REVERSE ENGINEERING DYNAMIC ANALYSIS

OXFORD BROOKES UNIVERSITY

COMP6016

Malware Analysis

Dr Muhammad Hilmi Kamarudin

WHY IS A DEBUGGER NEEDED?

OXFORD BROOKES UNIVERSITY

- Disassembler gives **static** results
  - Good overview of program logic
  - But need to “mentally execute” program
  - Difficult to jump to specific place in the code
- Debugger is **dynamic**
  - Can set break points
  - Can treat complex code as “black box”
  - Not all code disassembles correctly
- Disassembler **and** debugger both required for any serious Reverse Engineering task

SOURCE-LEVEL V. ASSEMBLY-LEVEL DEBUGGERS

OXFORD BROOKES UNIVERSITY

- Source-level debugger
  - Usually built into development platform
  - Can set breakpoints (which stop at lines of code)
  - Can step through program one line at a time
- Assembly-level debuggers (low-level)
  - Operate on assembly code rather than source code
  - Malware analysts are usually forced to use them, because they don't have source code

DYNAMIC PROGRAM ANALYSIS

OXFORD BROOKES UNIVERSITY

- Run the program and see what it is doing.
- Requires:
  - Dedicated machine - Not connected to the internet.
  - Or: Virtual machine.
    - However: Code can recognize whether it is running in VMWare.
  - Transport malware on a non-writable CD / DVD

DYNAMIC PROGRAM ANALYSIS

OXFORD BROOKES UNIVERSITY

- Run the programming, but keep track of the system calls that it makes with parameters.
  - More relevant calls (Unix):
    - open
    - read
    - write
    - Unlink
    - lstat
    - socket
    - close
  - Strace is a diagnostic, debugging and instructional userspace utility for Linux and has an option to intercepts all network related calls.

- Use fport, netstat, … to determine ports opened by the program.
- On Windows systems.
  - Use regmon
  - Use ListDlls
  - Use psList
    - to find out processes created by program.

DYNAMIC PROGRAM ANALYSIS

OXFORD BROOKES UNIVERSITY

- Intercept communication of program.
  - Need to generate a fake network.
  - E.g.: Static analysis reveals that the program tries to contact www.evil.org on the IRC port.
  - Hence, name an additional machine on separated net www.evil.org.

- Run program on a debugger.
  - Cutter
  - IDA Pro
  - OllyDbg
  - SoftIce

SANDBOX

OXFORD BROOKES UNIVERSITY

- All-in-one software for basic dynamic analysis
- Virtualized environment that simulates network services
- Examples: Norman Sandbox, GFI Sandbox, Anubis, Joe Sandbox, Cuckoo Sandbox, etc
- They are expensive but easy to use
- They produce a nice PDF report of results

SANDBOX AND VIRTUAL MACHINE

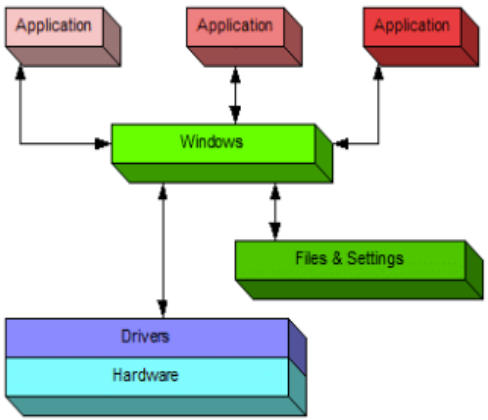


Fig 1: Windows: A conceptual diagram

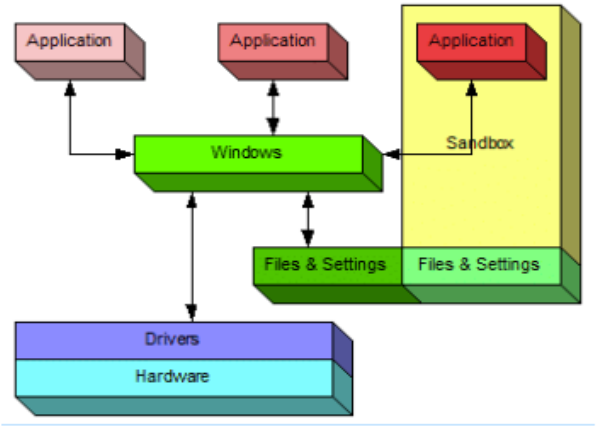


Fig 2: Windows Sandbox: A conceptual diagram

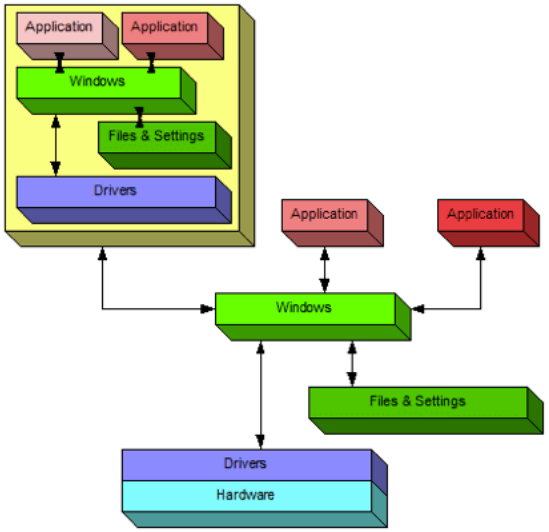


Fig 3: Windows VM: A conceptual diagram

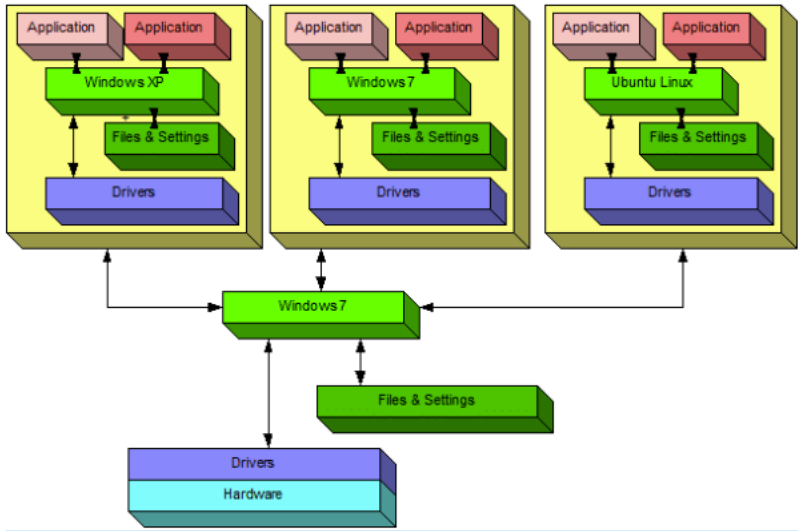


Fig 4: Windows with multiple VM's

OXFORD BROOKES UNIVERSITY

PROS AND CONS

- Sandbox not require additional RAM or space
- Fairly easy to setup
- Need to do little bit of config to get downloaded files from the sandbox
- Perfect isolated “virtual” machine
- Can run different OS than its host
- Virtual Machine- resource hungry.
- Involves installing OS from scratch

