

Computer Crime

Thomas Schwarz, SJ

Malware

Thomas Schwarz, SJ

Phases of an Attack

- Reconnaissance
 - Scanning, Searches, Social Engineering
- Vulnerability discovery
- Initial vulnerability exploitation
- Privilege escalation
- Attack permanence and anti-forensics

Threat Landscape

- Majority of attacks involves users
 - Majority of user-related attacks use some form of **social engineering**
- Majority of other attacks uses known vulnerabilities against unpatched systems

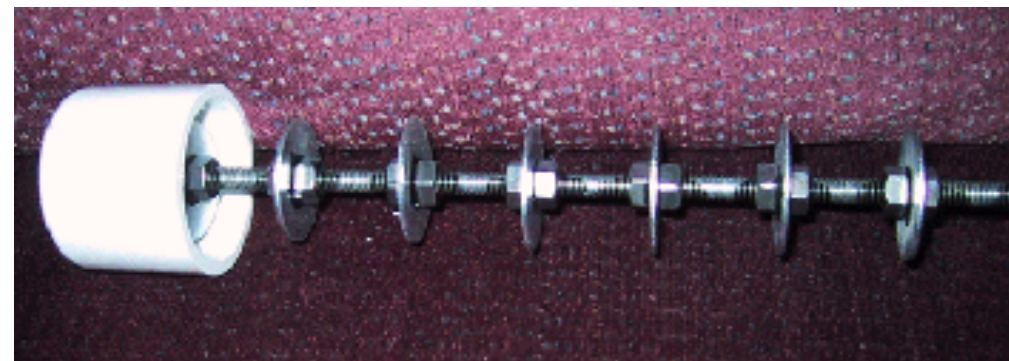
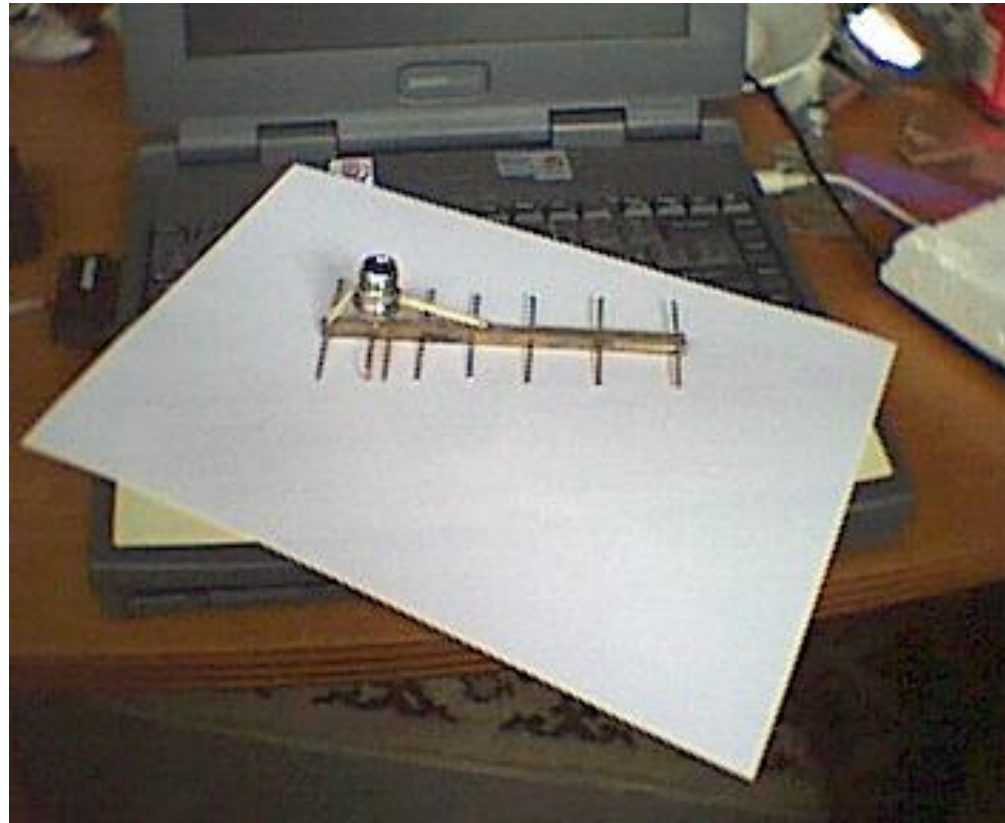
Scanning

- Scanning reads traffic on publicly accessible or protected media

Scanning

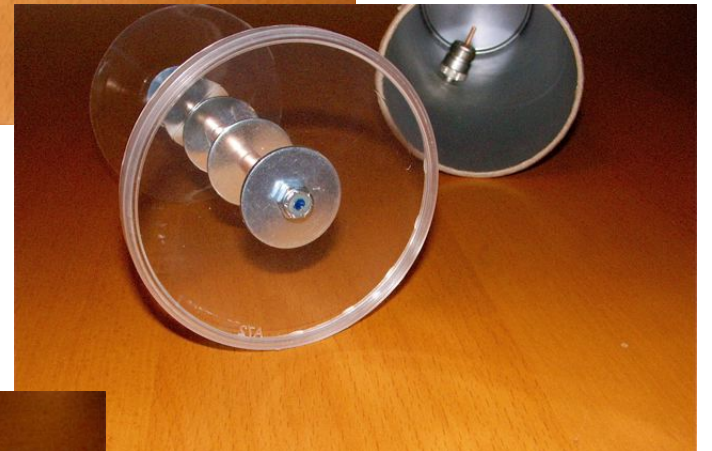
- Wireless scanning
 - Intercept wireless traffic
 - War driving — a geek sport:
 - Finding wireless access points and classify them
 - WLAN connections usually require a distance of at most 100 meters to function well
 - With antennas, signals can be received for kilometers
 - Omnidirectional antenna for discovery
 - Unidirectional antenna for scanning a target

War Driving



Homemade “cantennas”

Security Project at Santa Clara University



Security Project at Santa Clara University



Scanning

- Cantennas
 - Legality is unclear and might depend on state and local law
 - Software sends out a discovery message every 100 msec

Scanning

- Active wireless scanning:
 - Broadcasts 802.11 probes with ESSID “Any”
 - Clients reveals themselves by response
 - Implemented in netstumbler or since Windows XP2
- To discover packets
 - Put the wireless card in monitor mode
 - Known as rfmon
 - Analyze all received packets
 - Done by tools Wellenreiter and Kismet

Scanning

- Active wireless scanning
 - Some WLAN ignore probing packets with ESSID “ANY”
 - Need to find MAC address of access point
 - Tool then sends wireless de-authentication messages to the client with sender MAC of the access point
 - Clients now need to re-associate, which reveals the ESSID

Scanning

- Hardening against wireless scanning
 - ESSID should not give out information about the society
 - WLAN access points should ignore packets without ESSID
 - Use stronger authentication (than MAC addresses)
 - Use WPA instead of WEP
 - Lower the energy level of packet transmission

Scanning

- War-dialing
 - An old technology that still can find victims
 - Go through the phone number blocks of an organization to find modems
 - Which might use remote access
 - VNC, pcAnywhere, Mini remote control, Laplink gold,
...

Scanning

- Mapping a fixed net
 - Adversary has already gained access to the net
 - Techniques for information gathering:
 - Sweeping: ping all reachable addresses
 - Port mapping:
 - Discover / classify applications at an open port
 - Various methods, some stealthy, some usually logged

Scanning

- Port mapping
 - TCP scan: performs the TCP scan
 - TCP syn scan:
 - Adversary sends the initial SYN message, receives SYN-ACK, but does not follow up with ACK
 - Target drops connection
 - Usually not logged

Scanning

- Port mapping
 - TCP-protocol non-compliant scans
 - TCP-FIN
 - Adversary sends a FIN packet to a port
 - Target has to send RESET if the port is closed
 - Target has to not send anything if the port is open
 - Xmas tree scan
 - Adversary sends a packet with (contradictory) flags URG, ACK, PSH, RST, SYN, and FIN
 - Null scan
 - Adversary sends TCP packet without a flag
 - Closed port sends RESET, open port sends nothing

Scanning

- Port Scanning
 - All these scan messages should be intercepted by a firewall
- TCP-ACK scan
 - Adversary sends packets with ACK flag set
 - Stateless firewall will let these true
 - Target will respond with RESET
 - Absence of a response shows the existence of a state-full firewall

