CAL POLY

CSC 321: Introduction to Computer Security

Module 6: Malicious Software

Bret Hartman

Department of Computer Science and Software Engineering

California Polytechnic State University

E-mail: bahartma@calpoly.edu

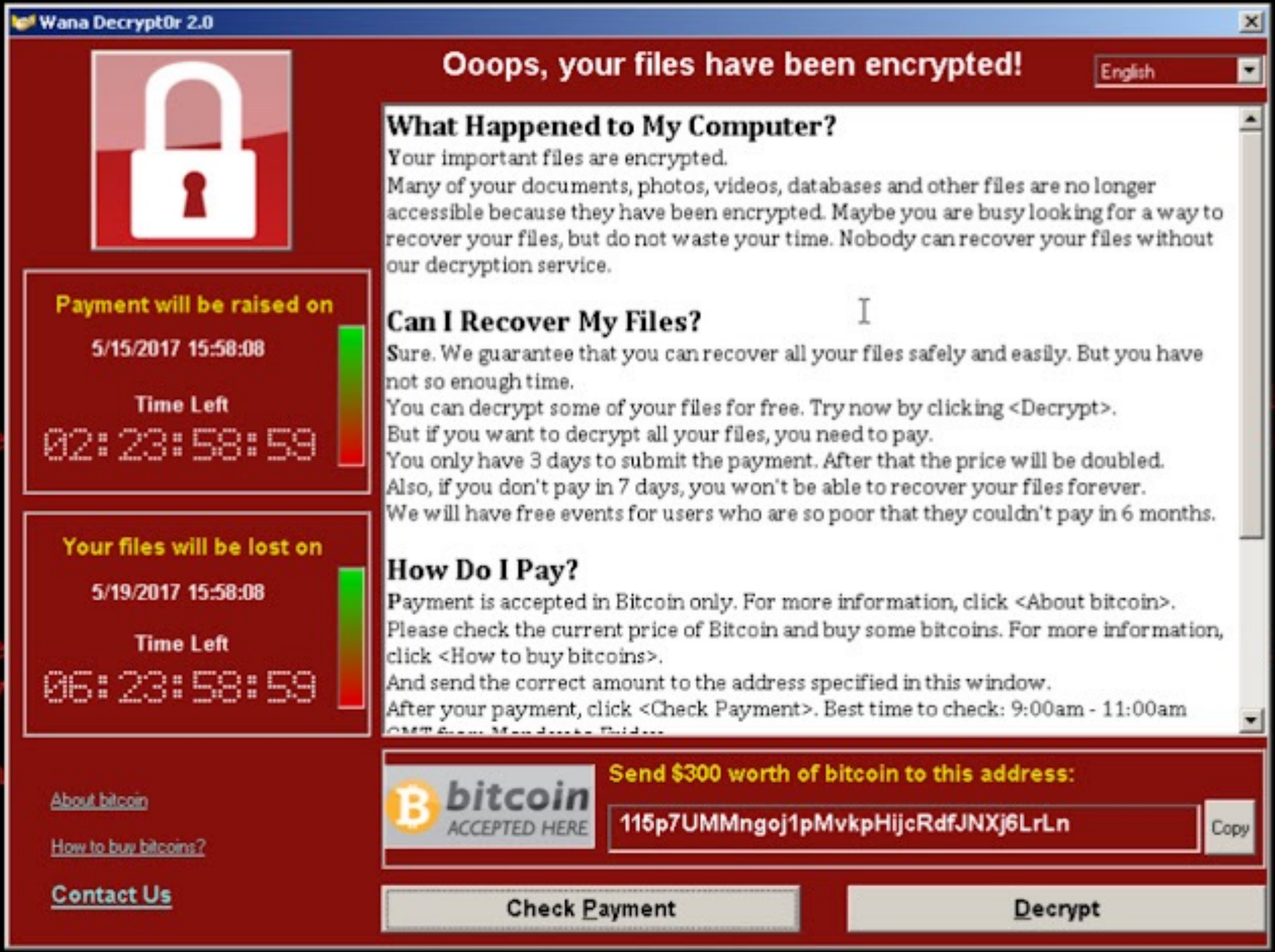
A special thanks to Dr. Bruce DeBruhl and Dr. Phoenix (Dongfeng) Fang, the authors of most of this material

Quiz 5 available
Lab 5 Access control due

What is software security

- The idea of engineering software so that it continues to function correctly even under malicious attack
- Consider the failure of the program, not only the user
- Aims to avoid security vulnerabilities by addressing security from the early stages of software development life cycle
- “Security is risk management”

May 12, 2017



Ooops, your files have been encrypted!

Payment will be raised on

5/15/2017 15:58:08

Time Left

02:23:58:59

Your files will be lost on

5/19/2017 15:58:08

Time Left

06:23:58:59

What Happened to My Computer?

Your important files are encrypted.

Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?

Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.

You can decrypt some of your files for free. Try now by clicking <Decrypt>.

But if you want to decrypt all your files, you need to pay.

You only have 3 days to submit the payment. After that the price will be doubled.

Also, if you don't pay in 7 days, you won't be able to recover your files forever.

We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?

Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.

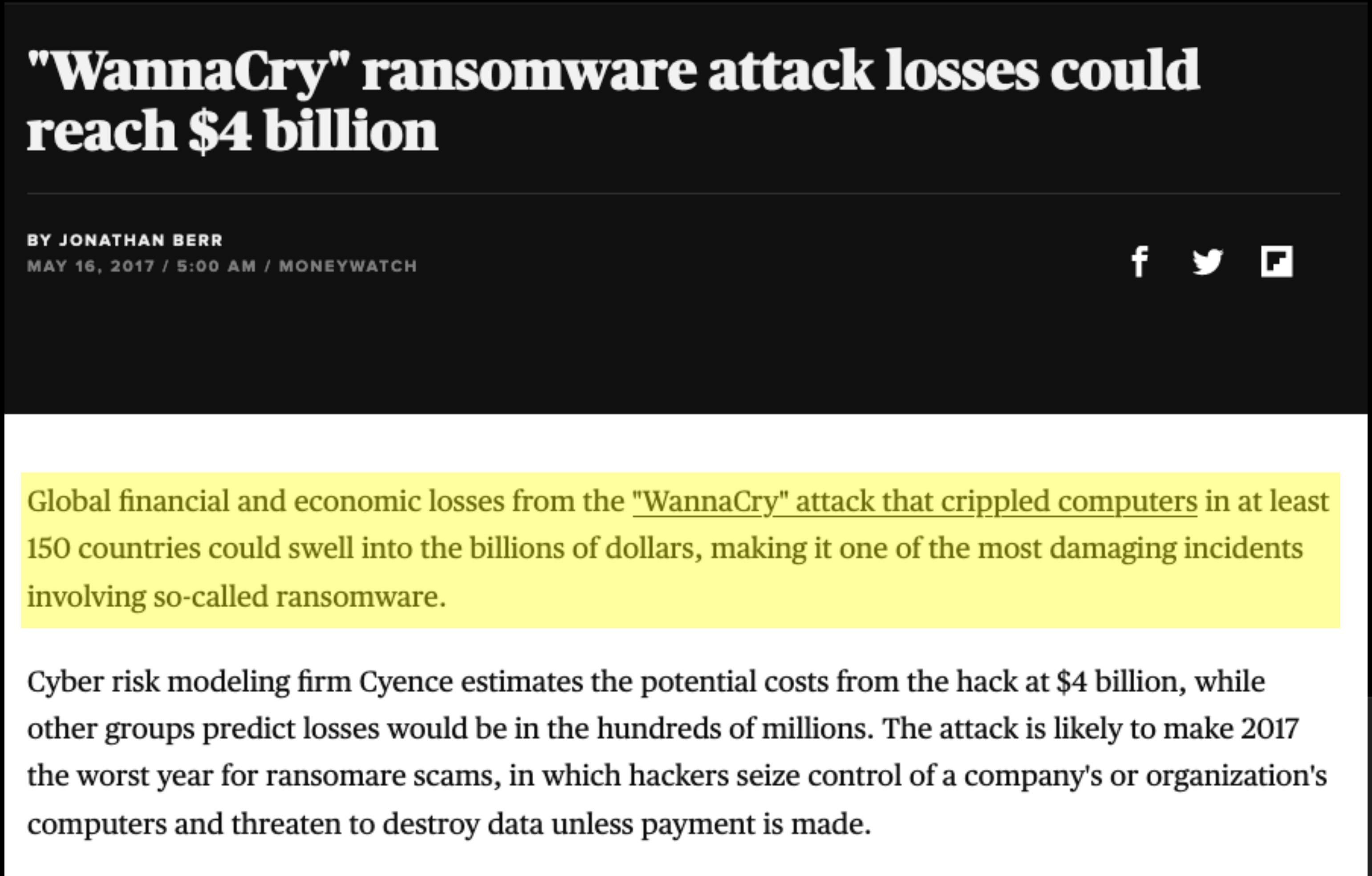
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.

And send the correct amount to the address specified in this window.

