Hacking Techniques & Intrusion Detection

Fall 2012/2013

Dr. Ali Al-Shemery

AKA: B!n@ry

Software Exploitation

Prepared by:

Dr. Ali Al-Shemery

Mr. Shadi Naif

Shellcode

```
/* the Aleph One shellcode */
"\x31\xc0\x31\xdb\xb0\x17\xcd\x80\xeb\x1f\x5e\x89"
"\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb"
"\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

Outline – Part 3

- Introduction
- System Calls
- Shellcode Basics
- Shellcode Types
- Considerations
- Useful Shellcode Tools

Shellcode?

- AKA bytecode
- Small piece of code used as the payload in the exploitation of a software vulnerability.
- Problems of writing shellcodes:
  - Not easy to write
  - Architecture and OS dependent
  - Must remove all string-delimiting characters

System Calls

- Kernel trap calls used by user-space programs to access kernel-space functions.

- Linux:
  - INT \x80, Sysenter, etc
- Windows
  - INT 0x2e, Sysenter, DLL(s), API(s), etc

- System Call # stored in EAX.
- 1st ARG in EBX, 2nd in ECX, and so on.

Shellcode Basics

- Spawning the process
  - Linux/Unix: execve
  - Windows: CreateProcess
- How child process deals with input and output is very important
- File descriptors (regardless of OS):
  - 0 for Standard Input (stdin)
  - 1 for Standard Output (stdout)
  - 2 for Standard Error (stderr)

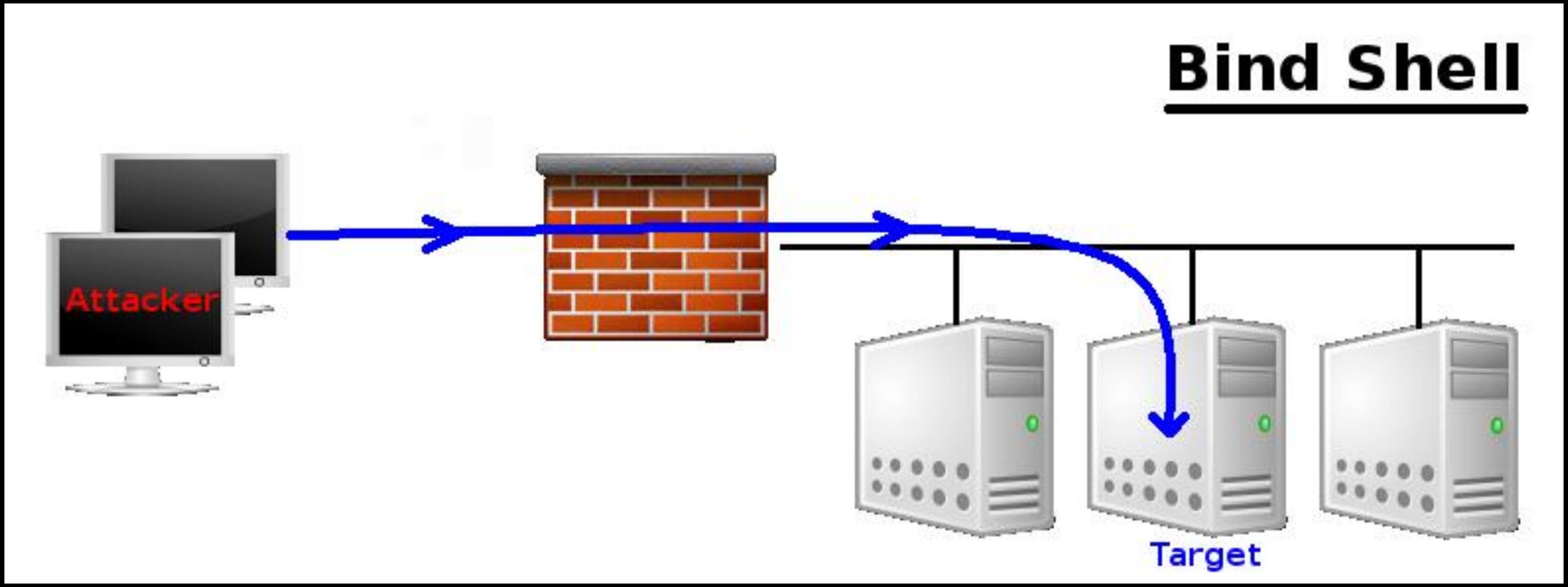
Shellcode Types

- Port Binding
- Reverse
- Find Socket
- Command Execution Code
- File Transfer
- Multistage
- System Call Proxy
- Process Injection
- Kernel Space

Port Binding Shellcode

- AKA “bind shell”
- Why/When to use this type of SC?
- What it does:
  - Create TCP socket
  - Bind socket to port (hardcoded and specified by the attacker)
  - Make socket Listen
  - Dup listening socket onto stdin, stdout, and stderr
  - Spawn command shell (bash, cmd.exe, etc)
- Attacker connects to that port to get control
- Problems:
  - Firewalls
