

Malware Analysis - DAY 3

Prep By Yohanes Syailendra

Today's Agenda

- ▶ Fundamental Reverse Engineering
- ▶ Malware Memory Analysis

Reverse Engineering

The background features abstract, overlapping geometric shapes in various shades of green, ranging from light lime to dark forest green. These shapes are primarily located on the right side of the frame, creating a modern, layered effect. The text 'Reverse Engineering' is centered on the left side of the image in a clean, sans-serif font.

Immunity Break Down

The screenshot displays the Immunity Debugger interface with three main windows open:

- CPU window (top left):** Shows assembly instructions for the SE handler installation. The instruction list includes operations like `PUSH EBP`, `MOV EBP,ESP`, `PUSH -1`, `PUSH avtst2.0040300C`, `PUSH avtst2.00401158`, `PUSH DWORD PTR FS:[0]`, `MOV DWORD PTR FS:[0],ESP`, `SUB ESP,0C`, `PUSH EBX`, `PUSH ESI`, `PUSH EDI`, `MOV DWORD PTR SS:[EBP-18],ESP`, `PUSH 20000000`, `CALL avtst2.00401010`, `POP ECX`, `MOV DWORD PTR DS:[404338],EAX`, `CALL avtst2.00401370`, `TEST EAX,EAX`, `JNZ SHORT avtst2.0040109C`, `PUSH 1`, `CALL avtst2.00401500`, `POP ECX`, `JMP avtst2.00401137`, `MOV DWORD PTR SS:[EBP-4],0`, `CALL avtst2.00401500`, `CALL avtst2.00401600`, `CALL avtst2.00401600`, `CALL avtst2.00401AE0`, `CALL avtst2.00401880`, `MOV EBX,avtst2.00403090`, `CMP EBX,avtst2.00403090`, and `IMP SHORT avtst2.0040100A`. The CPU window also displays `SE handler installation` and `CPU window shows instructions`.
- Registers window (top right):** Shows the state of the CPU registers. The `EAX` register contains `771A3378` (kernel32.BaseThreadInitThunk). The `EBP` register contains `0018FF94`. The `EIP` register contains `00401050` (avtst2.<ModuleEntryPoint>). The registers window also displays `Registers window is self explanatory`.
- Dump window (bottom left):** Shows a memory dump starting at address `00404000`. The dump includes hex values, ASCII characters, and their corresponding ASCII values. The dump window also displays `Dump window shows memory`.
- Stack window (bottom right):** Shows the contents of the stack. The stack contains various return addresses and pointers, including `771A338A` (kernel32.771A338A), `77979F72` (ntdll.77979F72), `77E6F11E` (ntdll.77E6F11E), `77979F45` (ntdll.77979F45), and `00401050` (avtst2.<ModuleEntryPoint>). The stack window also displays `Stack window shows contents of stack`.

The status bar at the bottom of the debugger shows `[22:57:39] Program entry point` and `Paused`.

Immunity Breakdown 2

The image shows a screenshot of the Immunity Debugger interface. The main window displays assembly code for the CPU's main thread in the module avtst2.exe. The code is color-coded and includes instructions such as PUSH, MOV, CALL, and JMP. The status bar at the bottom indicates the current instruction pointer (EBP) is 0018FF94. On the right side of the screenshot, several white arrows point from text labels to specific icons in the Immunity Debugger toolbar. These labels are: References, Hardware Breakpoints, Software Breakpoints, Call Stack, Show CPU window, Show Handles, Window Info, Show Threads, Show Memory Map, Show Loaded modules, and Show log.

Address	Disassembly
00401050	\$ 55 PUSH EBP
00401051	89E5 MOV EBP,ESP
00401053	6A FF PUSH -1
00401055	68 0C304000 PUSH avtst2.00401000
0040105A	68 58114000 PUSH avtst2.00401150
0040105F	64:FF35 000000 PUSH DWORD PTR FS:[0]
00401066	64:8925 000000 MOV DWORD PTR FS:[0],ESP
0040106D	83EC 0C SUB ESP,0C
00401070	53 PUSH EBX
00401071	56 PUSH ESI
00401072	57 PUSH EDI
00401073	8965 E8 MOV DWORD PTR SS:[EBP-4],ESP
00401076	68 00000002 PUSH 2000000
0040107B	E8 900C0000 CALL avtst2.00401010
00401080	59 POP ECX
00401081	A3 38434000 MOV DWORD PTR DS:[404338],EAX
00401086	E8 E5020000 CALL avtst2.00401070
0040108B	85C0 TEST EAX,EAX
0040108D	75 0D JNZ SHORT avtst2.0040109C
0040108F	6A 01 PUSH 1
00401091	E8 1A050000 CALL avtst2.00401030
00401096	59 POP ECX
00401097	E9 9B000000 JMP avtst2.00401130
0040109C	> C745 FC 000000 MOV DWORD PTR SS:[EBP-4],0
004010A3	E8 18050000 CALL avtst2.00401500
004010A8	E8 D3050000 CALL avtst2.00401680
004010AD	E8 1E060000 CALL avtst2.004016D0
004010B2	E8 290A0000 CALL avtst2.00401AE0
004010B7	E8 C40A0000 CALL avtst2.00401B80
004010BC	BB 90304000 MOV EBX,avtst2.00403090
004010C1	81FB 90304000 CMP EBX,avtst2.00403090
004010C7	73 0D JNB SHORT avtst2.004010D6