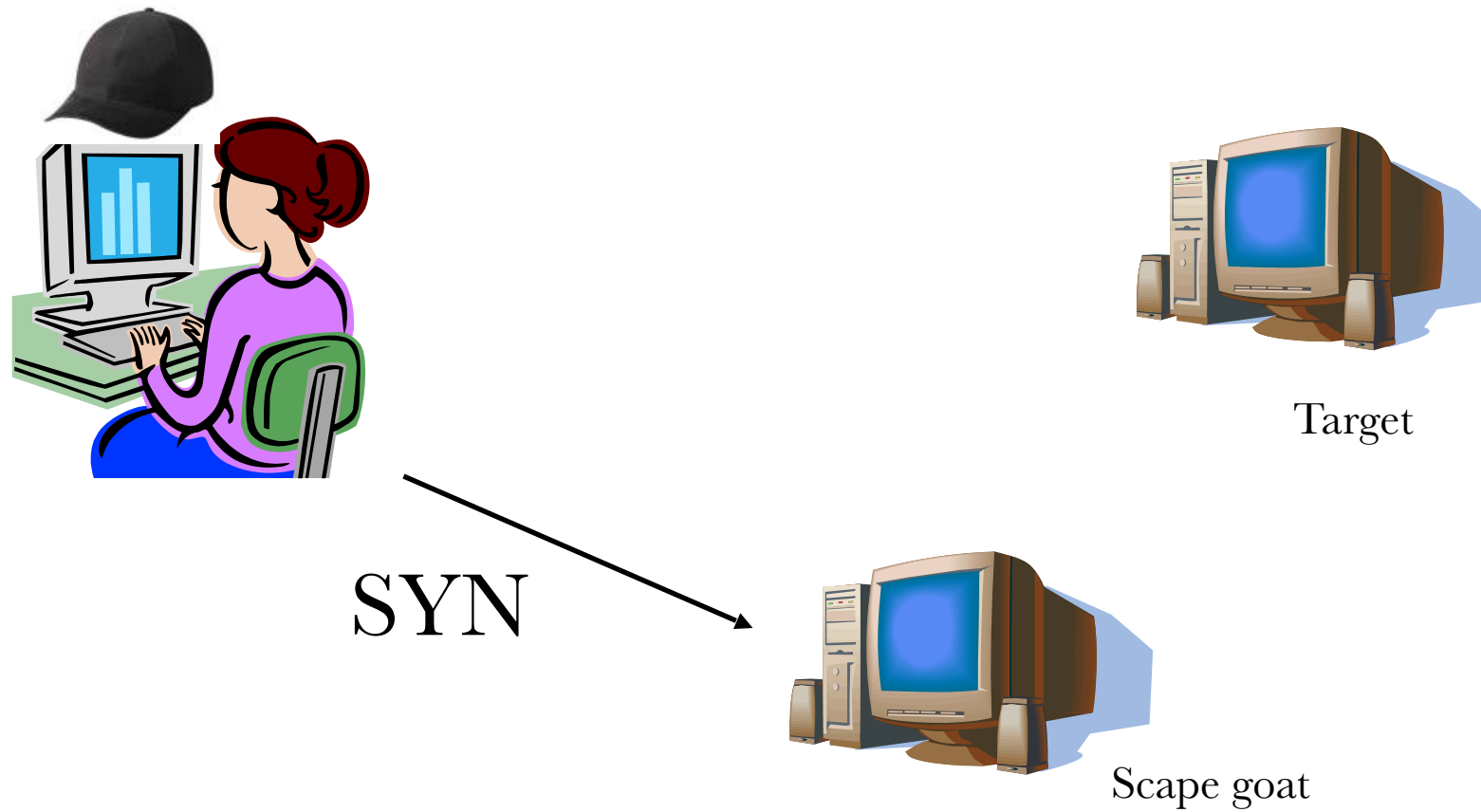
Scanning

- Port mapping
 - FTP Bounce scans
 - The scanning machine's address does not appear in a log
 - Uses an old ftp option for ftp printers
 - FTP server accept connections from a user but sends files to another system
 - Adversary requests a file to be sent to a port on the target
 - Error message to adversary gives port state:
 - Open port: FTP server has opened a connection but could not communicate
 - Closed port: FTP server could not communicate

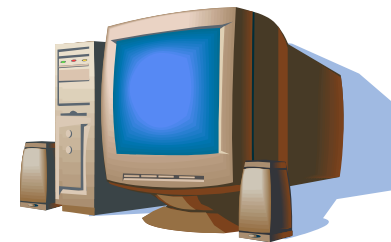
Scanning

- Idle scanning (another bounce example)
 - All IP packets contain a field IP-Identification
 - Used to identify fragments of a packet for reassembly at the receiver
 - Windows increments the IP
 - Adversary looks for a “zombie” as scape-goat
 - Adversary determines the IP of the zombie
 - Fakes message from zombie to target
 - When the target sends an ACK to the zombie (open port), the target