  - Not Invisible
  - Can’t distinguish between connections made to it

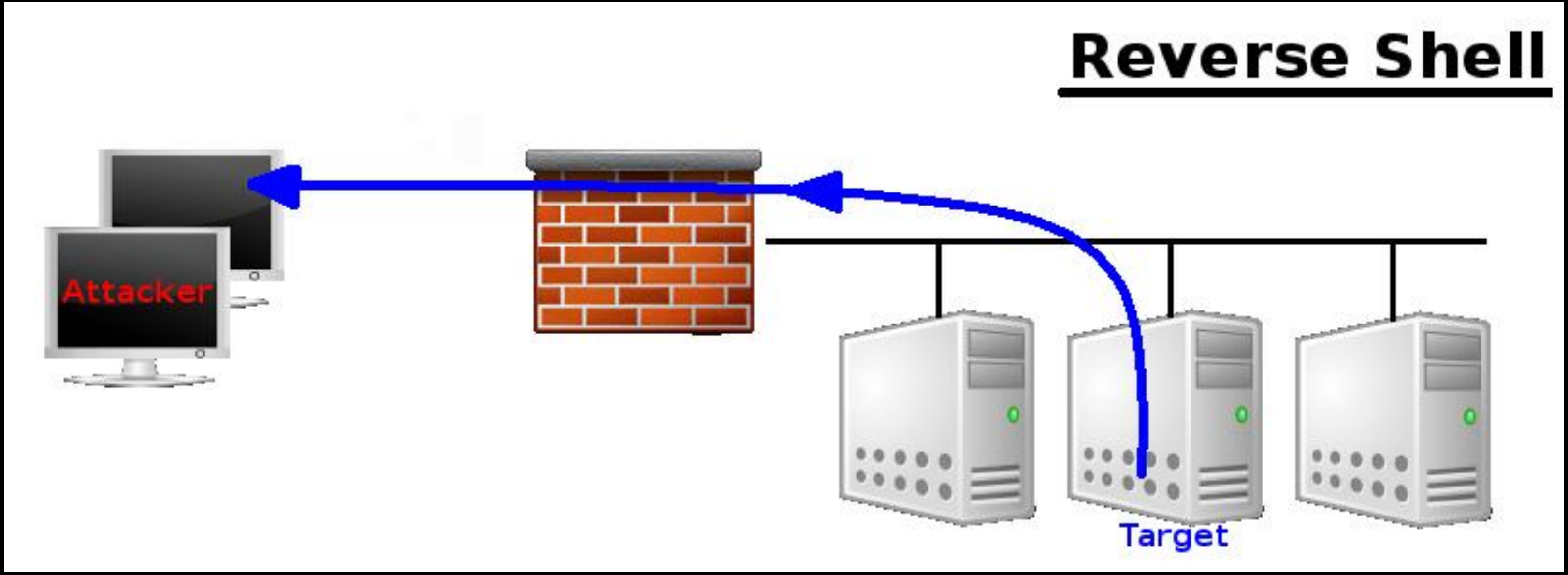
Port Binding Shellcode



Reverse Shellcode

- AKA ‘callback shellcode”, solves bind shell problems
- Why connect to the target, were we can make the target connect to us?
- What it does:
  - Create TCP socket
  - Make socket connect back to the attacker on IP+Port (hardcoded and specified by the attacker)
  - Connect to the IP and port
  - Dup the socket onto stdin, stdout, and stderr
  - Spawn command shell (bash, cmd.exe, etc)
- Problems
  - Outbound Filtering
  - Attacker must be listening on the specified port
  - Attacker behind NAT
  - Target behind some proxy
  - Not invisible too

Reverse Shellcode



Find Socket Shellcode

- Search for the file descriptor that represents attackers connection.
  - POSIX (file descriptors)
  - Windows (File Handlers)
- Query each descriptor to find which is remotely connected to the attackers computer.
- Hardcode the outbound port into the shellcode, makes find much easier on target.
- No new network connection (hard to detect)!

Find Socket Shellcode - 2

- Steps:
  - Find file descriptor for the network connection.
  - Duplicate the socket onto stdin, stdout, and stderr.
  - Spawn a new command shell process (will use original socket for I/O).
- Problem:
  - Attacker behind NAT device, can't control the outbound port from which his connection originated (P.S. won't know what file descriptor is used for his connection!)

Command Execution Shellcode

- Why create a network session when all needed to do is run a command?
  - ssh-copy-id to target
  - Adding/modifying a user account
  - Modify configuration file
- Steps:
  - Assemble command name
  - Assemble arguments required (if any!)
  - Invoke system call to execute the command
- Often very small

File Transfer Shellcode

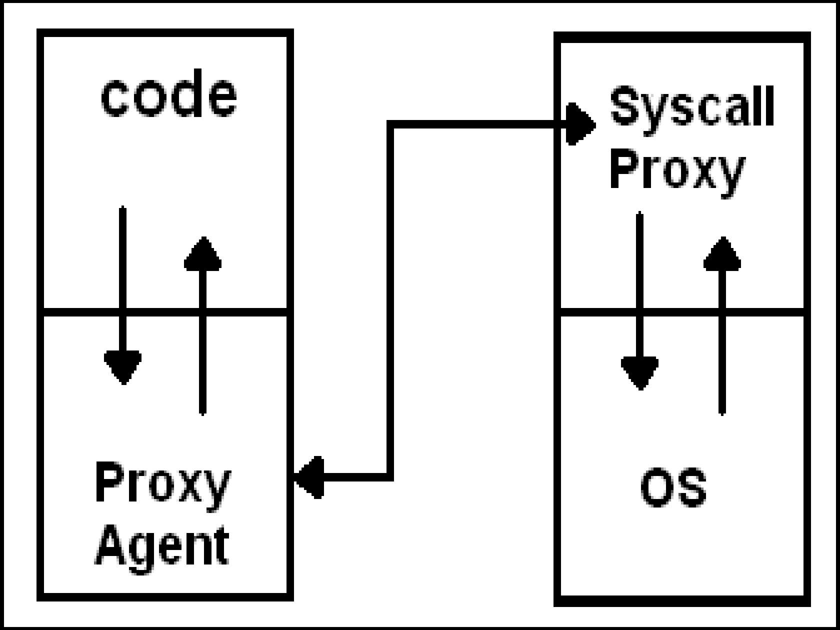
