

COMP3052.SEC Computer Security

Session 13: Software Vulnerabilities



Acknowledgements

- Some of the materials we use this semester may come directly from previous teachers of this module, and other sources ...
- Thank you to (amongst others):
 - Michel Valstar, Milena Radenkovic, Mike Pound, Dave Towey...

This Session

Malware

- Viruses
- Worms
- Trojans
- Ransomware
- Rootkits and Backdoors

Exploits

- Coding Flaws
- Common Vulnerabilities
- Buffer overflows
- ...



Danger: Malware Ahead!

Google Chrome has blocked access to this page

Content from valid.canardpc.com, a known malware distributor, has been inserted into this web page. Visiting this page now is very likely to infect your computer with malware.

Malware is malicious software that causes things like identity theft, financial loss, and permanent file deletion. [Learn more](#)



[Go back](#)

[Advanced](#)

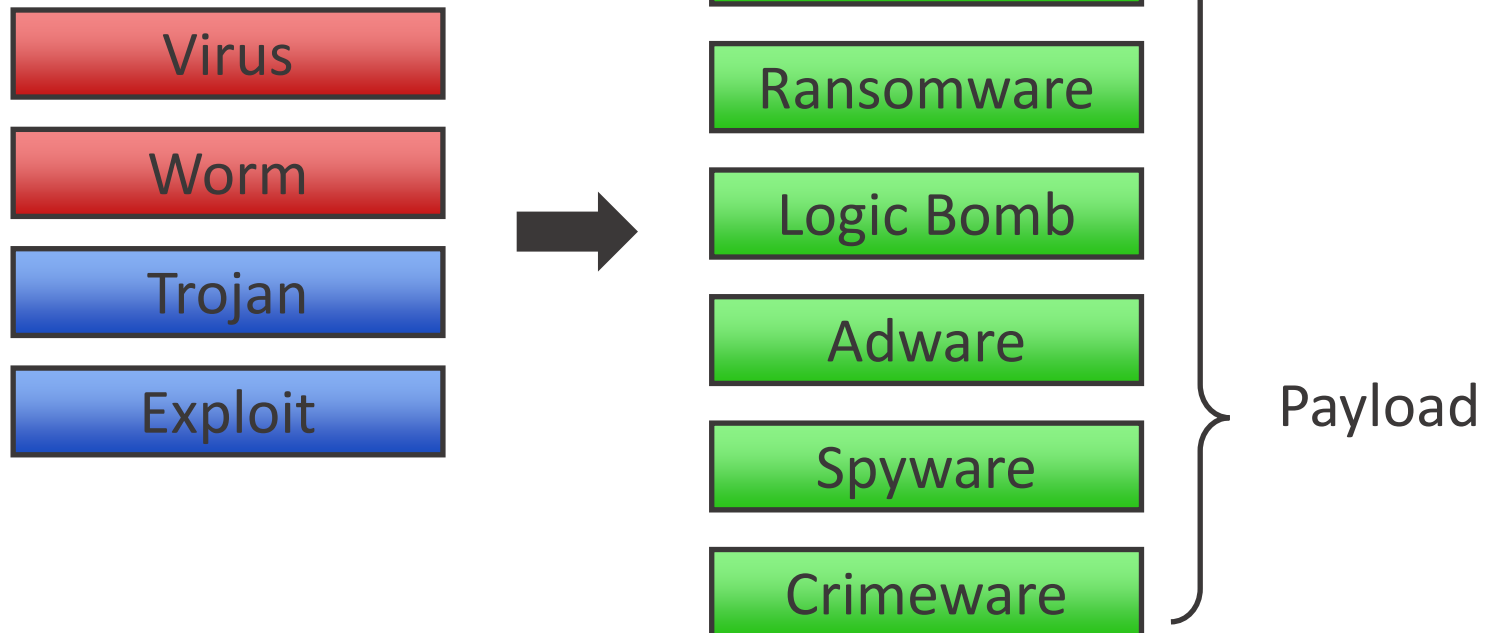
☐ Improve malware detection by sending additional data to Google when I encounter warnings like this. [Privacy policy](#)

Malware

Malicious Software

Malware

- A *very general* term, malware is usually categorised based on:
 - How it proliferates
 - What it does



Vectors

- Vectors are the mechanism through which malware infects a machine
- Usually the vector will be either a software vulnerability, or a human error
- In the case of human error, this often means someone has clicked “Yes” to something they shouldn’t have!