RUNNING MALWARES - LAUNCHING DLLS

- EXE files can be run directly, but DLLs can't
- Use Rundll32.exe (included in Windows)

  rundll32.exe *DLLname, Export arguments*

- Example
  - rip.dll has these exports: **Install** and **Uninstall**

  rundll32.exe rip.dll, Install

- Some functions use **ordinal** values instead of names, like

  rundll32.exe xyzzy.dll, #5

- It's also possible to modify the Portable Executable (PE) header and convert a DLL into an EXE

PROCESS MONITOR

- Monitors registry, file system, network, process, and thread activity
- All recorded events are kept, but you can filter the display to make it easier to find items of interest
- Don't run it too long or it will fill up all RAM and crash the machine

LAUNCHING CALC.EXE

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time of Day	Process Name	PID	Operation	Path	Result	Detail
1:17:48.5991893 PM	Explorer.EXE	3188	RegOpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\...	SUCCESS	Desired Access: Query Value
1:17:48.5992018 PM	Explorer.EXE	3188	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersi...	SUCCESS	
1:17:48.5998061 PM	Explorer.EXE	3188	CloseFile	C:\Windows\winsxs\x86_microsoft.windows.common-...	SUCCESS	
1:17:48.6001092 PM	calc.exe	2072	RegOpenKey	HKLM\Software\Microsoft\Windows\Windows Error ...	SUCCESS	Desired Access: Query Value
1:17:48.6001273 PM	calc.exe	2072	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows\Windows Err...	SUCCESS	Type: REG_DWORD, Length: 4, ...
1:17:48.6001350 PM	calc.exe	2072	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows\Windows Err...	SUCCESS	
1:17:48.6001722 PM	calc.exe	2072	ReadFile	C:\Windows\System32\calc.exe	SUCCESS	Offset: 103,424, Length: 32,768, I...
1:17:48.6011060 PM	calc.exe	2072	CreateFile	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	Desired Access: Read Attributes, ...
1:17:48.6011278 PM	calc.exe	2072	QueryBasicInfor...	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	CreationTime: 7/13/2009 4:29:14 ...
1:17:48.6011337 PM	calc.exe	2072	CloseFile	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	
1:17:48.6012132 PM	calc.exe	2072	CreateFile	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	Desired Access: Read Data/List ...
1:17:48.6012344 PM	calc.exe	2072	CreateFileMapp...	C:\Windows\System32\WindowsCodecs.dll	FILE LOCKED WI...	SyncType: SyncTypeCreateSecti...
1:17:48.6012901 PM	calc.exe	2072	CreateFileMapp...	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	SyncType: SyncTypeOther
1:17:48.6013372 PM	calc.exe	2072	Load Image	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	Image Base: 0x73aa0000, Image ...
1:17:48.6013796 PM	calc.exe	2072	CloseFile	C:\Windows\System32\WindowsCodecs.dll	SUCCESS	
1:17:48.6015378 PM	calc.exe	2072	RegOpenKey	HKCU\Software\Classes	SUCCESS	Desired Access: Maximum Allowe...
1:17:48.6015591 PM	calc.exe	2072	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
1:17:48.6015697 PM	calc.exe	2072	RegOpenKey	HKCU\Software\Classes\CLSID\{FAE3D380-FEA4-4...	NAME NOT FOUND	Desired Access: Read
1:17:48.6015797 PM	calc.exe	2072	RegOpenKey	HKCR\CLSID\{FAE3D380-FEA4-4623-8C75-C6B6111...	SUCCESS	Desired Access: Read
1:17:48.6015937 PM	calc.exe	2072	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
1:17:48.6016002 PM	calc.exe	2072	RegOpenKey	HKCU\Software\Classes\CLSID\{FAE3D380-FEA4-4...	NAME NOT FOUND	Desired Access: Read
1:17:48.6016130 PM	calc.exe	2072	RegOpenKey	HKCR\CLSID\{FAE3D380-FEA4-4623-8C75-C6B6111...	NAME NOT FOUND	Desired Access: Read