- Very simple, all needed is to upload a file to the target
- Steps:
  - Open new file on target
  - Read data from the network connection, and write it to the opened file (Note: connection obtained using previous discussed network shellcodes)
  - Repeat RW until file successfully transferred.
  - Close the open file
- Can be combined with a CE Shellcode

Multistage Shellcode

- Vulnerability contains un-sufficient space for injecting shellcode
- Consist of 2 or more shellcode stages
- Steps:
  - Stage1:
    - read more shellcode,
    - pass control to Stage2 shellcode
  - Stage2: accomplish the functionality required

System Call Proxy Shellcode

- AKA Syscall Proxy
- Technique first introduced by Maximiliano Caceres (CORE Impact creators) which can provide a real remote interface to the target's kernel
- Local process running has no idea it is running remotely!
- Syscall proxy payload can continue to run in the context of the exploited process.



System Call Proxy – Cont.

- Use many tools without installing anything on the target machine
- Memory resident ← Means What?
- Kernel Interface
- Request Local, Execute Remote
- Remote Debugging
- Others? use your own imagination!

Process Injection Shellcode

- Loading libraries of code running under a separate thread of execution within the context of an existing process on the target.
- Host process can be:
  - Process exploited.
  - Migrate to a complete different process.
- Injected library might never get written to the hard drive and harness in memory (hard even for forensics to discover)!
- Ex: Metasploit's Meterpreter (next week).

Important Stuff

- Disassemble
  - Maybe running a backdoor !
- Encoding
  - Bad char(s) is chasing you!
- Others?
  - Please add …

Assignments – Choose 2

- What is a Kernel Space Shellcode?
- Can we categories Metasploit’s Meterpreter as a Multi-Stage Shellcode?
- How can we debug a shellcode?

Debugging a Shellcode

