

『개발 보고서』

**IDS / IPS**

**2020. 08. 26.**

## 제·개정 이력

[illegible]

## - 목차 -

- 1. 개요.. 5
- 2. 구조.. 5
- 3. 사용법.. 6
  - 3.1. 프로그램 실행.. 6
    - 3.1.1. 수리카타 실행.. 6
- 4. 공격 목록.. 6
  - 4.1. TCP.. 6
    - 4.1.1. TCP SYN Flooding Attack.. 6
    - 4.1.2. TCP NULL Flooding Attack.. 6
    - 4.1.3. TCP ACK Flooding Attack.. 7
    - 4.1.4. TCP FIN Flooding Attack.. 8
    - 4.1.5. TCP PUSH Flooding Attack.. 8
    - 4.1.6. TCP URG Flooding Attack.. 9
    - 4.1.7. TCP RST Flooding Attack.. 10
    - 4.1.8. TCP X-mas Scan.. 10
    - 4.1.9. TCP NULL Scan.. 11
    - 4.1.10. TCP FIN Scan.. 11
    - 4.1.11. TCP Half(SYN) Open Scan.. 12
  - 4.2. UDP.. 12
    - 4.2.1. DNS Query Flooding.. 12
    - 4.2.2. UDP Port Scan.. 12
    - 4.2.3. NTF Reflection.. 13
  - 4.3. HTTP.. 13

4.3.1. HTTP GET Flooding.. 13

4.3.2. Slow HTTP Header Dos.. 14

4.3.3. XSS Attack.. 14

4.3.4. SQL Injection.. 15

4.4. ICMP.. 15

4.4.1. Ping of Death.. 15

4.4.2. ICMP Flooding.. 16

4.4.3. Land Attack.. 17

## 1. 개요

1) 목적: IDS/IPS 구현

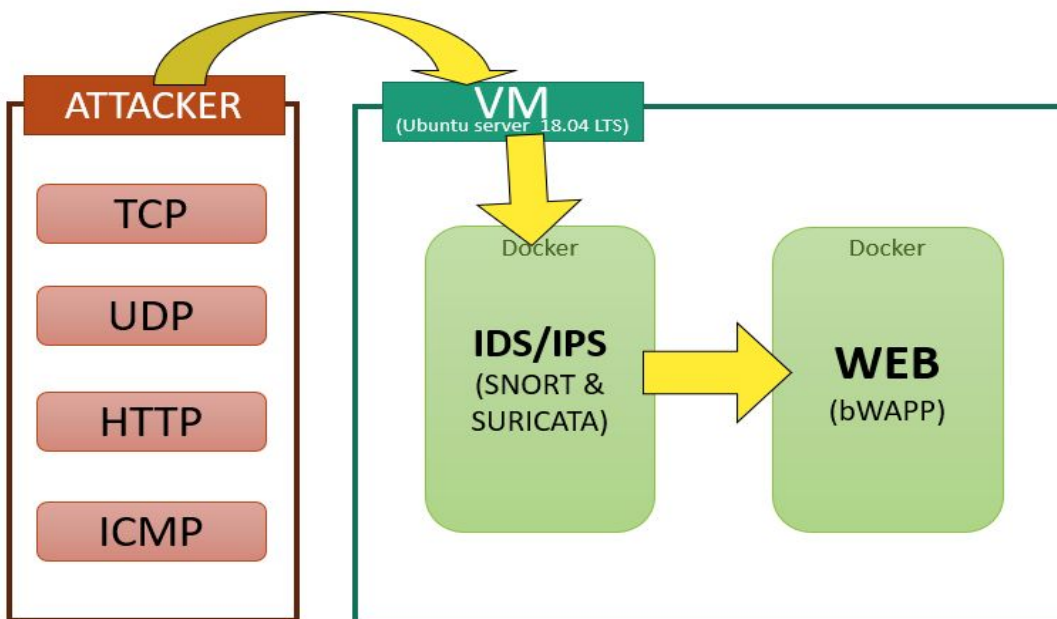
2) 개발환경

- Linux(Ubuntu 18.04 LTS)
- Virtual Machine(VMware / Virtual Box)
- Docker
- Suricata 5.0.3

3) 네트워크 환경 : NAT, Bridge

4) 구성 : VM ~ IDS/IPS(GUEST) ~ bWAPP(Docker)

## 2. 구조



[그림 1] 프로그램 구조

### 3. 사용법

#### 3.1. 프로그램 실행

##### 3.1.1. 수리카타 실행

- 명령어 : `suricata -c suricata.yaml -i eth0`

(suricata.yaml이 있는 폴더에서 실행 또는, 경로지정을 해야한다.)

### 4. 공격 목록

#### 4.1. TCP

##### 4.1.1. TCP SYN Flooding

###### 1) 공격분석

| No. | Time         | Source          | Destination     | Protocol | Length | Info  |
|-----|--------------|-----------------|-----------------|----------|--------|---|
| 1   | 0.0000000... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1133 → 0 [SYN] Seq=0 Win=512 Len=0          |
| 2   | 0.0001189... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 0 → 1133 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 3   | 0.0002553... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1134 → 0 [SYN] Seq=0 Win=512 Len=0          |
| 4   | 0.0002554... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1135 → 0 [SYN] Seq=0 Win=512 Len=0          |
| 5   | 0.0003065... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 0 → 1134 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 6   | 0.0003911... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 0 → 1135 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 7   | 0.0005225... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1136 → 0 [SYN] Seq=0 Win=512 Len=0          |
| 8   | 0.0005226... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1137 → 0 [SYN] Seq=0 Win=512 Len=0          |
| 9   | 0.0005227... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1138 → 0 [SYN] Seq=0 Win=512 Len=0          |
| 10  | 0.0005724... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 0 → 1136 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 11  | 0.0006330... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 0 → 1137 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 12  | 0.0006830... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 0 → 1138 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |

▶ Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0  
▶ Ethernet II, Src: VMware\_e3:00:19 (00:0c:29:e3:00:19), Dst: VMware\_26:13:d6 (00:0c:29:26:13:d6)  
▶ Internet Protocol Version 4, Src: 192.168.200.157, Dst: 192.168.200.128  
▶ Transmission Control Protocol, Src Port: 1135, Dst Port: 0, Seq: 0, Len: 0

|      |                   |                      |             |                 |
|------|-------------------|----------------------|-------------|-----------------|
| 0000 | 00 0c 29 26 13 d6 | 00 0c 29 e3 00 19    | 08 00 45 00 | ..)&..).-...E-  |
| 0010 | 00 28 a1 96 00 00 | 00 40 06 c6 ca c0 a8 | c8 9d c0 a8 | .(....@. ....   |
| 0020 | c8 80 04 6f 00 00 | 00 1e 9e ca 3e 0f 7c | b7 82 50 02 | ...o....>. ..P. |
| 0030 | 02 00 e7 28 00 00 | 00 00 00 00          |             | ...{(.... ....  |

- TCP SYN 패킷 다수가 감지된다.