Showing 128,723 of 253,268 events (50%) | Backed by virtual memory

PROCESS MONITOR TOOLBAR

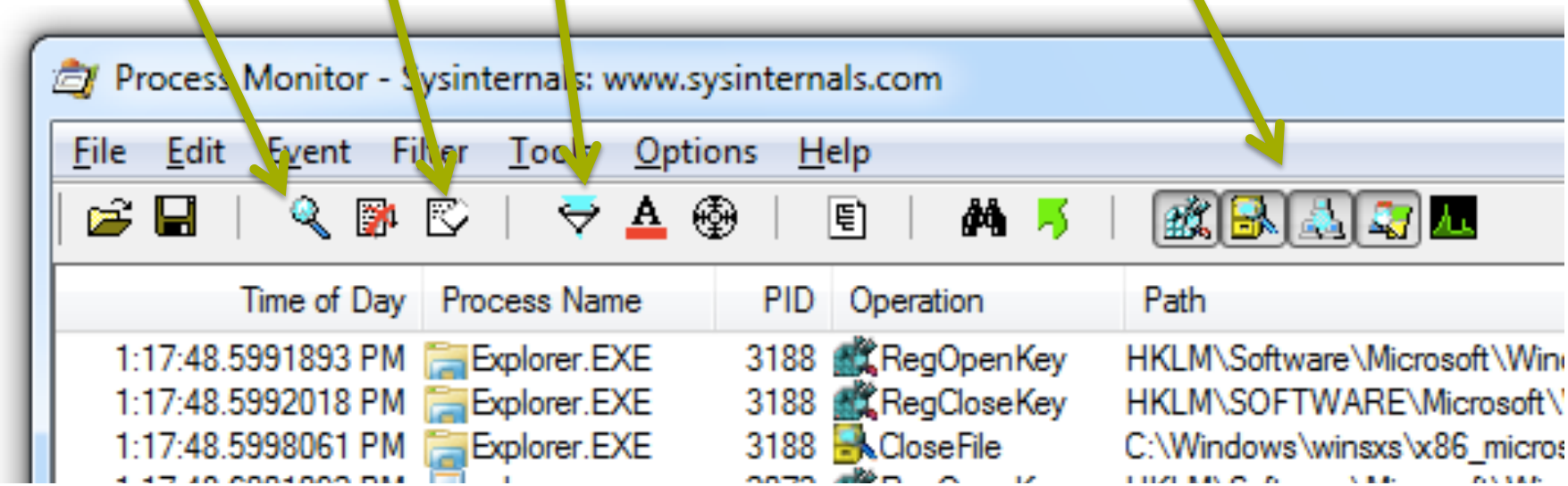
OXFORD BROOKES UNIVERSITY

Start/Stop Capture

Erase

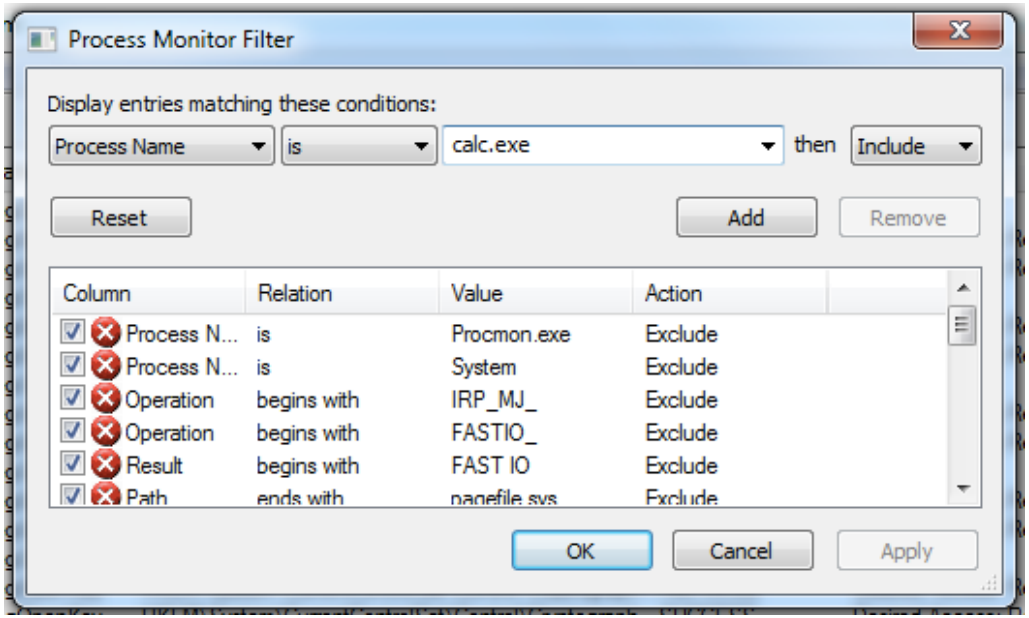
Filter

Default Filters
Registry, File system, Network, Processes



FILTERING EXCLUDE / INCLUDE

- Exclude
  - One technique: hide normal activity before launching malware
  - Right-click each Process Name and click **Exclude**
- Include
  - Most useful filters: Process Name, Operation, and Detail



VIEWING PROCESSES WITH PROCESS EXPLORER

OXFORD BROOKES UNIVERSITY

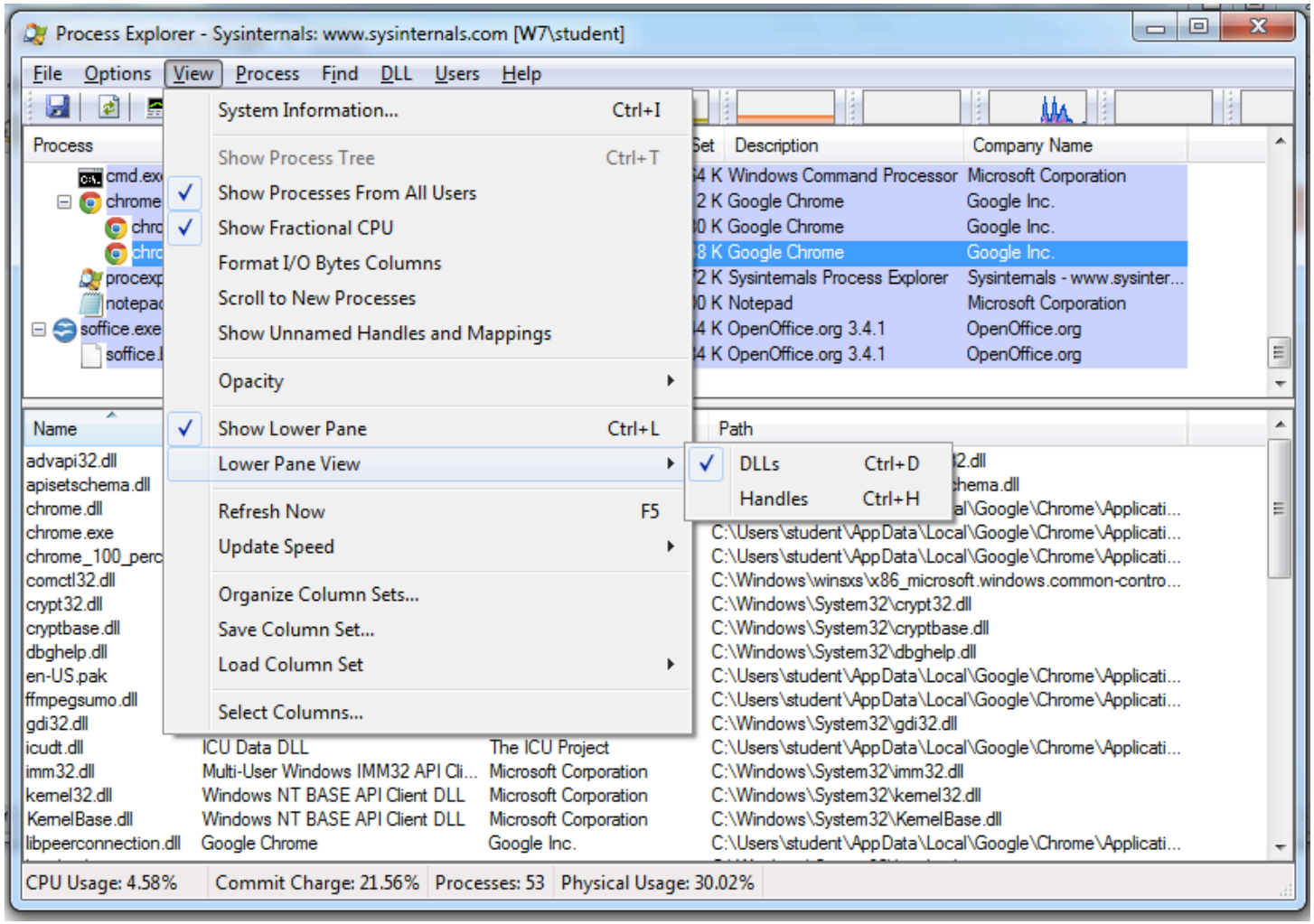
Process Explorer - Sysinternals: www.sysinternals.com [W7\student]