EBP=0018FF94

- References
- Hardware Breakpoints
- Software Breakpoints
- Call Stack
- Show CPU window
- Show Handles
- Window Info
- Show Threads
- Show Memory Map
- Show Loaded modules
- Show log

Intro to Assembly

Common Instructions:

Registers:

are sections of memory that can be quickly accessed on the CPU die
EIP (Instruction pointer) and ESP (Stack pointer) are used for pointing to locations in memory while the majority of the other registers are used for general purposes

There is also a flags register that can state various information about the CPU

Stack:

A section of memory that contains currently used data

Intro to Assembly

Registers:

EAX	Primary Accumulator - stores function return values
ECX	Count Register - Counter for string and loop operations
EDX	Data Register - I/O pointer
EBX	Base Register - Base pointer to the data section
ESP	Stack pointer
EBP	Base Pointer - Stack frame base pointer
ESI	Source Index- Source pointer for string operations
EDI	Destination Index - Destination pointer for string operations
EIP	Instruction Pointer - Pointer to next instruction to execute

Intro to Assembly

Common Instructions:

NOP - No Operation

PUSH - Moves a word/Dword/Qword or register (not EIP) onto the stack

POP - Removes a Dword off the stack and puts it in a register

CALL - Transfers control to a different function in a way that control can be returned back

(A call can take place using an absolute or relative address)

RET - Used to return from a function

MOV - Can move a register to a memory / memory to a register, an immediate to register / immediate to memory

LEA - copy the result of one operand (register/memory/address/constant) to another

CMP - Compares two operands

JMP - Moves control to absolute or relative address

The following conditional jumps perform a JMP based on the condition of the previous CMP

JE - When equal | JNE - When not equal || JZ - When zero | JNZ - When not zero

JG - When greater than | JGE - When greater than or equal || JL - When less than |

JLE - When less than or equal to

Install GCC

- ▶ Test if there's gcc installation
 - ▶ `#!/locate glibc =>`
 - ▶ It should be `:"/usr/share/man/man7/glibc.7.gz"`
 - ▶ `#!/gcc →` should be:
 - ▶ "gcc: fatal error: no input files" => already installed
 - ▶ To install GCC :
 - ▶ `#!/apt-get install gcc`

Create Hello file.c

```
$ nano file.c
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Haloo");
```

```
    return 0;
```

```
}
```

Open at edb

- ▶ Search for Helloo string and replace with another string
- ▶ Edit the string to another

Search for the Flag

- ▶ R2 [filename]
- ▶ Type 'aa' → to start analyze all
- ▶ Type 'pdf@main' → to find the int main function
- ▶ Find the flag:
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/test>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/wow2>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/test2>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/test3>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/test4>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/test5>

Search For the Flag

- ▶ Open Windows
- ▶ Install the tools:
 - ▶ Immunity Debugger
 - ▶ Die it Easy 0.95
 - ▶ IDAPro
- ▶ Check the Password Flag
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/App1.exe>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/App2.exe>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/App4.exe>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/App5.exe>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/App6.exe>
 - ▶ <http://libra.syailendra.my.id/download/malware-analysis/App7.exe>

Memory Analysis

The background features a series of overlapping, semi-transparent green triangles and polygons of various shades, ranging from light lime green to dark forest green. These shapes are primarily located on the right side of the slide, creating a modern, abstract design.