Idle Scanning

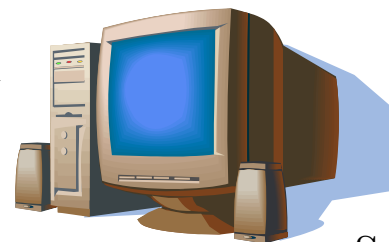


Idle Scanning



Target

SYN-ACK, IP-ID = 5

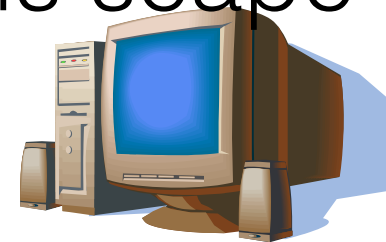


Scape goat

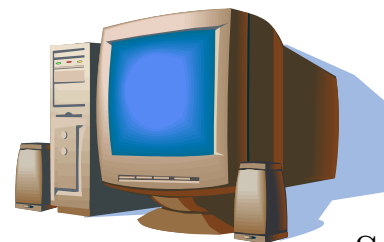
Idle Scanning

SYN to TCP port 12345

Source is scape goat

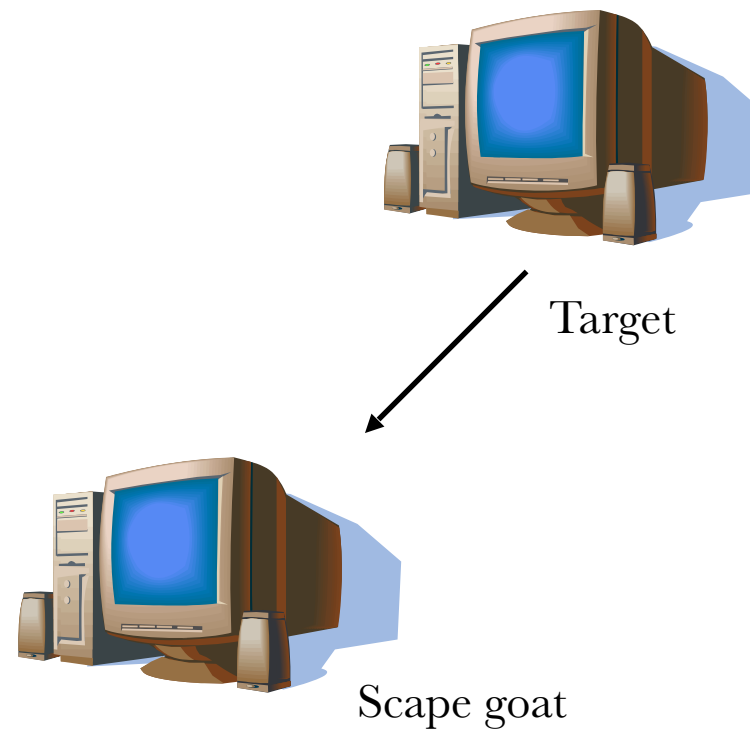


Target

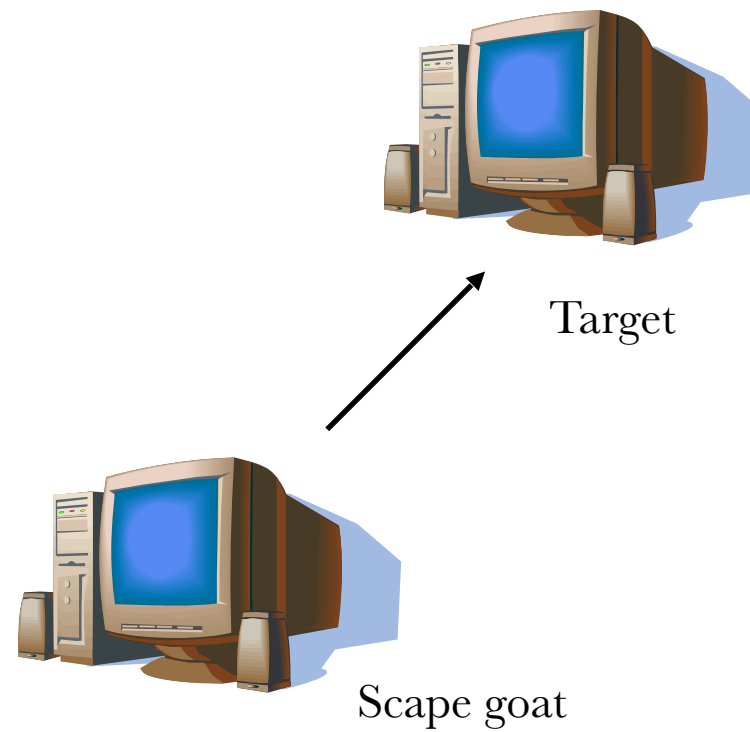


Scape goat

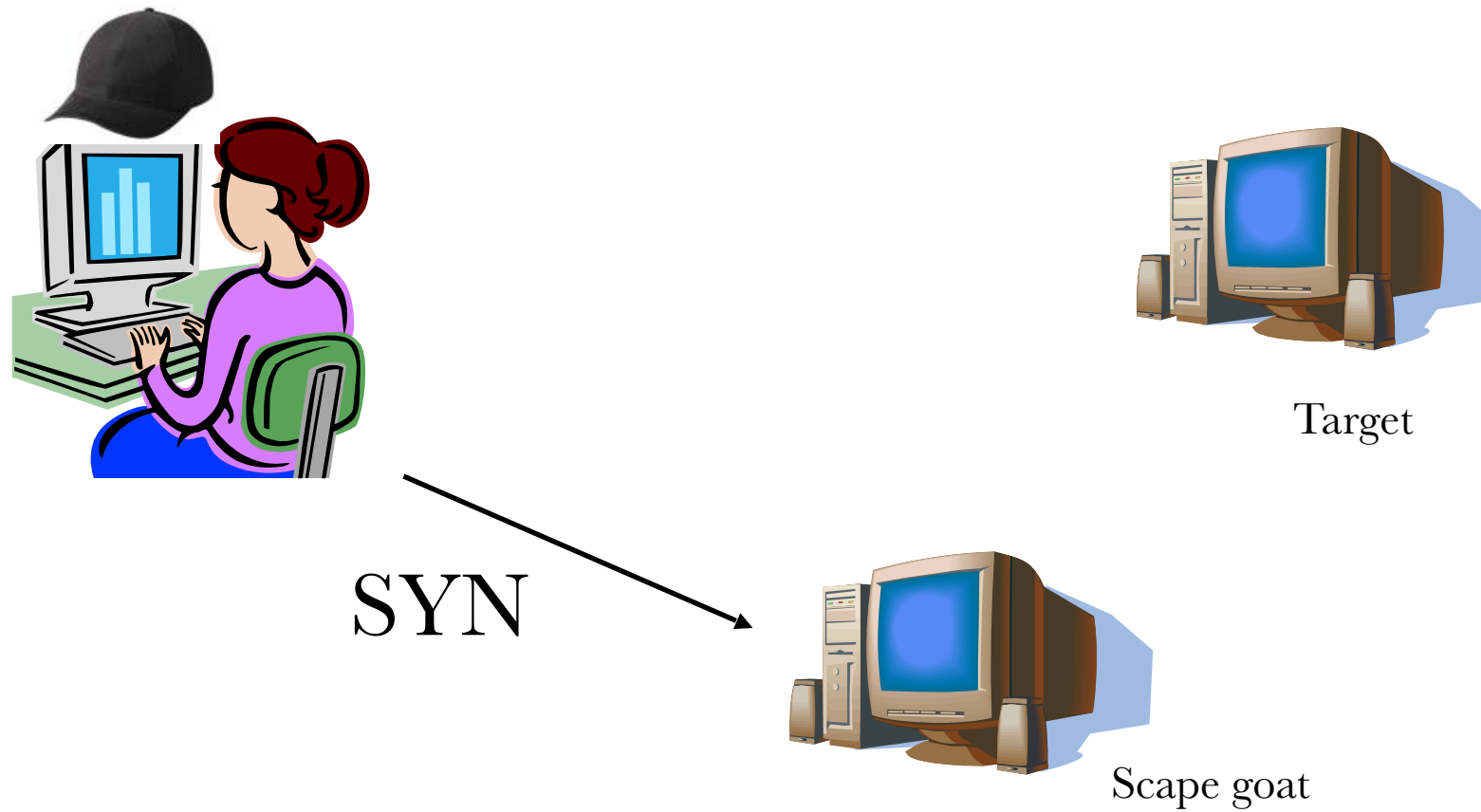
Idle Scanning



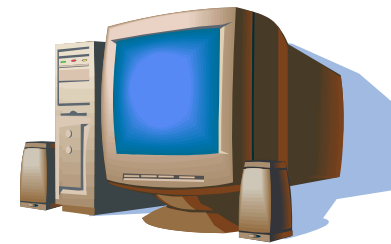
Idle Scanning



Idle Scanning

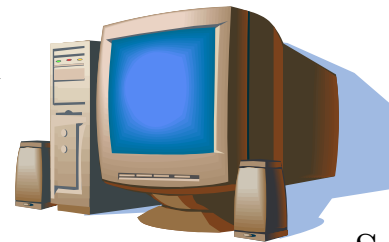


Idle Scanning

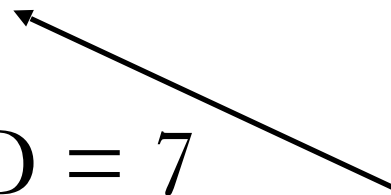


Target

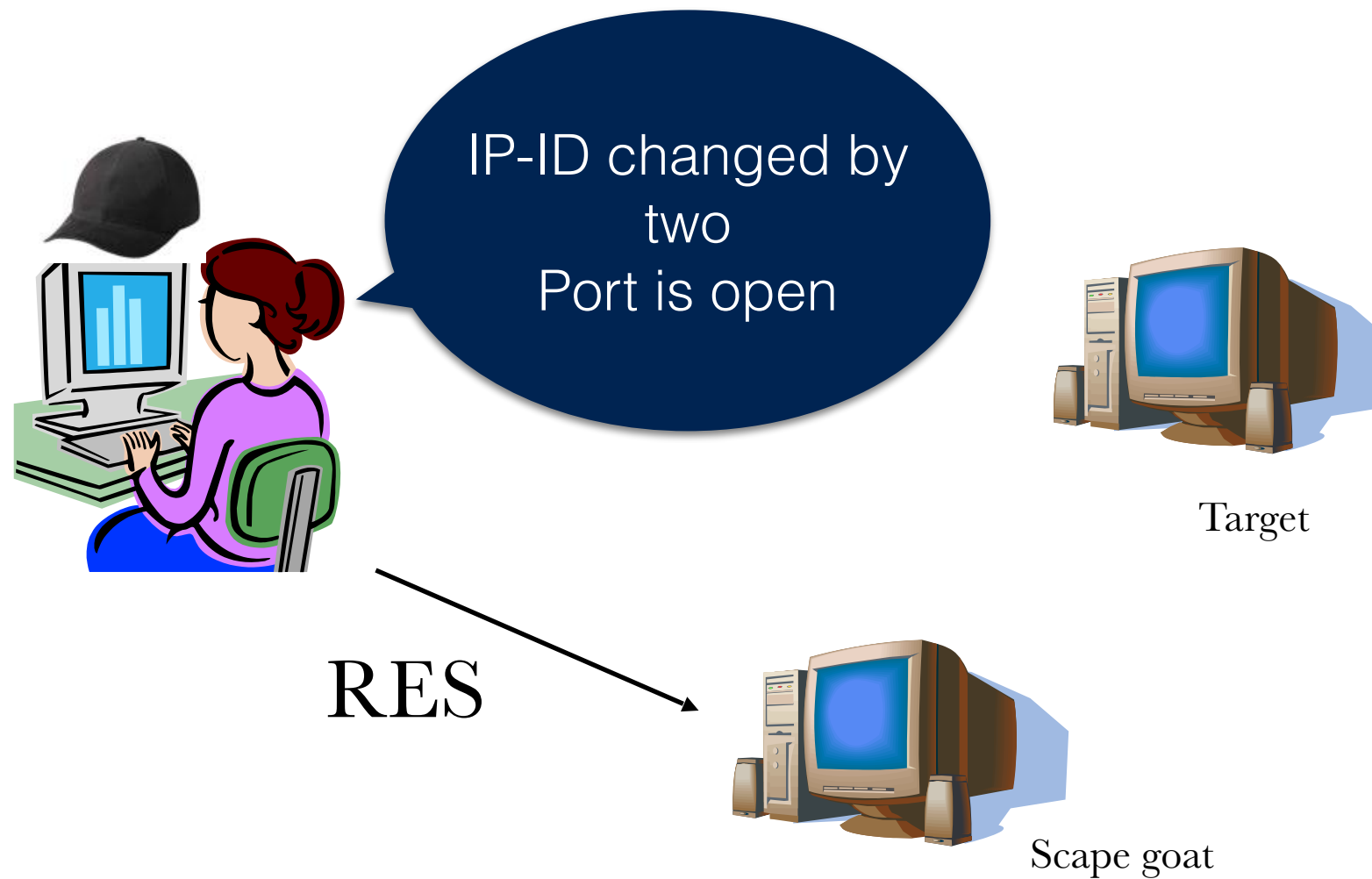
SYN-ACK, IP-ID = 7



Scape goat



Idle Scanning



IDLE SCANNING

- Methods for scanning can be sophisticated
- Packet inspection can find all scanning packets
- IDS should be able to alert to scanning

Sniffers



- Sniffers collect traffic from a LAN
 - Passive: Sniffer does not change net configurations
 - Stealth: Sniffer has no proper network activity
 - Active: Sniffer changes net configurations
 - Example: Changes router to route traffic through the sniffing machine

Sniffers

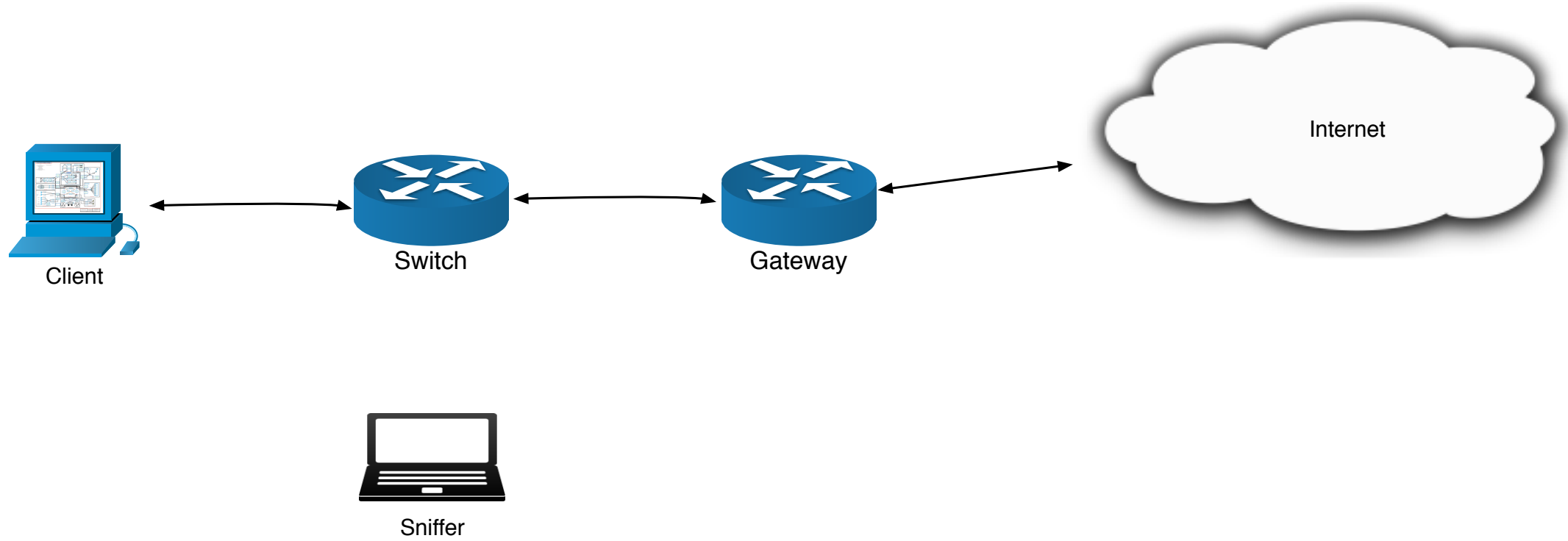


- Active sniffer:
 - MAC flooding
 - Switches have a table to associate MAC addresses and physical switch ports
 - Sniffer floods the switch with random MAC addresses
 - Table overflows: Switch becomes a hub

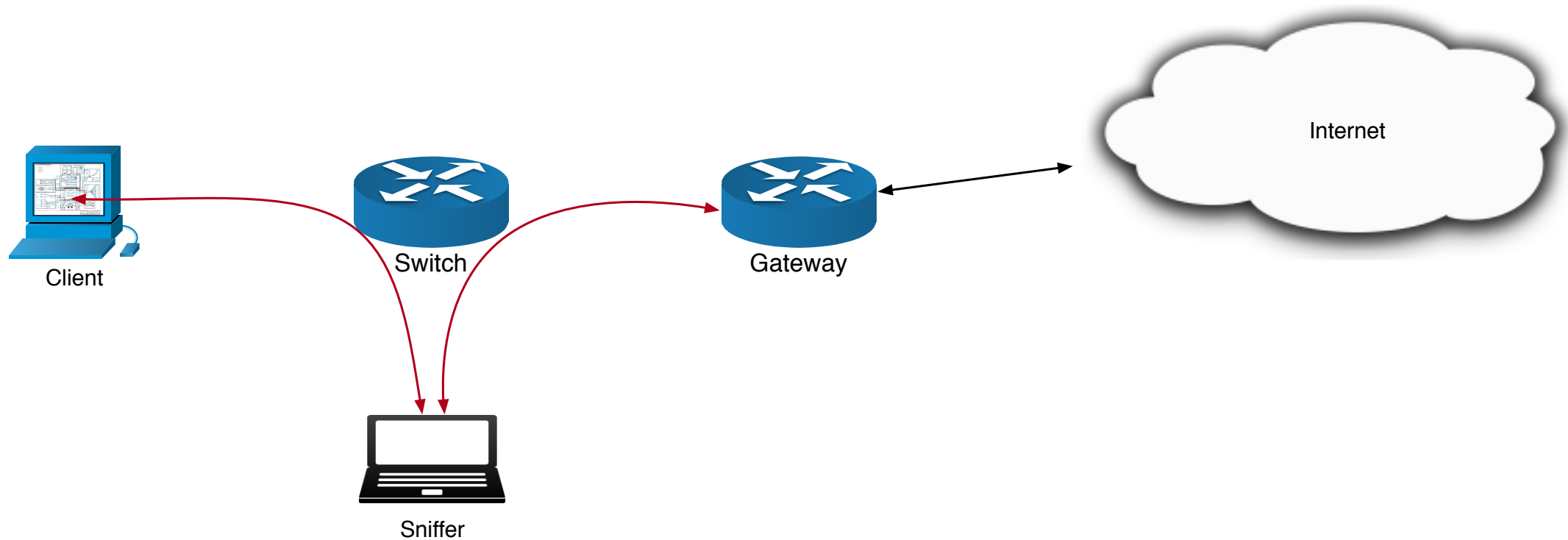
Sniffers



- Active sniffing
 - ARP spoofing
 - Use false Address Resolution Protocol messages in order to change the ARP mapping