File Options View Process Find Users Help

Process	PID	CPU	Private Bytes	Working Set	Description	Company Name
System Idle Process	0	96.81	0 K	24 K		
System	4	0.09	48 K	560 K		
Interrupts	n/a	0.88	0 K	0 K	Hardware Interrupts and DPCs	
smss.exe	260		224 K	748 K	Windows Session Manager	Microsoft Corporation
csrss.exe	348	< 0.01	1,252 K	3,164 K	Client Server Runtime Process	Microsoft Corporation
wininit.exe	400		892 K	3,084 K	Windows Start-Up Application	Microsoft Corporation
services.exe	504	0.01	3,972 K	6,640 K	Services and Controller app	Microsoft Corporation
svchost.exe	652		2,700 K	6,024 K	Host Process for Windows S...	Microsoft Corporation
dllhost.exe	1716		6,176 K	4,804 K	COM Surrogate	Microsoft Corporation
WmiPrvSE.exe	740		1,804 K	4,736 K	WMI Provider Host	Microsoft Corporation
svchost.exe	724	< 0.01	2,972 K	6,012 K	Host Process for Windows S...	Microsoft Corporation
svchost.exe	772		13,776 K	11,760 K	Host Process for Windows S...	Microsoft Corporation
audiodg.exe	3200		14,960 K	13,972 K	Windows Audio Device Grap...	Microsoft Corporation
svchost.exe	912		37,940 K	42,292 K	Host Process for Windows S...	Microsoft Corporation
dwm.exe	3248	0.74	61,892 K	27,976 K	Desktop Window Manager	Microsoft Corporation
svchost.exe	936	0.02	20,836 K	29,900 K	Host Process for Windows S...	Microsoft Corporation
svchost.exe	1116	0.03	5,136 K	8,340 K	Host Process for Windows S...	Microsoft Corporation
svchost.exe	1260	0.06	10,840 K	11,960 K	Host Process for Windows S...	Microsoft Corporation
spoolsv.exe	1352		5,392 K	7,436 K	Spooler SubSystem App	Microsoft Corporation
svchost.exe	1388		6,752 K	8,720 K	Host Process for Windows S...	Microsoft Corporation
svchost.exe	1500		2,472 K	4,712 K	Host Process for Windows S...	Microsoft Corporation
gogoc.exe	1592	< 0.01	1,216 K	3,920 K	gogoCLIENT	gogo6, Inc.
vmtoolsd.exe	1728	0.07	7,260 K	10,368 K	VMware Tools Core Service	VMware, Inc.
svchost.exe						

CPU Usage: 3.19% | Commit Charge: 21.92% | Processes: 57 | Physical Usage: 30.24%

DLL MODE

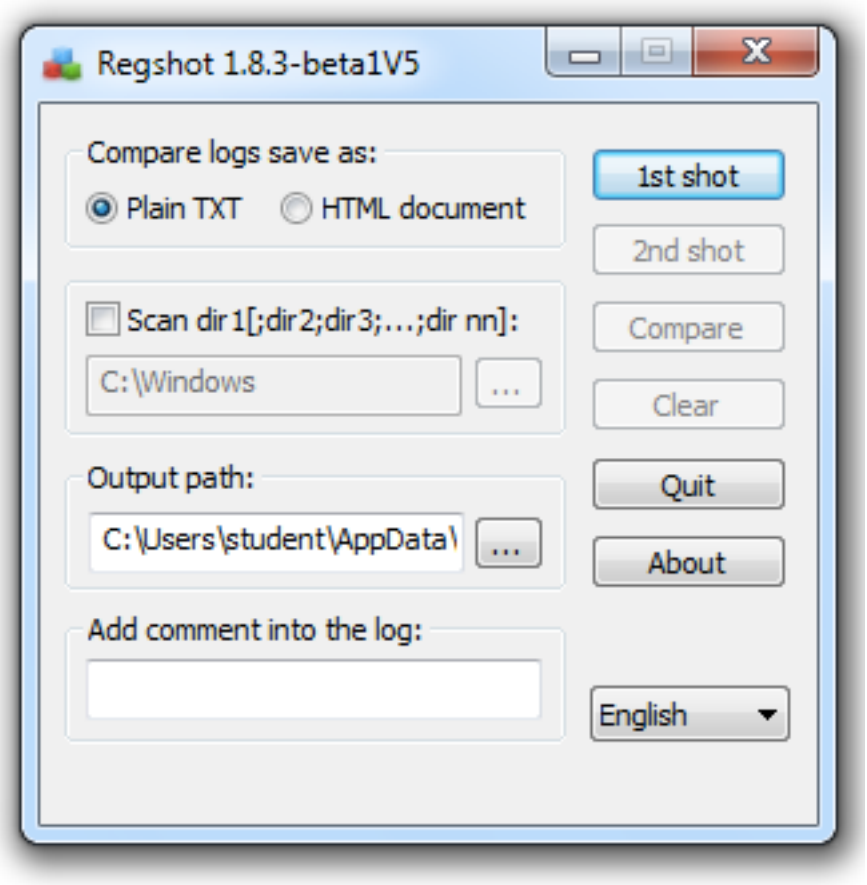

Process Explorer - Sysinternals: www.sysinternals.com [W7\student]
File Options View Process Find DLL Users Help
System Information... Ctrl+I
Show Process Tree Ctrl+T
Show Processes From All Users
Show Fractional CPU
Format I/O Bytes Columns
Scroll to New Processes
Show Unnamed Handles and Mappings
Opacity
Show Lower Pane Ctrl+L
Lower Pane View
DLLs Ctrl+D
Handles Ctrl+H
Refresh Now F5
Update Speed
Organize Column Sets...
Save Column Set...
Load Column Set
Select Columns...
Process
Description
Company Name
Name
Path
CPU Usage: 4.58%
Commit Charge: 21.56%
Processes: 53
Physical Usage: 30.02%

DETECTING MALICIOUS DOCUMENTS

OXFORD BROOKES UNIVERSITY

- Open the document (e.g. PDF) on a system with a vulnerable application
- Watch Process Explorer to see if it launches a process
- The Image tab of that process's Properties sheet will show where the malware is

COMPARING REGISTRY SNAPSHOTS WITH REGSHOT



KERNEL V. USER-MODE DEBUGGING

OXFORD BROOKES UNIVERSITY

- User mode:
  - Debugger runs on the same system as the code being analyzed
  - Debugging a single executable
  - Separated from other executables by the OS
- Kernel mode:
  - Requires two computers, because there is only one kernel per computer
  - If the kernel is at a breakpoint, the system stops
  - One computer runs the code being debugged
  - Other computer runs the debugger
  - OS must be configured to allow kernel debugging
  - Two machines must be connected

USING A DEBUGGER - TWO WAYS

- Start the program with the debugger
  - It stops running immediately prior to the execution of its entry point
- Attach a debugger to a program that is already running
  - All its threads are paused
  - Useful to debug a process that is affected by malware
  - Further read: http://www.qnx.com/developers/docs/qnxcar2/index.jsp?topic=%2Fcom.qnx.doc.neutrino.prog%2Ftopic%2Fusing_gdb_AlreadyRunning.html

STEPPING-OVER V. STEPPING-INTO

- Single step executes one instruction
- **Step-over** call instructions
  - Completes the call and returns without pausing
  - Decrease the amount of code you need to analyze
  - Might miss important functionality, especially if the function never returns
- **Step-into** a call
  - Moves into the function and stops at its first command