Volatility

```
jniето@behindthefirewalls:/home/volatility-2.1$ python vol.py -f zeus.vmem pstree
Volatile Systems Volatility Framework 2.1
Name                               Pid  PPid  Thds  Hnds  Time
-----
0x810b1660:System                   4    0     58   379  1970-01-01 00:00:00
. 0xff2ab020:smss.exe                544   4      3    21  2010-08-11 06:06:21
.. 0xff1ec978:winlogon.exe           632  544    24   536  2010-08-11 06:06:23
... 0xff255020:lsass.exe              688  632    21   405  2010-08-11 06:06:24
... 0xff247020:services.exe           676  632    16   288  2010-08-11 06:06:24
.... 0xff1b8b28:vmtoolsd.exe          1668  676     5   225  2010-08-11 06:06:35
..... 0xff224020:cmd.exe                124  1668     0  ----- 2010-08-15 19:17:55
.... 0x80ff88d8:svchost.exe            856  676    29   336  2010-08-11 06:06:24
.... 0xff1d7da0:spoolsv.exe            1432  676    14   145  2010-08-11 06:06:26
.... 0x80fbf910:svchost.exe            1028  676    88  1424  2010-08-11 06:06:24
..... 0x80f60da0:wuauclt.exe            1732  1028     7   189  2010-08-11 06:07:44
..... 0x80f94588:wuauclt.exe            468  1028     4   142  2010-08-11 06:09:37
..... 0xff364310:wscntfy.exe            888  1028     1    40  2010-08-11 06:06:49
.... 0xff217560:svchost.exe            936  676    11   288  2010-08-11 06:06:24
.... 0xff143b28:TPAutoConnSvc.e        1968  676     5   106  2010-08-11 06:06:39
..... 0xff38b5f8:TPAutoConnect.e       1084  1968     1    68  2010-08-11 06:06:52
.... 0xff22d558:svchost.exe            1088  676     7    93  2010-08-11 06:06:25
.... 0xff218230:vmacthlp.exe           844  676     1    37  2010-08-11 06:06:24
.... 0xff25a7e0:alg.exe                216  676     8   120  2010-08-11 06:06:39
.... 0xff203b80:svchost.exe            1148  676    15   217  2010-08-11 06:06:26
.... 0xff1fdc88:VMUpgradeHelper        1788  676     5   112  2010-08-11 06:06:38
.. 0xff1ecda0:csrss.exe               608  544    10   410  2010-08-11 06:06:23
0xff3865d0:explorer.exe             1724  1708    13   326  2010-08-11 06:09:29
. 0xff374980:VMwareUser.exe          452  1724     8   207  2010-08-11 06:09:32
. 0xff3667e8:VMwareTray.exe          432  1724     1    60  2010-08-11 06:09:31
```


Download the images

- ▶ <http://libra.syailendra.my.id/download/malware-analysis/cridex.zip>
- ▶ <http://libra.syailendra.my.id/download/malware-analysis/zaptftis.rar>

Volatility

- ▶ `./vol.py imageinfo -f <Destination of the memory Dump>`
- ▶ `./vol.py -profile=WinXPSP2x86 pslist -f <Destination of the memory Dump>` → show all running process
- ▶ `./vol.py -profile=WinXPSP2x86 kdbgscan -f <Destination of the memory Dump>` → show kernel debugger block (show hidden process)
- ▶ `./vol.py -profile=WinXPSP2x86 kpcrscan -f <Destination of the memory Dump>` → show processor specific data
- ▶ `./vol.py -profile=WinXPSP2x86 dlllist -f <Destination of the memory Dump>` → show all running dll
- ▶ `./vol.py -profile=WinXPSP2x86 dlldump -D <Destination Directory> -f <memory image location>` → Dump all DLL into folder

Volatility

- ▶ `./vol.py -profile=WinXPSP2x86 psscan-D <Destination Directory> -f <memory image location>` → scan all process
- ▶ `./vol.py -profile=WinXPSP2x86 -f <memory image location>` → Show all process in a tree
- ▶ `./vol.py -profile=WinXPSP2x86 connection -f <memory image location>`
→ Show all running connection. `./vol.py -profile=WinXPSP2x86 sockets -f <memory image location>` → show all open sockets (ports)
- ▶ `./vol.py -profile=WinXPSP2x86 hivescan -f <memory image location>` → search for any injected process
- ▶ `./vol.py -profile=WinXPSP2x86 hivelist -f <memory image location>` → search for any injected process on virtual memory
- ▶ `./vol.py -profile=WinXPSP2x86 svcscan -f <memory image location>` → show all services on memory

Thank You

The background features abstract, overlapping geometric shapes in various shades of green, ranging from light lime to dark forest green. These shapes are primarily located on the right side of the frame, creating a modern, layered effect against the white background.