###### 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg: "Warning! TCP SYN Flooding Detection"; flags: S; threshold:type threshold, track by\_src, count 20, seconds 1; sid:1000001;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg: "TCP SYN Flooding Detection"; flags: S; threshold:type threshold, track by\_src, count 40, seconds 1; sid:1000001;)

##### 4.1.2. TCP NULL Flooding

###### 1) 공격분석

| No.   | Time                    | Source                        | Destination     | Protocol | Length | Info |
|---|-------------------------|-------------------------------|-----------------|----------|--------|------|
| 172...  | 3.3129742...            | 192.168.200.157               | 192.168.200.128 | IPv4     | 60     |      |
| 172...  | 3.3129743...            | 192.168.200.157               | 192.168.200.128 | IPv4     | 60     |      |
| 172...  | 3.3130200...            | 192.168.200.157               | 192.168.200.128 | IPv4     | 60     |      |
| 172...  | 3.3130200...            | 192.168.200.157               | 192.168.200.128 | IPv4     | 60     |      |
| 172...  | 3.3130315...            | 192.168.200.157               | 192.168.200.128 | IPv4     | 60     |      |
| 172...  | 3.3130315...            | 192.168.200.157               | 192.168.200.128 | IPv4     | 60     |      |
| 172...  | 3.3130725...            | 192.168.200.157               | 192.168.200.128 | IPv4     | 60     |      |
| Identification: 0x699b (27035)                |                         |                               |                 |          |        |      |
| Flags: 0x0000                                 |                         |                               |                 |          |        |      |
| Fragment offset: 0                            |                         |                               |                 |          |        |      |
| Time to live: 64                              |                         |                               |                 |          |        |      |
| Protocol: TCP (6)                             |                         |                               |                 |          |        |      |
| Header checksum: 0xfed9 [validation disabled] |                         |                               |                 |          |        |      |
| 0000  | 00 0c 29 26 13 d6       | 00 0c 29 e3 00 19 08 00 45 00 | ..)&... ).....E |          |        |      |
| 0010  | 00 14 69 9b 00 00 40 06 | fe d9 c0 a8 c8 9d c0 a8       | ..i...@.....    |          |        |      |
| 0020  | c8 80 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00       | .....           |          |        |      |

- TCP 프로토콜을 이용하여 공격자인 클라이언트가 서버에 TCP 헤더의 Flags를 NULL(0x00)로 세팅하여 대량의 패킷을 보내 서버의 자원을 소모시키게 된다.

## 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Warning! NULL Flooding Detection"; flags:0; threshold:type threshold, track by\_src, count 10, seconds 1; sid:1000002;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"NULL Flooding Detection"; flags:0; threshold:type threshold, track by\_src, count 20, seconds 1; sid:1000002;)

## 4.1.3. TCP ACK Flooding

### 1) 공격분석

| No.  | Time                    | Source                        | Destination       | Protocol | Length | Info  |
|--|-------------------------|-------------------------------|-------------------|----------|--------|---|
| 1  | 0.0000000...            | 21.208.43.255                 | 192.168.200.128   | TCP      | 60     | 2784 → 80 [ACK] Seq=1 Ack=1 Win=512 Len=0     |
| 2  | 0.0000428...            | 192.168.200.128               | 21.208.43.255     | TCP      | 54     | 80 → 2784 [RST] Seq=1 Win=0 Len=0             |
| 3  | 0.6144832...            | 192.168.200.128               | 251.154.78.32     | TCP      | 58     | 80 → 1842 [SYN, ACK] Seq=0 Ack=1 Win=64240... |
| 4  | 1.0009817...            | 162.113.215.208               | 192.168.200.128   | TCP      | 60     | 2785 → 80 [ACK] Seq=1 Ack=1 Win=512 Len=0     |
| 5  | 1.0010514...            | 192.168.200.128               | 162.113.215.208   | TCP      | 54     | 80 → 2785 [RST] Seq=1 Win=0 Len=0             |
| 6  | 1.6387630...            | 192.168.200.128               | 244.164.96.222    | TCP      | 58     | 80 → 1843 [SYN, ACK] Seq=0 Ack=1 Win=64240... |
| 7  | 2.0019447...            | 254.44.217.20                 | 192.168.200.128   | TCP      | 60     | 2786 → 80 [ACK] Seq=1 Ack=1 Win=512 Len=0     |
| 8  | 2.0019943...            | 192.168.200.128               | 254.44.217.20     | TCP      | 54     | 80 → 2786 [RST] Seq=1 Win=0 Len=0             |
| 9  | 3.0028114...            | 17.110.17.0                   | 192.168.200.128   | TCP      | 60     | 2787 → 80 [ACK] Seq=1 Ack=1 Win=512 Len=0     |
| 10   | 3.0028894...            | 192.168.200.128               | 17.110.17.0       | TCP      | 54     | 80 → 2787 [RST] Seq=1 Win=0 Len=0             |
| Acknowledgment number: 1 (relative ack number) |                         |                               |                   |          |        |   |
| Acknowledgment number (raw): 2113562086        |                         |                               |                   |          |        |   |
| 0101 ... = Header Length: 20 bytes (5)         |                         |                               |                   |          |        |   |
| Flags: 0x010 (ACK)                             |                         |                               |                   |          |        |   |
| Window size value: 512                         |                         |                               |                   |          |        |   |
| [Calculated window size: 512]                  |                         |                               |                   |          |        |   |
| [Window size scaling factor: -1 (unknown)]     |                         |                               |                   |          |        |   |
| Checksum: 0xcea2 [unverified]                  |                         |                               |                   |          |        |   |
| [Checksum status: Unverified]                  |                         |                               |                   |          |        |   |
| 0000   | 00 0c 29 26 13 d6       | 00 0c 29 e3 00 19 08 00 45 00 | ..)&... ).....E   |          |        |   |
| 0010   | 00 28 43 a7 00 00 40 06 | 33 be a2 71 d7 d0 c0 a8       | .(C...@. 3..q.... |          |        |   |
| 0020   | c8 80 0a e1 00 50 0b 9b | e1 19 7d fa 65 e6 50 10       | .....P...}.e.P    |          |        |   |

- TCP ACK Flooding은 공격자가 서버에 TCP 헤더의 플래그를 ACK(0x10)으로 설정한 후 많은 패킷을 전송한다.