After ARP Poisoning, the client thinks that the Gateway is at the address of the Sniffer



Sniffers



- ARP Poisoning
 - ARP resolves IP address into MAC addresses
 - Given an IP address but no MAC address, we send out a broadcast ARP message to the LAN
 - ARP-REQ: Who-has-IP-10.0.0.5?
 - Destination answers with
 - ARP-RESP: I have 14:98:1:1:34:0
 - The answer is put in a local ARP database
 - Usually age out
 - ARP packets are not authenticated

Sniffers



- ARP-Poisoning
 - Create false ARP responses
 - Target now associates the IP-address of the gateway with the MAC address of the adversary

Sniffers



- ARP-Poisoning counter measures:
 - Use static ARP addresses
 - Never age out
 - Can be set by the user
 - Useful to prevent ads from advertisers by setting their IP address to a not-existing MAC
 - Does not scale
 - Configure to ignore all ARP reply packets

Sniffers



- ARP-Poisoning detection
 - Cross-check all ARP responses on a LAN
 - at the OS (Anticap, Antidote) or in switches
 - Sniff all traffic for ARP messages and build check against a database
 - MAC with multiple IP addresses is suspicious
- Replace the ARP protocol with S-ARP

Sniffers



- Port Stealing
 - Ethernet switch associates MAC-addresses to its ports
 - If a packet is received, switch looks at the sender MAC and associates it to the port
 - Adversary sends (many) packets to the switch with target's MAC address
 - Switch updates association table
 - Sends traffic intended for target to adversary
 - Can resend after waiting:
 - Target has re-associated its MAC to its port at the switch

Sniffers



- Port Stealing
 - Use enterprise class switches with port security

Sniffers



- DNS spoofing:
 - DNS messages are (not yet) authenticated
 - Adversary intercepts DNS demands from the target and sends a false DNS response
 - Adversary becomes MitM

Sniffers



- TCP-nicing
 - To throttle bandwidth of a sniffed connection:
 - Adversary inserts small TCP windows announcements
 - Target lowers speed of TCP-flow



Backdoors

Backdoors



- Programs bypassing normal authentication procedure
 - Voluntary backdoors
 - E.g. ATM machines used to have a maintenance PIN
 - Installed after penetration to allow adversary continued access
 - E.g. Sony / BMG rootkit installed with purchased music CDs which collected usage data

Backdoors

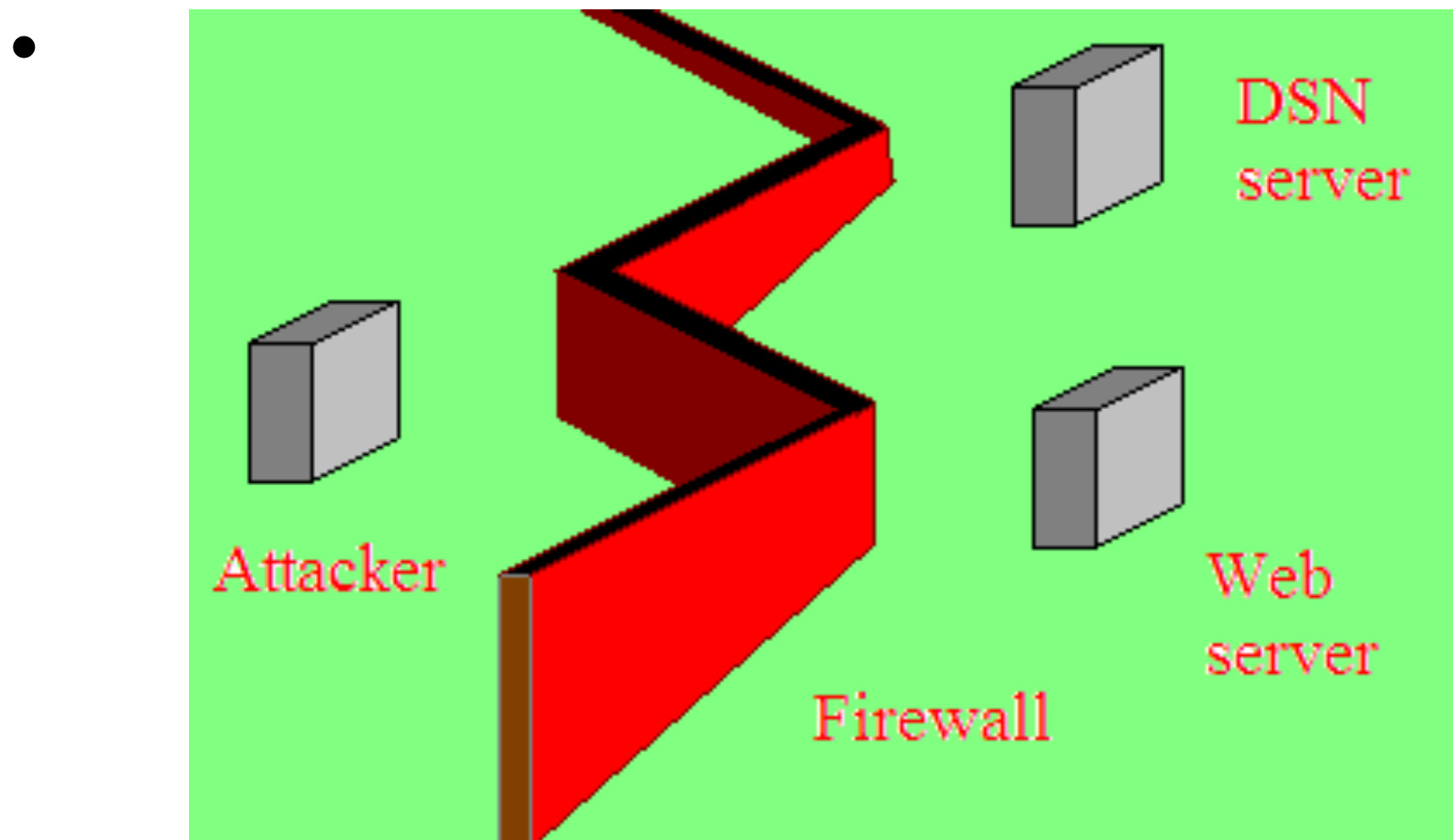


- Cd00r
 - Sniffer looks for TCP packets with destination ports X, Y, Z
 - When they arrive in this order:
 - System opens connection
 - Originally: Shoveling a TCP shell

Backdoors



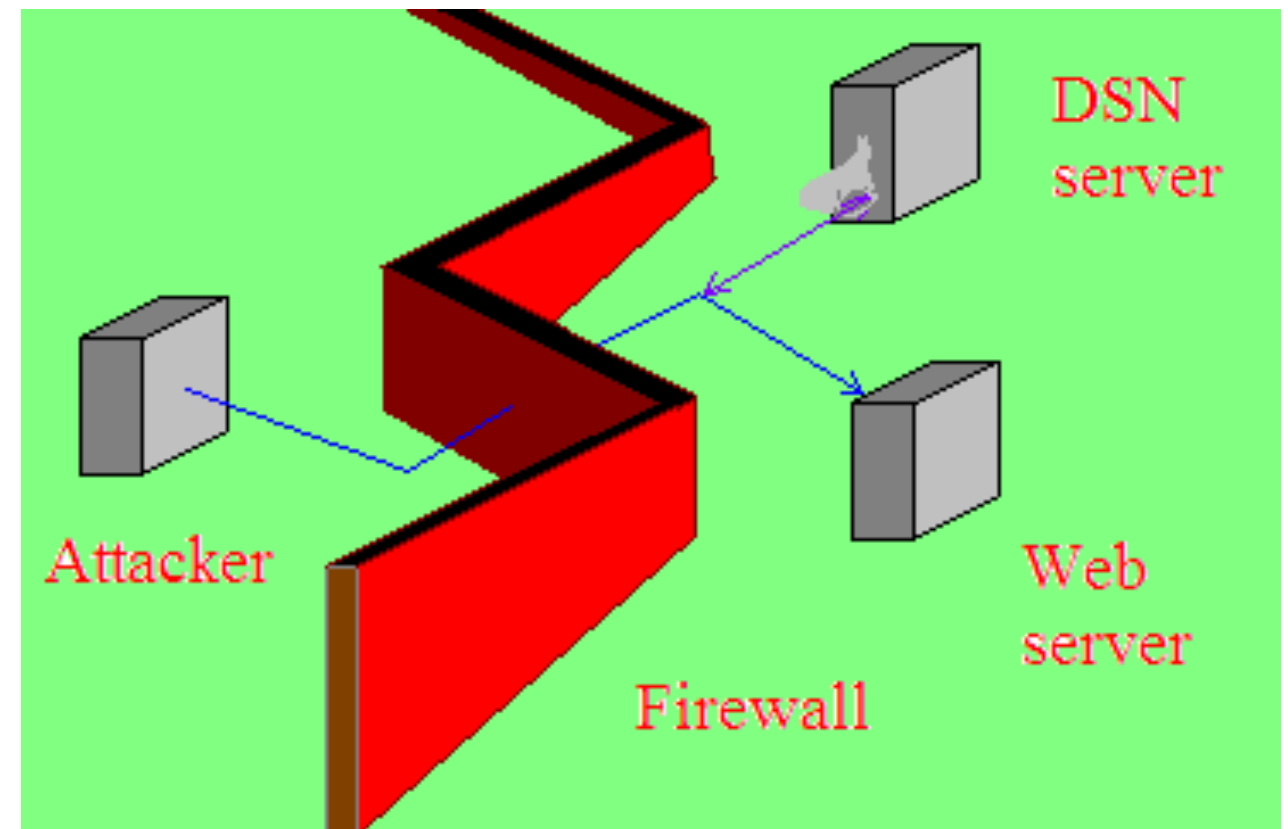
- Backdoor traffic can be hidden
 - Example: Put sniffer in promiscuous mode



Backdoors



- Adversary sends traffic to web server
- However, DSN server sniffs traffic
- Responds back to adversary
- Security team will suspect the web server and concentrate on it





Malware Properties

- Auto-replication
 - In contrast to passive replication
- Population increase
- Parasitic
 - Does the malware needs another application
 - Example: MS Word Macro Virus

Malware Types



- Logic Bomb
 - Consists of trigger
 - “Thomas Schwarz is no longer on the payroll”
 - And payload
 - “All student grades at Marquette are changed to F”
- Sometimes installed by a system administrator

Malware Types



- Trojan Horse
 - Program that appears to have a legitimate functionality
 - But also has a hidden functionality
 - Old vulnerability — Trojan login screen
 - Program presents itself as a login screen
 - Run by previous user
 - New user logs in
 - Login information is stored
 - Program exists and brings up the true login screen

