

Malware Analysis (CS6038)

Week 01.1 Introductions

Scott Nusbaum

nusbausa@ucmail.uc.edu

January 15, 2019

Overview

- Who am I
- Syllabus review
- What is expected of you
 - Homework
 - Grading
 - Exams
- Question and Answers

Instructor Introduction

- [TrustedSec](#) -- Senior Incident Response & Research Consultant
- [LinkedIn](#)
- [Twitter](#) (@_snus)
- nusbausa@ucmail.uc.edu
- Office Hours -- On Request



Disclaimer

You will learn a number of techniques helpful for attacking systems, as well as other techniques that may facilitate other illegitimate goals. Many of these will be older, but still present concepts that continue to apply today. You are responsible for using this knowledge for academic and research purposes only, within the scope of the examples & assignments provided in this class and in your research efforts at The University. Abuse of these methods for illegal purposes is against policy. You will learn how attacks are employed, in order to become better analysts yourselves.

Syllabus

- Dates and topics are subject to change
- <http://class.snusbaum.com/>



Grading

NO CHEATING!!! Do your own work.

Warning, some of the homework might contain answers unique to each student. Submitting the answer of a student other than yourself will be considered cheating.

Next Class

- Bring a storage device large enough to hold a 4.5Gb zip file
 - Windows 10 OVA image for VirtualBox
- I will bring in a router to aid in the distribution of the file.

Does everyone understand?



(silent)



Good, I'll move on then

Reason #1,487 why it's a good idea to ask questions

Questions - Extensions

- Exe
- Pdf
- Jpg
- Eml
- Png
- Pcap
- Java
- Class
- Com
- dll
- Js
- Doc
- Docx
- Plist
- Rtf
- Csv
- Dat
- Ppt
- Pptx
- jar
- Tar
- gz
- zip
- Xml
- Bmp
- Xls
- Xlsx
- Db
- Sql
- Apk
- Bat

Questions – Magic Numbers

- What are magic numbers?
- Why are they useful?



Questions – Magic Numbers

- List of common [Magic numbers](#)
- Linux command "file"

Questions – What is this program?

The screenshot shows a network traffic analysis tool interface. The top part displays a list of 15 network packets with columns for No., Time, Source, Destination, Protocol, Dest Port, Length, and Info. Packet 10 is highlighted in red, indicating a reset (RST) from the source to the destination.

No.	Time	Source	Destination	Protocol	Dest Port	Length	Info
1	0.000000	192.168.0.33	192.168.0.21	TCP	8080	62	58031 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK...
2	0.000040	192.168.0.21	192.168.0.33	TCP	58031	62	8080 → 58031 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MS...
3	0.000328	192.168.0.33	192.168.0.21	TCP	8080	60	58031 → 8080 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.001057	192.168.0.33	192.168.0.21	HTTP	8080	963	POST / HTTP/1.1 (application/x-www-form-urlencoded)
5	0.001093	192.168.0.21	192.168.0.33	TCP	58031	54	8080 → 58031 [ACK] Seq=1 Ack=910 Win=30906 Len=0
6	0.002780	192.168.0.21	192.168.0.33	TCP	58031	71	8080 → 58031 [PSH, ACK] Seq=1 Ack=910 Win=30906 Len=17...
7	0.002889	192.168.0.21	192.168.0.33	HTTP	58031	217	HTTP/1.0 200 OK (text/html)
8	0.003212	192.168.0.33	192.168.0.21	TCP	8080	60	58031 → 8080 [ACK] Seq=910 Ack=181 Win=64060 Len=0
9	0.003537	192.168.0.33	192.168.0.21	TCP	8080	60	58031 → 8080 [ACK] Seq=910 Ack=182 Win=64060 Len=0
10	0.004236	192.168.0.33	192.168.0.21	TCP	8080	60	58031 → 8080 [RST, ACK] Seq=910 Ack=182 Win=0 Len=0
11	0.009862	192.168.0.33	192.168.0.21	TCP	8080	62	58032 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK...
12	0.009881	192.168.0.21	192.168.0.33	TCP	58032	62	8080 → 58032 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MS...
13	0.010145	192.168.0.33	192.168.0.21	TCP	8080	60	58032 → 8080 [ACK] Seq=1 Ack=1 Win=64240 Len=0
14	0.010921	192.168.0.33	192.168.0.21	HTTP	8080	828	POST / HTTP/1.1 (application/x-www-form-urlencoded)
15	0.010945	192.168.0.21	192.168.0.33	TCP	58032	54	8080 → 58032 [ACK] Seq=1 Ack=775 Win=30186 Len=0

