Offensive Software Exploitation

SEC-300-01/CSI-301-02

Ali Hadi

@binaryz0ne

Bug Hunting…

a quick road to bug hunting …

Wait …

- Before we proceed into exploitation, do you know what we mean by a:
  - “Vulnerability” or “Security hole” ?

- *INFOSEC 101* …. ☺

Bug Hunting

- Bug hunting is the process of finding bugs in software or hardware “cited [1]”

- Security bugs (aka software security vulnerabilities and security holes) allows attackers to:
  - Remotely compromise systems
  - Escalate local privileges
  - Cross privilege boundaries
  - Wreak havoc on a system!

4 Fun & Profit

- Finding security bugs was done for fun and to get media attention
- Today, organizations are paying for security researchers to identify bugs
  - Bounty programs (Google, FaceBook, Twitter, RedHat, etc)
  - Zero Day Initiative (ZDI)
  - iDefense
  - Tipping Point
  - Pwn2Own
  - Others? Please add

Taking Advantages of Bugs

- Software that take the advantages of a software vulnerability are called “exploits”
- Exploiting a widely used application, os, protocol, etc … will grab huge media coverage and attention
  - Road to become a Hacking Star ☺

Exploits Language

- No specific language for writing exploits
- Exploits can be written using any programming language
  - C, C++, Perl, JavaScript, Assembly, and PYTHON !
- I prefer Python for it's simplicity and for the huge range of libraries that could be used for creating a PoC or working exploit

Bug Hunting Formal Process

- Writing software is a *human art*, and two different coders may code the same function with the same requirements differently!
- For that reason IMO, Bug Hunting is a *human art* too!
- No formal process to finding bugs in SW, but there are a couple of techniques that can be used for bug discovery

Common Techniques

- Static Analysis
  - Static Code Analysis
  - Reverse Engineering
- Dynamic Analysis
  - Debugging
  - Fuzzing
- Each technique has it’s pros and cons
  - Bug hunters mix it up

General Bug Hunting Methodology

Understand the Application

- Read specs / documentation
  - understand purpose or business logic
- Examine attack surface
  - inputs, configuration
- Identify target components an attacker would hit
  - think like an attacker to defend better:
    - try to hit the Database for SQLi?
    - try to upload a file?
    - try to spawn a shell?

What Leads to Bugs?

- Miscalculations
- Failure to validate input
- Programmer failure to understand an API
- Failure to validate results: operations, functions, etc
- Application state failures
- Complex protocols
- Complex file formats
- Complex encoding / decoding / expansion
- etc