Malware Types

- Virus
 - Auto-replicant
 - Parasitic — runs when another binary runs
 - Infects media or other programs
 - Name comes from SF novella
 - “The Scarred Man” Gregory Benford, 1970



Malware Types

- Worm
 - Auto-replicating
 - Population increases sometimes dramatically
 - Not parasitic
 - Distributed over the net

Malware Types

- Rabbits
 - Version 1: Increases so quickly that system resources are used up
 - Example: Fork bomb
 - Process that forks itself without restriction
 - Fill up process table and renders system useless
 - Version 2: Malware that infects other systems, and then deletes itself
 - Difficult to find because the malware switches from system to system in an installation

Malware Types

- Spyware
 - Malware that collects information and sends it to someone else
- Adware
 - Malware that installs advertising
- Clickware
 - Malware that visits websites, pretending to come from certain pages

Virus

- Virus consists of
 - Infection mechanism (possibly multiple)
 - Trigger (optional)
 - Payload (optional)

Virus

- Can be classified by infection target
 - Boot sectors
 - Files
 - Macros

Virus

- Virus can be classified according to openness
 - No hiding
 - Cryptographic protection
 - Virus consists of
 - penetrator (to install itself on the system)
 - decryption engine
 - encrypted malware
 - Stealth
 - Oligomorphism, Polymorphism, Metamorphism

Virus: Boot sector infection

- Boot sector:
 - Contains code that executes when the system starts
- Volume Boot Sector
 - First sector in a non-partitioned storage medium
 - First sector inside a partition
- Master Boot Sector
 - First sector in a partitioned storage medium

Virus: Boot sector infection

- Boot process:
 - Bootstrap loader
 - Loads the software that starts the OS
 - Multi-stage bootstrap loader

Virus: Boot sector infection

- Original boot sequence on IBM-PC
 - Executes instruction at F000;FFF0 of BIOS
 - Jumps to bootstrap code
 - Executes Power-On-Self-Test (POST)
 - Goes through a list of predefined storage media
 - When it finds a bootable medium, starts the code in the boot sector
 - Example: MBR of a hard drive
 - MBR has the address of the bootable partition
 - Loads boot sector of the bootable partition
 - Boot sector moves OS kernel to memory and then initializes it.

Virus: Boot sector infection

- Boot sector infector
 - Copies itself into MBR or VBR
 - Usually after making a copy of the true BR
 - Example: Michelangelo — April 1991
 - Moves original boot sector to a secure location
 - Infects all diskettes inserted into the computer
 - Payload: overwrites file system with zeroes
 - No longer in the wild

Virus: Boot sector infection

- Example: Stoned virus
 - 1988, originally only infected 360KB diskettes
 - Infects computers booted with an infected diskette
 - Moves MBR
 - Overwrites MBR
 - Infected system sometimes stop and presents a message, originally “Your computer is now stoned”
 - Variant found in Seagate drives in 1995
 - 2014: Signature was inserted into a bitcoin blockchain, causing Microsoft Security Essentials to remove the file, which is then reloaded

Virus: Boot sector infection

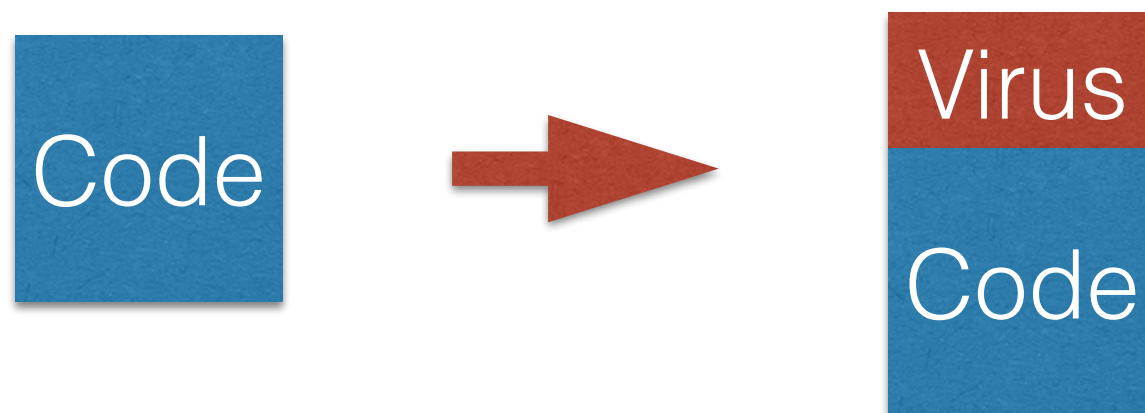
- Boot sector infectors
 - Extinguished in the wild
 - Diskettes are no longer used (to boot)
 - OS no longer allows writes to the boot sector without authorization
 - BIOS also can enable boot sector protection

Virus: USB / Pen Drive

- Uses automatic execution mechanism
 - autorun.inf has a list of files for automatic execution until Windows 7
- All USB automatically install drivers
 - Altered drivers can install memory resident software

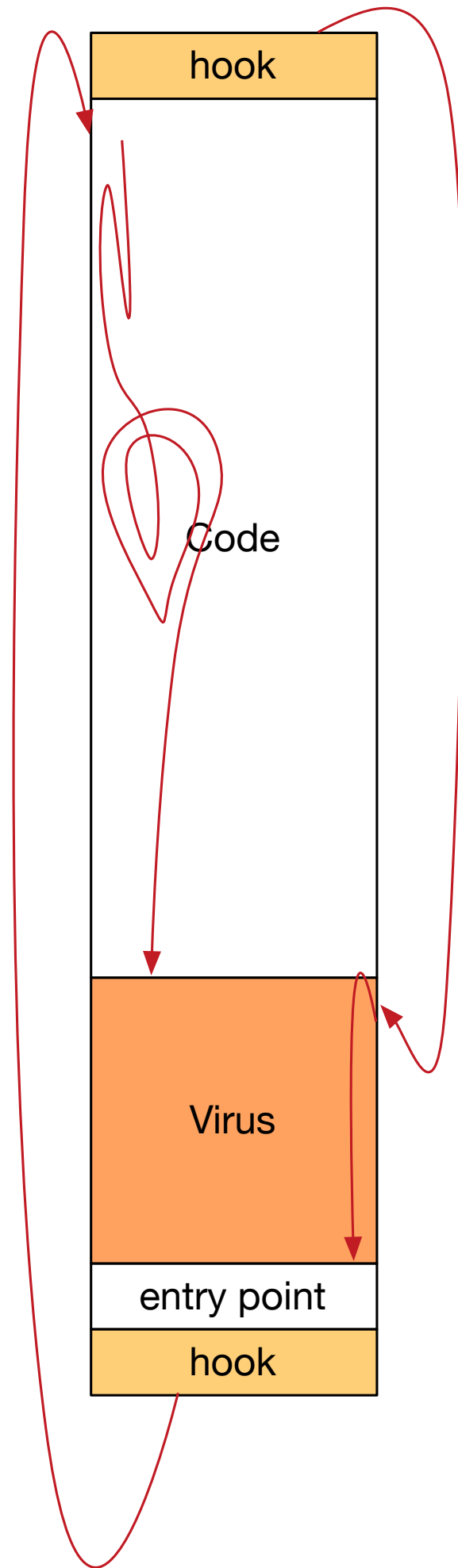
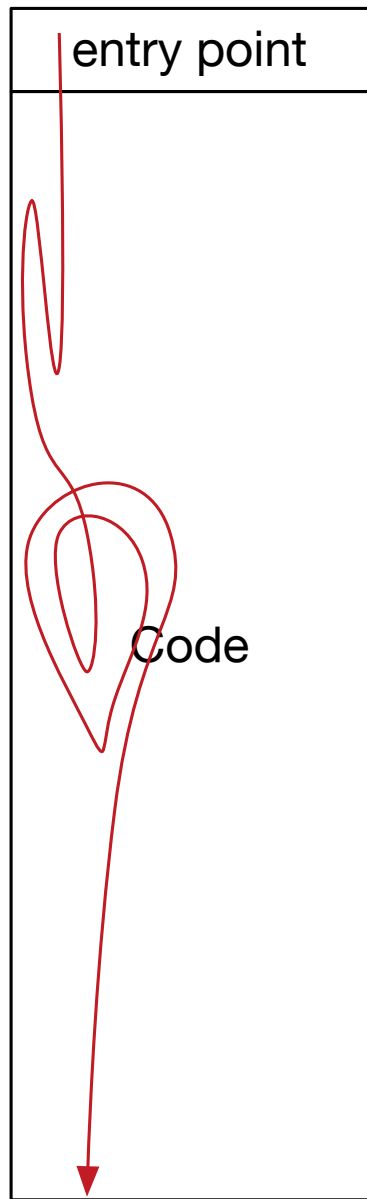
Virus: File infection

- Virus infects an executable
 - Virus is placed in an executable
 - At the beginning:
 - Typical entry point for programs
 - Rest of file placed afterwards



Virus: File infectors

- Virus can be placed at the end
 - To pass control to the virus
 - Save first instruction
 - Place a trampoline (hook): jump to the virus
 - After virus executes
 - places original first instruction into its original position
 - jump to safeguarded first instruction, then jump to rest of code



Trampoline example

- http://www.binaryguard.com/bgc/malware/sandbox/2015/11/09/dissecting_inline_hooks.html

74D56F01	8BFF	MOV EDI,EDI
74D56F03	55	PUSH EBP
74D56F04	8BEC	MOV EBP,ESP
74D56F06	83EC 10	SUB ESP,10
74D56F09	56	PUSH ESI
74D56F0A	57	PUSH EDI

- Beginning of normal code.
- Need to move five bytes to make room for the long jump (This is a CISC architecture)

74D56F01	- E9 FA902A98	JMP 0D000000
74D56F06	83EC 10	SUB ESP,10
74D56F09	56	PUSH ESI
74D56F0A	57	PUSH EDI

- This is the code after the change
- The target code stays after byte 5

0D000000	60	Save registers	PUSHAD
0D000001	90		NOP
0D000002	90		NOP
0D000003	90		NOP
0D000004	90		NOP
0D000005	90		NOP
0D000006	90		NOP
0D000007	90	Malicious code	NOP
0D000008	90	gets executed	NOP
0D000009	90	here in this block	NOP
0D00000A	90		NOP
0D00000B	90		NOP
0D00000C	90		NOP
0D00000D	90		NOP
0D00000E	90		NOP
0D00000F	61	Restore registers	POPAD
0D000010	8BFF	Execute code	MOV EDI,EDI
0D000012	55	Trampled by	PUSH EBP
0D000013	8BEC	hook	MOV EBP,ESP
0D000015	-E9 EC6ED567		JMP WS2_32.74D56F06