Payloads

- Payloads are the actual malware deposited on the machine, or the harmful results
- They range in severity:
 - Essentially do nothing
 - Messages and adverts
 - Recruited into a botnet or mail spam
 - Stealing private information
 - System destruction



Viruses

- A virus is a piece of self-replicating code
- Propagates by attaching itself to a disk, file or document, which would usually be executable
- When the file is run, the virus runs, and attempts to proliferate
- Installs without the user's knowledge or consent
- Requires human intervention to spread



Virus Attachment Mechanisms

- Historically, viruses installed themselves in the boot sectors of disk partitions
- Executable files may be sent as email attachments and require social engineering to entice users
- Script files for system admins including Windows batch files and UNIX shell scripts
- Documents containing macros
 - Older office formats, newer macro enabled workbooks

Notable Viruses

- 1981: Elk Cloner, the first known computer virus outbreak affects Apple II computers
- 1986: Brain, the first MS-DOS computer virus
- 1989: Ghostball, the first multipartite virus – affects both EXEs and the boot sector
- 1995: First macro virus, Concept, affects MS Word documents
- 1996: First linux virus, Staog, uses bugs in the Linux kernel

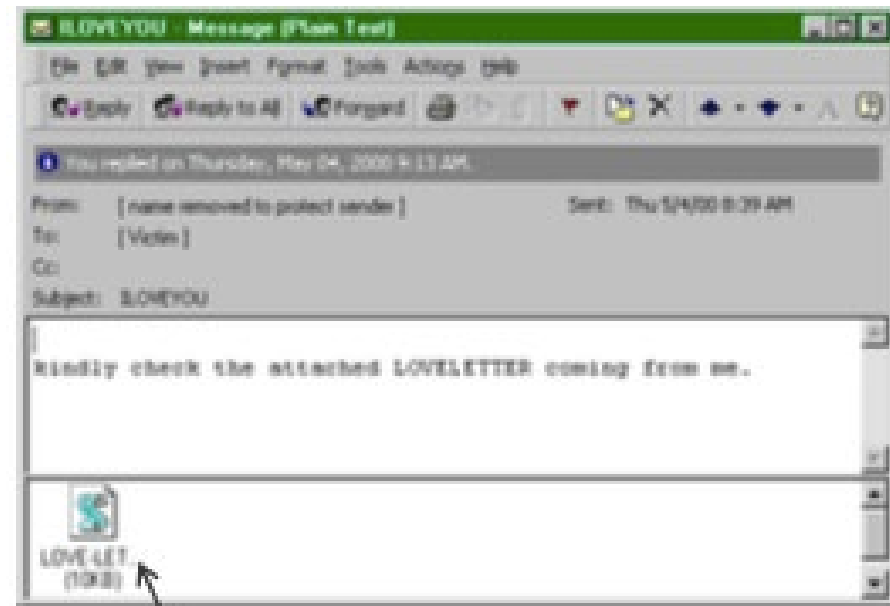
Worms

- Viruses traditionally require a human to spread
- Worms are self-replicating and stand-alone programs
 - Do not require human intervention
- Scanning worms or email worms
- Exploit known software vulnerabilities in order to spread