Debugging…

Debugger

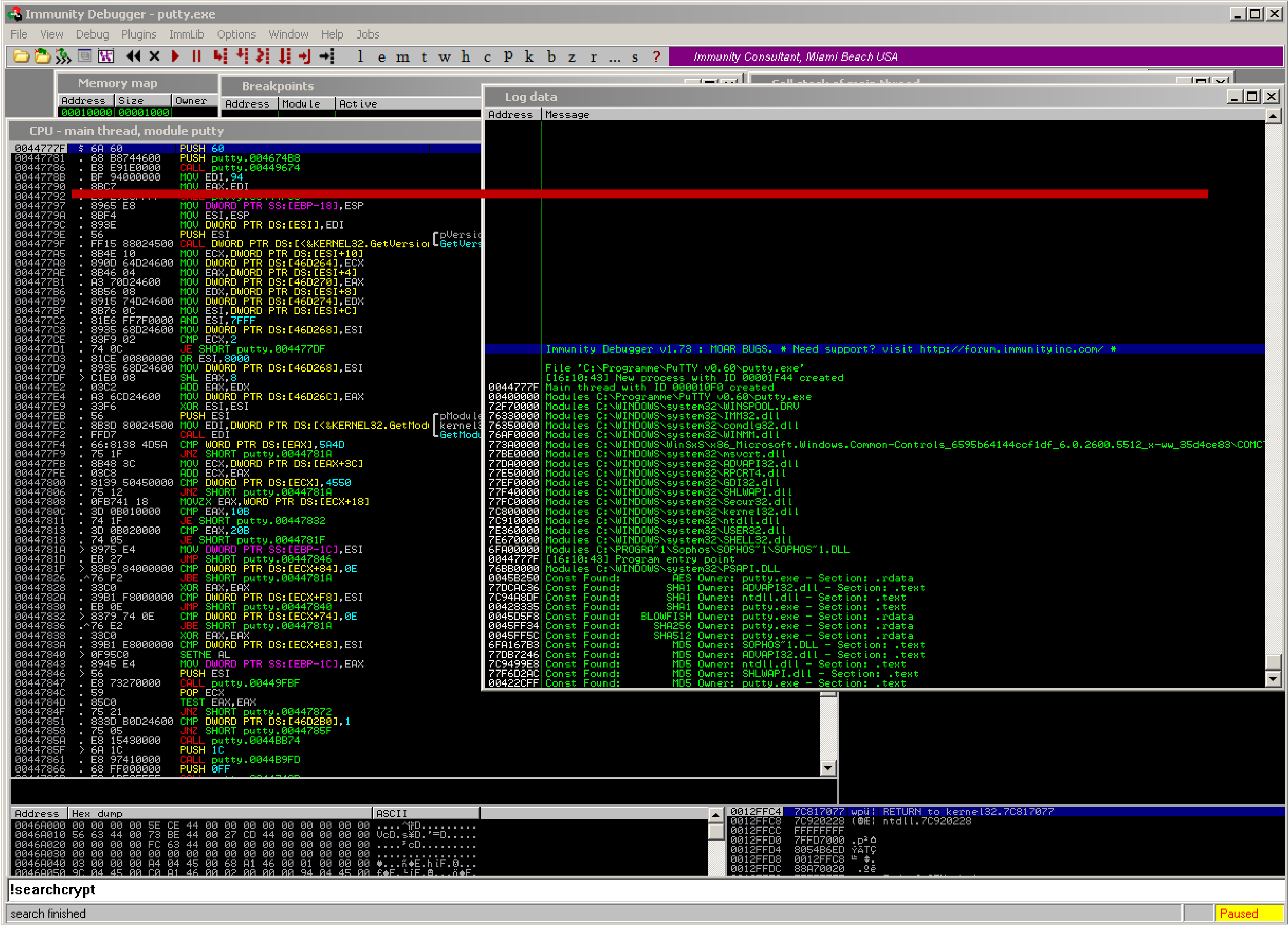
- A computer program that lets you run your program, line by line and examine the values of variables or look at values passed into functions and let you figure out why it isn't running the way you expected it to.

Why use Debuggers

- Debuggers offer sophisticated functions such as:
  - Running a program step by step (single-stepping mode),
  - Stopping (breaking) (pausing the program to examine the current state) at some event or specified instruction by means of a breakpoint,
  - Tracking the values of variables,
  - Tracking the values of CPU registers,
  - Attach to a process,
  - View the process's Memory map,
  - Load memory dump (post-mortem debugging),
  - Disassemble program instructions,
  - Change values at runtime,
  - Continue execution at a different location in the program to bypass a crash or logical error.

Common Debuggers

- GNU Debugger (GDB)
- Microsoft Windows Debugger (Windbg)
- OllyDbg
- Immunity Debugger
  - Based on Ollydbg
- Microsoft Visual Studio Debugger
- Interactive DisAssembler (IDA Pro)