PAUSING EXECUTION WITH BREAKPOINTS

OXFORD BROOKES UNIVERSITY

- A program that is paused at a **breakpoint** is called **broken**
- Example
  - You can't tell where this call is going
  - Set a breakpoint at the call and see what's in eax

Example 9-3. Call to EAX

```
00401008   mov     ecx, [ebp+arg_0]
0040100B   mov     eax, [edx]
0040100D   call    eax
```

PAUSING EXECUTION WITH BREAKPOINTS

OXFORD BROOKES UNIVERSITY

- This code calculates a filename and then creates the file
- Set a breakpoint at CreateFileW and look at the stack to see the filename

Example 9-4. Using a debugger to determine a filename

```
0040100B  xor     eax, esp
0040100D  mov     [esp+0D0h+var_4], eax
00401014  mov     eax, edx
00401016  mov     [esp+0D0h+NumberOfBytesWritten], 0
0040101D  add     eax, 0FFFFFFFEh
00401020  mov     cx, [eax+2]
00401024  add     eax, 2
00401027  test    cx, cx
0040102A  jnz     short loc_401020
0040102C  mov     ecx, dword ptr ds:a_txt ; ".txt"
00401032  push    0                ; hTemplateFile
00401034  push    0                ; dwFlagsAndAttributes
00401036  push    2                ; dwCreationDisposition
00401038  mov     [eax], ecx
0040103A  mov     ecx, dword ptr ds:a_txt+4
00401040  push    0                ; lpSecurityAttributes
00401042  push    0                ; dwShareMode
00401044  mov     [eax+4], ecx
00401047  mov     cx, word ptr ds:a_txt+8
0040104E  push    0                ; dwDesiredAccess
00401050  push    edx              ; lpFileName
00401051  mov     [eax+8], cx
00401055 1call    CreateFileW ; CreateFileW(x,x,x,x,x,x,x)
```

ENCRYPTED DATA

- Suppose malware sends encrypted network data
- Set a breakpoint before the data is encrypted and view it

Example 9-5. Using a breakpoint to view data before the program encrypts it

```
004010D0  sub     esp, 0CCh
004010D6  mov     eax, dword_403000
004010DB  xor     eax, esp
004010DD  mov     [esp+0CCh+var_4], eax
004010E4  lea     eax, [esp+0CCh+buf]
004010E7  call    GetData
004010EC  lea     eax, [esp+0CCh+buf]
004010EF  call    EncryptData
004010F4  mov     ecx, s
004010FA  push    0                ; flags
004010FC  push    0C8h             ; len
00401101  lea     eax, [esp+0D4h+buf]
00401105  push    eax              ; buf
00401106  push    ecx              ; s
00401107  call    ds:Send
```

EXCEPTIONS

- Used by debuggers to gain control of a running program
- Breakpoints generate exceptions
- Exceptions are also caused by
  - Invalid memory access
  - Division by zero
  - Other conditions

FIRST- AND SECOND-CHANCE EXCEPTIONS

OXFORD BROOKES UNIVERSITY

- When a exception occurs while a debugger is attached
  - The program stops executing
  - The debugger is given **first chance** at control
  - Debugger can either handle the exception, or pass it on to the program
  - If it's passed on, the program's exception handler takes it
- If the application doesn't handle the exception
- The debugger is given a **second chance** to handle it
  - This means the program would have crashed if the debugger were not attached
- In malware analysis, first-chance exceptions can usually be ignored
- Second-chance exceptions cannot be ignored
  - They usually mean that the malware doesn't like the environment in which it is running

COMMON EXCEPTIONS

OXFORD BROOKES UNIVERSITY

- INT 3 (Software breakpoint)
- Single-stepping in a debugger is implemented as an exception
  - If the **trap flag** in the flags register is set,
  - The processor executes one instruction and then generates an exception
- Memory-access violation exception
  - Code tries to access a location that it cannot access, either because the address is invalid or because of access-control protections
- Violating Privilege Rules
  - Attempt to execute privileged instruction with outside privileged mode
  - In other words, attempt to execute a kernel mode instruction in user mode
  - Or, attempt to execute Ring 0 instruction from Ring 3

MODIFYING EXECUTION WITH A DEBUGGER

OXFORD BROOKES UNIVERSITY

- Skipping a function
  - You can change control flags, the instruction pointer, or the code itself
  - You could avoid a function call by setting a breakpoint where at the call, and then changing the instruction pointer to the instruction after it
    - This may cause the program to crash or malfunction

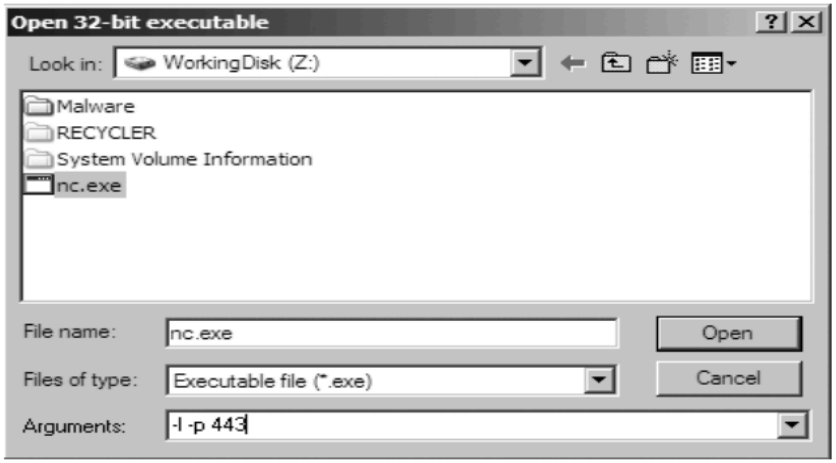
- Testing a function
  - You could run a function directly, without waiting for the main code to use it
    - You will have to set the parameters
    - This destroys a program's stack
    - The program won't run properly when the function completes

OLLYDBG - OVERVIEW

- OllyDbg was developed more than a decade ago
- First used to crack software and to develop exploits
- The OllyDbg 1.1 source code was purchased by Immunity and rebranded as Immunity Debugger
- The two products are very similar
- You can load EXEs or DLLs directly into OllyDbg
- If the malware is already running, you can attach OllyDbg to the running process