Notable Worms

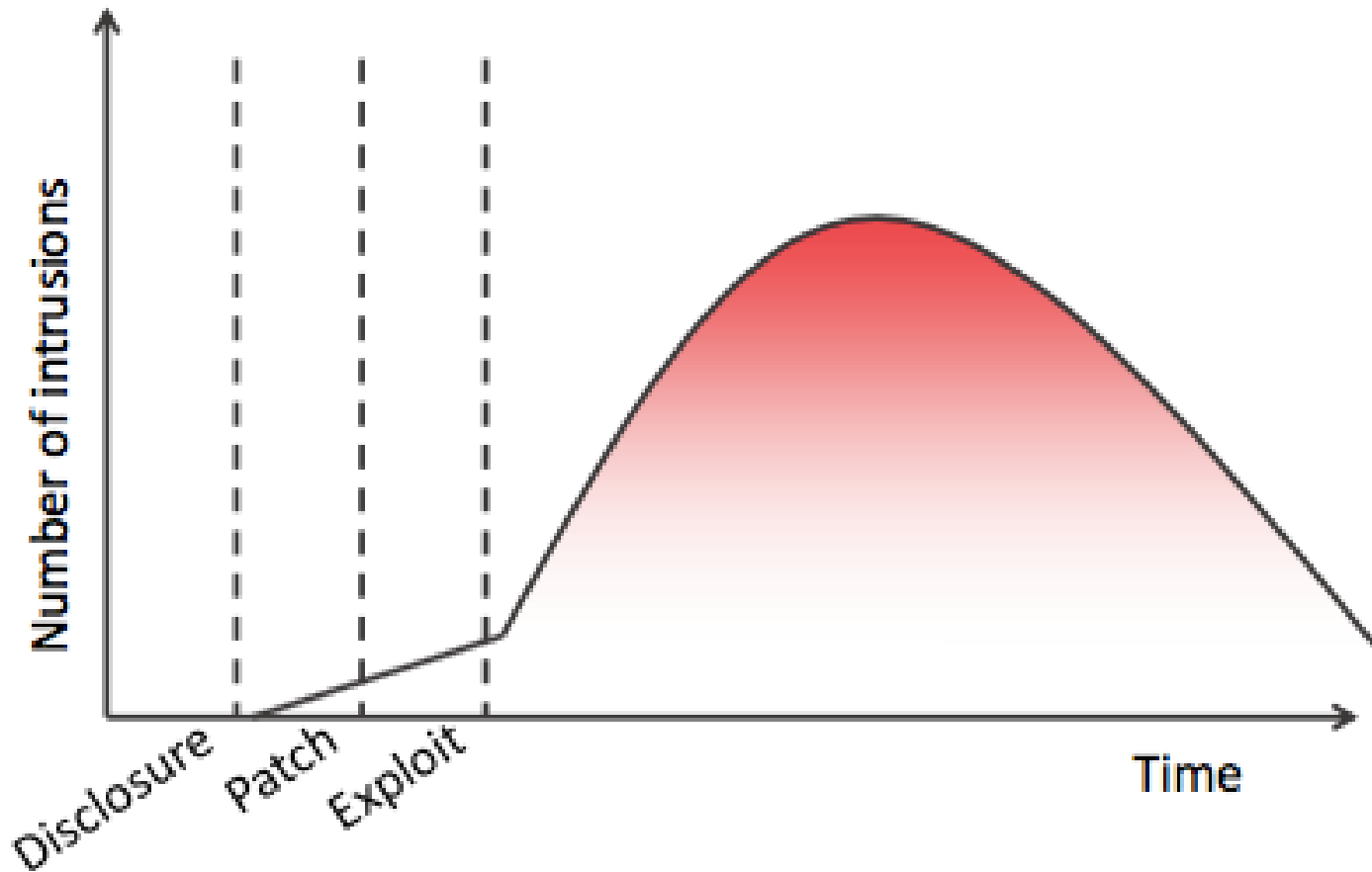
- 1988: The Morris Worm, affects BSD Unix machines. One of the first known buffer overruns
- 2000: The ILOVEYOU worm, one of the most damaging worms ever, shows how powerful social engineering can be