- 이를 통해 서버가 정상적인 서비스를 못하게 하거나, 지연시키는 공격 방법이다.

## 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Warning! ACK Flooding Detection"; flags:A; threshold:type threshold, track by\_src, count 10, seconds 1; sid:1000003;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"ACK Flooding Detection"; flags:A; threshold:type threshold, track by\_src, count 20, seconds 1; sid:1000003;)

### 4.1.4. TCP FIN Flooding

#### 1) 공격분석

| Time   | Source       | Destination     | Protocol        | Length | Info                     |
|--------|--------------|-----------------|-----------------|--------|--------------------------|
| 135... | 6.8763273... | 192.168.200.157 | 192.168.200.128 | TCP    | 60 65324 → 80 [FIN] S... |
| 135... | 6.8764208... | 192.168.200.128 | 192.168.200.157 | TCP    | 54 80 → 65324 [RST, A... |
| 135... | 6.8763273... | 192.168.200.157 | 192.168.200.128 | TCP    | 60 65325 → 80 [FIN] S... |
| 135... | 6.8764324... | 192.168.200.128 | 192.168.200.157 | TCP    | 54 80 → 65325 [RST, A... |
| 135... | 6.8763274... | 192.168.200.157 | 192.168.200.128 | TCP    | 60 65326 → 80 [FIN] S... |
| 135... | 6.8764714... | 192.168.200.128 | 192.168.200.157 | TCP    | 54 80 → 65326 [RST, A... |
| 135... | 6.8763274... | 192.168.200.157 | 192.168.200.128 | TCP    | 60 [TCP Previous segm... |
| 135... | 6.8764840... | 192.168.200.128 | 192.168.200.157 | TCP    | 54 [TCP ACKed unseen ... |
| 135... | 6.8763274... | 192.168.200.157 | 192.168.200.128 | TCP    | 60 [TCP Retransmissio... |

|  |
|--|
| ▶ Acknowledgment number: 2052886612        |
| Acknowledgment number (raw): 2052886612    |
| 0101 .... = Header Length: 20 bytes (5)    |
| ▶ <b>Flags: 0x001 (FIN)</b>                |
| Window size value: 512                     |
| [Calculated window size: 512]              |
| [Window size scaling factor: -1 (unknown)] |

|      |          |                |                         |                             |
|------|----------|----------------|-------------------------|-----------------------------|
| 0000 | 00 0c 29 | 26 13 d6 00 0c | 29 e3 00 19 08 00 45 00 | .. )& . . . . ) . . . . E . |
| 0010 | 00 28 2c | 70 00 00 40 06 | 2b c8 00 c8 c8 0d 00 c8 | 2u 0                        |

- TCP FIN Flooding은 DOS 공격의 한 방법으로, TCP 프로토콜의 특성을 이용하여 플래그를 FIN(0x01)으로 조작한 패킷을 다수 전송하여 공격한다.

## 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Warning! FIN Flooding Detection"; flags:F; threshold:type threshold, track by\_src, count 10, seconds 1; sid:1000004;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"FIN Flooding Detection"; flags:F; threshold:type threshold, track by\_src, count 20, seconds 1; sid:1000004;)

### 4.1.5. TCP PUSH Flooding

#### 1) 공격분석



|        |              |                 |                 |     |                            |
|--------|--------------|-----------------|-----------------|-----|----------------------------|
| 458... | 12.769352... | 192.168.200.157 | 192.168.200.128 | TCP | 60 8507 → 80 [PSH] Seq=... |
| 458... | 12.769359... | 192.168.200.128 | 192.168.200.157 | TCP | 54 80 → 8507 [RST, AC...]  |
| 458... | 12.769352... | 192.168.200.157 | 192.168.200.128 | TCP | 60 8508 → 80 [PSH] Seq=... |
| 458... | 12.769386... | 192.168.200.128 | 192.168.200.157 | TCP | 54 80 → 8508 [RST, AC...]  |
| 458... | 12.769352... | 192.168.200.157 | 192.168.200.128 | TCP | 60 8509 → 80 [PSH] Seq=... |
| 458... | 12.769394... | 192.168.200.128 | 192.168.200.157 | TCP | 54 80 → 8509 [RST, AC...]  |

Acknowledgment number (raw): 1624367468  
 0101 .... = Header Length: 20 bytes (5)  
 ▶ Flags: 0x008 (PSH)  
 window size value: 512  
 [Calculated window size: 512]  
 [Window size scaling factor: -1 (unknown)]

|      |   |                 |
|------|---|-----------------|
| 0000 | 00 0c 29 26 13 d6 00 0c 29 e3 00 19 08 00 45 00 | ..)&... )....E. |
| 0010 | 00 28 31 a7 00 00 40 06 36 ba c0 a8 c8 9d c0 a8 | .(1...@. 6..... |

- TCP PUSH Flooding 공격은 TCP 헤더의 플래그를 PUSH(0x08)로 하여 서버의 리소스를 소모시키는 공격 방법이다.