Immunity Debugger - putty.exe

File View Debug Plugins ImmLib Options Window Help Jobs

Immunity Consultant, Miami Beach USA

Memory map

```
Address  Size     Owner
00010000 00001000
```

Breakpoints

```
Address Module Active
```

CPU - main thread, module putty

```
0044777F $ 6A 60          PUSH 60
00447781 . 68 B8744600    PUSH putty.004674B8
00447786 . E8 E91E0000    CALL putty.00449674
0044778B . BF 94000000    MOV EDI,94
00447790 . 8BC7           MOV EAX,EDI
00447792 [redacted]
00447797 . 8965 E8        MOV DWORD PTR SS:[EBP-18],ESP
0044779A . 8BF4           MOV ESI,ESP
0044779C . 893E           MOV DWORD PTR DS:[ESI],EDI
0044779E . 56             PUSH ESI                                 [pVersio
0044779F . FF15 88024500  CALL DWORD PTR DS:[<&KERNEL32.GetVersio  LGetVers
004477A5 . 8B4E 10        MOV ECX,DWORD PTR DS:[ESI+10]
004477A8 . 890D 64D24600  MOV DWORD PTR DS:[46D264],ECX
004477AE . 8B46 04        MOV EAX,DWORD PTR DS:[ESI+4]
004477B1 . A3 70D24600    MOV DWORD PTR DS:[46D270],EAX
004477B6 . 8B56 08        MOV EDX,DWORD PTR DS:[ESI+8]
004477B9 . 8915 74D24600  MOV DWORD PTR DS:[46D274],EDX
004477BF . 8B76 0C        MOV ESI,DWORD PTR DS:[ESI+C]
004477C2 . 81E6 FF7F0000  AND ESI,7FFF
004477C8 . 8935 68D24600  MOV DWORD PTR DS:[46D268],ESI
004477CE . 83F9 02        CMP ECX,2
004477D1 . 74 0C          JE SHORT putty.004477DF
004477D3 . 81CE 00800000  OR ESI,8000
004477D9 . 8935 68D24600  MOV DWORD PTR DS:[46D268],ESI
004477DF > C1E0 08        SHL EAX,8
004477E2 . 03C2           ADD EAX,EDX
004477E4 . A3 6CD24600    MOV DWORD PTR DS:[46D26C],EAX
004477E9 . 33F6           XOR ESI,ESI
004477EB . 56             PUSH ESI                                 [pModule
004477EC . 8B3D 80024500  MOV EDI,DWORD PTR DS:[<&KERNEL32.GetMod  [kernel3
004477F2 . FFD7           CALL EDI                                 LGetModu
004477F4 . 66:8138 4D5A   CMP WORD PTR DS:[EAX],5A4D
004477F9 . 75 1F          JNZ SHORT putty.0044781A
004477FB . 8B48 3C        MOV ECX,DWORD PTR DS:[EAX+3C]
004477FE . 03C8           ADD ECX,EAX
00447800 . 8139 50450000  CMP DWORD PTR DS:[ECX],4550
00447806 . 75 12          JNZ SHORT putty.0044781A
00447808 . 0FB741 18      MOVZX EAX,WORD PTR DS:[ECX+18]
0044780C . 3D 0B010000    CMP EAX,10B
00447811 . 74 1F          JE SHORT putty.00447832
00447813 . 3D 0B020000    CMP EAX,20B
00447818 . 74 05          JE SHORT putty.0044781F
0044781A > 8975 E4        MOV DWORD PTR SS:[EBP-1C],ESI
0044781D . EB 27          JMP SHORT putty.00447846
0044781F > 83B9 84000000  CMP DWORD PTR DS:[ECX+84],0E
00447826 .^76 F2          JBE SHORT putty.0044781A
00447828 . 33C0           XOR EAX,EAX
0044782A . 39B1 F8000000  CMP DWORD PTR DS:[ECX+F8],ESI
00447830 . EB 0E          JMP SHORT putty.00447840
00447832 > 8379 74 0E     CMP DWORD PTR DS:[ECX+74],0E
00447836 .^76 E2          JBE SHORT putty.0044781A
00447838 . 33C0           XOR EAX,EAX
0044783A . 39B1 E8000000  CMP DWORD PTR DS:[ECX+E8],ESI
00447840 > 0F95C0         SETNE AL
00447843 . 8945 E4        MOV DWORD PTR SS:[EBP-1C],EAX
00447846 > 56             PUSH ESI
00447847 . E8 73270000    CALL putty.00449FBF
0044784C . 59             POP ECX
0044784D . 85C0           TEST EAX,EAX
0044784F . 75 21          JNZ SHORT putty.00447872
00447851 . 833D B0D24600  CMP DWORD PTR DS:[46D2B0],1
00447858 . 75 05          JNZ SHORT putty.0044785F
0044785A . E8 15430000    CALL putty.0044BB74
0044785F > 6A 1C          PUSH 1C
00447861 . E8 97410000    CALL putty.0044B9FD
00447866 . 68 FF000000    PUSH 0FF
```