After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Send $300 worth of bitcoin to this address:

115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn

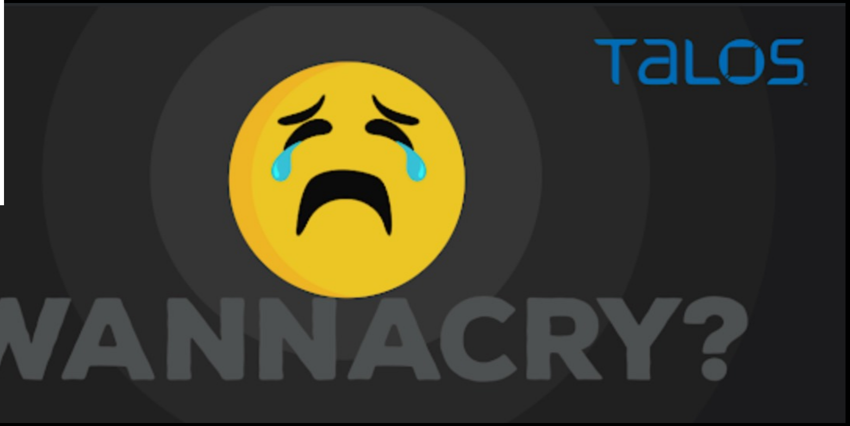


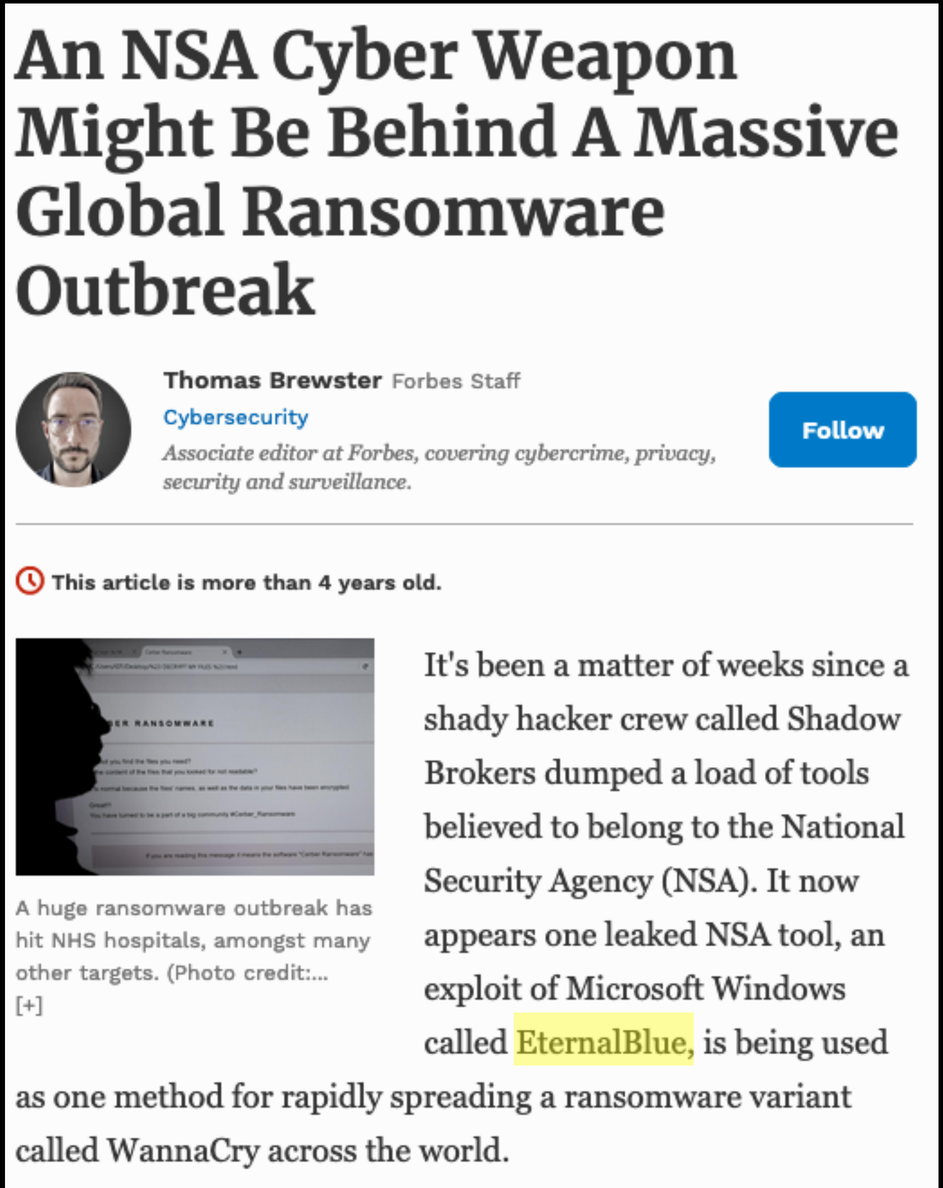
"WannaCry" ransomware attack losses could reach $4 billion

BY JONATHAN BERR
MAY 16, 2017 / 5:00 AM / MONEYWATCH

Global financial and economic losses from the "WannaCry" attack that crippled computers in at least 150 countries could swell into the billions of dollars, making it one of the most damaging incidents involving so-called ransomware.

Cyber risk modeling firm Cyence estimates the potential costs from the hack at $4 billion, while other groups predict losses would be in the hundreds of millions. The attack is likely to make 2017 the worst year for ransomare scams, in which hackers seize control of a company's or organization's computers and threaten to destroy data unless payment is made.





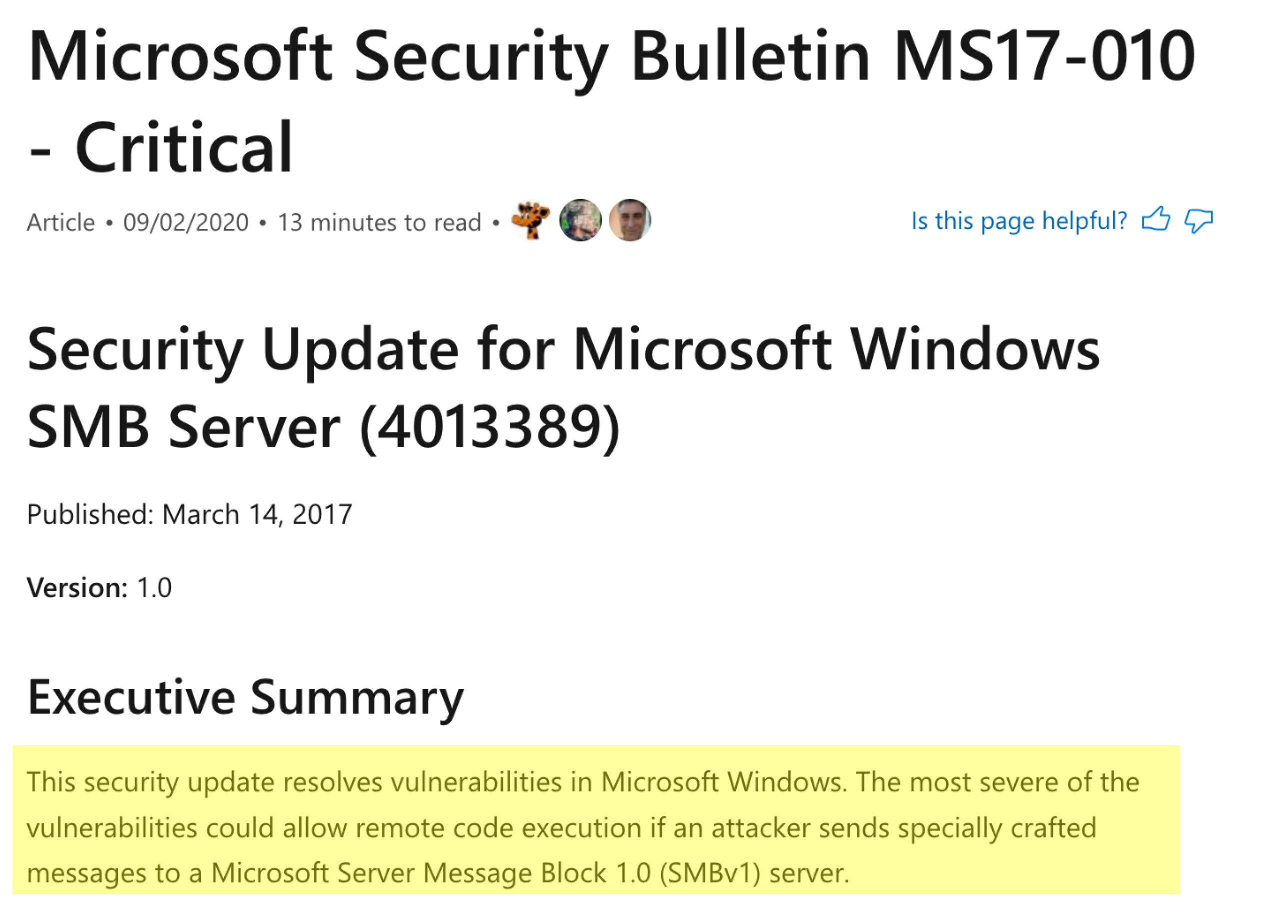
An NSA Cyber Weapon Might Be Behind A Massive Global Ransomware Outbreak

Thomas Brewster Forbes Staff
Cybersecurity
Associate editor at Forbes, covering cybercrime, privacy, security and surveillance.

This article is more than 4 years old.