## 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Warning! PUSH Flooding Detection"; flags:P; threshold:type threshold, track by\_src, count 10, seconds 1; sid:1000005;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"PUSH Flooding Detection"; flags:P; threshold:type threshold, track by\_src, count 20, seconds 1; sid:1000005;)

## 4.1.6. TCP URG Flooding

### 1) 공격분석

| No.    | Time         | Source          | Destination     | Protocol | Length | Info                     |
|--------|--------------|-----------------|-----------------|----------|--------|--------------------------|
| 432... | 6.8241526... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 48385 → 80 [URG] Seq=... |
| 432... | 6.8242333... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 80 → 48385 [RST, ACK]... |
| 432... | 6.8241527... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 48386 → 80 [URG] Seq=... |
| 432... | 6.8242378... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 80 → 48386 [RST, ACK]... |
| 432... | 6.8241527... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 48387 → 80 [URG] Seq=... |
| 432... | 6.8242584... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 80 → 48387 [RST, ACK]... |

[Next sequence number: 2988904750 (relative sequence number)]  
 ▶ Acknowledgment number: 1473677036  
 Acknowledgment number (raw): 1473677036  
 0101 .... = Header Length: 20 bytes (5)  
 ▶ Flags: 0x020 (URG)  
 window size value: 512

|      |   |                 |
|------|---|-----------------|
| 0000 | 00 0c 29 26 13 d6 00 0c 29 e3 00 19 08 00 45 00 | ..)&... )....E. |
| 0010 | 00 28 9f 07 00 00 40 06 c9 59 c0 a8 c8 9d c0 a8 | .(...@. Y.....  |
| 0020 | c8 80 hd 02 00 50 21 aa c7 2a 57 d6 86 ec 50 20 | .....PI. *W...P |

- TCP URG Flooding 공격은 공격자가 헤더의 플래그를 URG(0x20)으로 설정한 후 대량의 패킷을 보내서 이를 처리하는 서버를 마비시키는 공격 방법이다.

## 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Warning! URG Flooding Detection"; flags:U; threshold:type threshold, track by\_src, count 10, seconds 1; sid:1000006;)

- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"URG Flooding Detection"; flags:U; threshold:type threshold, track by\_src, count 20, seconds 1; sid:1000006;)

#### 4.1.7. TCP Reset Flooding

##### 1) 공격분석

| No. | Time         | Source          | Destination     | Protocol | Length | Info                  |
|-----|--------------|-----------------|-----------------|----------|--------|-----------------------|
| 3   | 0.0003571... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 2863 → 80 [RST] Se... |
| 4   | 0.0003895... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 2721 → 80 [RST] Se... |
| 5   | 0.0003895... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 2722 → 80 [RST] Se... |
| 6   | 0.0003895... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 2777 → 80 [RST] Se... |
| 7   | 0.0003896... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 2723 → 80 [RST] Se... |
| 8   | 0.0003896... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 2724 → 80 [RST] Se... |

|  |                               |
|--|-------------------------------|
| Acknowledgment number (raw): 1483046532    |                               |
| 0101                                       | = Header Length: 20 bytes (5) |
| Flags: 0x004 (RST)                         |                               |
| Window size value: 512                     |                               |
| [Calculated window size: 512]              |                               |
| [Window size scaling factor: -1 (unknown)] |                               |

|      |                         |                         |                  |
|------|-------------------------|-------------------------|------------------|
| 0000 | 00 0c 29 26 13 d6 00 0c | 29 e3 00 19 08 00 45 00 | ..)&... ).....E. |
| 0010 | 00 28 9a 60 00 00 40 06 | ce 00 c0 a8 c8 9d c0 a8 | .(. .@. ....     |
| 0020 | c8 80 0a d9 00 50 05 3e | a4 e5 58 65 7e 84 50 04 | ....P.> ..Xe~.P. |
| 0030 | 02 00 0f 3b 00 00 00 00 | 00 00 00 00 00 00       | ...; .....       |

- TCP Reset Flooding은 위조 TCP 리셋 패킷을 전송하여 세션을 변조하고 종료시키는 공격 방법이다.
- 공격자는 수많은 Reset 패킷(RST)을 전송하여 서버의 자원을 소모시킨다.

##### 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Warning! RESET Flooding Detection"; flags:R; threshold:type threshold, track by\_src, count 10, seconds 1; sid:1000007;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"RESET Flooding Detection"; flags:R; threshold:type threshold, track by\_src, count 20, seconds 1; sid:1000007;)

#### 4.1.8. TCP Xmas Scan : OK

##### 1) 공격분석

| No. | Time         | Source          | Destination     | Protocol | Length | Info                                     |
|-----|--------------|-----------------|-----------------|----------|--------|--|
| 3   | 0.0005797... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1392 → 80 [FIN, PSH, URG] Seq=1 Win=5... |
| 4   | 0.0006473... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 80 → 1392 [RST, ACK] Seq=1 Ack=2 Win=... |

```

Sequence number (raw): 2104080771
[Next sequence number: 2 (relative sequence number)]
▶ Acknowledgment number: 2130470132
Acknowledgment number (raw): 2130470132
0101 .... = Header Length: 20 bytes (5)
▶ Flags: 0x029 (FIN, PSH, URG)
Window size value: 512
[Calculated window size: 512]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x7aae [unverified]
[Checksum Status: Unverified]
0000 00 0c 29 26 13 d6 00 0c 29 e3 00 19 08 00 45 00  ..)&....).....E.

```

- TCP 플래그 값을 모두 설정하거나, 일부 값을 설정하여 패킷을 보내는 스캔방법이다.
- 보통 TCP 헤더 flag에 FIN, PUSH, URG를 설정하여 패킷을 보낸다. 이 3개의 flag가 설정된 패킷을 공격 대상이 받으면, 해당 포트가 오픈된 경우에는 패킷을 DROP 한다.
- 닫혀 있는 경우에는 RST/ACK 패킷을 보낸다. 이러한 응답 유형으로 공격 대상의 포트를 확인할 수 있다.