OPENING AN EXE

- File, Open
- Add command-line arguments if needed

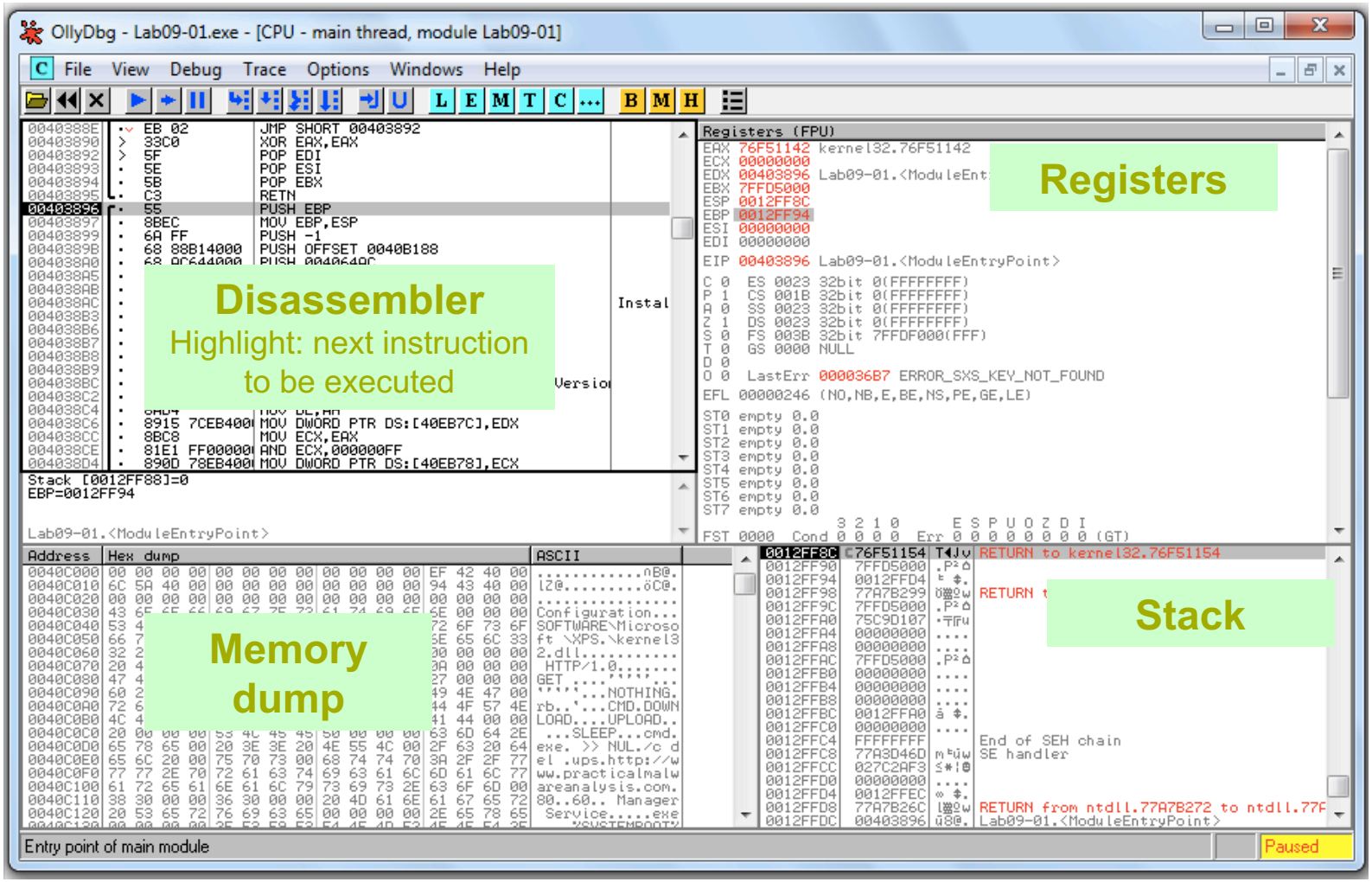


- OllyDbg will stop at the entry point, WinMain, if it can be determined
- Otherwise it will break at the entry point defined in the PE Header
  - Configurable in Options, Debugging Options

ATTACHING TO A RUNNING PROCESS

- File, Attach
- OllyDbg breaks in and pauses the program and all threads
  - If you catch it in DLL, set a breakpoint on access to the entire code section to get to the interesting code

THE OLLYDBG INTERFACE



MEMORY MAP

Memory map

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00010000				Map	RW	RW	
00020000	00010000				Map	RW	RW	
0012D000	00001000				Priv	RW Gua	RW Gua	
0012E000	00002000			Stack of main thr	Priv	RW	RW	
00130000	00004000				Map	R	R	
00140000	00001000				Priv	RW	RW	
00150000	00067000				Map	R	R	\Device\HarddiskVolume1\Windows\System32\locale.nls
001C0000	00001000				Priv	RW	RW	
001D0000	00001000				Priv	RW	RW	
00240000	00003000				Priv	RW	RW	
002A0000	00008000				Priv	RW	RW	
00400000	00001000	Lab09-01		PE header	Img	R	RWE Cop	
00401000	0000A000	Lab09-01	.text	Code	Img	R E	RWE Cop	
0040B000	00001000	Lab09-01	.rdata	Imports	Img	R	RWE Cop	
0040C000	00005000	Lab09-01	.data	Data	Img	RW Cop	RWE Cop	
00420000	00005000				Map	R	R	
004E0000	00003000				Map	R	R	
004F0000	00101000			GDI handles	Map	R	R	
00600000	0008B000				Map	R	R	
75C60000	00001000	KERNELBA		PE header	Img	R	RWE Cop	
75C61000	00044000	KERNELBA			Img	R E	RWE Cop	
75CA5000	00002000	KERNELBA			Img	RW	RWE Cop	
75CA7000	00004000	KERNELBA			Img	R	RWE Cop	
75EB0000	00001000	NSI		PE header	Img	R	RWE Cop	
75EB1000	00002000	NSI			Img	R E	RWE Cop	
75EB3000	00001000	NSI			Img	RW	RWE Cop	
75EB4000	00002000	NSI			Img	R	RWE Cop	
75EC0000	00001000	SHELL32		PE header	Img	R	RWE Cop	
75EC1000	003C8000	SHELL32			Img	R E	RWE Cop	
76289000	00007000	SHELL32			Img	RW Cop	RWE Cop	
76290000	00879000	SHELL32			Img	R	RWE Cop	
76B10000	00001000	USER32		PE header	Img	R	RWE Cop	
76B11000	00068000	USER32			Img	R E	RWE Cop	
76B79000	00001000	USER32			Img	RW	RWE Cop	
76B7A000	0005F000	USER32			Img	R	RWE Cop	
76BE0000	00001000	sechost		PE header	Img	R	RWE Cop	
76BE1000	00013000	sechost			Img	R E	RWE Cop	
76BF4000	00003000	sechost			Img	RW Cop	RWE Cop	
76BF7000	00002000	sechost			Img	R	RWE Cop	

- EXE and DLLs are identified
- Double-click any row to show a memory dump
- Right-click, View in Disassembler

VIEWING THREADS AND STACKS

- View, Threads
- Right-click a thread to "Open in CPU", etc.

