 확실히 나쁜 시도이다. 매우 허접하게 쓰여진 파이어월 규칙에선 동작하지만, 아래 말로된 ruleset에선 생각해 보아야 한다.

state를 지키면서 172.26.1.2의 80번 포트로 가는 tcp 패킷은 통과
state를 지키면서 172.26.1.4의 53번 포트로 가는 tcp 패킷은 통과
state를 지키면서 172.26.1.6의 25번 포트로 가는 tcp 패킷은 통과
그 외에껀 다 버린다.

명령:

nmap -sP 172.26.1.0/29 -g 53

tcpdump Output:

```
10:52:02.083065 192.168.5.20 > 172.26.1.0: ICMP: echo request
10:52:02.083260 192.168.5.20.domain > 172.26.1.0.http: . ack 2177885259 win 3072
10:52:02.083301 192.168.5.20 > 172.26.1.1: ICMP: echo request
10:52:02.083346 192.168.5.20.domain > 172.26.1.1.http: . ack 2684323392 win 3072
10:52:02.083384 192.168.5.20 > 172.26.1.2: ICMP: echo request
10:52:02.083421 192.168.5.20.domain > 172.26.1.2.http: . ack 1438652920 win 3072
10:52:02.083459 192.168.5.20 > 172.26.1.3: ICMP: echo request
10:52:02.083496 192.168.5.20.domain > 172.26.1.3.http: . ack 771338950 win 3072
10:52:02.083534 192.168.5.20 > 172.26.1.4: ICMP: echo request
10:52:02.083570 192.168.5.20.domain > 172.26.1.4.http: . ack 3541039396 win 3072
10:52:02.083608 192.168.5.20 > 172.26.1.5: ICMP: echo request
10:52:02.083645 192.168.5.20.domain > 172.26.1.5.http: . ack 2586779353 win 3072
10:52:02.083683 192.168.5.20 > 172.26.1.6: ICMP: echo request
10:52:02.083719 192.168.5.20.domain > 172.26.1.6.http: . ack 45434507 win 3072
10:52:02.083757 192.168.5.20 > 172.26.1.7: ICMP: echo request
10:52:02.083794 192.168.5.20.domain > 172.26.1.7.http: . ack 1886752887 win 3072
10:52:02.084616 172.26.1.2.http > 192.168.5.20.domain: R 1438652920:1438652920(0) win 0 (DF)
10:52:02.084845 172.26.1.4.http > 192.168.5.20.domain: R 3541039396:3541039396(0) win 0 (DF)
10:52:02.085219 172.26.1.6.http > 192.168.5.20.domain: R 45434507:45434507(0) win 0 (DF)
```

특별한 규칙을 가진 stateful 파이어월에서 DMZ안 어느 호스트로의 DNS 트래픽을 허용하는 것은 꽤 안전해 보인다. 위 tcpdump의 출력물이 보여주듯, 이 규칙은 DNS 트래픽뿐만 아니라, DMZ를 스캔하려고 보내는 TCP 패킷 또한 허용한다. 파이어월 ruleset안 논리적 오류와 애매한 규칙은 보기 드물지 않으며, 호스트 발견에서 매우 유용한 것임을 알 수 있다.

3.2 Customizing ICMP Messages

파이어월 대부분이 ICMP echo request 메세지를 버리긴 하지만, 다른 형태의 ICMP 트래픽은 방해받지 않고 통과될 수 있다. 게다가 nmap의 지금 버전은 2가지 다른 형태의 ICMP 메세지를 보낼 수 있다.

nmap에서 --PP 옵션의 사용으로 ICMP timestamp request (type 13)을 보낼 수 있으며, 답장으론 ICMP timestamp replies (type 14)을 받는다. 그 호스트가 살아 있다고 가정하고, 만약 살아 있다면 받는 패킷이 14번 형태의 ICMP 패킷이다.