A huge ransomware outbreak has hit NHS hospitals, amongst many other targets. (Photo credit:... [+]

It's been a matter of weeks since a shady hacker crew called Shadow Brokers dumped a load of tools believed to belong to the National Security Agency (NSA). It now appears one leaked NSA tool, an exploit of Microsoft Windows called EternalBlue, is being used as one method for rapidly spreading a ransomware variant called WannaCry across the world.



Microsoft Security Bulletin MS17-010 - Critical

Article • 09/02/2020 • 13 minutes to read •

Security Update for Microsoft Windows SMB Server (4013389)

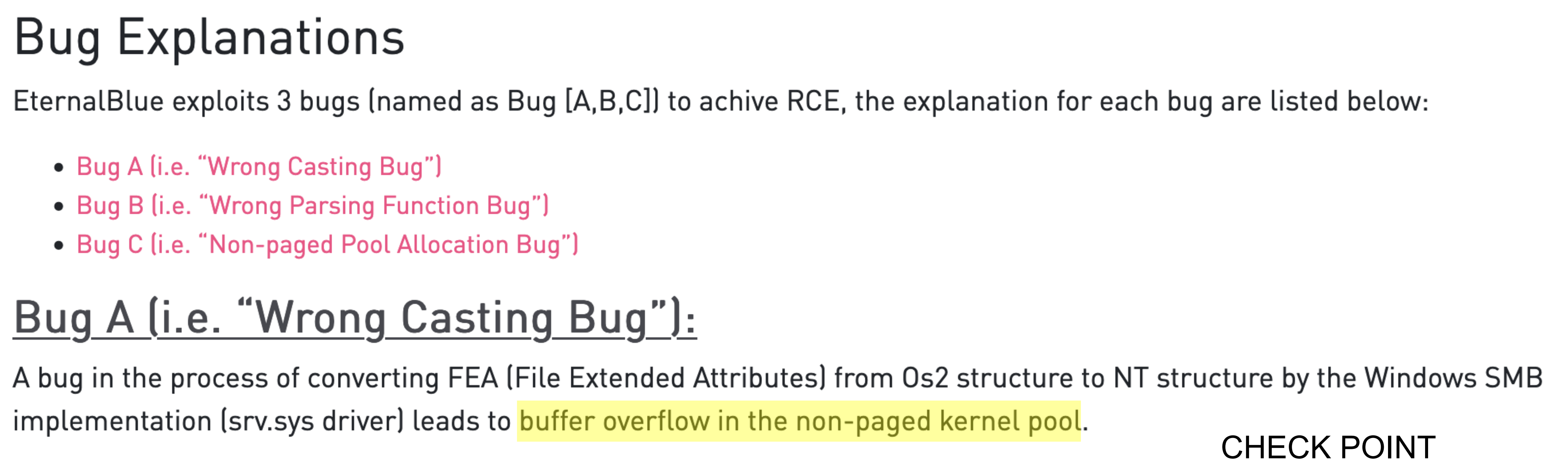
Published: March 14, 2017

Version: 1.0

Executive Summary

This security update resolves vulnerabilities in Microsoft Windows. The most severe of the vulnerabilities could allow remote code execution if an attacker sends specially crafted messages to a Microsoft Server Message Block 1.0 (SMBv1) server.

Eternal Blue leads to Remote Code Execution (RCE)



Bug Explanations

EternalBlue exploits 3 bugs (named as Bug [A,B,C]) to achive RCE, the explanation for each bug are listed below:

- Bug A (i.e. “Wrong Casting Bug”)
- Bug B (i.e. “Wrong Parsing Function Bug”)
- Bug C (i.e. “Non-paged Pool Allocation Bug”)

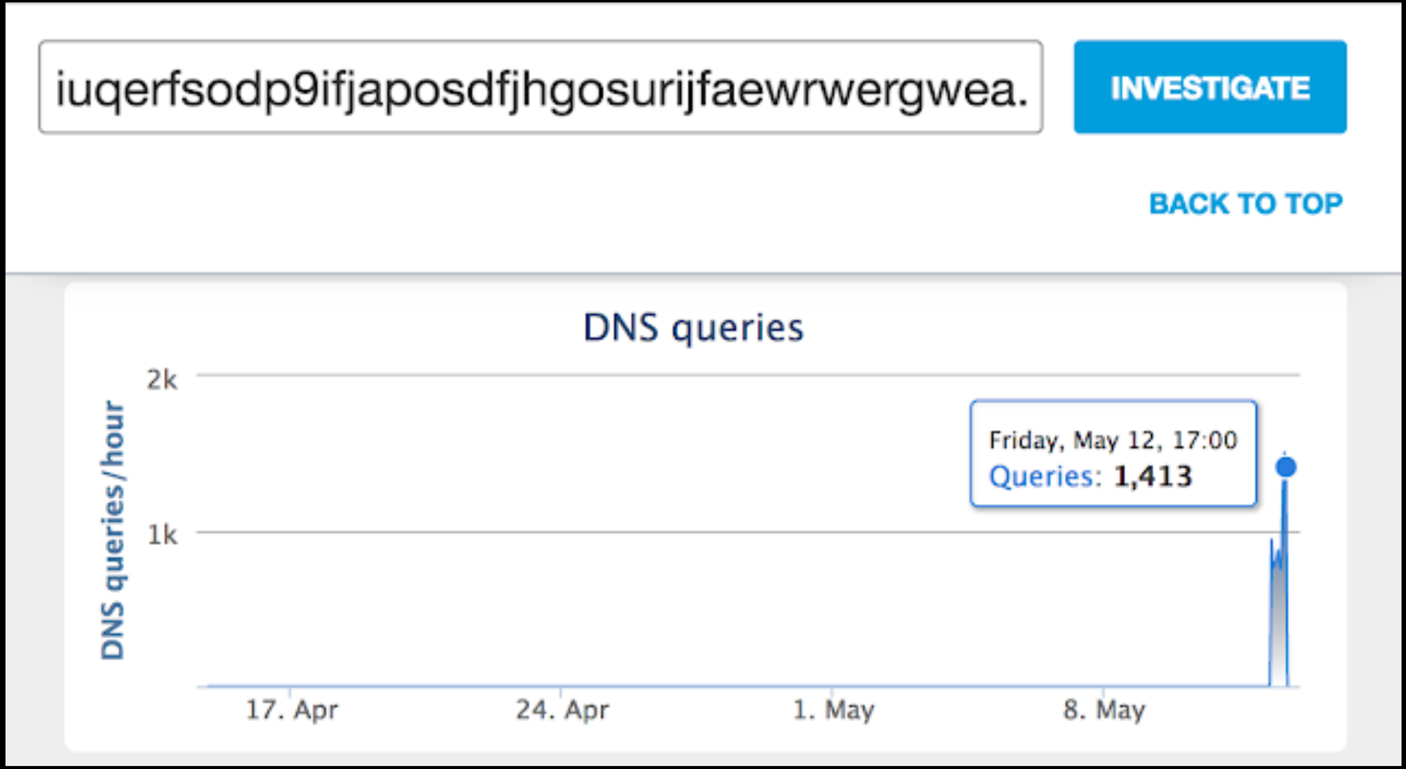
Bug A (i.e. “Wrong Casting Bug”):

A bug in the process of converting FEA (File Extended Attributes) from Os2 structure to NT structure by the Windows SMB implementation (srv.sys driver) leads to buffer overflow in the non-paged kernel pool.

CHECK POINT

WannaCry countermeasures

- Up to date patch as directed by Microsoft!
- Backups
- Security products
  - Anti-malware
  - Firewalls
  - Detection of kill switch domain
  - Network analytics



https://blog.talosintelligence.com/2017/05/wannacry.html

Malware damage

- Malware could cause
  - Information theft
  - Destruction
  - Denial of service
- Resulting in
  - Data Loss
  - Account Theft
  - Privacy violations
  - Financial Losses

Malware propagation mechanisms

- Viruses
  - Malicious software attached to a document, file, or program, and shared by people
- Worms
  - Malicious software that rapidly replicates and spreads to any device within the network
- Trojans
  - Malicious software disguised as a useful software program

Viruses phases and classification

- Phases:
    - Dormant
    - Propagation
    - Triggering
    - Execution
- Virus classification by target
    - Boot sector infector, file infector, macro virus, multipartite virus
- Virus classification by concealment and mutation
    - Polymorphic – evasion based variable encryption key
    - Metamorphic – evasion based on code rewrite

Propagation – worms

- Like virus but self replicating
- Why is self-replication useful?
- Allows for
  - Rapid infection
  - DDoS attacks
- Examples using different self-replication strategies
  - 1984: Ken Thomson Reflections on Trusting Trust – scary experiment
  - 1988: Morris worm – early days of Internet
  - 2005: Samy worm – mostly harmless
  - 2017: WannaCry, Not Petya – massively damaging

Trojan horses

- Involves social engineering
- A useful, or apparently useful, program or utility containing hidden code, which can perform unwanted or harmful function
- Examples
  - Free version of word processor, chat client, email client, …

Malware payloads

- Logic Bombs and Backdoors
- Ransomware
- Spyware, Adware
- Botnets running on distributed machines
- Rootkits installed in privileged area