▼ Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
▼ Ethernet II, Src: IntelCor_72:11:f4 (44:85:00:72:11:f4), Dst: PcsCompu_d6:af:a7 (08:00:27:d6:af:a7)
▼ Internet Protocol Version 4, Src: 192.168.0.33, Dst: 192.168.0.21
▼ Transmission Control Protocol, Src Port: 58031, Dst Port: 8080, Seq: 0, Len: 0

```
0000  08 00 27 d6 af a7 44 85 00 72 11 f4 08 00 45 00  ...'.D.r...E.
0010  00 30 5d 19 40 00 80 06 1c 28 c0 a8 00 21 c0 a8  .0]@... (..!..
0020  00 15 e2 af 1f 90 ca df 23 a6 00 00 00 00 70 02  .....#....p.
0030  fa f0 15 e2 00 00 02 04 05 b4 01 01 04 02      .....

```

Questions – What does this show?

The screenshot displays the Immunity Debugger interface for the memory module `14_embedded_exe.mem_mod`. On the left, a list of symbols is shown with columns for index, name, block, and size. The central area features a control flow graph (CFG) with nodes representing code blocks and arrows indicating the flow of execution. The right pane provides a detailed view of the selected code block, showing assembly instructions and their comments.

Symbol List:

Idx	Name	Block	Size
0	EntryPoint	1	17
136	sub_401011	1	80
137	sub_401061	2	38
253	sub_build_ofb_function_table	6	105
1	sub_wraps_decode_of_functions	7	195
2	sub_4011b8	1	25
138	sub_4011d1	10	147
139	sub_401264	6	64
3	sub_4014e6	4	75
140	sub_401531	4	56
141	sub_40156b	9	169
142	sub_401614	6	89
4	sub_40166d	24	269
5	sub_401750	1	7
143	sub_401794	26	521
6	sub_40199d	11	141
144	sub_401a2a	37	556
7	sub_401ab0	1	7
8	sub_401b40	1	7
145	sub_401c64	7	191
146	sub_401d23	13	137
147	sub_401dac	7	60
9	sub_401de0	2	4
148	sub_401dec	12	190
10	sub_401eb0	35	581
149	sub_4020fb	5	66
11	sub_40212d	10	122
255	procedures		

Code Block Details (loc_40184a):

```
00401846 5A1F          push  eax
00401850 6A2C          push  0x2c
00401852 8007FFFFFF   call  sub_wraps_decode_of_functions ; sub_wraps_decode_of_functi
00401857 8945CB       mov   dword [ebp+0x38], eax
0040185A       push  dword [dword_43b022] ; dword_43b022
00401863 8945CC       mov   dword [ebp+0x34], eax
00401866 6A2C          push  0x2c
00401868       call  sub_wraps_decode_of_functions ; sub_wraps_decode_of_functi
0040186B 8945CB       mov   dword [ebp+0x38], eax
00401875 FF75CC       push  dword [dword_43b040]
00401878 FF75CC       call  dword [dword_43b040]
0040187A 8945D0       mov   dword [ebp+0x38], eax
00401881 8045D0       push  0x40
00401884 58           push  eax
00401887 6A35       push  0x35
00401889       call  sub_wraps_decode_of_functions ; sub_wraps_decode_of_functi
0040188E 58           push  eax
00401891       call  dword [ebp+0x2c], eax
```

Questions – Disassemblers

- Do you know what a disassembler is?
- Who has used at least one the following:
 - IDA Pro
 - Hopper
 - BinaryNinja
 - Objdump
 - Radare2

Questions

- What is the difference between a disassembler and a decompiler?
- How has used at least one of the following:
 - ILSpy
 - .NET Reflector
 - JADx
 - JEB Decompiler