LOVE-LETTER-FOR-YOU.txt.vbs

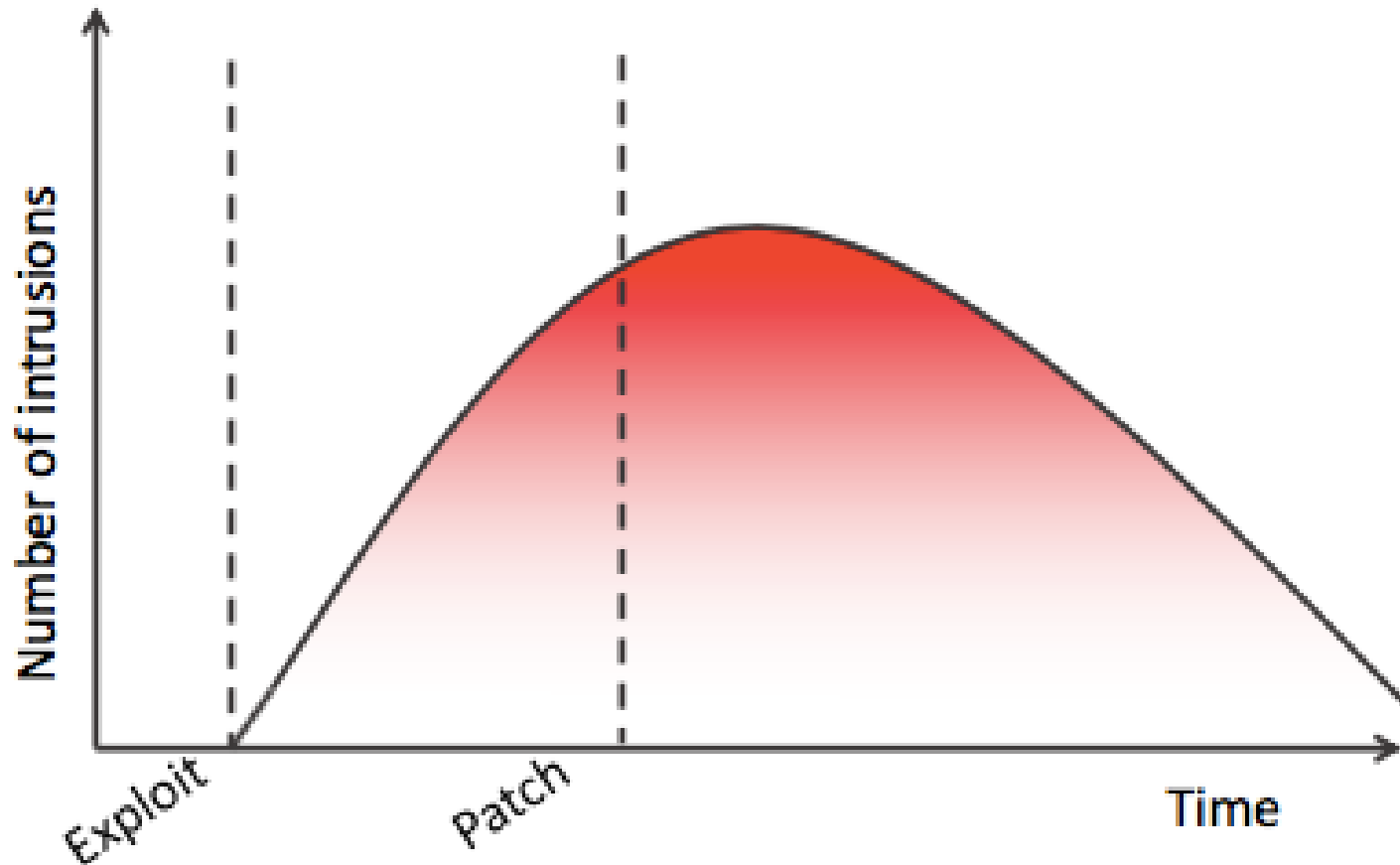
Exploit Life Cycle

- Many exploits are reversely engineered from patches, or developed simultaneously to patches