Logic bombs and backdoors

- Code inserted by insider that fires under certain logic
    - Logic bomb triggers on system conditions
    - Backdoor triggers externally
- Results could be harmless Easter egg or serious damage
    - Job security
    - Extortion
- Detect using static analysis and code review
    - May hide to avoid triggering during testing

Ransomware

- Holding a computer system captive while demanding a ransom

- Restricts user access to the computer
    - Encrypting files on the hard drive
    - Locking down the system
    - Displaying messages to force user to pay to regain access

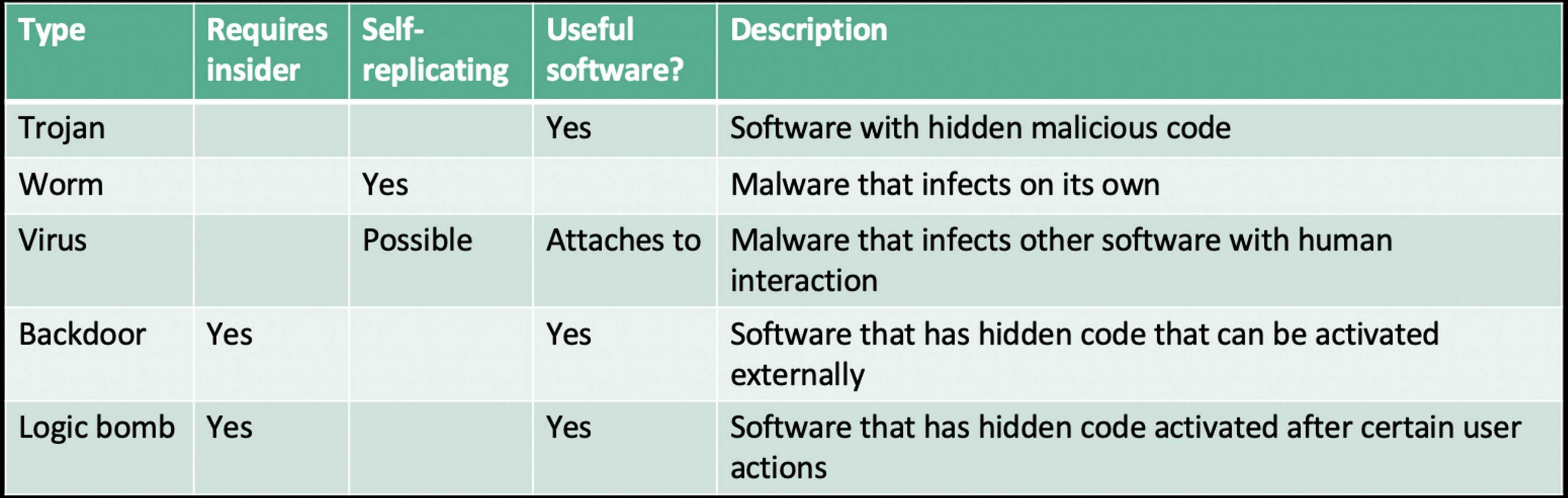
Spyware

- Installed on computers to collect information about users without their knowledge
- Typically hidden from the users and can be difficult to detect
- Keyloggers record keystrokes
- Lurk on your computer to steal important information, like your passwords and login and other personal identification information and then send it off to someone else

Adware

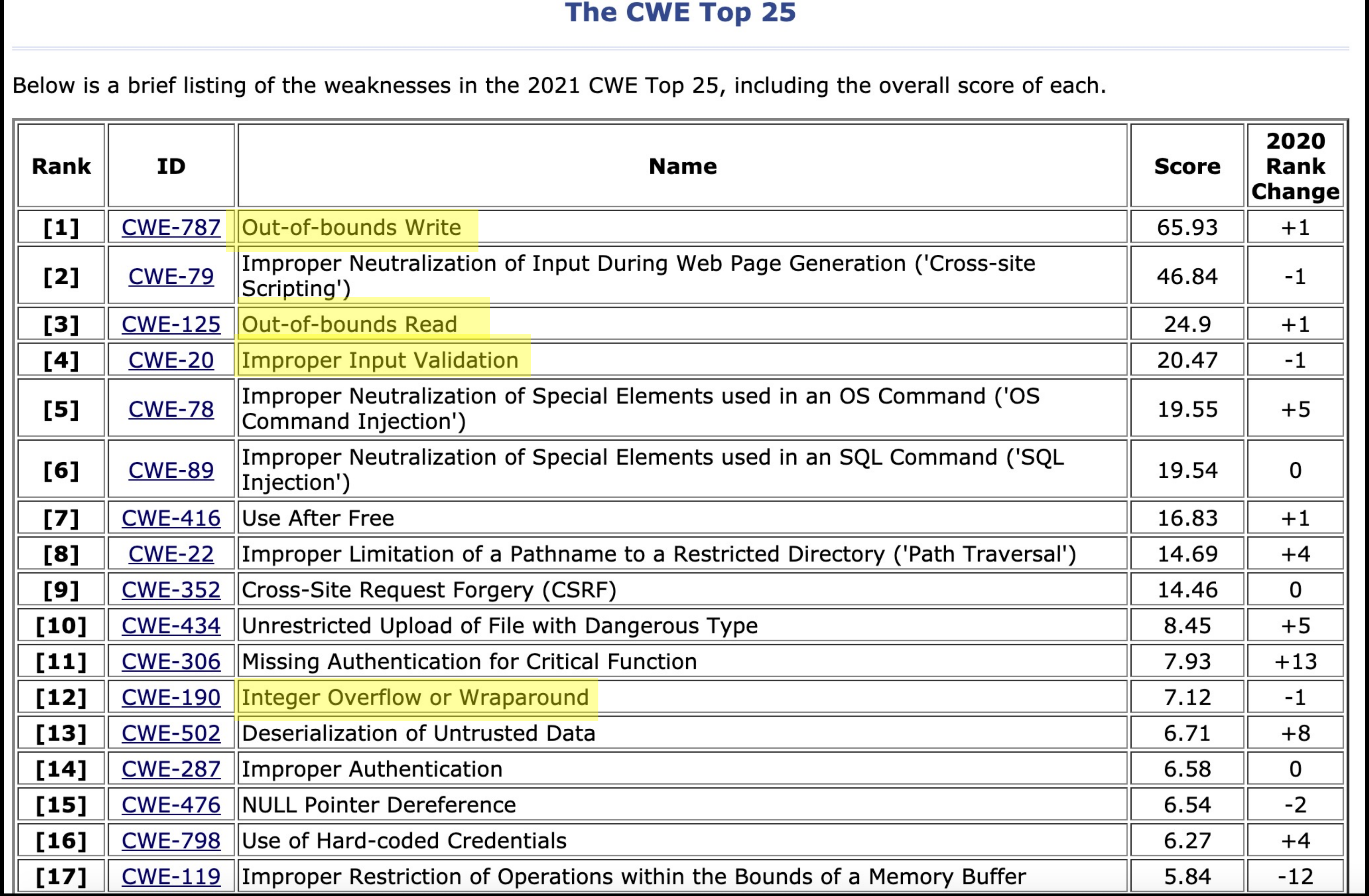
- Short for advertising-supported software, automatically delivers advertisements
- Common examples: pop-up ads on websites and advertisements that are displayed by software
- Often exists in software and applications offer “free” version

Malware summary



Type	Requires insider	Self-replicating	Useful software?	Description
Trojan			Yes	Software with hidden malicious code
Worm		Yes		Malware that infects on its own
Virus		Possible	Attaches to	Malware that infects other software with human interaction
Backdoor	Yes		Yes	Software that has hidden code that can be activated externally
Logic bomb	Yes		Yes	Software that has hidden code activated after certain user actions

Vulnerabilities – Common Weakness Enumeration



The CWE Top 25

Below is a brief listing of the weaknesses in the 2021 CWE Top 25, including the overall score of each.

Rank	ID	Name	Score	2020 Rank Change
[1]	CWE-787	Out-of-bounds Write	65.93	+1
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	CWE-125	Out-of-bounds Read	24.9	+1
[4]	CWE-20	Improper Input Validation	20.47	-1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	CWE-416	Use After Free	16.83	+1
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	CWE-306	Missing Authentication for Critical Function	7.93	+13
[12]	CWE-190	Integer Overflow or Wraparound	7.12	-1
[13]	CWE-502	Deserialization of Untrusted Data	6.71	+8
[14]	CWE-287	Improper Authentication	6.58	0
[15]	CWE-476	NULL Pointer Dereference	6.54	-2
[16]	CWE-798	Use of Hard-coded Credentials	6.27	+4
[17]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	5.84	-12

Out-of-bounds write

- Writes data past the end, or before the beginning, of the intended buffer
- Caused by type safety failure
- Frequently used by viruses and worms
- Common consequences?
- Likelihood?