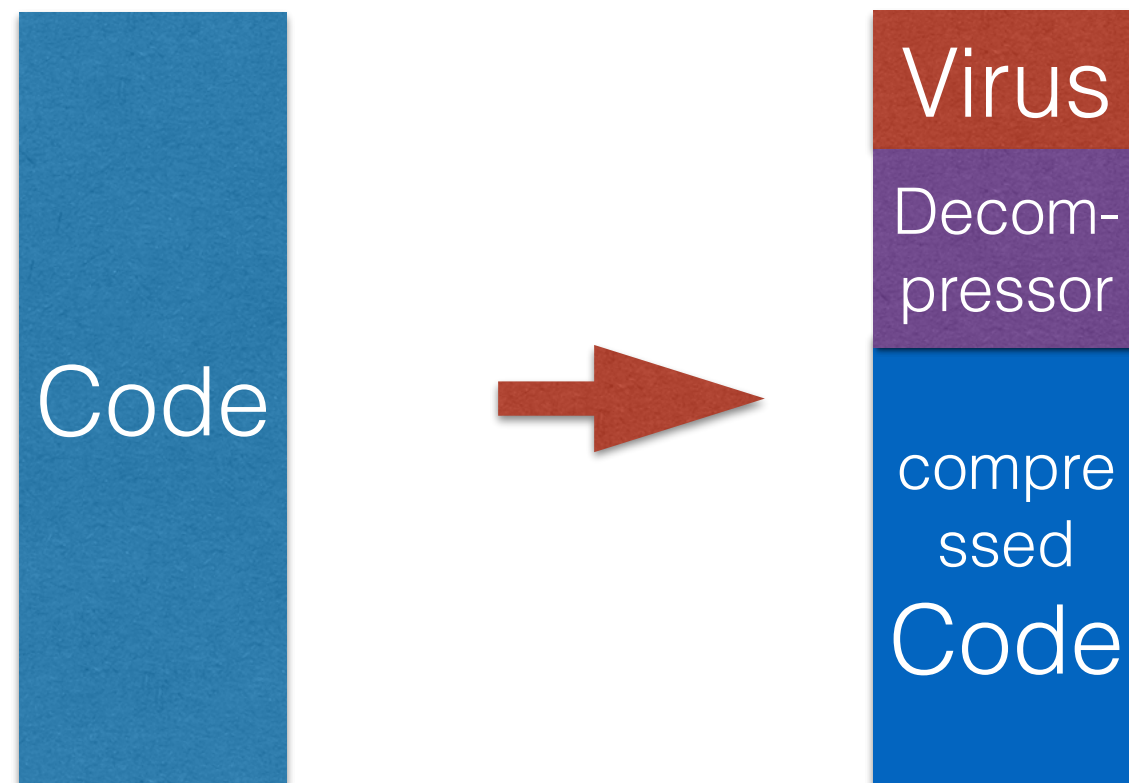
- End of malicious code:
 - Restore registers with POPAD
 - Execute first instruction
 - Jump back

Virus: File infectors

- Virus can overwrite the code
 - Cavity virus: Needs to identify a useless / empty part of the original code and inserts itself there
 - File slack (= non-assigned part of file)
 - Overwriting virus:
 - Superfluous data such as error messages
 - Can Put overwritten code into a companion file
 -

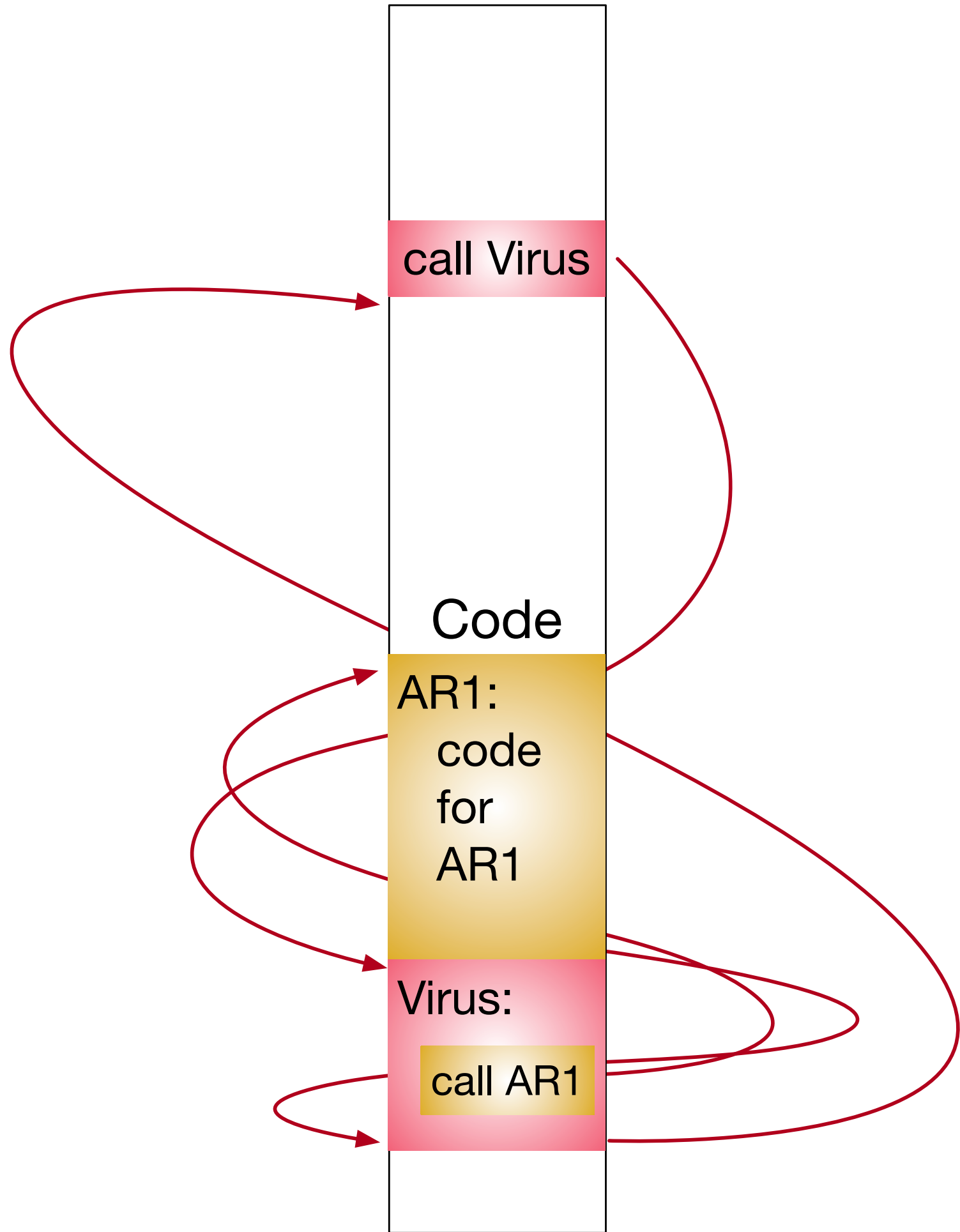
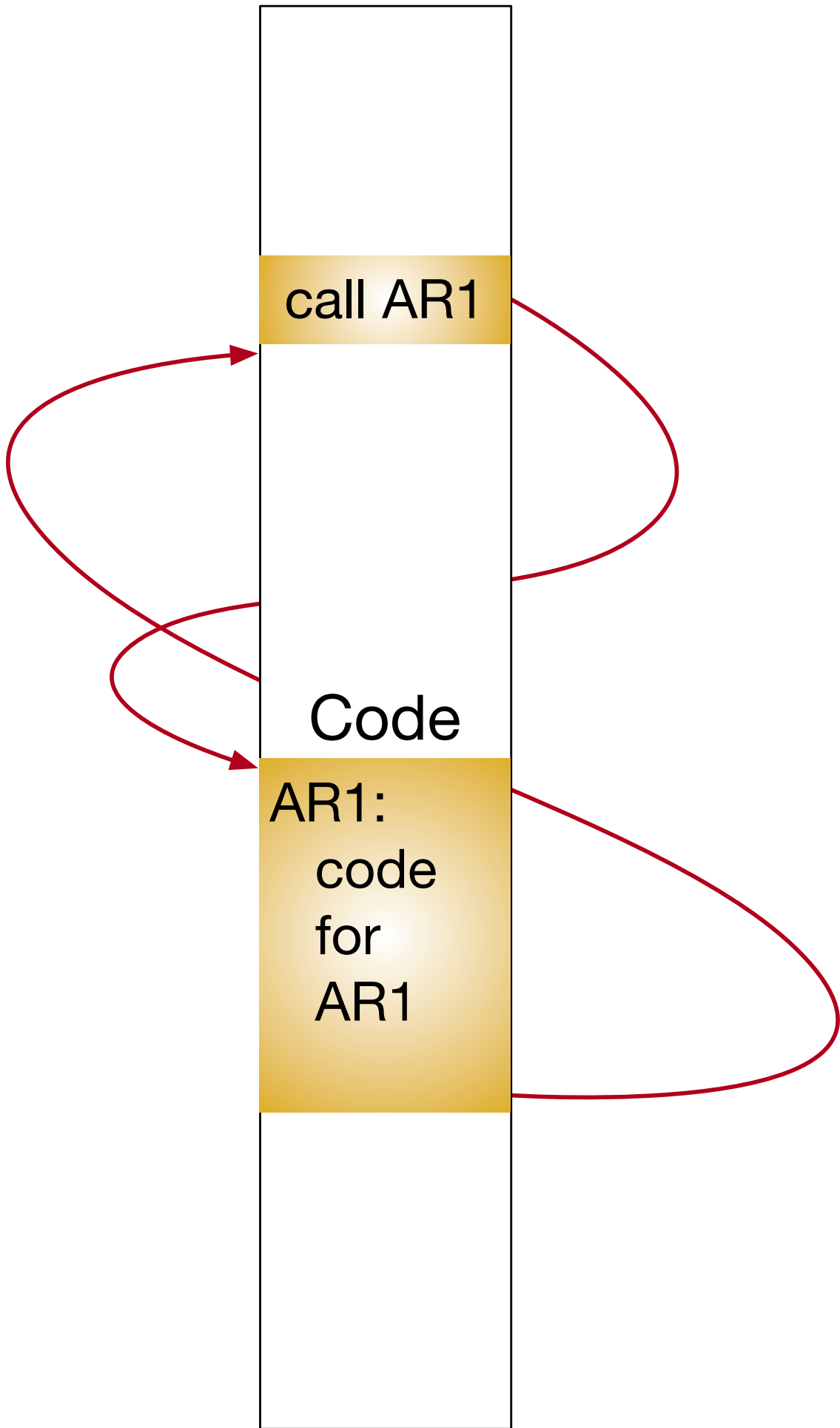
Virus: File infectors

- Compressing virus
 - Uses compression and decompresses after virus execution



Virus: File infectors

- Entry Point Obfuscation Virus
 - Stealth:
 - Anti-virus emulator will not find the virus if the virus does not execute soon
 - Entry-point checking is part of anti-virus heuristics
 - Overwrite the target of a function call
 - Function call now results in calling the virus