Questions - File Specs

offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00000000	0x5A4D (MZ)		lastsize		PagesInFile		relocations		headerSizeInParagraph		MinExtraParagraphNeeded		MaxExtraParagraphNeeded		Initial (relative) SS	
0x00000010	Initial (relative) SP		checksum		Initial IP		Initial (relative) CS		FileAddOfRelocTable		OverlayNumber		reserved		reserved	
0x00000020	reserved		reserved		OEMIdentifier		OEMInformation		reserved		reserved		reserved		reserved	
0x00000030	reserved		reserved		reserved		reserved		reserved		reserved		0x80 (offset to PE signature)			
0x00000040	This block contains instructions to display the message "This program cannot be run in DOS mode" when run in MS-DOS															
0x00000050																
0x00000060																
0x00000070																
0x00000080	0x00004550 (PE\0\0 - PE Signature)				Target Machine		NumberOfSections		TimeDateStamp			PointerToSymbolTable (0 for image)				
0x00000090	NumberOfSymbols (0 for image)				SizeOfOptionalHeaders		Characteristics		0x10B (exe)	lnMajVer	lnMnrVer	SizeOfCode				
0x000000A0	SizeOfInitializedData				SizeOfUninitializedData				AddressOfEntryPoint				BaseOfCode			
0x000000B0	BaseOfData				ImageBase				SectionAlignment				FileAlignment			
0x000000C0	MajorOSVersion		MinorOSVersion		MajorImageVersion		MinorImageVersion		MajorSubsystemVersion		MinorSubsystemVersion		Win32VersionValue			
0x000000D0	SizeOfImage				SizeOfHeaders				CheckSum				CheckSum		DllCharacteristics	
0x000000E0	SizeOfStackReserve				SizeOfStackCommit				SizeOfHeapReserve				SizeOfHeapCommit			

Questions – File Specs

```
00000000: 7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 .ELF.....
00000010: 02 00 3e 00 01 00 00 00 80 08 40 00 00 00 00 ..>.....@
00000020: 40 00 00 00 00 00 00 00 f8 33 00 00 00 00 00 @.....3
00000030: 00 00 00 00 40 00 38 00 08 00 40 00 1c 00 1b ...@.8...@
00000040: 06 00 00 00 05 00 00 00 40 00 00 00 00 00 00 .....@
00000050: 40 00 40 00 00 00 00 00 40 00 40 00 00 00 00 @.@...@@
00000060: c0 01 00 00 00 00 00 00 c0 01 00 00 00 00 00 .....
00000070: 08 00 00 00 00 00 00 00 03 00 00 00 04 00 00 .....
00000080: 00 02 00 00 00 00 00 00 00 02 40 00 00 00 00 .....@
00000090: 00 02 40 00 00 00 00 00 1c 00 00 00 00 00 00 ..@.....
000000a0: 1c 00 00 00 00 00 00 00 01 00 00 00 00 00 00 .....
000000b0: 01 00 00 00 05 00 00 00 00 00 00 00 00 00 00 .....
000000c0: 00 00 40 00 00 00 00 00 00 00 40 00 00 00 00 ..@.....@
000000d0: bc 28 00 00 00 00 00 00 bc 28 00 00 00 00 00 (.....(
```


Questions – What does this do?

The screenshot shows a debugger interface with the following components:

- Assembly View:** Displays assembly instructions with their addresses and comments. The current instruction is `add rsp,38` at address `0000000077C87983`. Other instructions include `jmp ntd11.77C87983`, `nop`, `lea rax,qword ptr ds:[77D0D6A0]`, and `mov r11,rsp`.
- Registers View:** Shows the state of various registers. `RAX` is `0000000000000000`, `RCX` is `0000000077C2BF8A`, `R8` is `000000000022F508`, and `R9` is `0000000077CE03F0`.
- Command Line:** Shows the command `.text:0000000077C87981 ntd11.d11:$A7981 #A6D81`.
- Memory Dump:** Shows a hex dump of memory starting at address `0000000077BE1000`. The dump includes hex values and their corresponding ASCII characters.
- Status Bar:** Indicates the debugger is **Paused** and that a **System breakpoint reached!** message is displayed.