명령:

```
nmap -sP -PP 172.26.1.4
```

tcpdump Output:

```
13:32:05.780376 192.168.5.20 > 172.26.1.4: icmp: time stamp query id 47345 seq 0 (DF)
13:32:05.781066 172.26.1.4 > 192.168.5.20: icmp: time stamp reply id 47345 seq 0 : org 0x0
recv 0x3c339bd xmit 0x3c339bd
```

ICMP address mask (netmask) requests (type 17)을 보낼려면 -PM 옵션을 써야하며 답장으론 ICMP address mask reply (type 18)을 받는다. 다시 한번 말하지만 18번 형태의 패킷을 받는다면, 그 호스트는 살아 있는 것이다.

명령:

```
nmap -sP -PM 172.26.1.4
```

tcpdump Output:

```
13:37:11.452204 192.168.5.20 > 172.26.1.4: icmp: address mask request (DF)
```

참고: 뭐 파이어월이 ICMP 패킷을 통과 시킨다 해도, 대부분의 운영체제는 그 요구에 받아들일 것이고, 그 다음 조용히 그 패킷을 제거할 것이다. 일반적인 라우터는 netmask request에 응할 것이다.

3.3 Bringing It All Together

지금까지 우리는 nmap 발견 작업이 어떻게 이루어 지는지, 그리고 customize의 방법에 대해, 또 나머지 nmap의 특성도 알아 보았다. 이제 마지막 상황에 대해 집중하기로 하자. 위에서 보여준 "좀 나쁜 경우"의 4가지 상황에서, DMZ의 모든 호스트를 찾기 위한 방법을 샅샅히 알아볼 것이다.

파이어월 Ruleset

state를 지키면서 172.26.1.2의 80번 포트로 가는 tcp 패킷은 통과
state를 지키면서 172.26.1.4의 53번 포트로 가는 tcp 패킷은 통과
state를 지키면서 172.26.1.6의 25번 포트로 가는 tcp 패킷은 통과
그 외에껀 다 버린다.

명령:

```
nmap -sP -PS80 172.26.1.0/29
```

tcpdump Output:

```
11:03:11.198359 192.168.5.20.49989 > 172.26.1.0.http: S 3857711107:3857711107(0) win 1024
11:03:11.198465 192.168.5.20.49989 > 172.26.1.1.http: S 3508535339:3508535339(0) win 1024
11:03:11.198506 192.168.5.20.49989 > 172.26.1.2.http: S 1017118803:1017118803(0) win 1024
11:03:11.198544 192.168.5.20.49989 > 172.26.1.3.http: S 832045179:832045179(0) win 1024
11:03:11.198582 192.168.5.20.49989 > 172.26.1.4.http: S 2873622691:2873622691(0) win 1024
11:03:11.198620 192.168.5.20.49989 > 172.26.1.5.http: S 1101529291:1101529291(0) win 1024
11:03:11.198658 192.168.5.20.49989 > 172.26.1.6.http: S 99614963:99614963(0) win 1024
11:03:11.198696 192.168.5.20.49989 > 172.26.1.7.http: S 3741843739:3741843739(0) win 1024
11:03:11.199453 172.26.1.2.http > 192.168.5.20.49989: S 92167158:92167158(0) ack
1017118804 win 5840 <mss 1460> (DF)
```

예상했겠지만, 웹 서버 (172.26.1.2) 만 답장을 주었다. TCP 핑의 소스 포트를 25로 바꿀 것이다.

명령:

nmap -sP -PS25 172.26.1.0/29

tcpdump Output:

```
11:05:06.000617 192.168.5.20.38849 > 172.26.1.0.smtp: S 3544186883:3544186883(0) win 2048
11:05:06.000720 192.168.5.20.38849 > 172.26.1.1.smtp: S 4056940587:4056940587(0) win 2048
11:05:06.000759 192.168.5.20.38849 > 172.26.1.2.smtp: S 3272605779:3272605779(0) win 2048
11:05:06.000797 192.168.5.20.38849 > 172.26.1.3.smtp: S 4168614011:4168614011(0) win 2048
11:05:06.000835 192.168.5.20.38849 > 172.26.1.4.smtp: S 586154147:586154147(0) win 2048
11:05:06.000872 192.168.5.20.38849 > 172.26.1.5.smtp: S 3571974347:3571974347(0) win 2048
11:05:06.000910 192.168.5.20.38849 > 172.26.1.6.smtp: S 667418867:667418867(0) win 2048
11:05:06.000948 192.168.5.20.38849 > 172.26.1.7.smtp: S 902824219:902824219(0) win 2048
11:05:06.001909 172.26.1.6.smtp > 192.168.5.20.38849: S 203047548:203047548(0) ack
667418868 win 5840 <mss 1460> (DF)
```