## 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Xmas Scan Detection";flags:FPU;sid:1000008;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"Xmas Scan Detection";flags:FPU;sid:1000008;)

## 4.1.9. TCP NULL Scan

### 1) 공격분석

| No. | Time         | Source          | Destination     | Protocol | Length | Info                                     |
|-----|--------------|-----------------|-----------------|----------|--------|--|
| 3   | 0.6019435... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1574 → 80 [<None>] Seq=1 Win=512 Len=0   |
| 4   | 0.6019902... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 80 → 1574 [RST, ACK] Seq=1 Ack=1 Win=... |
| 5   | 3.5512919... | 192.168.200.157 | 224.0.0.251     | MDNS     | 180    | Standard query response 0x0000 PTR, c... |
| 6   | 3.5537066... | 192.168.200.157 | 192.168.200.1   | DNS      | 87     | Standard query 0x75e2 A daisy.ubuntu...  |
| 7   | 3.5598592... | 192.168.200.1   | 192.168.200.157 | DNS      | 119    | Standard query response 0x75e2 A dais... |

```

[TCP Segment Len: 0]
Sequence number: 1 (relative sequence number)
Sequence number (raw): 1852576128
[Next sequence number: 1 (relative sequence number)]
▶ Acknowledgment number: 1807696101
Acknowledgment number (raw): 1807696101
0101 .... = Header Length: 20 bytes (5)
▶ Flags: 0x000 (<None>)
Window size value: 512
[Calculated window size: 512]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x686e [unverified]

```

- TCP 헤더 flag에 어떤 flag도 설정하지 않고, 패킷을 보내는 스캔 방법이다.
- 이러한 유형의 패킷에 대해서 공격 대상은 X-mas Scan과 동일하게 응답한다.

## 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Null Scan Detection";flags:0;sid:1000009;)

- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"Null Scan Detection";flags:0;sid:1000009;)

#### 4.1.10. TCP FIN Scan

##### 1) 공격분석

| No. | Time         | Source          | Destination     | Protocol | Length | Info                                     |
|-----|--------------|-----------------|-----------------|----------|--------|--|
| 1   | 0.0000000... | 192.168.200.157 | 192.168.200.128 | TCP      | 60     | 1514 → 80 [FIN] Seq=1 Win=512 Len=0      |
| 2   | 0.0000573... | 192.168.200.128 | 192.168.200.157 | TCP      | 54     | 80 → 1514 [RST, ACK] Seq=1 Ack=2 Win=... |
| 3   | 4.1369406... | 192.168.200.157 | 224.0.0.251     | MDNS     | 180    | Standard query response 0x0000 PTR, c... |
| 4   | 4.1391896... | 192.168.200.157 | 192.168.200.1   | DNS      | 87     | Standard query 0xb968 A daisy.ubuntu...  |
| 5   | 4.1439057... | 192.168.200.1   | 192.168.200.157 | DNS      | 119    | Standard query response 0xb968 A dais... |

[TCP Segment Len: 0]  
 Sequence number: 1 (relative sequence number)  
 Sequence number (raw): 952345987  
 [Next sequence number: 2 (relative sequence number)]  
 Acknowledgment number: 1397667208  
 Acknowledgment number (raw): 1397667208  
 0101 .... = Header Length: 20 bytes (5)  
 Flags: 0x001 (FIN)  
 Window size value: 512  
 [Calculated window size: 512]  
 [Window size scaling factor: -1 (unknown)]

- 스텔스 스캔(Stealth Scan) 중 하나로, TCP 헤더를 조작하여 특수한 패킷을 만들어 보낸 후 그에 대한 응답으로 포트의 활성화 여부를 판단한다.
- 세션을 완전히 연결하지 않기 때문에 로그가 남지 않을 수 있다
- TCP 헤더 내에 FIN 플래그를 설정하여 전송한다.

##### 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Fin Scan Detection";flags:F;sid:1000010;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"Fin Scan Detection";flags:F;sid:1000010;)

#### 4.1.11. TCP SYN(Half Open) Scan

##### 1) 공격분석

- 세션을 완전히 연결하지 않고 포트의 활성화 여부를 판단하는 스캔 방법이다.
- 스텔스 스캔 중 하나로 포트가 열려있는 경우 공격자가 SYN 패킷을 보내고 공격 대상으로부터 SYN/ACK 패킷을 받으면, 그 즉시 RST 패킷을 보내 연결을 끊어서 로그를 남기지 않는다.
- 한편, 포트가 닫혀 있는 경우, 공격자가 SYN 패킷을 보내면 공격 대상으로부터 RST/ACK 패킷을 받는다.

##### 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET any (msg:"Syn Scan Detection"; flags:S; threshold:type both, track by\_dst, count 100, seconds 1; sid:1000011;)
- (IPS) drop tcp any any -> \$HOME\_NET any (msg:"Syn Scan Detection"; flags:S; threshold:type both, track by\_dst, count 100, seconds 1; sid:1000011;)



## 4.2. UDP

### 4.2.1. DNS Query Flooding

#### 1) 공격분석

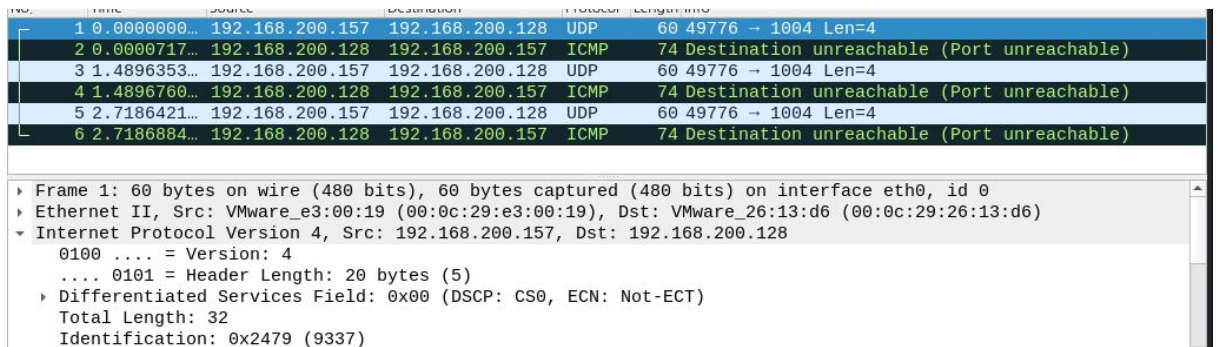
- DNS 쿼리 데이터를 다량으로 서버에 전송하여 DNS의 정상적인 서비스를 방해한다.
- 네트워크 인프라에 대한 대역폭 소진 공격으로써 동일 네트워크를 사용하는 모든 서비스에 대한 접속 장애를 유발한다.