Example – buffer overflow

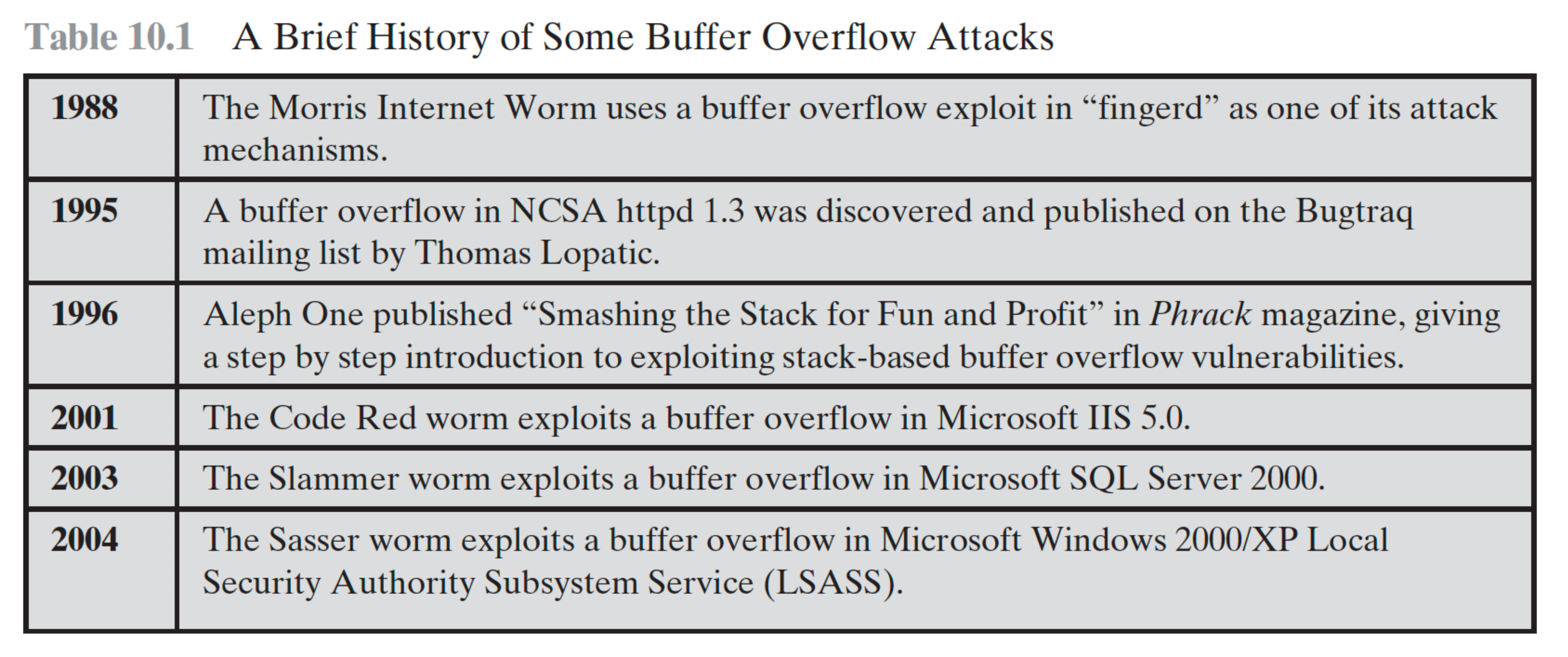
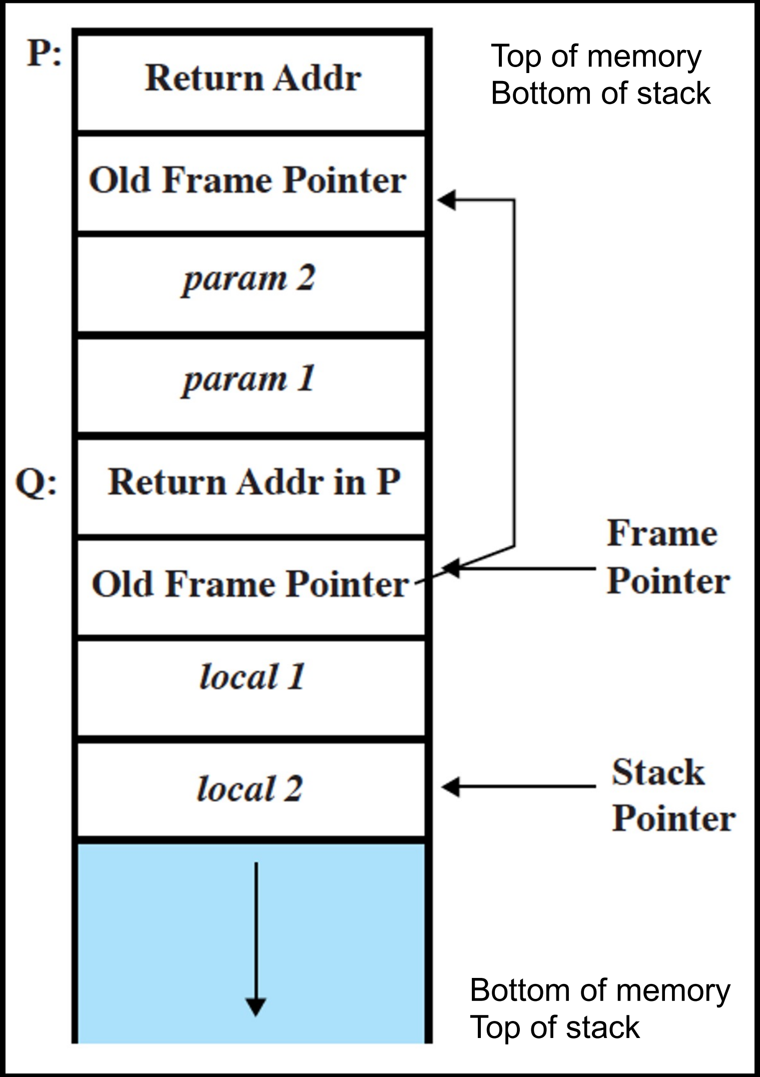


Table 10.1 A Brief History of Some Buffer Overflow Attacks

1988	The Morris Internet Worm uses a buffer overflow exploit in “fingerd” as one of its attack mechanisms.
1995	A buffer overflow in NCSA httpd 1.3 was discovered and published on the Bugtraq mailing list by Thomas Lopatic.
1996	Aleph One published “Smashing the Stack for Fun and Profit” in *Phrack* magazine, giving a step by step introduction to exploiting stack-based buffer overflow vulnerabilities.
2001	The Code Red worm exploits a buffer overflow in Microsoft IIS 5.0.
2003	The Slammer worm exploits a buffer overflow in Microsoft SQL Server 2000.
2004	The Sasser worm exploits a buffer overflow in Microsoft Windows 2000/XP Local Security Authority Subsystem Service (LSASS).

Stack-based buffer overflow

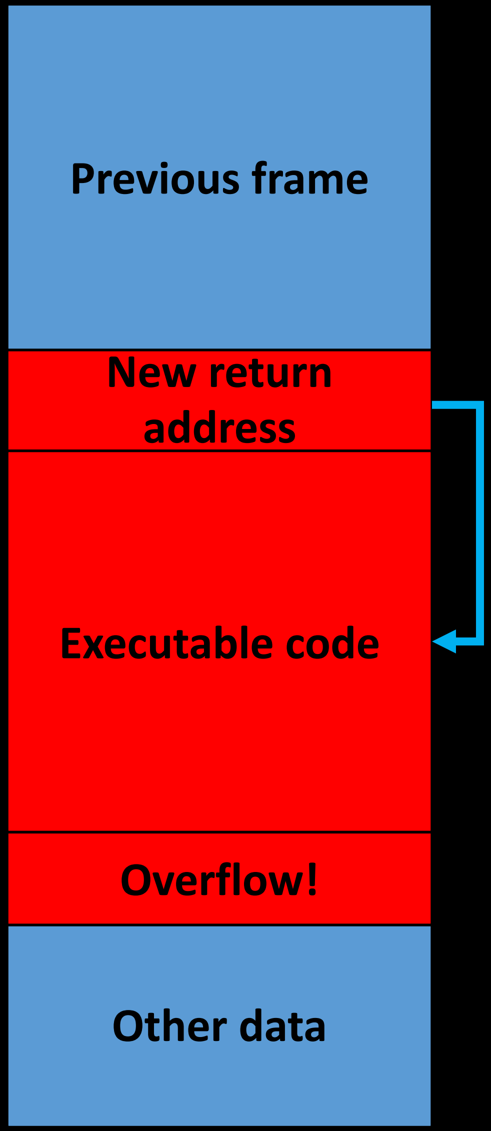
- Smashing the stack (paper by the same name)
  - Used by Morris Worm (finger Daemon vulnerability)
- Exploit local variables
  - Rewrite return address
  - Do whatever you want