Threads

Ord	Ident	Window's title	Last error	Entry	TIB	Suspend	Priority	User time	System time
Main	00000F34	Cisco Packet Trac	ERROR_SUCCESS (00		7FFDF000	0.	Normal	1.1544 s	0.2964 s
2.	00000488		ERROR_SUCCESS (00	7029345E	7FFDE000	0.	Normal	0.0000 s	0.0000 s
3.	000007C4		ERROR_SUCCESS (00	76E6EB16	7FFDD000	0.	Normal	0.0000 s	0.0000 s
4.	00000414		ERROR_SUCCESS (00	76E6D34E	7FFDC000	0.	Normal	0.0000 s	0.0000 s
5.	00000A80		ERROR_SUCCESS (00	76E6D34E	7FFDB000	0.	Normal	0.0000 s	0.0000 s
6.	0000093C		ERROR_SUCCESS (00	768DC89D	7FFDA000	0.	Normal	0.0000 s	0.0000 s
7.	000008C8		ERROR_SUCCESS (00	749E6F14	7FFD9000	0.	High	0.0000 s	0.0000 s

EACH THREAD HAS ITS OWN STACK

- Visible in Memory Map

Memory map

Address	Size	Owner	Section	Contains	Type	Access	Initial
05050000	00800000				Priv	RW	RW
05850000	00A80000				Priv	RW	RW
06820000	003FC000				Map	R	R
06D1D000	00002000				Priv	RW Gua	RW Gua
06D1F000	00001000			Stack of thread 2. (00000488)	Priv	RW	RW
06E1D000	00002000				Priv	RW Gua	RW Gua
06E1F000	00001000			Stack of thread 3. (000007C4)	Priv	RW	RW
06F10000	00BBD000				Priv	RW	RW
07AD0000	006B5000				Priv	RW	RW
0828D000	00002000				Priv	RW Gua	RW Gua
0828F000	00001000			Stack of thread 4. (00000414)	Priv	RW	RW
0838D000	00002000				Priv	RW Gua	RW Gua
0838F000	00001000			Stack of thread 5. (00000A80)	Priv	RW	RW
0848C000	00002000				Priv	RW Gua	RW Gua
0848E000	00002000			Stack of thread 6. (0000093C)	Priv	RW	RW
0858D000	00002000				Priv	RW Gua	RW Gua
0858F000	00001000			Stack of thread 7. (000008C8)	Priv	RW	RW
08630000	00019000				Priv	RW	RW
08670000	0021F000				Map	RW	RW
088B0000	01C57000				Priv	RW	RW
0A510000	001E6000				Priv	RW	RW

EXECUTING CODE

Table 10-1. OllyDbg Code-Execution Options

Function	Menu	Hotkey	Button
Run/Play	Debug ▸ Run	F9	
Pause	Debug ▸ Pause	F12	
Run to selection	Breakpoint ▸ Run to Selection	F4	
Run until return	Debug ▸ Execute till Return	CTRL-F9	
Run until user code	Debug ▸ Execute till User Code	ALT-F9	
Single-step/step-into	Debug ▸ Step Into	F7	
Step-over	Debug ▸ Step Over	F8	

RUN AND PAUSE

- You could Run a program and click Pause when it's where you want it to be
- But that's sloppy and might leave you somewhere uninteresting, such as inside library code
- Setting breakpoints is much better

RUN AND RUN TO SELECTION

- Run is useful to resume execution after hitting a breakpoint
- Run to Selection will execute until just before the selected instruction is executed
  - If the selection is never executed, it will run indefinitely

EXECUTE TILL RETURN

- Pauses execution until just before the current function is set to return
- Can be useful if you want to finish the current function and stop
- But if the function never ends, the program will continue to run indefinitely

EXECUTE TILL USER CODE

- Useful if you get lost in library code during debugging
- Program will continue to run until it hit compiled malware code
  - Typically the `.text` section

STEPPING THROUGH CODE

- F7 -- Single-step (also called step-into)
- F8 -- Step-over
  - Stepping-over means all the code is executed, but you don't see it happen
- Some malware is designed to fool you, by calling routines and never returning, so stepping over will miss the most important part

BREAKPOINTS

OXFORD BROOKES UNIVERSITY

Types of Breakpoints

- Software breakpoints
- Hardware breakpoints
- Conditional breakpoints
- Breakpoints on memory

- F2 – Add or remove a breakpoint

SOFTWARE BREAKPOINTS

OXFORD BROOKES UNIVERSITY

- Useful for string decoders
- Malware authors often obfuscate strings
  - With a **string decoder** that is called before each string is used
- Put a breakpoint at the end of the decoder routine
- The string becomes readable on the stack
  Each time you press Play in OllyDbg, the program will execute and will break when a string is decoded for use
- This method will only reveal strings as they are used

Example 10-2. A string decoding breakpoint

```
push offset "4NNpTNHLKIXoPm7iBhUAjvRKNaUVBlr"
call String_Decoder
...
push offset "ugKLdNlLT6emldCeZi72mUjieuBqdfZ"
call String_Decoder
...
```

HARDWARE BREAKPOINTS

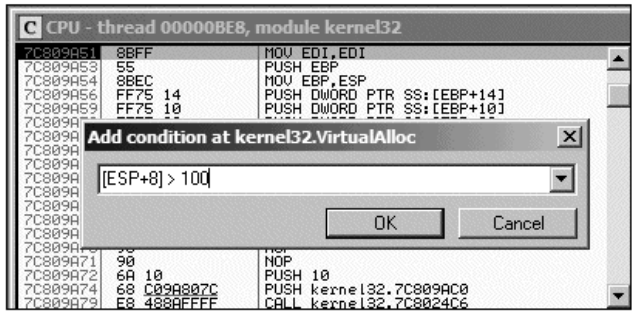
- Don't alter code, stack, or any target resource
- Don't slow down execution
- But you can only set 4 at a time
- Click **Breakpoint**, "**Hardware**, **on Execution**"
- You can set OllyDbg to use hardware breakpoints by default in Debugging Options
  - Useful if malware uses anti-debugging techniques

CONDITIONAL BREAKPOINTS

OXFORD BROOKES UNIVERSITY

- Breaks only when a condition is true
- Ex: Poison Ivy backdoor
  - Poison Ivy allocates memory to house the shellcode it receives from Command and Control (C&C) servers
  - Most memory allocations are for other purposes and uninteresting
  - Set a conditional breakpoint at the VirtualAlloc function in Kernel32.dll

1. Right-click in the disassembler window on the first instruction of the function, and select **Breakpoint ▶ Conditional**. This brings up a dialog asking for the conditional expression.
2. Set the expression and click **OK**. In this example, use **`[ESP+8]>100`**.
3. Click **Play** and wait for the code to break.