#### 2) Rule

- (IPS) drop udp any any -> \$HOME\_NET 53 (msg:"DNS query flooding"; content:"www"; nocase; threshold:type both, track by\_src, count 100, seconds 1; sid:3000001;)

### 4.2.2. Port Scan

#### 1) 공격 분석



| No. | Time       | Source          | Destination     | Protocol | Length | Info                                       |
|-----|------------|-----------------|-----------------|----------|--------|--|
| 1   | 0.00000000 | 192.168.200.157 | 192.168.200.128 | UDP      | 60     | 49776 -> 1004 Len=4                        |
| 2   | 0.0000717  | 192.168.200.128 | 192.168.200.157 | ICMP     | 74     | Destination unreachable (Port unreachable) |
| 3   | 1.4896353  | 192.168.200.157 | 192.168.200.128 | UDP      | 60     | 49776 -> 1004 Len=4                        |
| 4   | 1.4896760  | 192.168.200.128 | 192.168.200.157 | ICMP     | 74     | Destination unreachable (Port unreachable) |
| 5   | 2.7186421  | 192.168.200.157 | 192.168.200.128 | UDP      | 60     | 49776 -> 1004 Len=4                        |
| 6   | 2.7186884  | 192.168.200.128 | 192.168.200.157 | ICMP     | 74     | Destination unreachable (Port unreachable) |

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0  
Ethernet II, Src: VMware\_e3:00:19 (00:0c:29:e3:00:19), Dst: VMware\_26:13:d6 (00:0c:29:26:13:d6)  
Internet Protocol Version 4, Src: 192.168.200.157, Dst: 192.168.200.128  
0100 .... = Version: 4  
.... 0101 = Header Length: 20 bytes (5)  
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
Total Length: 32  
Identification: 0x2479 (9337)

- 개방되지 않은 Port에 Packet을 날리면 ICMP unreachable packet을 돌려 준다.
- 이를 통해 개방된 포트가 뭐가 있는지 모두 알 수 있다.

#### 2) Rule

- (IPS) drop icmp \$HOME\_NET any -> any any ( msg:"udp scan"; itype:3; icode:3; drop icmp \$HOME\_NET any -> any any ( msg:"udp scan"; itype:3; icode:3;

### 4.2.3. NTP Reflection

- #### 1) NTP server(반사서버)에 공격자의 IP 즉 Source IP를 Victim의 IP로 변조하여 monlist 명령을 요청하면 응답이 증폭되어 전송되므로 공격이 된다.

#### 2) Rule

- (IPS) drop udp \$HOME\_NET 123 -> any any (msg:"ntp"; content:"|2a|"; threshold:type both, track by\_dst, count 100, seconds 1; sid:3000003;)

## 4.3. HTTP

### 4.3.1. HTTP GET Flooding

#### 1) 공격 분석

- 일반적으로 DDoS 공격은 웹 서버(Web Service)를 대상으로 DDoS 공격이 발생되며, TCP 세션 연결 이후 발생하는 일반적인 공격 형태이다. 정상적인 TCP 연결 과정 이후 정상적으로 보이는 HTTP Transaction 과정이 수행되는 DDoS 공격 기법이다.

#### 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET 80 (msg: "Warning HTTP GET flooding Detection"; content: "GET / HTTP/1."; nocase; depth:20; threshold:type threshold, track by\_src, count 5, seconds 1; sid:2000009;)
- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg: "HTTP GET flooding Detection"; content: "GET / HTTP/1."; nocase; depth:20; threshold:type threshold, track by\_src, count 10, seconds 1; sid:2000009;)

### 4.3.2. Slow HTTP Header Dos

#### 1) 공격 분석

| No.  | Time         | Source          | Destination     | Protocol | Length | Info   |
|------|--------------|-----------------|-----------------|----------|--------|--|
| 4356 | 10.041380... | 192.168.200.157 | 192.168.200.128 | TCP      | 75     | GET / HTTP/1.1 [TCP segment of a reassemb... |
| 4357 | 10.041380... | 192.168.200.157 | 192.168.200.128 | TCP      | 74     | GET / HTTP/1.1 [TCP segment of a reassemb... |
| 4358 | 10.041380... | 192.168.200.157 | 192.168.200.128 | TCP      | 75     | GET / HTTP/1.1 [TCP segment of a reassemb... |
| 4359 | 10.041380... | 192.168.200.157 | 192.168.200.128 | TCP      | 74     | GET / HTTP/1.1 [TCP segment of a reassemb... |
| 4360 | 10.041380... | 192.168.200.157 | 192.168.200.128 | TCP      | 75     | GET / HTTP/1.1 [TCP segment of a reassemb... |
| 4361 | 10.041380... | 192.168.200.157 | 192.168.200.128 | TCP      | 76     | GET / HTTP/1.1 [TCP segment of a reassemb... |
| 4362 | 10.041380... | 192.168.200.157 | 192.168.200.128 | TCP      | 75     | GET / HTTP/1.1 [TCP segment of a reassemb... |
| 4363 | 10.041380... | 192.168.200.157 | 192.168.200.128 | TCP      | 75     | GET / HTTP/1.1 [TCP segment of a reassemb... |

|   |
|---|
| Frame 4372: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, id 0   |
| Ethernet II, Src: VMware_e3:00:19 (00:0c:29:e3:00:19), Dst: VMware_26:13:d6 (00:0c:29:26:13:d6) |
| Internet Protocol Version 4, Src: 192.168.200.157, Dst: 192.168.200.128                         |
| Transmission Control Protocol, Src Port: 56072, Dst Port: 80, Seq: 175, Ack: 1, Len: 8          |
| Source Port: 56072  |
| Destination Port: 80  |
| [Stream index: 321]   |

|      |                         |                         |     |                |
|------|-------------------------|-------------------------|-----|----------------|
| 0000 | 00 0c 29 26 13 d6 00 0c | 29 e3 00 19 08 00 45 00 | ... | &... )...E     |
| 0010 | 00 3c ff e1 40 00 40 06 | 28 6b c0 a8 c8 9d c0 a8 | ... | <...@... (k... |
| 0020 | c8 80 db 08 00 50 64 8c | f6 63 56 30 a6 8d 80 18 | ... | ...Pd...cV0... |
| 0030 | 01 f6 2b b7 00 00 01 01 | 08 0a 4e f6 28 7f 63 1b | ... | ...+...N...c   |
| 0040 | 3d 12 58 2d 66 3a 20 75 | 0d 0a                   | ... | =>X-f: u...    |