Zero-Day Exploits

- An exploit that is previously unknown – by far the most dangerous



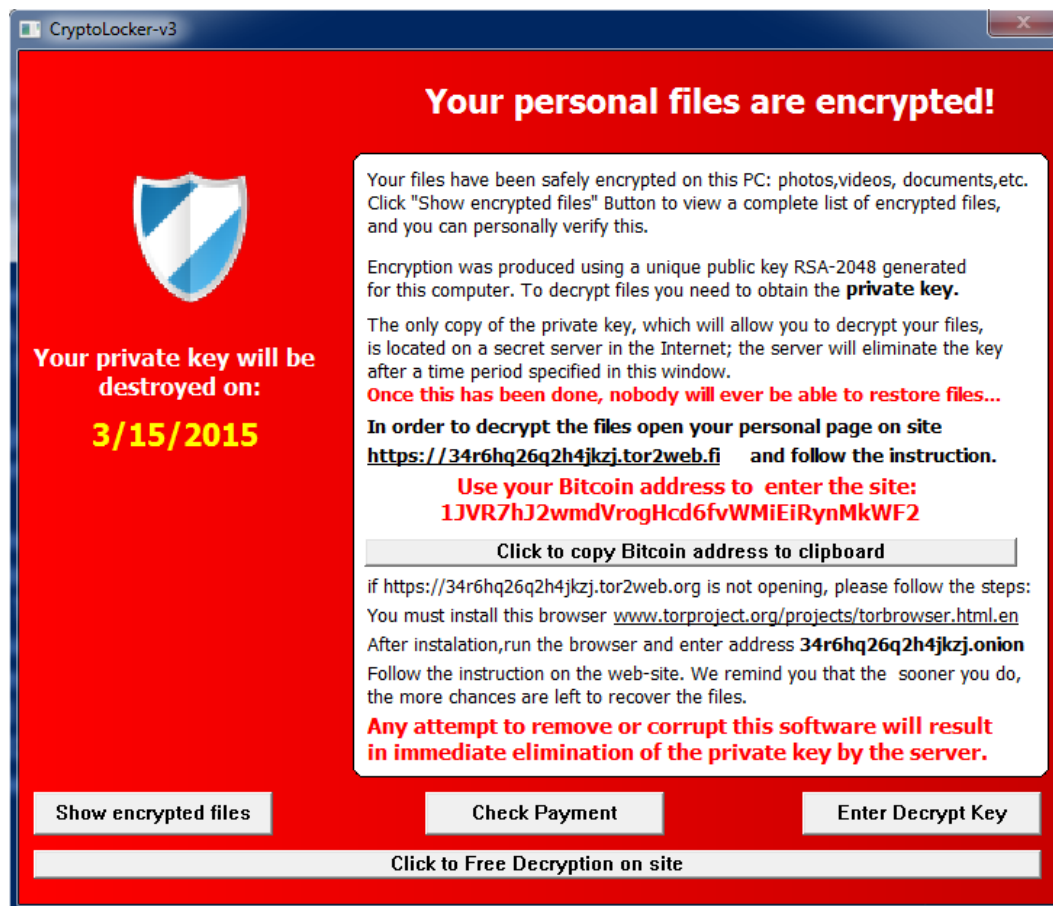
Trojans

- A malicious program pretending to be a legitimate application
 - For example, an application to “clean” hardware, but instead infects the host with adware and spyware
- Often obtained in email attachments or on malicious websites
- Don’t replicate themselves – user error



Ransomware

- Since about 2013, the rise of **Ransomware**



Ransomware

- Usually encrypts or blocks access to files and demands a ransom
- It is a clever attack, because if an anti-virus (AV) system removes it, it is often too late
- Usually distributed on malicious websites, or to already infected machines
- The file decryption keys are protected by encrypting using the public key of a C&C server

Ransomware Variants

- Most of the challenge in successfully using ransomware is tricking a user into running it, and bypassing AV and browser protections

- Fake emails
- Malicious web pages
- Obfuscated Javascript (JS) attachments
- Deployed using exploit kits

From: c.makee@duvalschools.com

To: yourmail@nottingham.edu.cn

Subject: My Resume

Hi, my name is Carlos Mckee

Please find my resume in the attachment

Thank you,

Carlos

Backdoors

- Once you have access to a system, a backdoor can be used to provide easy access
- These often work in a client-server architecture, awaiting commands from a central server
 - Remote Administration Tool (RAT)
- More discrete methods will involve subtly changing existing software, or the kernel itself
- Reverse shells – connect back for instructions

Rootkits

- Hide the presence of malicious code by hiding it from the operating system's process table
- Often activated before or during a boot
- Used to hide other malware from standard countermeasures
- Often installs inside the kernel itself, hooking into system call tables, or loads from the master boot record

Exploits

- The easiest way of getting access to a machine is having the user to install something for you – not always possible!
- Failing this, we need an exploit that defeats the security perimeter put in place by the operating system

How is this allowed to happen?!

- There are numerous reasons that exploits can be found in software
 - Programming errors
 - Unchecked user input
 - Incorrect assumptions
 - Weak APIs
 - Bad protocols
- Damage is often worse due to the widespread use of some libraries

Bad Software

- Bad software is everywhere:
 - NASA Mars Lander (\$165m) – Conversion of units
 - Ariane-5 Rocket (\$500m) – Integer overflow
 - Cancer radiation treatment (6 patients died) – Radiation measure malfunction
- Just search google for “worst software bugs”!
 - ... or look back at some of our slides in SQA, FSE,...