Questions

- What is this?

dGhpcyBpcyBhIHRlc3QgaG93IG1hbnkgb2YgeW9
1IGNhbiBkZXRlcmlpbmUgd2hhdCBpcyBiZWlu
ZyBzYWlkIGJ5IHRoaXMgbWVzc2FnZT8/Cg==

Questions – What is this?

```
1 %PDF-1.3^M
2 1 0 obj^M
3 [/PDF /Text /ImageB /ImageC /ImageI]^M
4 endobj^M
5 6 0 obj^M
6 << /Length 2187 /Filter /FlateDecode >> stream^M
7 x<9c>-<9b>msÛ6^RÇBg&&B^A3õ<8b>Û0Àx~ðÛÛ^L-Ñ<91>z¶ãJt^ [Ûõ^øb¥õ<8d>eµ²ó|ý[<80> ^EI$X^YJ2ã<88>^FðÃí^?±
8 ,TÝ<86>iÏX=Û_<88>A^[E^LÛ0³6Ë^ZÌ$ç<84>`B^váfCq<81>eÝÇ^P<8d>@A<9f>çèl8ç`¿%Û^H^L¿d~6<8c>P,`@ÛÂT*íE<8
9 ^K#Pq<85>8@<85><80>q8Ã<9c>;â^A)?^X_ç¿<8e>Ç}4%}]^Vù^T]Û;Ãð^aX<8f>Ð^G4Ëóþ »F? â^?`8Ey±^_F(<83><95>ÃÛ
10 ã<8b>i>ù%+Æ<93>;^D0\<9b>»^C8ÑÝ+v^?õçýðõ+<8c>±ÃB^U8ÑB+²>£-ð-9çþÝY<93>»ý<84><95>Û<97><92>ÍV<96>W%Ëõ
11 È±]É^@L<95>2^B-Í§^E<9a><8c>ÇÛkEV2PB.¥A<94>@cOÛ¶g/%1Ç<90><92>Ãq|ãÐÛ.e@!J^Y1wH^Y<80>@RF@_ájq_w{<93>
12 pL^Ãj^L^[Û~<92>VEDIcf-^U0Q0^X0^[^Vw{*ZÑ^U<8d>ñ^PíPÑç(HiÝÓ<83>4E<8d>Ãj:<81>ý<9e>iP4 S^U]1%5<9c>-
13 <9a><9a>#jg²^UË^CGMñb²^Fb³<8c>5r<87><8a>^A<97>²^b<8d>;$ÍVðäUYÓ;³,t<<82><96>PñË<8a>þCÍÀLÛ3<82>iÐ
14 [u0<8a>6P~AÍ@1^MõÍíý³óÍÝ <9a>Îõ× <içý^C<86>õ\<87>^M
15 endstream^M
16 endobj^M
17 2 0 obj^M
18 << /Type /Page /Parent 7 0 R /MediaBox [0 0 595.44 841.68] /Contents 6 0 R /Resources << /ProcSet
19 endobj^M
20 3 0 obj^M
21 << /Type /XObject /Subtype /Image /ColorSpace /DeviceRGB /BitsPerComponent 8 /Filter /FlateDecode
22
```



Questions –What does this do?

```
00401029
00401029 loc_401029:
00401029 mov     ecx, offset loc_40107C
0040102E add     ecx, 79h
00401031 mov     eax, offset loc_40107C
00401036 mov     dl, [ebp+buf]
```

```
00401039
00401039 loc_401039:
00401039 mov     bl, [eax]
0040103B xor     bl, dl
0040103D add     bl, 22h
00401040 mov     [eax], bl
00401042 inc     eax
00401043 cmp     eax, ecx
00401045 jl     short loc_401039
```