```
char shellcode[] =
“Insert shellcode/bytecode here";

int main(int argc, char **argv)
{
  int (*func)();
  func = (int (*)()) code;
  (int)(*func)();
}
```

Useful Tools

- GCC: gcc -c shellcode.s
- Objdump: objdump -d shellcode.o
- LD: ld binary.o -o binary
- NASM: nasm -f elf64 shellcode.asm
- strace: trace system calls and signals
- Corelan’s pveWritebin.pl and pveReadbin.pl
- BETA3 --decode
- Ndisasm
- Immunity Debugger
- GDB

Summary

- What Shellcodes are, and problems that face shellcode developers,
- Types of Shellcodes,
- Why it’s important to disassemble a shellcode you didn’t write,
- Why sometimes you need to encode your shellcode,
- List of useful tools related to shellcode development.

References (1)

- Papers/Presentations/Links:
  - ShellCode, http://www.blackhatlibrary.net/Shellcode
  - Introduction to win32 shellcoding, Corelan, http://www.corelan.be/index.php/2010/02/25/exploit-writing-tutorial-part-9-introduction-to-win32-shellcodeing/
  - Hacking/Shellcode/Alphanumeric/x64 printable opcodes, http://skypher.com/wiki/index.php/Hacking/Shellcode/Alphanumeric/x64_printable_opcodes
  - Learning Assembly Through Writing Shellcode, http://www.patternsinthevoid.net/blog/2011/09/learning-assembly-through-writing-shellcode/
  - Shellcoding for Linux and Windows Tutorial, http://www.vividmachines.com/shellcode/shellcode.html
  - Unix Assembly Codes Development, http://pentest.cryptocity.net/files/exploitation/asmcodes-1.0.2.pdf
  - Win32 Assembly Components, http://pentest.cryptocity.net/files/exploitation/winasm-1.0.1.pdf

References (2)

- Papers/Presentations/Links:
  - 64-bit Linux Shellcode, http://blog.markloiseau.com/2012/06/64-bit-linux-shellcode/
  - Writing shellcode for Linux and *BSD, http://www.kernel-panic.it/security/shellcode/index.html
  - Understanding Windows's Shellcode (Matt Miller's, aka skape)
  - Metasploit's Meterpreter (Matt Miller, aka skape)
  - Syscall Proxying fun and applications, csk @ uberwall.org
  - X86 Opcode and Instruction Reference, http://ref.x86asm.net/
  - Shellcode: the assembly cocktail, by Samy Bahra, http://www.infosecwriters.com/hhworld/shellcode.txt

References (3)

- Books:
  - Grayhat Hacking: The Ethical Hacker's Handbook, 3rd Edition
  - The Shellcoders Handbook,
  - The Art of Exploitation, 2nd Edition,
- Shellcode Repositories:
  - Exploit-DB: http://www.exploit-db.com/shellcodes/
  - Shell Storm: http://www.shell-storm.org/shellcode/
- Tools:
  - BETA3 - Multi-format shellcode encoding tool, http://code.google.com/p/beta3/
  - X86 Opcode and Instruction Reference, http://ref.x86asm.net/
  - bin2shell, http://blog.markloiseau.com/wp-content/uploads/2012/06/bin2shell.tar.gz