Memory Management

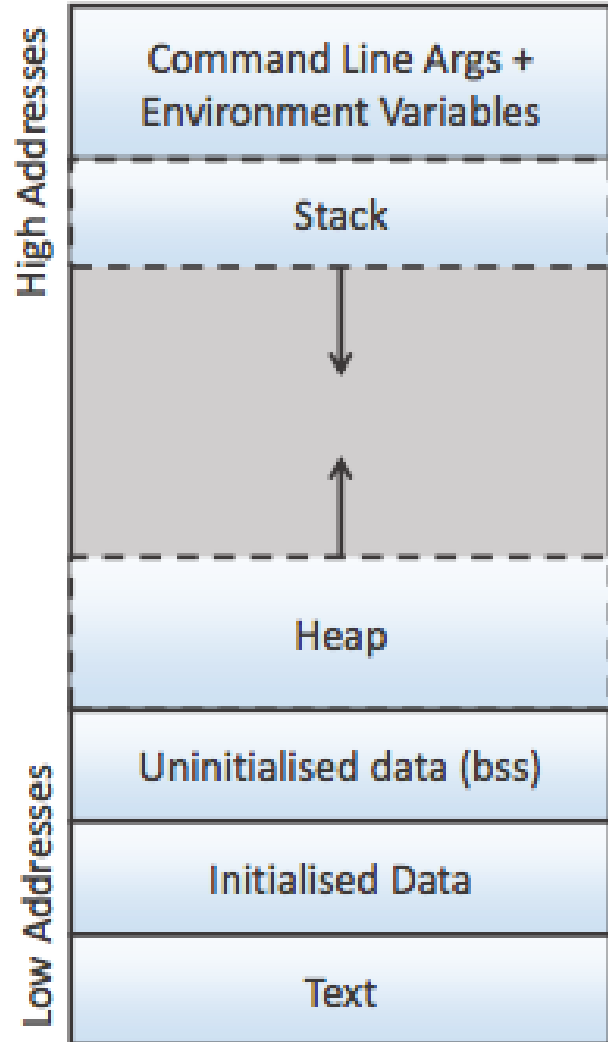
- In C and C++, the programmer performs memory management
- Flexible, powerful, fast, but dangerous
 - Buffer Overruns
 - Stack Overruns
 - Heap Overruns
- Memory-managed languages avoid this, but of course may have their own vulnerabilities

Buffer Overflows

- When a program is executed, contiguous blocks of memory can be allocated to store arrays (buffers)
- If data is written into a buffer that exceeds its size, an overflow occurs
- The data will overwrite the memory beyond the buffer – Bad!

Program Memory

- Memory is stored in a virtual address space from 0x00000000 to 0xFFFFFFFF (32-bit)
- Parts of the program are held in different regions by convention
- Different restrictions are placed on these regions

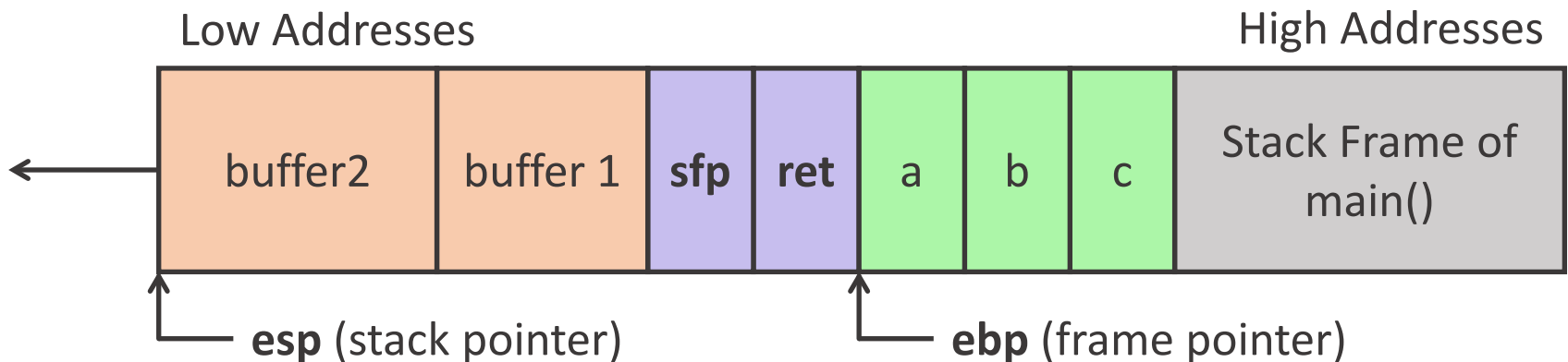


The Stack

- The stack holds information on local variables and functions calls (stack frames)
- A function call will push a new frame onto the stack
- A return will pop it off, and go to **ret**

```
void function(int a, int b, int c)
{
    char buffer1[5];
    char buffer2[10];
}

void main()
{
    function(1,2,3);
}
```