Log data

```
Address  Message
         Immunity Debugger v1.73 : MOAR BUGS. * Need support? visit http://forum.immunityinc.com/ *

         File 'C:\Programme\PuTTY v0.60\putty.exe'
         [16:10:43] New process with ID 00001F44 created
0044777F Main thread with ID 000010F0 created
00400000 Modules C:\Programme\PuTTY v0.60\putty.exe
72F70000 Modules C:\WINDOWS\system32\WINSPOOL.DRV
76330000 Modules C:\WINDOWS\system32\IMM32.dll
76350000 Modules C:\WINDOWS\system32\comdlg32.dll
76AF0000 Modules C:\WINDOWS\system32\WINMM.dll
773A0000 Modules C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.5512_x-ww_35d4ce83\COMC
77BE0000 Modules C:\WINDOWS\system32\msvcrt.dll
77DA0000 Modules C:\WINDOWS\system32\ADVAPI32.dll
77E50000 Modules C:\WINDOWS\system32\RPCRT4.dll
77EF0000 Modules C:\WINDOWS\system32\GDI32.dll
77F40000 Modules C:\WINDOWS\system32\SHLWAPI.dll
77FC0000 Modules C:\WINDOWS\system32\Secur32.dll
7C800000 Modules C:\WINDOWS\system32\kernel32.dll
7C910000 Modules C:\WINDOWS\system32\ntdll.dll
7E360000 Modules C:\WINDOWS\system32\USER32.dll
7E670000 Modules C:\WINDOWS\system32\SHELL32.dll
6FA00000 Modules C:\PROGRA~1\Sophos\SOPHOS~1\SOPHOS~1.DLL
0044777F [16:10:43] Program entry point
76BB0000 Modules C:\WINDOWS\system32\PSAPI.DLL
0045B250 Const Found:        AES Owner: putty.exe - Section: .rdata
77DCAC36 Const Found:       SHA1 Owner: ADVAPI32.dll - Section: .text
7C94A8DF Const Found:       SHA1 Owner: ntdll.dll - Section: .text
00428335 Const Found:       SHA1 Owner: putty.exe - Section: .text
0045D5F8 Const Found:   BLOWFISH Owner: putty.exe - Section: .rdata
0045FF34 Const Found:     SHA256 Owner: putty.exe - Section: .rdata
0045FF5C Const Found:     SHA512 Owner: putty.exe - Section: .rdata
6FA167B3 Const Found:        MD5 Owner: SOPHOS~1.DLL - Section: .text
77DB7246 Const Found:        MD5 Owner: ADVAPI32.dll - Section: .text
7C9499E8 Const Found:        MD5 Owner: ntdll.dll - Section: .text
77F6D2AC Const Found:        MD5 Owner: SHLWAPI.dll - Section: .text
00422CFF Const Found:        MD5 Owner: putty.exe - Section: .text
```

```
Address  Hex dump                                         ASCII
0046A000 00 00 00 00 5E CE 44 00 00 00 00 00 00 00 00 00  ....^╬D.........
0046A010 56 63 44 00 73 BE 44 00 27 CD 44 00 00 00 00 00  VcD.s¥D.'=D.....
0046A020 00 00 00 00 FC 63 44 00 00 00 00 00 00 00 00 00  ....³cD.........
0046A030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0046A040 03 00 00 00 A4 04 45 00 68 A1 46 00 01 00 00 00  ♥...ñ♦E.híF.☺...
```

```
0012FFC4  7C817077  wpü|  RETURN to kernel32.7C817077
0012FFC8  7C920228  (☻Æ|  ntdll.7C920228
0012FFCC  FFFFFFFF
0012FFD0  7FFD7000  .p²⌂
0012FFD4  8054B6ED  ÝÂTÇ
0012FFD8  0012FFC8  ╚ ↕.
0012FFDC  88A70020   .ºê
```

!searchcrypt

search finished

Paused

?