Questions

- What does this code do?

```
{  
    char *d = dest;  
    const char *s = src;  
    while (len--)  
        *d++ = *s++;  
    return dest;  
}
```

```

memory:
0000000000000068a 55          push    rbp
0000000000000068b 4889E5     mov     rbp, rsp
0000000000000068e 48897DE8   mov     qword [rbp+var_18], rdi
00000000000000692 488975E0   mov     qword [rbp+var_20], rsi
00000000000000696 488955D8   mov     qword [rbp+var_28], rdx
0000000000000069a 488B45E8   mov     rax, qword [rbp+var_18]
0000000000000069e 488945F0   mov     qword [rbp+var_10], rax
000000000000006a2 488B45E0   mov     rax, qword [rbp+var_20]
000000000000006a6 488945F8   mov     qword [rbp+var_8], rax
000000000000006aa EB1D      jmp     loc_6c9

-----
loc_6ac:
000000000000006ac 488B55F8   mov     rdx, qword [rbp+var_8]
000000000000006b0 488D4201   lea    rax, qword [rdx+1]
000000000000006b4 488945F8   mov     qword [rbp+var_8], rax
000000000000006b8 488B45F0   mov     rax, qword [rbp+var_10]
000000000000006bc 488D4801   lea    rcx, qword [rax+1]
000000000000006c0 48894DF0   mov     qword [rbp+var_10], rcx
000000000000006c4 0FB612    movzx  edx, byte [rdx]
000000000000006c7 8810      mov     byte [rax], dl

-----
loc_6c9:
000000000000006c9 488B45D8   mov     rax, qword [rbp+var_28]
000000000000006cd 488D50FF   lea    rdx, qword [rax-1]
000000000000006d1 488955D8   mov     qword [rbp+var_28], rdx
000000000000006d5 4885C0    test   rax, rax
000000000000006d8 75D2      jne    loc_6ac

-----
000000000000006da 488B45E8   mov     rax, qword [rbp+var_18]
000000000000006de 5D        pop     rbp
000000000000006df C3        ret

; endp

```



```
memory:
0000055d 55          push     ebp
0000055e 89E5       mov     ebp, esp
00000560 83EC10    sub     esp, 0x10
00000563 E8DC000000 call    __x86.get_pc_thunk.ax
00000568 05701A0000 add     eax, 0x1a70
0000056d 8B4508    mov     eax, dword [ebp+__dest]
00000570 8945F8    mov     dword [ebp+var_8], eax
00000573 8B450C    mov     eax, dword [ebp+__src]
00000576 8945FC    mov     dword [ebp+var_4], eax
00000579 EB17     jmp     loc_592
```

loc_57b:

```
0000057b 8B55FC    mov     edx, dword [ebp+var_4]
0000057e 8D4201    lea    eax, dword [edx+1]
00000581 8945FC    mov     dword [ebp+var_4], eax
00000584 8B45F8    mov     eax, dword [ebp+var_8]
00000587 8D4801    lea    ecx, dword [eax+1]
0000058a 894DF8    mov     dword [ebp+var_8], ecx
0000058d 0FB612    movzx  edx, byte [edx]
00000590 8810     mov     byte [eax], dl
```

loc_592:

```
00000592 8B4510    mov     eax, dword [ebp+__n]
00000595 8D50FF    lea    edx, dword [eax-1]
00000598 895510    mov     dword [ebp+__n], edx
0000059b 85C0     test   eax, eax
0000059d 75DC     jne    loc_57b
```

```
0000059f 8B4508    mov     eax, dword [ebp+__dest]
000005a2 C9       leave  esp
000005a3 C3       ret
```

; endp

Questions

- What are the following?
 - HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
 - HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce
 - HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
 - \Users\Foo\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Startup\

Questions

- What is this and what does it do?
 - Set-NetFirewallProfile -Profile
Domain,Public,Private -Enabled False
 - Set-MpPreference -DisableRealtimeMonitoring
\$true

Questions – What are these?

- Cuckoo
- VirusTotal.com
- <http://contagiodump.blogspot.com/>
- VirusBay.io
- VirusShare.com

Questions

```
public boolean a();
Code:
  0: aload_0
  1: getfield      #45          // Field d:Lcom/google/ctf/shallweplayagame/a$a;
  4: getstatic    #43          // Field com/google/ctf/shallweplayagame/a$a.a:Lcom/google/ctf/shallweplayagame/a$a;
  7: if_acmpne    12
 10: iconst_1
 11: ireturn
 12: iconst_0
 13: ireturn

public void setValue(com.google.ctf.shallweplayagame.a$a);
Code:
  0: aload_0
  1: aload_1
  2: bipush      1
  4: invokevirtual #203        // Method a:(Lcom/google/ctf/shallweplayagame/a$a;I)V
  7: return
```

Questions

- What is this?

mshta.exe

```
javascript:hw44jPL="7maatEa";jP7=new%20ActiveXObject("WScript.Shell");vM0LgE="j";BE4Uk=jP7.RegRead("HKLM\\software\\0XhyxVPNt6\\YXfi2uwWS");L4E1Z="uH";eval(BE4Uk);XGJz6ED="hHxrwk7";
```