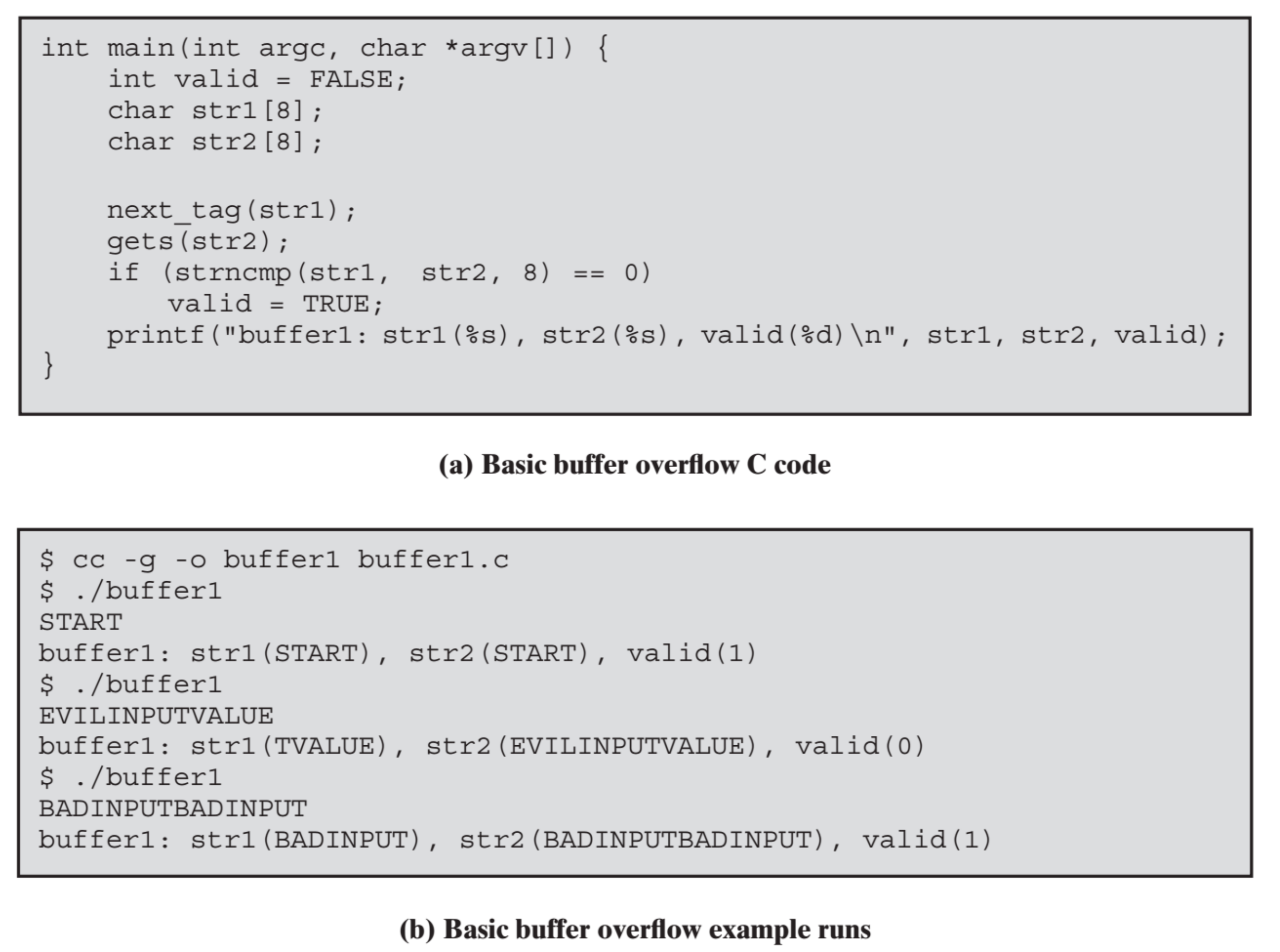
Buffer overflow

To execute code

- Place code to be executed
- Convince the program to jump to the code



Buffer overflow example (Stallings)



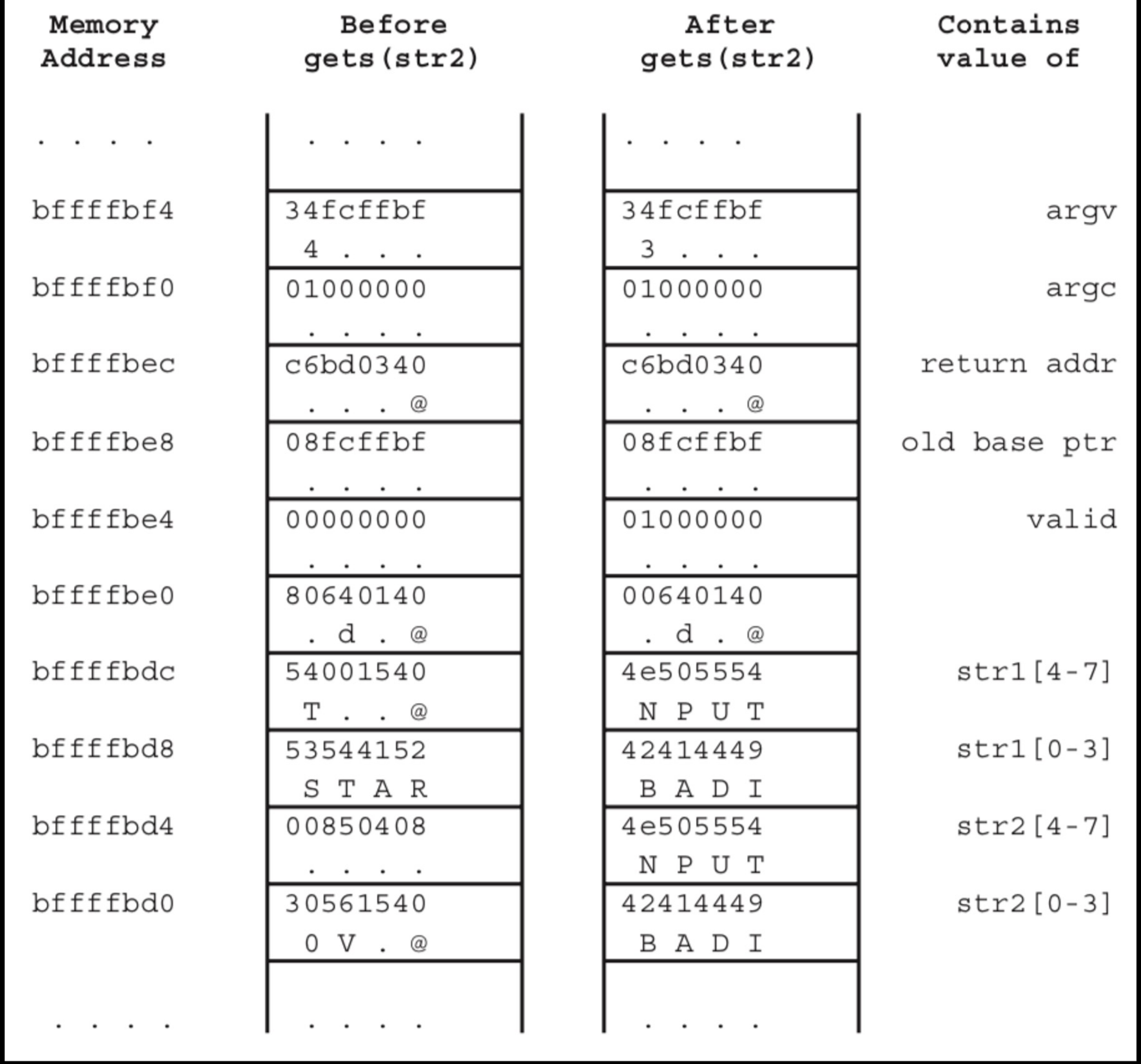
```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1,  str2, 8) == 0)
       valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

(a) Basic buffer overflow C code

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

(b) Basic buffer overflow example runs



Memory Address	Before gets(str2)	After gets(str2)	Contains value of
. . . .	. . . .	. . . .	
bffffbf4	34fcffbf 4 . . .	34fcffbf 3 . . .	argv
bffffbf0	01000000	01000000	argc
bffffbec	c6bd0340 . . . @	c6bd0340 . . . @	return addr
bffffbe8	08fcffbf	08fcffbf	old base ptr
bffffbe4	00000000	01000000	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
bffffbdc	54001540 T . . @	4e505554 N P U T	str1[4-7]
bffffbd8	53544152 S T A R	42414449 B A D I	str1[0-3]
bffffbd4	00850408	4e505554 N P U T	str2[4-7]
bffffbd0	30561540 0 V . @	42414449 B A D I	str2[0-3]
. . . .	. . . .	. . . .	

Shellcode

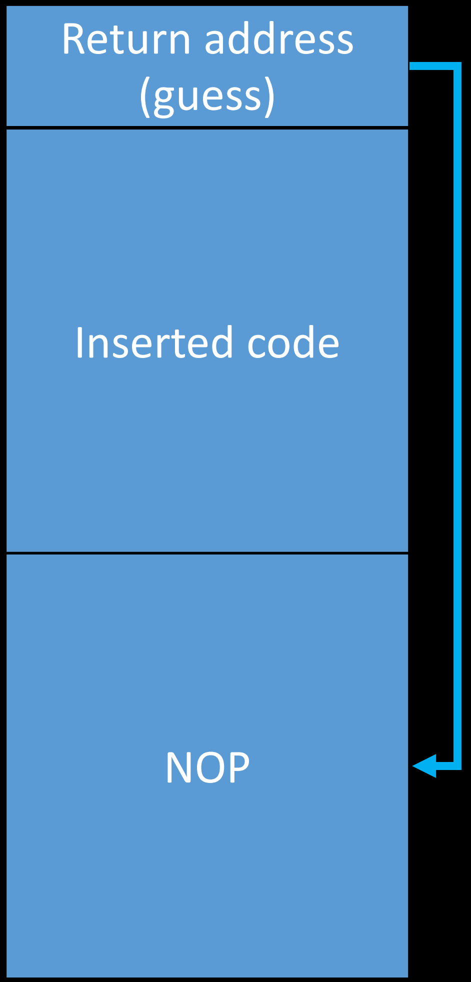
- Machine code inserted by attacker
- Saved via buffer overflow
- Traditionally transferred control to shell
  - Specific to processor and OS
  - Traditionally needed assembly skill
  - Automated tools now exist – Metasploit

NOP slide

What if we don’t know the precise location of inserted code?

NOP slide

- Guess good enough
- Slide all the way down
- Eventually reach inserted code



What we discussed

- Software security
- Malware damage
- Malware propagation
  - Viruses
  - Worms
  - Trojans
- Malware payloads
  - Logic bombs / backdoors
  - Ransomware
  - Spyware
- Out-of-bounds attacks
  - Smash the stack buffer overflow
  - Shellcode
  - NOP slide

What’s next

- Module 6 Malicious Software – continued
- Readings