MEMORY BREAKPOINTS

OXFORD BROOKES UNIVERSITY

- Code breaks on access to specified memory location
- OllyDbg supports software and hardware memory breakpoints
- Can break on read, write, execute, or any access
- Right-click memory location, click **Breakpoint**, **"Memory, on Access**“
- You can only set one memory breakpoint at a time
- OllyDbg implements memory breakpoints by changing the attributes of memory blocks
- This technique is not reliable and has considerable overhead
- Use memory breakpoints sparingly

VIEWING ACTIVE BREAKPOINTS

OXFORD BROOKES UNIVERSITY

- View, Breakpoints, or click B icon on toolbar

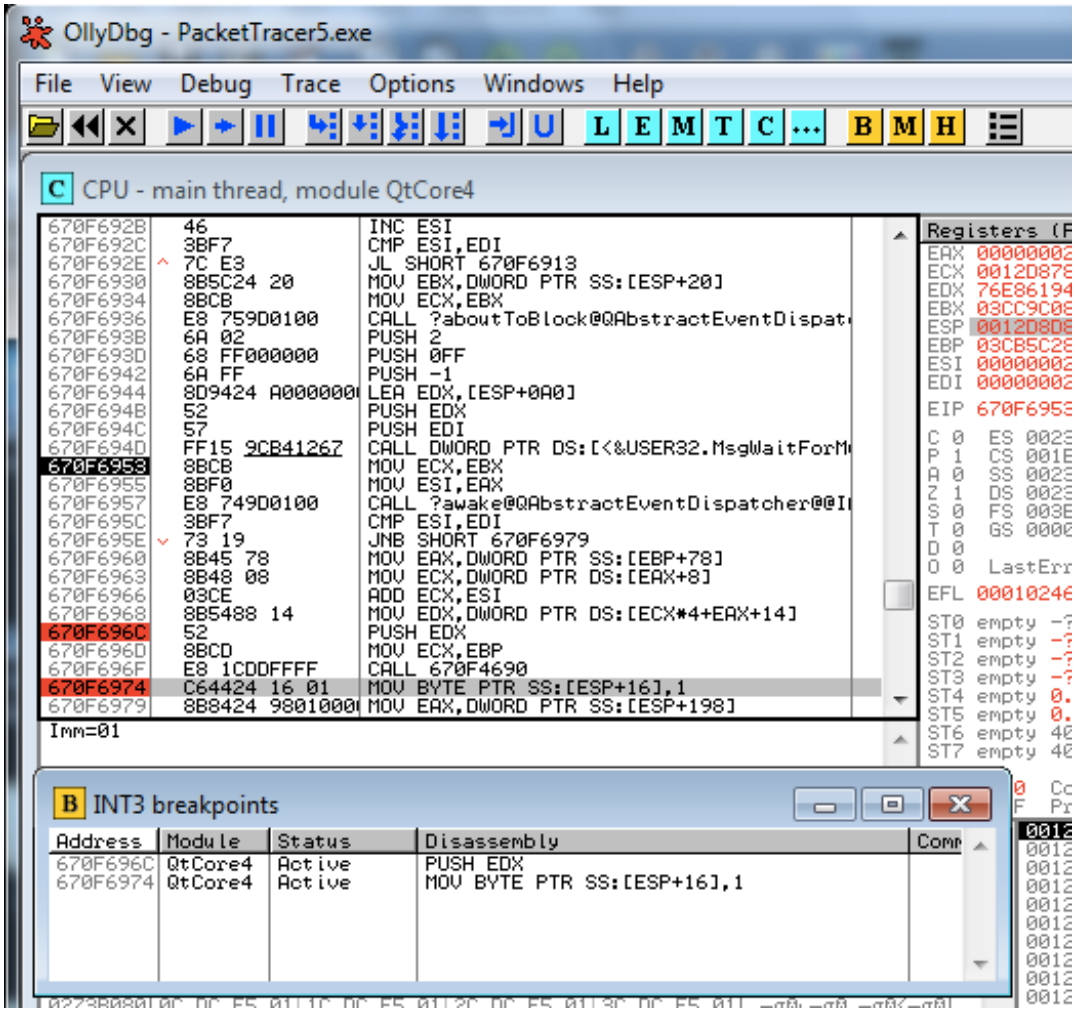


Table 10-2. OllyDbg Breakpoint Options

Function	Right-click menu selection	Hotkey
Software breakpoint	Breakpoint ▸ Toggle	F2
Conditional breakpoint	Breakpoint ▸ Conditional	SHIFT-F2
Hardware breakpoint	Breakpoint ▸ Hardware, on Execution	
Memory breakpoint on access (read, write, or execute)	Breakpoint ▸ Memory, on Access	F2 (select memory)
Memory breakpoint on write	Breakpoint ▸ Memory, on Write	

SAVING BREAKPOINTS

- When you close OllyDbg, it saves your breakpoints
- If you open the same file again, the breakpoints are still available

PATCHING - BINARY EDIT

OXFORD BROOKES UNIVERSITY

OllyDbg - Lab09-01.exe - [CPU - main thread, module ntdll]

File View Debug Plugins Options Window Help

```
77AB4440  8B5424 0C     MOV EDX,DWORD PTR SS:[ESP+C]
77AB4444  8B4C24 04     MOV ECX,DWORD PTR SS:[ESP+4]
77AB4448  85D2          TEST EDX,EDX
77AB444A  74 4F         JE SHORT ntdll.77AB449B
77AB444C  33C0          XOR EAX,EAX
77AB444E  8A4424 08     MOV AL,BYTE PTR SS:[ESP+8]
77AB4452  57            PUSH EDI
77AB4453  8BF9          MOV EDI,ECX
77AB4455  83FA 04       CMP EDX,4
77AB4458  72 31         JB SHORT ntdll.77AB448B
77AB445A  F7D9          NEG ECX
77AB445C  83E1 03       AND ECX,3
77AB445F  74 0C         JE SHORT ntdll.77AB446D
77AB4461  2BD1          SUB EDX,
77AB4463  8807          MOV BYTE
77AB4465  83C7 01       ADD EDI,
77AB4468  83E9 01       SUB ECX,
77AB446B  75 F6         JNZ SHOR
77AB446D  8BC8          MOV ECX,
77AB446F  C1E0 08       SHL EAX,
77AB4472  03C1          ADD EAX,
77AB4474  8BC8          MOV ECX,
77AB4476  C1E0 10       SHL EAX,
77AB4479  03C1          ADD EAX,
77AB447B  8BCA          MOV ECX,
77AB447D  83E2 03       AND EDX,
77AB4480  C1E9 02       SHR ECX,
77AB4483  74 06         JE SHORT
77AB4485  F3:AB         REP STOS
77AB4487  85D2          TEST EDX
77AB4489  74 0A         JE SHORT
77AB448B  8807          MOV BYTE
77AB448D  83C7 01       ADD EDI,
```

Registers (FP

```
EAX 00000000
ECX 00000000
EDX 00000030
EBX 00000000
ESP 0012FD28
EBP 0012FDF8
ESI 00000000
EDI 0012FD98
EIP 77AB4476
C 0  ES 0023
P 1  CS 001B
A 0  SS 0023
Z 1  DS 0023
S 0  FS 003B
T 0  GS 0000
```

- Backup
- Copy
- Binary
  - Edit Ctrl+E
  - Fill with 00's
  - Fill with NOPs
  - Binary copy
  - Binary paste
- Assemble Space
- Label :
- Comment ;
- Breakpoint
- Run trace

DISCOVERING VULNERABILITIES

Two Primary Methods:

1. **Source Code Auditing**
   - Requires source code

2. **Reverse Engineering**
   - Can be done without source code.
   - need binaries
   - hard

OXFORD BROOKES UNIVERSITY

Thank you