- Can we modify executables using a debugger?
- Write an example showing howto modify a Windows EXE file.
  - For example bypass a whole check routine or set of instructions!!!

Determining Exploitability

- This process requires experience of debugging security issues, but some steps can be taken to gain a good idea of how exploitable an issue is...
- Look for any cases where data is written to a controllable address – this is key to controlling code execution and the majority of such conditions will be exploitable
- Verify whether any registers have been overwritten, if they do not contain part data sent from the fuzzer, step back in the disassembly to try and find where the data came from

Determining Exploitability – Cont.

- If the register data is controllable, point the register which caused the crash to a page of memory which is empty, fill that page with data (e.g., 'aaaaa...')
- Repeat and step through each operation, until another crash occurs, reviewing all branch conditions which are controlled by data at the location of the (modified) register to ensure that they are executed

Determining Exploitability – Cont.

- Are saved return address/stack variables overwritten?
- Is the crash in a heap management function?
- Are the processor registers derived from data sent by the fuzzer (e.g. 0x61616161)?
- Is the crash triggered by a read operation?
- Can we craft a test case to avoid this?
- Is the crash triggered by a write operation?
- Do we have full or partial control of the faulting address?
- Do we have full or partial control of the written value?

Language To Use

- Don’t listen to others
- Just choose one, whatever you like
- But if you still need a recommendation? Python is my answer
- Many hack libraries are written in python
- If you use C for fuzzing because you just want to feel you’re l33t, you’re wrong!
- *Realize what is your goal*

SUMMARY

- Described what bug hunting means, and who's hunting nowadays
- What we mean by taking advantages of bugs
- What language to use to write an exploit
- Why there isn't a bug hunting formal process for vulnerability discovery
- Common Techniques for bug hunting
- What and why do we need a debugger and the most popular debuggers

References

- A Bug Hunter’s Diary, Tobias Klein, No Starch Press
- Fuzz Testing, http://en.wikipedia.org/wiki/Fuzz_testing
- Fuzzing: Brute Force Vulnerability Discovery, Michael Sutton, et al, Addison-Wesely
- University of Wisconsin Fuzz Testing (the original fuzz project)
- Fuzzing 101, NYU/Poly.edu, Mike Zusman, http://pentest.cryptocity.net/fuzzing/
- Fuzzing for Security Flaws, John Heasman, Stanford University
- EVERYONE HAS HIS OR HER OWN FUZZER, BEIST (BEISTLAB/GRAYHASH), www.codeengn.com
- An Introduction to SPIKE, the Fuzzer Creation Kit, Dave Aitel, http://www.docstoc.com/docs/2687423/An-Introduction-to-SPIKE-the-Fuzzer-Creation-Kit---PowerPoint
- Common Vulnerablities and Exposures, http://cve.mitre.org/
- Common Weakness Enumeration, http://cwe.mitre.org/
- Seven kingdoms of weaknesses Taxonomy, http://cwe.mitre.org/documents/sources/SevenPerniciousKingdomsTaxonomyGraphic.pdf
- Common Configuration Enumeration, http://cce.mitre.org/
- http://nvd.nist.gov/home.cfm
- Exploit Database, http://exploit-db,com
- http://www.security-database.com/toolswatch/+-Fuzzers-+.html
- http://caca.zoy.org/wiki/zzuf
- https://code.google.com/p/ouspg/wiki/Radamsa