    - Anderson Chapter 4 on Access Control (especially 4.4 What goes wrong)
    - Smashing the Stack
    - Format String Vulnerabilities
    - Polymorphic malware
    - Optional: The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls (on the x86)
    - Optional: ROP
    - Optional: Malware
- You should be working on 6a Lab Leviathan / 6b Lab Microcorruption due next Tuesday
- #breach-of-the-week – participate on slack!
- Office hours Thurs 11:00am-12:00pm in 192-333 or M/W/F on zoom

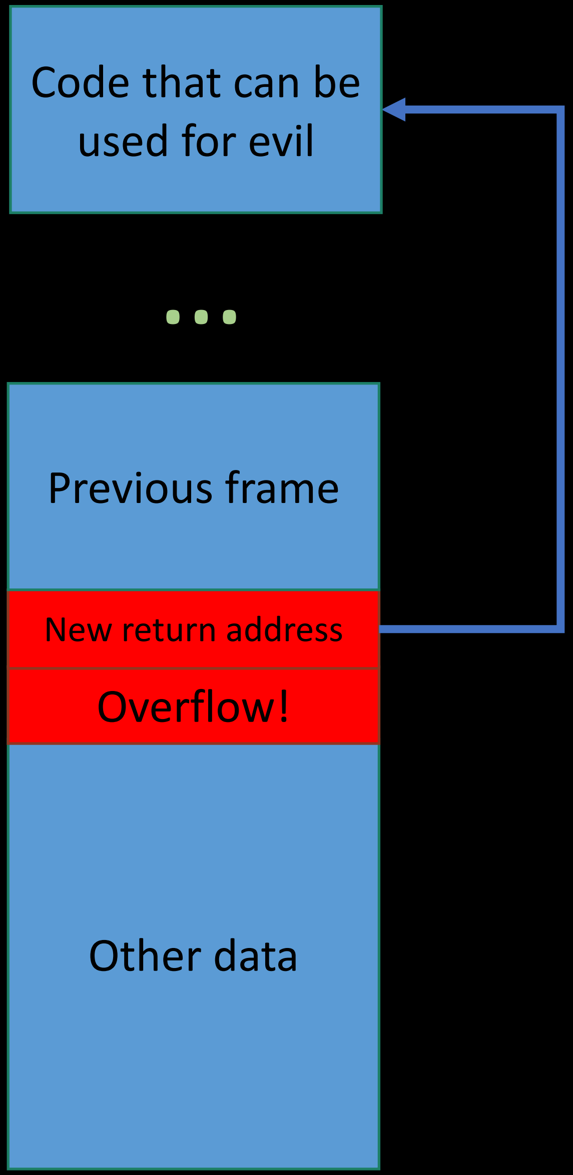
Breach of the Week!

Other forms of overflow attacks

Return-Oriented Programming (ROP)

- Advanced version of smashing the stack buffer overflow for non-executable stack
  - Find code that does something
  - Convince the program to jump

- Calls standard system calls via libc or machine instruction sequences
- Construct desired parameters before calling
- Attackers can chain commands
- Needs exact memory position



Heap spraying/overflow

- Attack buffer in the heap
  - Typically above program code
  - Location of dynamic data structures

- No return address
  - Can not transfer control
  - Can exploit data or function pointers

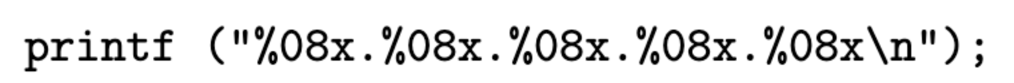
Integer overflow

- Numbers are represented with bits
  - 8-bits
    - Unsigned numbers between 0 and 255
    - Signed numbers – 128 and 127

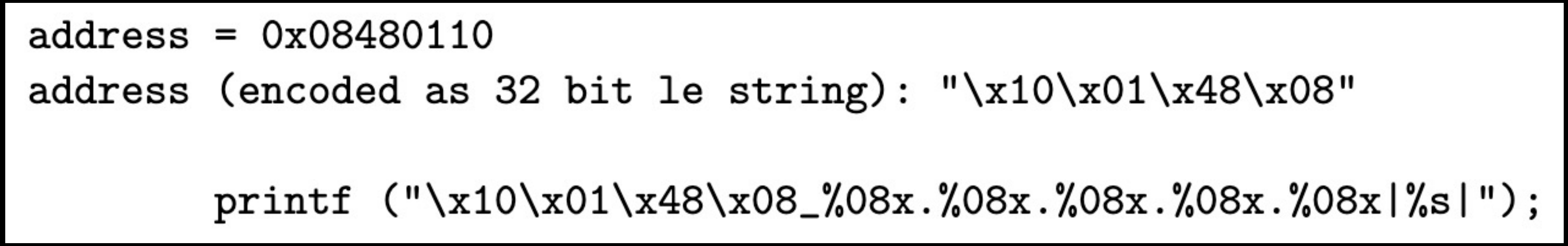
- Assume 8-bit signed numbers
  - What is 110+60?
    - 170
      - Does not exist
- Can cause lots of problems: Y2K, Y2038 (signed 32-bit integer)

Format string vulnerabilities

- User submitted data of an input string is evaluated as a command
- Format parameters and their consequences:
  - “%x” Read hex value from the stack
  - “%s” Read character string by reference from address in process memory
  - “%n” Write an integer by reference to address in the process’ memory
- Fairly easy compile-time detection