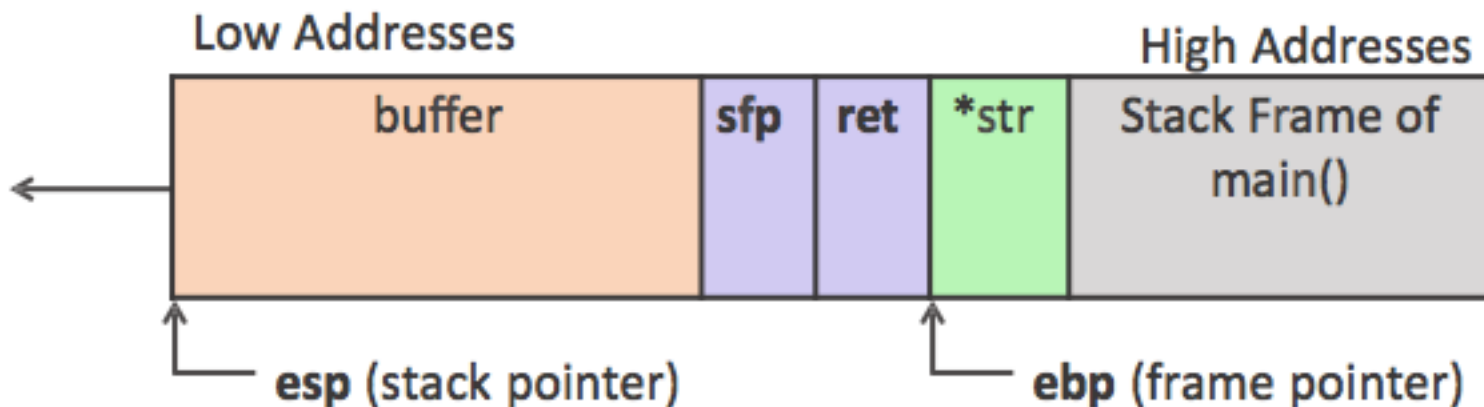
Stack Smashing

- In C and C++, low level functions like strcpy perform no bounds checking at all
- If str is long, we can write into other memory

```
void function(char *str)
{
    char buffer[128];
    strcpy(buffer, str);
}
```

Copy str into local buffer

Allocate local buffer

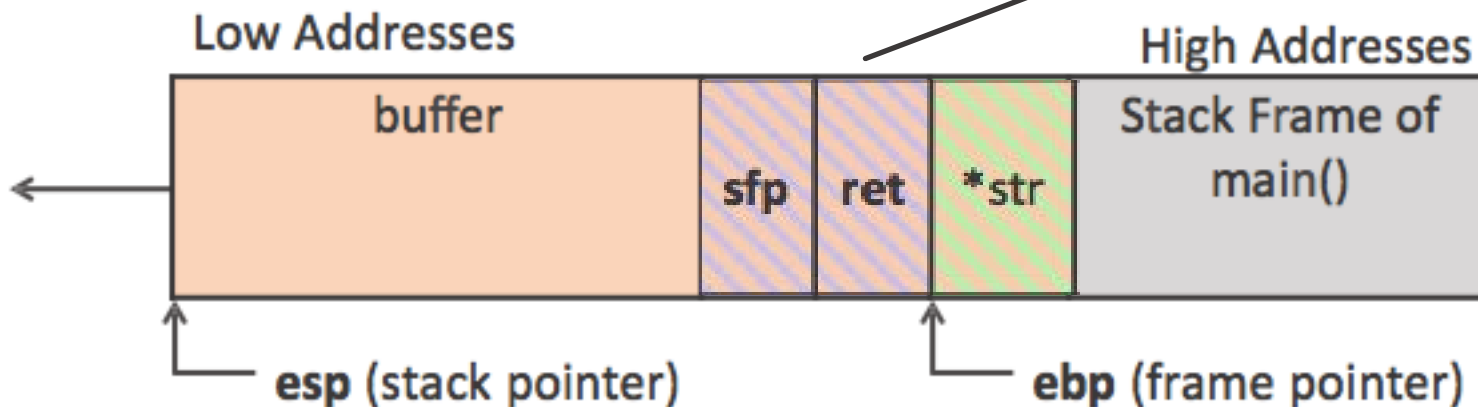


Stack Smashing

- By crafting the string `str`, we can overwrite the buffer and the return address with custom exploit code!