- 개행문자를 '\r\n(0d 0a)'로 전송하여 불완전한 헤더를 전송 // \r\n\r\n이 올때까지 대기하게 됨
- 헤더 정보가 모두 전달되지 않은 것으로 판단하여 연결을 장시간 유지

#### 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET 80 (msg:"Slow HTTP Header Dos Detection"; flow:established, to\_server ; content:"|0d 0a|"; nocase; sid:2000010;)
- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg:"Slow HTTP Header Dos Detection"; flow:established, to\_server ; content:"|0d 0a|"; nocase; sid:2000010;)

### 4.3.3. XSS Attack

#### 1) 공격 분석

| No. | Time         | Source          | Destination     | Protocol | Length | Info  |
|-----|--------------|-----------------|-----------------|----------|--------|---|
| 2   | 0.0000000... | 192.168.200.1   | 192.168.200.128 | TCP      | 60     | 13981 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS... |
| 3   | 0.0000643... | 192.168.200.128 | 192.168.200.1   | TCP      | 58     | 80 → 13980 [SYN, ACK] Seq=0 Ack=1 Win=6424... |
| 4   | 0.0001039... | 192.168.200.128 | 192.168.200.1   | TCP      | 58     | 80 → 13981 [SYN, ACK] Seq=0 Ack=1 Win=6424... |
| 5   | 0.0003105... | 192.168.200.1   | 192.168.200.128 | HTTP     | 1575   | POST /bwAPP/xss_stored_1.php HTTP/1.1 (ap...  |
| 6   | 0.0003387... | 192.168.200.128 | 192.168.200.1   | TCP      | 54     | 80 → 13980 [ACK] Seq=1 Ack=1522 Win=62780 ... |
| 7   | 0.0004175... | 192.168.200.128 | 192.168.200.1   | HTTP     | 4642   | HTTP/1.1 200 OK (text/html)                   |

Key: entry

Value [truncated]: <?r\n\r\n\$cookie=\$\_GET['data'];\r\n\r\n\$time=date("y-m-d H:i:s"); // \$time에 연월분...

▶ Form item: "blog" = "submit"

▶ Form item: "entry\_add" = ""

|      |                         |                         |                         |                  |
|------|-------------------------|-------------------------|-------------------------|------------------|
| 0300 | 3d                      | 25 33 43 25 33 46 25    | 30 44 25 30 41 25 30 44 | %3C%3F% 0D%0A%0D |
| 0310 | 25 30 41 25 32 34 63 6f | 6f 6b 69 65 25 33 44 25 | %0A%24co okie%3D%       |                  |
| 0320 | 32 34 5f 47 45 54 25 35 | 42 25 32 37 64 61 74 61 | 24_GET%5 B%27data       |                  |
| 0330 | 25 32 37 25 35 44 25 33 | 42 25 30 44 25 30 41 25 | %27%5D%3 B%0D%0A%       |                  |
| 0340 | 30 44 25 30 41 25 32 34 | 61 74 69 6d 65 25 33 44 | 0D%0A%24 \$time%3D      |                  |
| 0350 | 64 61 74 65 25 32 38 25 | 32 32 79 2d 6d 2d 64 2b | date%28% 22y-m-d+       |                  |
| 0360 | 48 25 33 41 69 25 33 41 | 73 25 32 32 25 32 39 25 | H%3Ai%3A s%22%29%       |                  |

- 서버에 <script> 악성스크립트 </script>를 삽입하여 데이터베이스에 저장 시킴
- 악의적인 스크립트로 인해 웹 페이지의 원래 코드의 의도와 다르게 실행되도록 함(클라이언트)
- 탐지시 대소문자를 구분하지 않도록 nocase keyword 사용

## 2) Rule

- (IDS) alert tcp any any -> \$HOME\_NET 80 (msg:"XSS Detection"; content:"|25 33 43|script>"; nocase; sid:2000001;)
- (IDS) alert tcp any any -> \$HOME\_NET 80 (msg:"XSS Detection"; content:"%3cscript%3e"; nocase; sid:2000002; )
- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg:"XSS Detection"; content:"|25 33 43|script>"; nocase; sid:2000001;)
- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg:"XSS Detection"; content:"%3cscript%3e"; nocase; sid:2000002;)

## 4.3.4. SQL Injection

### 1) 공격 분석

- 악의적인 SQL문을 삽입하여 데이터베이스가 비정상적인 행위를 하도록 함
- Error base, Time base, Union SQL, Blind SQL 등등 인젝션 기법이 다양

### 2) Rule

- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg:"SQL UNION SELECT Detection"; flow:established,to\_server; content:"UNION"; nocase; http\_uri; content:"SELECT"; nocase; http\_uri; pcre:"/UNION.+SELECT/"; sid:2000003;)
- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg:"SQL user in uri Detection"; flow:established,to\_server; content:"SELECT"; nocase; http\_uri; content:"user"; nocase; http\_uri; pcre:"/SELECT[^a-z].+user/"; sid:2000004; )
- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg:"SQL sleep Detection"; flow:established,to\_server; content:"SELECT"; nocase; http\_uri; content:"sleep |28|"; nocase; http\_uri; pcre:"/bSELECT.\*?\bsleep\x28/"; sid:2000005;)
- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg:"SQL information schema Detection"; flow:established,to\_server; content:"information\_schema"; nocase; http\_uri; sid:2000006;)
- (IPS) drop tcp any any -> \$HOME\_NET 80 (msg:"SQL SELECT FROM Detection"; flow:established,to\_server; content:"SELECT"; nocase; http\_uri; content:"FROM"; nocase; http\_uri; pcre:"/SELECT.\*FROM/"; sid:2000007; )

## 4.4. ICMP

#### 4.4.1. Ping of Death

## 1) 공격 분석

[illegible]

- Ping of Death 공격시 Data의 값이 58585858.. 같은 수로 반복된다.