```
printf ("%08x.%08x.%08x.%08x.%08x\n");
```



```
address = 0x08480110
address (encoded as 32 bit le string): "\x10\x01\x48\x08"

        printf ("\x10\x01\x48\x08_%08x.%08x.%08x.%08x.%08x|%s|");
```

Stopping malicious code

Address vulnerabilities across the Secure Development Lifecycle (SDL)

- Development/compile time – Eliminate code vulnerabilities

- Distribution time – Stop malware from propagating

- Run time – Stop malware from running

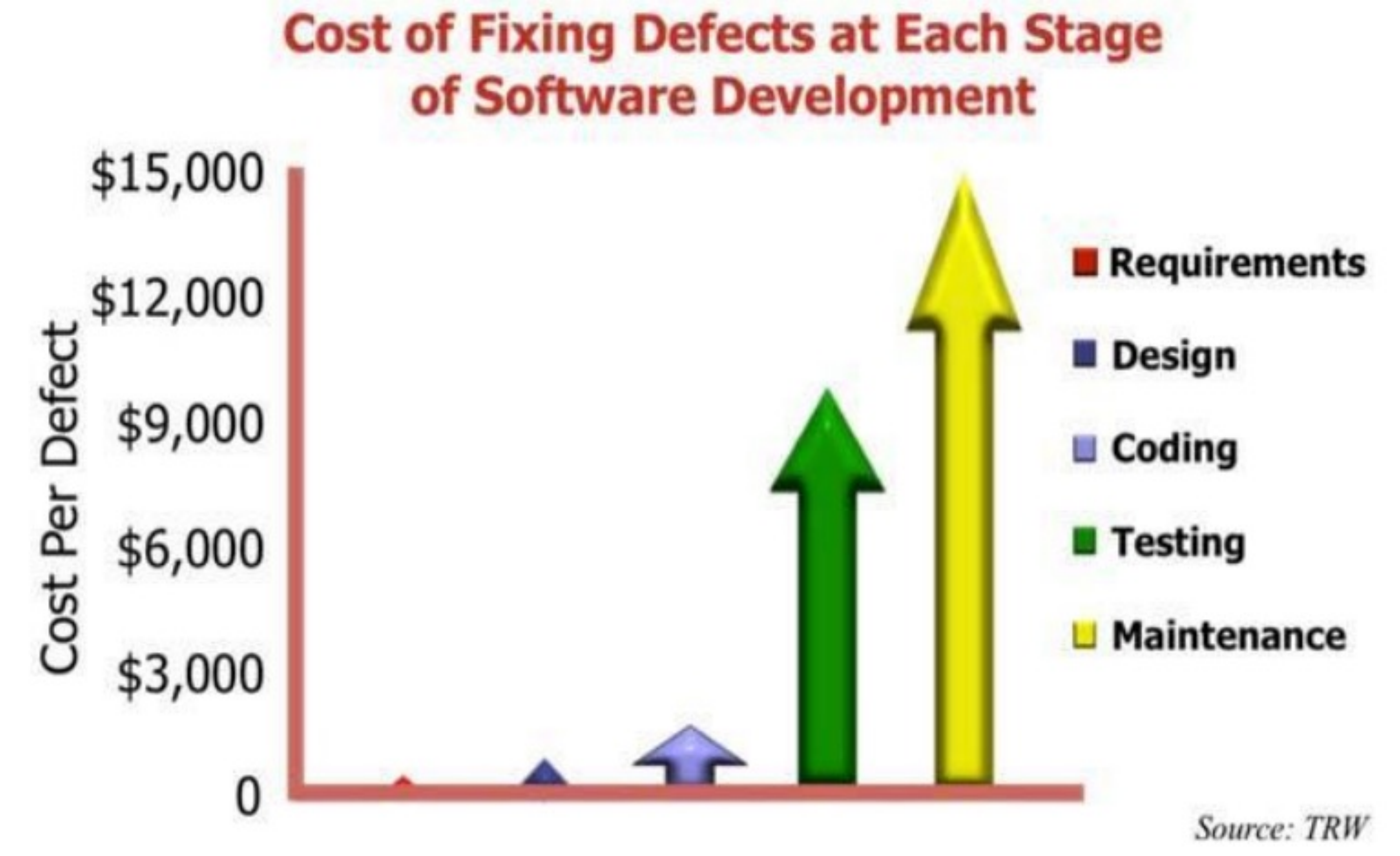
Secure Development Lifecycle (SDL) (Anderson)

- Requirements – Risk assessment and quality gates
- Design – Threat modeling and secure system design principles
- Implementation – Use approved tools, avoid unsafe functions, static analysis on code
- Verification – Dynamic analysis on code, fuzz testing
- Release – Incident response plan and final security review

From DevOps to DevSecOps

- DevOps
  - Minimize separation between development and production
  - Especially in container-based Software as a Service (SaaS) applications
- DevSecOps
  - Brings Security Development Lifecycle into DevOps

Fixing defects: the earlier the better



Source: TRW

Development/compile time defenses – safe languages and libraries

- Programming language – Avoid C, C++, and other loosely typed languages

- Safe libraries -- If you must program in C, avoid unsafe routines:
  strcpy(char *dest, const char *src)
  strcat(char *dest, const char *src)
  getwd(char *buf)
  gets(char *s)
  scanf(const char *format, ...)
  sprintf(char *str, const char *format, ...)

Development/compile time defenses – defensive programming (Stallings)

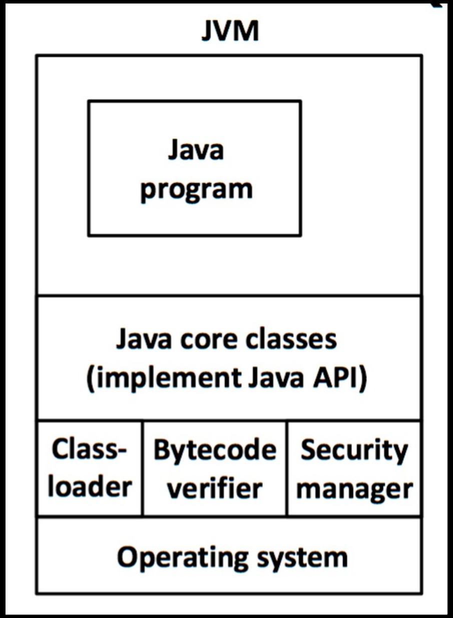
- Handle erroneous inputs
  - Explicitly validate all external inputs, fail gracefully
- Understand system calls and libraries
  - Conditions vary by OS and operations
  - Verify under race conditions
- Handle program output
  - Only output what is needed
  - Explicitly identify output encoding
- Store in secure and expected ways
  - Don’t store secrets in source code
  - Encrypt secrets and protect keys

Distribution time defenses

- Digitally sign software to enable verification of where the software comes from
  - Hard to get people to check
  - Can social engineer signers
- Anti-virus / anti-malware
  - Create known virus fingerprint
  - Polymorphism – evasion based variable encryption key
  - Metamorphism – evasion based on code rewrite
  - Behavioral analytics to detect new unknown suspicious behavior
  - All anti-malware will miss some new attacks or generate false positives

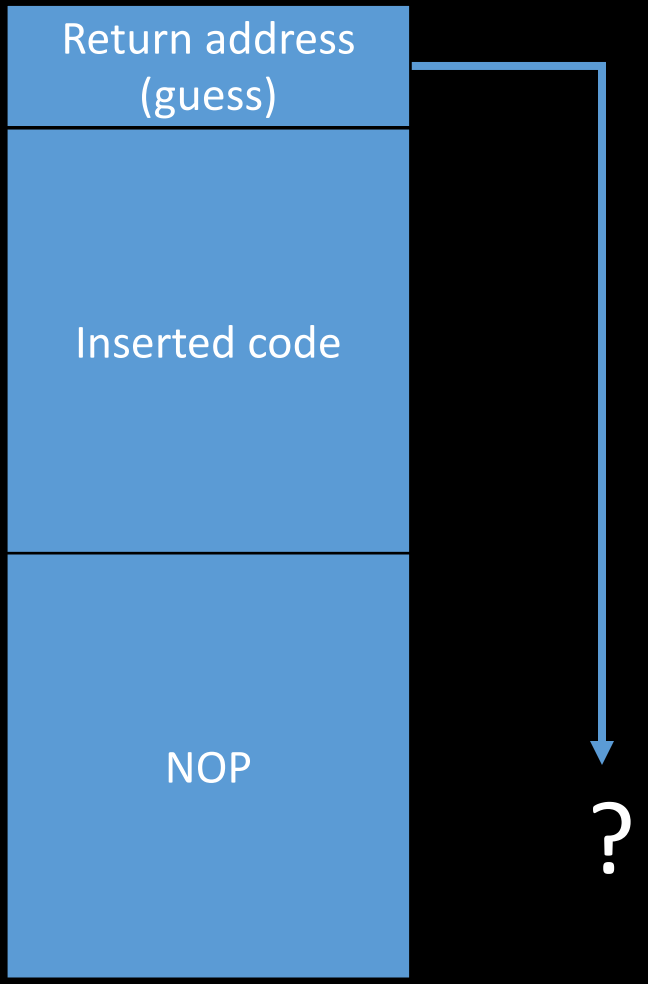
Run time defenses – analysis and controls

- Dynamic analysis
  - Testing and evaluation of an application during runtime
- Non-executable address space
  - Virtual memory support to make some regions of memory non-executable
  - Don’t run code on stack, but won’t work in all cases (JIT, JVM, recursion in C)
  - Won’t protect against ROP
- Confinement and access control
  - Isolate code to minimize attack surface
  - Operating system isolation
  - Virtual Machines
  - Containers
  - Many others



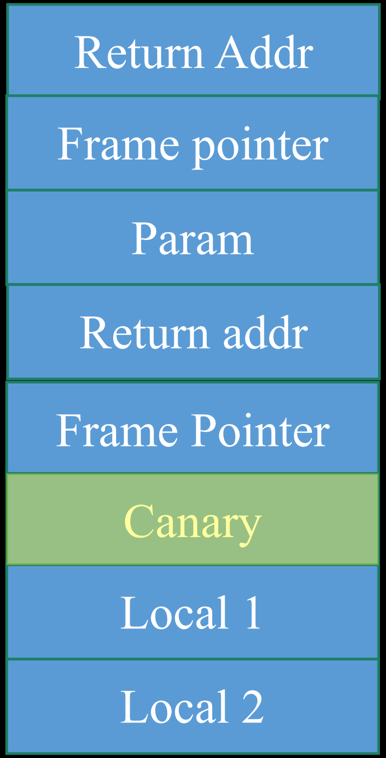
Run time defenses – memory randomization

- Address space layout randomization (ASLR)
- Remember a buffer overflow with NOP slide
- Could be anywhere in 32 bit memory



Run time defenses – canary

- Idea: Put something in the stack to detect changes
- If the canary dies, kill the program
- Terminator
  - Require null before return address
- Random
  - Changes every program run



Return Addr

Frame pointer

Param

Return addr

Frame Pointer

Canary

Local 1

Local 2

What can users do to protect your computer?

- Practice caution when working with files from unknown or questionable sources
- Do not open e-mail if you do not recognize the sender
- Download files only from reputable internet sites
- Install firewall
- Install protection software
- Detect command and control of malware

Possible malware symptoms

- Strange computer behavior
- Emails/messages being sent automatically and without user’s knowledge (a friend receives a strange email from you that you did not send)
- There seems to be a lot of network activity when you are not using the network
- The available memory on your computer is lower than it should be
- Programs or files appear or disappear without your knowledge
- File names are changed
- Or nothing happens at all

What we discussed

- Out-of-bounds attacks – continued
    - Return-oriented programming
    - Heap spraying
    - Integer overflow
    - Format string
- Stopping malicious code across the Secure Development Lifecycle
    - Development/compile time
    - Distribution time
    - Run time

What’s next

- Module 7 Network Security
- Readings
  - Anderson Chapter 21 on Network Attack and Defence (especially 21.2 Network protocols and service denial; 21.4 Defense against network attack; 21.5 Cryptography: the ragged boundary)
- Quiz 6, 6a Lab Leviathan / 6b Lab Microcorruption due on Tuesday
- #breach-of-the-week – participate on slack!
- Office hours Thurs 11:00am-12:00pm in 192-333 or M/W/F on zoom