Virus: File infectors

- Companion Virus
 - Exploits the way executables are selected
 - MS Windows: prefers .COM over .EXE
 - If a .COM and a .EXE file share the same name, then the .COM file is executed
 - Companion virus has the same name as a popular program (notepad.exe) but different extension (notepad.com)

Virus

- Code Virus
 - Virus changes source code of a system file
 - If altered file is compiled and executed on a new host, the virus has spread

Macro-Virus

- A tale of the woesome consequences of mingling data and programs
 - Macros: code that is part of a document
 - Used in MS Office
 - Written in Visual Basic for Applications (VBA)
 - Virus creates new macro or changes old one

Macro Virus

- Concept (1995-1997)
 - Infects the metadata file normal.dot for Word (<Word2007)
 - Creates PayLoad and FileSaveAs macros
 - All documents saved with Save-As are infected

Macro Virus

- Laroux (1996 —)
 - Excel macro virus
 - Consists of macros auto_open and check_files
 - auto_open executes when a spreadsheet is opened, followed by check_files
 - virus looks for PERSONAL.XLS
 - Apparently, was a PoC and had no payload

Macro Virus

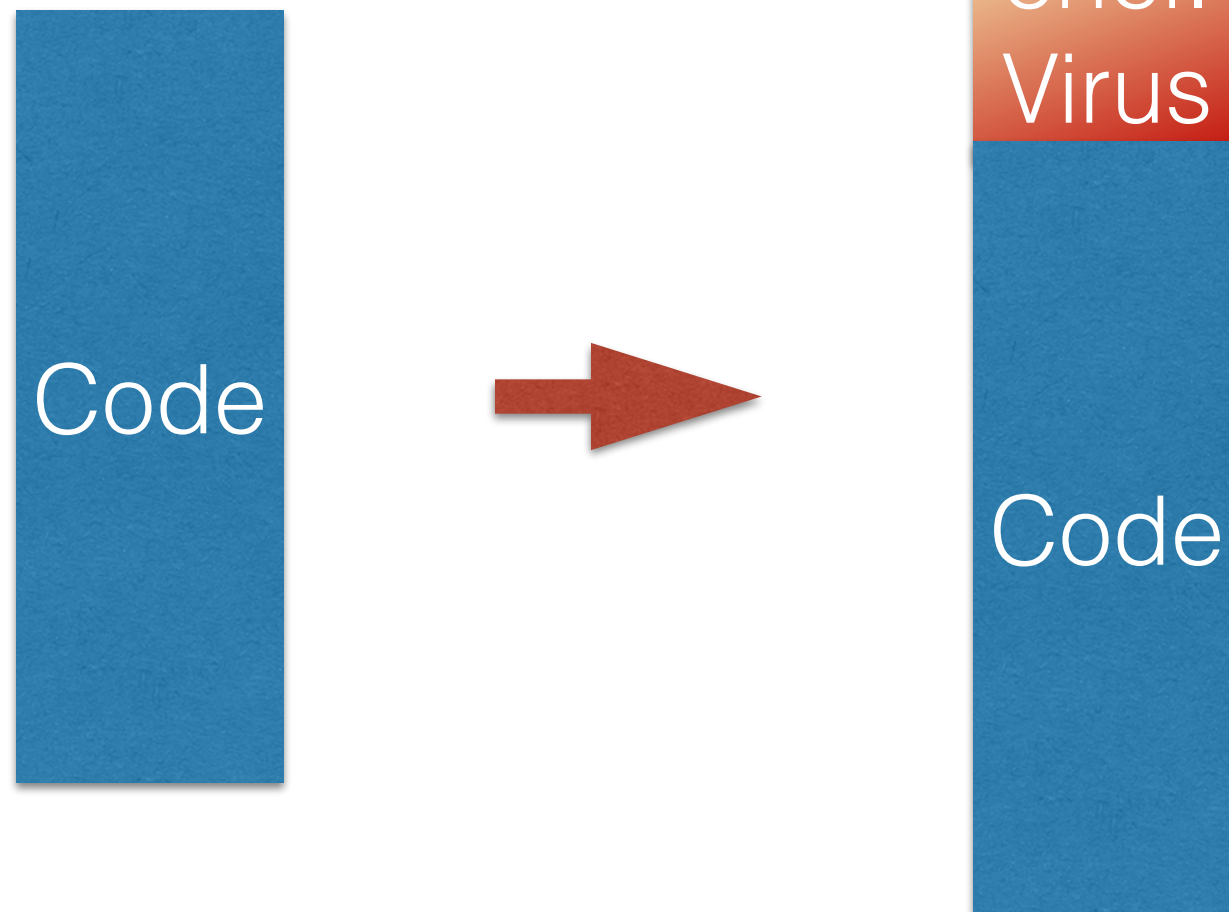
- Macro virus protection
 - Forced MS to abandon Macros as a generic tool
 - More modern versions of MS Office have security level for macro execution
 - High security level only allow signed macros to execute
 - Warn of macros being loaded
 - Implemented scanning for Concept virus

Virus Obfuscation

- No obfuscation
 - Virus writers hope for success before the virus is captured, analyzed, and scan procedures are developed

Virus Obfuscation

- Encryption:
 - Virus body is encrypted
 - Decrypter inserted



Virus Obfuscation

- Obfuscation via Encryption
 - Encryption can be simple (XOR encryption) or sophisticated (AES)
 - Key can be fixed or vary

Virus Obfuscation

- Oligomorphism
 - Use different decrypters - encryption methods
 - Virus appears in several versions
- Polymorphism
 - Change keys, use different decrypters
 - Virus appears in many versions

Virus Obfuscation

- Metamorphism
 - Virus code changes
 - E.g. decrypter changes in addition to the key
 - There are many equivalent ways to obtain the same goal, especially in Assembly
 - To set a register to zero:
 - Load zero, XOR with itself, AND with itself, Load a register with zero contents, copy register elsewhere then AND the two, ...

Virus Obfuscation

- There are many equivalent ways to obtain the same goal, especially in Assembly
 - Reorder instructions
 - Remap registers
 - Reorder data
 - Spaghetti code
 - Insert junk code
 - Generate code during execution
 - Use concurrent threads

Virus Obfuscation

- Invisible virus
 - Hides infection by antiforensics:
 - Changes time-stamps back
 - Use rootkit techniques such as intercepting system calls ...

Virus Obfuscation

- Visible virus:
 - Virus payload makes all executables appear infected
 - Antivirus system kills the system
 - (SWATing system)

Anti-Virus Techniques

- Malware handling
 - Detection
 - Containment
 - Identification
 - Disinfection \ Eradication
 - Recovery

Static Malware Protection

- Protect all system files
 - Tripwire:
 - Calculate cryptographic hashes of all system files and store them in a secure place
 - Periodically scan system files and compare hashes
 - Restore changes system files
 - Updates are now more complicated
 - Used in MS Window systems

Static Anti-Virus Techniques

- On demand / on access scanning
 - Signatures:
 - Identify short strings that identify the existence of a virus in an executable
 - with a minimal false positive rate
 - Signature based scanning looks for these scan strings
 - Theoretical question: How best to scan for a large number of scan strings

Static Anti-Virus Techniques

- On-demand / on-access scanning
 - Positive and negative heuristics

Static Anti-Virus Techniques

- Boosters (positive heuristics)
 - Garbage code
 - Loops as if deciphering / decompression goes on
 - Self-modifying code
 - Manipulation of interrupt vectors
 - Unusual statements, esp. those not created by a compiler
 - String constants with obscenities or virus
 - Distance between entry point and beginning / end of file
 - Spectral analysis (frequency distribution of statements)

Static Anti-Virus Techniques

- Stoppers (negative heuristics)
 - User input
 - Pop-up GUI

Static Anti-Virus Techniques

- Machine learning techniques:
 - Neural nets
 - Data mining technology
 - Trained with examples of good and of infected executables

Static Anti-Virus Techniques

- Integrity Protection
 - Tripwire:
 - Calculates crypt. secure hashes of all system files
 - Stores them in an unchangeable directory
 - Such as a CD-ROM
 - Periodically scans all system files
 - System update:
 - Check integrity
 - Update / patch
 - Create new integrity data

Dynamic Anti-Virus Techniques

- Blockers based on behavior
 - Have a software monitor in real time
 - Look for suspicious activities
 - Examples:
 - Self-modifying code
 - Jumping behavior of an appended virus
 - Avoids some false positives by implementing resource ownerships by executables

Dynamic Anti-Virus Techniques

- Emulation
 - Analyze code before letting it execute
 - Emulation uses dynamic heuristics
 - All the static heuristics and behavior
 - Emulation can detect a decryption loop
 - Executable executes code that was recently written
 - Uses more than 24B
 - Emulation can use file signatures on modified code

Dynamic Anti-Virus Techniques

- Emulator architecture
 - Single-stepping (can be detected by supervised / debugged executable)
 - Instead: complete emulation of CPU, memory, hardware, and OS (clock is now virtual)
- Emulation control — most important decision is when to stop
 - Dynamic criteria: # of statements, emulation time, proportion of statements that modify memory, presence of stoppers

Dynamic Anti-Virus Techniques

- Emulator:
 - Post-emulation analysis
 - Use a histogram of statements
 - Detect dead code
 - Indicates changed executable

Dynamic Anti-Virus Techniques



- Emulator-reentry
 - Virus code can be specific to CPU type
 - Need to emulate with various CPUs
 - Install an interrupt handler
- Emulating file access
 - Use sacrificial goats — artificial files — to detect infected behavior

Virus Verification

- Verification
 - To reduce false positives
 - Necessary for disinfection
 - Difficult with polymorphic virus
 - Can use emulation results
 - Sometimes can break weak cryptography
- Identification
 - File signatures
 - Known virus copy

Virus Quarantine

- Quarantine isolates infected file
 - Usually maintains a quarantine directory
 - Can use weak cryptography to prevent executables in quarantine from being runnable
 - Can make them invisible

Disinfection

- Restore from backup
- Counteract specific virus actions

Anti-Anti-Virus Techniques

- Virus can attack anti-virus software
- Virus uses code obfuscation to make work harder for anti-virus labs
- Use information about virus detection techniques for countermeasures

Anti-Anti-Virus Techniques

- Virus deactivates antivirus software
 - Comes with a list of process names
 - Ganda-virus revises programs in “autorun”
 - Changes first statement to “return”
 - Reduces priority of anti-virus software
 - Disables automatic updates

Anti-Anti-Virus Techniques

- Entry-point-obfuscation
 - Virus does not start at the beginning
 - Rather, is started by call-hooking
 - “Ganda” hooks `APIExitProcess`

Anti-Anti-Virus Techniques

- Virus survives emulation
 - Waiting: Virus does not do anything bad for a long time
 - Random replication
 - Entry point obfuscation
 - Be smarter than emulation
 - Restructure code to not trigger boosters