## 2) Rule

- (IDS) alert icmp any any -> \$HOME\_NET any (msg:"Warning Ping of Death Attack!!!"; dsiz>1000; itype:8; icode:0; detection\_filter:track by\_src, count 30, seconds 1; sid:4000002; classtype:denial-of-service;)
- (IPS) drop icmp any any -> \$HOME\_NET any (msg:"Warning Ping of Death Attack!!!"; dsiz>1000; itype:8; icode:0; detection\_filter:track by\_src, count 30, seconds 1; sid:4000002; classtype:denial-of-service;)

#### 4.4.2. ICMP Flooding

## 1) 공격분석



| No.     | Time         | Source          | Destination     | Protocol | Length | Info                              |
|---------|--------------|-----------------|-----------------|----------|--------|-----------------------------------|
| 1767... | 63.860953837 | 192.168.200.128 | 192.168.200.255 | ICMP     | 60     | Echo (ping) request id=0xa016,... |
| 1767... | 63.860953885 | 192.168.200.1   | 192.168.200.128 | ICMP     | 60     | Echo (ping) reply id=0xa016,...   |
| 1767... | 63.860953929 | 192.168.200.1   | 192.168.200.128 | ICMP     | 60     | Echo (ping) reply id=0xa016,...   |
| 1767... | 63.861095275 | 192.168.200.128 | 192.168.200.255 | ICMP     | 60     | Echo (ping) request id=0xa016,... |
| 1767... | 63.861095370 | 192.168.200.128 | 192.168.200.255 | ICMP     | 60     | Echo (ping) request id=0xa016,... |
| 1767... | 63.861095424 | 192.168.200.1   | 192.168.200.128 | ICMP     | 60     | Echo (ping) reply id=0xa016,...   |
| 1767... | 63.861095478 | 192.168.200.1   | 192.168.200.128 | ICMP     | 60     | Echo (ping) reply id=0xa016,...   |
| 1767... | 63.861095557 | 192.168.200.128 | 192.168.200.255 | ICMP     | 60     | Echo (ping) request id=0xa016,... |
| 1767... | 63.861095615 | 192.168.200.128 | 192.168.200.255 | ICMP     | 60     | Echo (ping) request id=0xa016,... |
| 1767... | 63.861095675 | 192.168.200.1   | 192.168.200.128 | ICMP     | 60     | Echo (ping) reply id=0xa016,...   |
| 1767... | 63.861140286 | 192.168.200.1   | 192.168.200.128 | ICMP     | 60     | Echo (ping) reply id=0xa016,...   |

|                                   |  |  |  |  |  |  |
|-----------------------------------|--|--|--|--|--|--|
| Internet Control Message Protocol |  |  |  |  |  |  |
| Type: 8 (Echo (ping) request)     |  |  |  |  |  |  |
| Code: 0                           |  |  |  |  |  |  |
| Checksum: 0x2fc3 [correct]        |  |  |  |  |  |  |
| [Checksum Status: Good]           |  |  |  |  |  |  |

|      |                         |                         |        |         |
|------|-------------------------|-------------------------|--------|---------|
| 0000 | ff ff ff ff ff 00 0c    | 29 e3 00 19 08 00 45 00 | .....) | .....E  |
| 0010 | 00 1d 05 d6 00 00 40 01 | 62 39 c0 a8 c8 80 c0 a8 | .....@ | .....b9 |
| 0020 | c8 ff 08 00 2f c3 a0 16 | d0 25 58 00 00 00 00 00 | ...../ | .....%X |
| 0030 | 00 00 00 00 00 00 00 00 | 00 00 00 00             | .....  | .....   |

- 단시간 내에 다량의 패킷이 수신 되는게 특징이다.

## 2) Rule

- (IDS) alert icmp any any -> \$HOME\_NET any (msg:"Warning ICMP Flooding!!!"; threshold:type both, track by\_src, count 3, seconds 1; sid:4000003;)
- (IPS) drop icmp any any -> \$HOME\_NET any (msg:"ICMP Flooding!!!"; threshold:type both, track by\_src, count 20, seconds 5; sid:4000003;)

## 4.4.3. Land Attack

### 1) 공격분석

| No. | Time         | Source          | Destination     | Protocol | Length | Info                              |
|-----|--------------|-----------------|-----------------|----------|--------|-----------------------------------|
| 19  | 49.909420308 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 20  | 49.909420427 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 21  | 49.909420479 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 22  | 49.909567152 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 23  | 49.909567220 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 24  | 49.909567286 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 25  | 49.909698064 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 26  | 49.909827184 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 27  | 49.910148982 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 28  | 49.910212966 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 29  | 49.910213036 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |
| 30  | 49.910213095 | 192.168.200.128 | 192.168.200.128 | ICMP     | 60     | Echo (ping) request id=0xcc16,... |

|                                    |  |  |  |  |  |  |
|------------------------------------|--|--|--|--|--|--|
| Identifier (BE): 52246 (0xcc16)    |  |  |  |  |  |  |
| Identifier (LE): 5836 (0x16cc)     |  |  |  |  |  |  |
| Sequence number (BE): 768 (0x0300) |  |  |  |  |  |  |
| Sequence number (LE): 3 (0x0003)   |  |  |  |  |  |  |
| [No response seen]                 |  |  |  |  |  |  |

|      |                         |                         |        |        |
|------|-------------------------|-------------------------|--------|--------|
| 0000 | 00 0c 29 26 13 d6 00 0c | 29 e3 00 19 08 00 45 00 | .....) | .....E |
| 0010 | 00 1c 8f 95 00 00 40 01 | d8 f9 c0 a8 c8 80 c0 a8 | .....@ | .....  |
| 0020 | c8 80 08 00 28 e9 cc 16 | 03 00 00 00 00 00 00 00 | .....( | .....  |
| 0030 | 00 00 00 00 00 00 00 00 | 00 00 00 00             | .....  | .....  |

- Source IP와 Destination IP가 동일하다.

## 2) Rule

- (IDS) alert icmp any any -> \$HOME\_NET any (msg: "Warning Land Attack Detect"; sameip; sid:4000004;)
- (IPS) drop icmp any any -> \$HOME\_NET any (msg: "Land Attack Detect"; sameip; threshold:type both, track by\_src, count 20, seconds 5; sid:4000004;)