SMTP 서버 (172.26.1.6)가 답장을 주었다. 또 다시, DNS 서버를 조사하기 위해 수신측 TCP 포트를 53번으로 바꾼다.

명령:

nmap -sP -PS53 172.26.1.0/29

tcpdump Output:

```
11:06:52.601522 192.168.5.20.51592 > 172.26.1.0.domain: S 2862088195:2862088195(0) win 4096
11:06:52.601623 192.168.5.20.51592 > 172.26.1.1.domain: S 3921149995:3921149995(0) win 4096
11:06:52.601662 192.168.5.20.51592 > 172.26.1.2.domain: S 3150970963:3150970963(0) win 4096
11:06:52.601699 192.168.5.20.51592 > 172.26.1.3.domain: S 4262985851:4262985851(0) win 4096
11:06:52.601736 192.168.5.20.51592 > 172.26.1.4.domain: S 3247440035:3247440035(0) win 4096
11:06:52.601773 192.168.5.20.51592 > 172.26.1.5.domain: S 1634730187:1634730187(0) win 4096
11:06:52.601811 192.168.5.20.51592 > 172.26.1.6.domain: S 947388659:947388659(0) win 4096
11:06:52.601848 192.168.5.20.51592 > 172.26.1.7.domain: S 2042102043:2042102043(0) win 4096
11:06:52.602957 172.26.1.4.domain > 192.168.5.20.51592: S 328239288:328239288(0) ack
3247440036 win 5840 <mss 1460> (DF)
```

DNS 서버 (172.26.1.4)가 답장을 주었기 때문에, 모든 호스트가 사로잡혔다. 하지만 잠시

만 기다려 보자. DMZ안 서비스와 호스트를 알고 있다면, TCP 핑을 어느 포트로 보내야 할지 알 수 있다. 외부 평가를 수행할 때, 고객이 우리에게 그 시스템의 정보와 제공되는 서비스를 말해주는 것은 매우 가망 없는 이야기다. 그러한 정보를 알아 내는 것이 어렵긴 하지만, 호스트를 발견하는 것은 가능한 이야기다.

위에서 처럼 다른 목적지와 소스 포트를 가진 TCP 핑 sweep을 사용하여 파이어월과 부딪히며 성공적인 발견을 해야만 한다. 목적지 포트중 대부분은 인터넷 서비스부터 택하는 것이 좋은 시작이다. 소스 포트로 좋은 것은 20, 53 번이 있다.(왜 좋을까요 ?) 스크립트는 이러한 과정을 자동화 하고 데이터를 분석할 수 있다. 위 개념을 증명하기 펄 스크립트를 만들어 보았다. :) http://www.moonpie.org/tools/discover.tgz

4. 결론

이 문서로 인해 유용한 정보가 제공되고, 공부를 하게 하면서, 호스트를 찾는 방법론에 있어 좀더 면밀히 검토하는 어떠한 기회가 되었음 좋겠다. 인터넷으로 접근 가능한 호스트를 숨길 수 있는 확실한 규칙을 어떻게 만들어야 하는지, 이 문서가 파이어월 관리자를 위한 안내서가 되었음 좋겠다.

너무 나도 강력한 open source 포트 스캐너, nmap만드는데 노력해주신 Fyodor 님께 고맙단 인사를 드리고 싶습니다. nmap에 관한 정보가 더 필요하시다면 http://www.nmap.org 와 http://www.insecure.org 로 가보시기 바랍니다. 감사합니다.