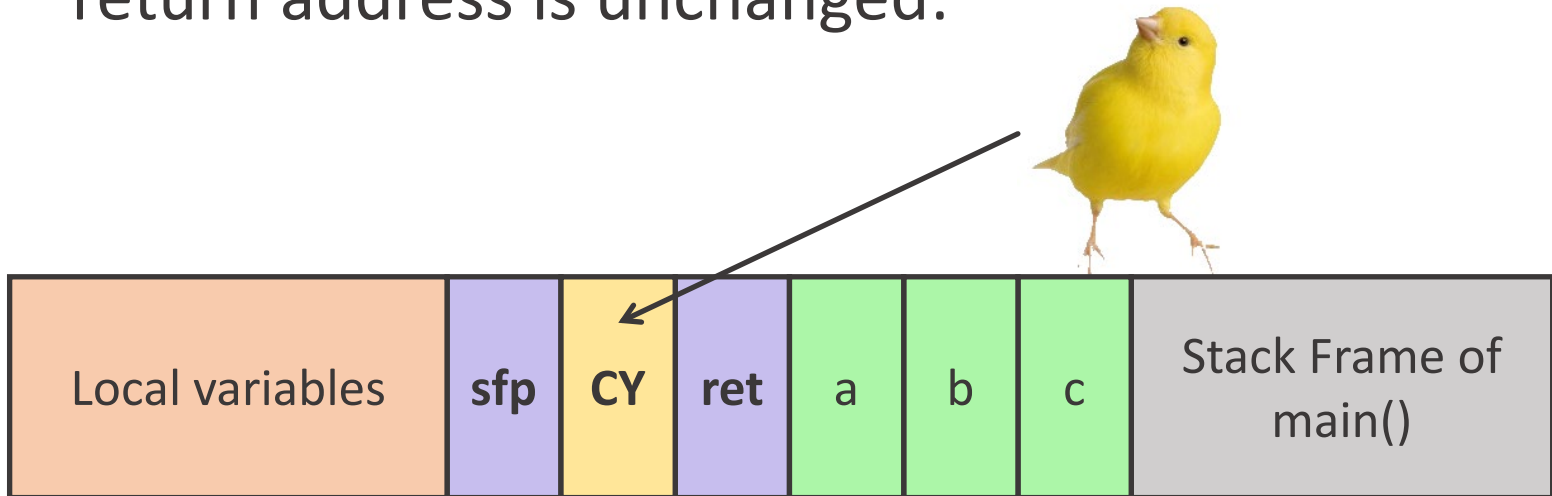
```
void function(char *str)
{
    char buffer[128];
    strcpy(buffer, str);
}
```

Return value
overwritten!



Protection: Canaries

- Stack canaries modify the prologue and epilogue of all functions to check a value in front of the return address is unchanged:



- If you can work out the canary value, there is no issue. You could also corrupt the Structured Exception Handler (SEH)

Data Execution Prevention (NX)

- Modern operating systems (where possible) will mark the stack as non-executable.
 - NX on AMD, XD on Intel, XN on arm
- An NX stack means that adding in our exploit code won't work
- We can circumvent this using a return-to-libc attack

Further Protection

- To defeat ret2libc, various 0x0 null bytes are inserted into standard libraries
- Developers also restrict access to obvious system calls
- Address Space Layout Randomisation (ASLR) moves the address of library and programs around
- They don't have to move too much before your hand-crafted **ret** addresses will break

Race Conditions

- With concurrent threads or processes, timing can lead to security vulnerabilities:

Victim

```
if (!access("file", W_OK))
{
    exit(1);
}

fd = open("file", O_WRONLY);
write(fd, buffer,
      sizeof(buffer));
```

Attacker



Race Conditions

- With concurrent threads or processes, timing can lead to security vulnerabilities:

Victim

```
if (!access("file", W_OK))
{
    exit(1);
}

fd = open("file", O_WRONLY);
write(fd, buffer,
      sizeof(buffer));
```

Attacker

```
...
symlink("/etc/passwd", "file");
...
```

r00t::0:0:0wned:/root:/bin/bash

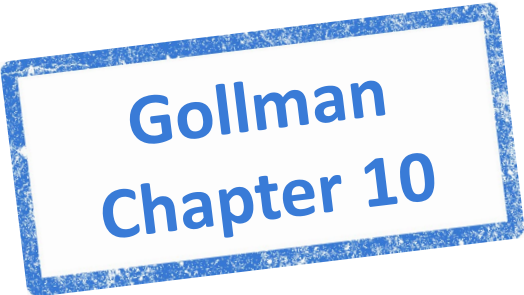
Summary

- **Malware**

- Viruses
- Worms
- Trojans
- Ransomware
- Rootkits and Backdoors

- **Exploits**

- Coding Flaws
- Common Vulnerabilities
- Buffer overflows
- ...



**Gollman
Chapter 10**