Anti-Anti-Virus Techniques

- Detection of emulation
 - Use of undocumented CPU statements
 - Use of different CPU features
 - Use up emulation memory
 - Try to catch running in emulator
 - For example, clock returns are unrealistic
 - Use obscure libraries
 - Access peripherals
 - Target individual emulators

Anti-Anti-Virus Techniques

- Armoring
 - Use anti-engineering techniques to make virus analysis difficult
 - Detect debugger and behave differently

Anti-Anti-Virus Techniques

- Anti-integrity protection
 - Use polymorphism to avoid simple check-sums
 - Slow virus: only infect files that are about to be changed anyway

Anti-Anti-Virus Techniques

- Avoidance
 - Only attack in areas where anti-software does not check

Worms

- Worms
 - propagate through the network
 - do not require (usually) action by users



Worms

- Worm components
 - Warhead
 - Propagation machine
 - Target selection mechanism
 - Scanner
 - Payload



Worms

- Warhead:
 - Code that exploits a vulnerability
 - Morris worm:
 - Vulnerabilities in sendmail
 - Vulnerabilities in finger
 - Vulnerabilities in ssh/rexec
 - Weak passwords



Worms

- Warhead:
 - Exploits a vulnerability to run commands on the remote system
 - Buffer overflows
 - File sharing systems
 - Email
 - Common configuration errors
 - Known default passwords



Worms

- Propagation Machine
 - After gaining access, the worm can execute commands
 - Worm now needs to transfer itself
 - Not all worms have propagation machines because they are so tiny
 - Other mechanisms
 - FTP, TFTP, HTTP, SMB (MS Server Message Block / Unix server with Samba)



Worms

- Target Selection Mechanism
 - Worm looks for new targets
 - Can use
 - Email address book
 - Server lists
 - Lists of trusted systems
 - Network neighborhood
 - DNS searches intercepted
 - Random IP addresses



Worms

- Scanner
 - Once targets are selected the worm scans them for original vulnerability



Worms

- Payload
 - Specific actions in the interest of the attacker
 - Open a backdoor
 - Participate in a DDoS attack
 - Become part of a botnet
 - Do complicated calculations
 - Brute force attacks
 - Calculate pi (happened in Phoenix, brought down 911 system)



Worms

- Worm propagation is limited by
 - system diversity
 - Example: “Tiny Worm” only attacked systems running software from a medium-sized company, but was able to infect almost all of them
 - causing too many failures
 - creating network congestion by fast worms



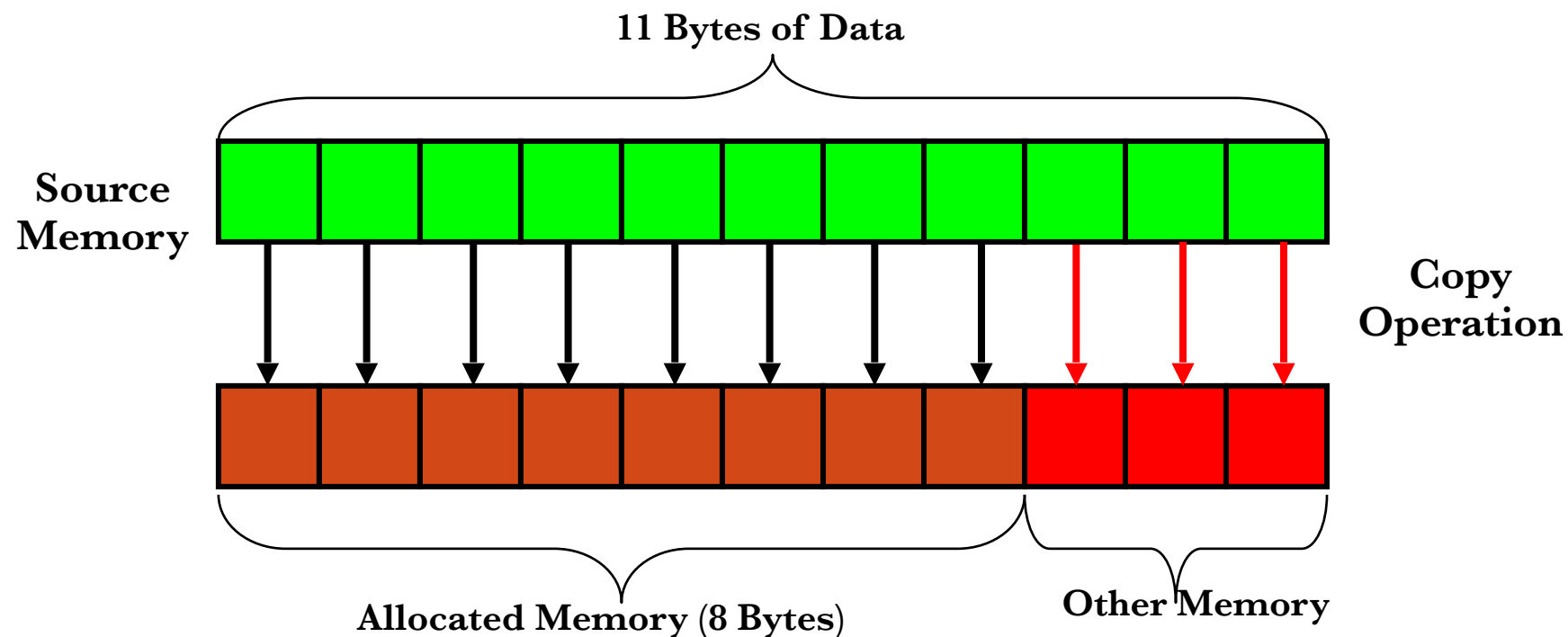
Worms

- Tendencies:
 - More multi-platform worms
 - Exploiting more vulnerabilities in the same worm
 - Zero-day vulnerabilities
 - Fast propagation worms
 - E.g. Warhol/Flash use pre-scanning
 - Polymorphic / metamorphic worms

Attacks

Buffer Overflows

- Buffer overflows overwrite outside of allocated memory



Buffer Overflows



- Buffer overflows happen when buffer boundaries are not checked
 - Standard C library functions do not check buffer boundaries
 - Programmers are not trained to check programmatically either

Buffer Overflows

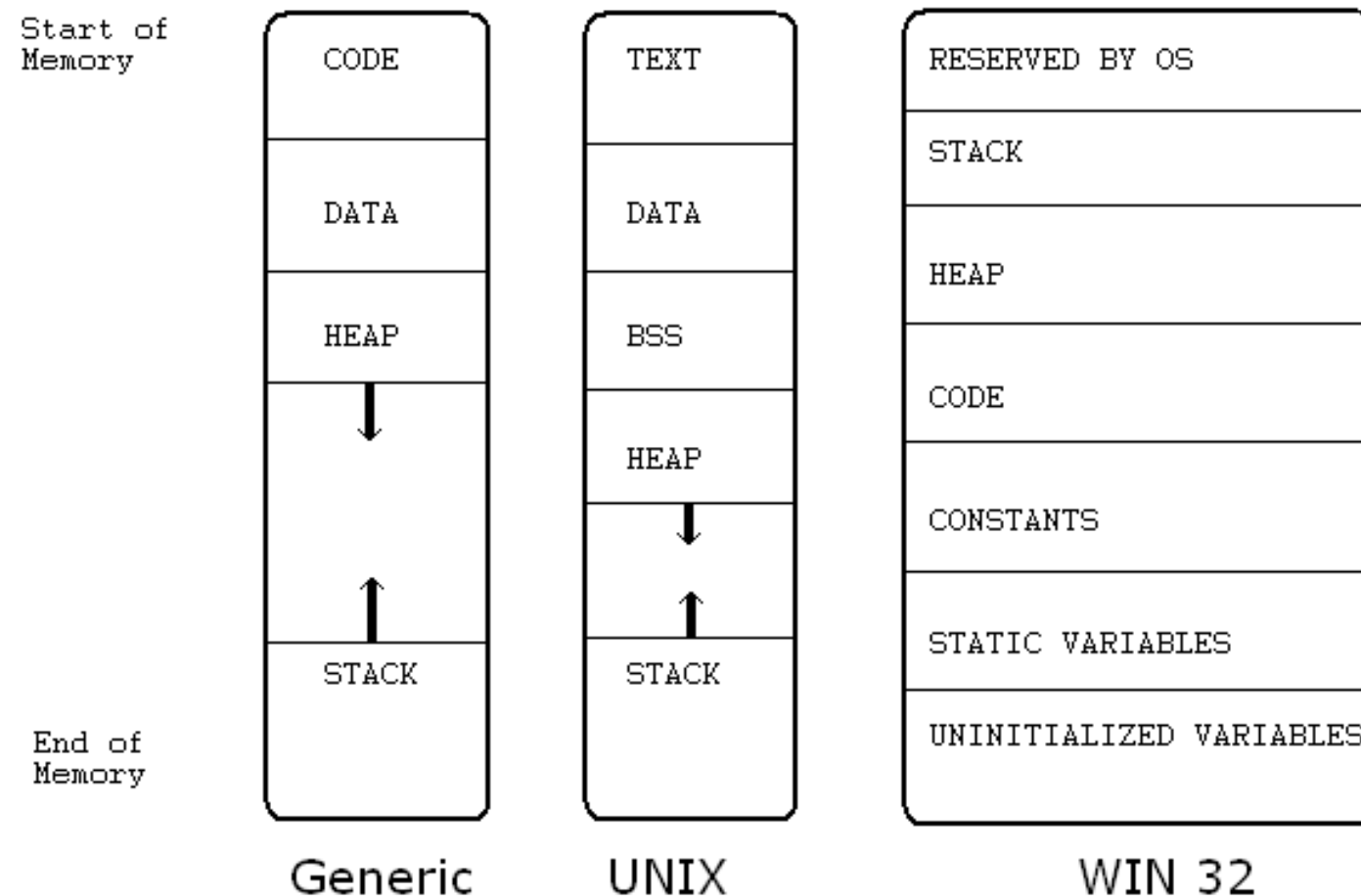


- Might constitute an error (bug) — Programmatic error
- Might constitute a vulnerability — the bug could be exploited
- Allow an exploit

Buffer Overflows



- Memory organization



Code or Text:
Instructions and read only data

Data: Initialized data, uninitialized data, static variables, global variables

Heap: Dynamically allocated variables

Stack: Local variables, return addresses, etc.

Buffer Overflows



- Stack management
 - When a function is called
 - Stack stores return address
 - Stack stores arguments and return values
 - Stack stores local variables of the sub-routine
 - Stack stores values of saved registers
 - This is called a frame
 - Frame address is in the frame register
 - Frame register is not the Base point register (ebp)

Buffer Overflow



```
#include <iostream>
bool IsPasswordOkay()
{
    char Password[100];

    gets(Password);
    if (!strcmp(Password, "badprog"))
        return true;
    else return false;
}

void main()
{
    bool PwStatus;

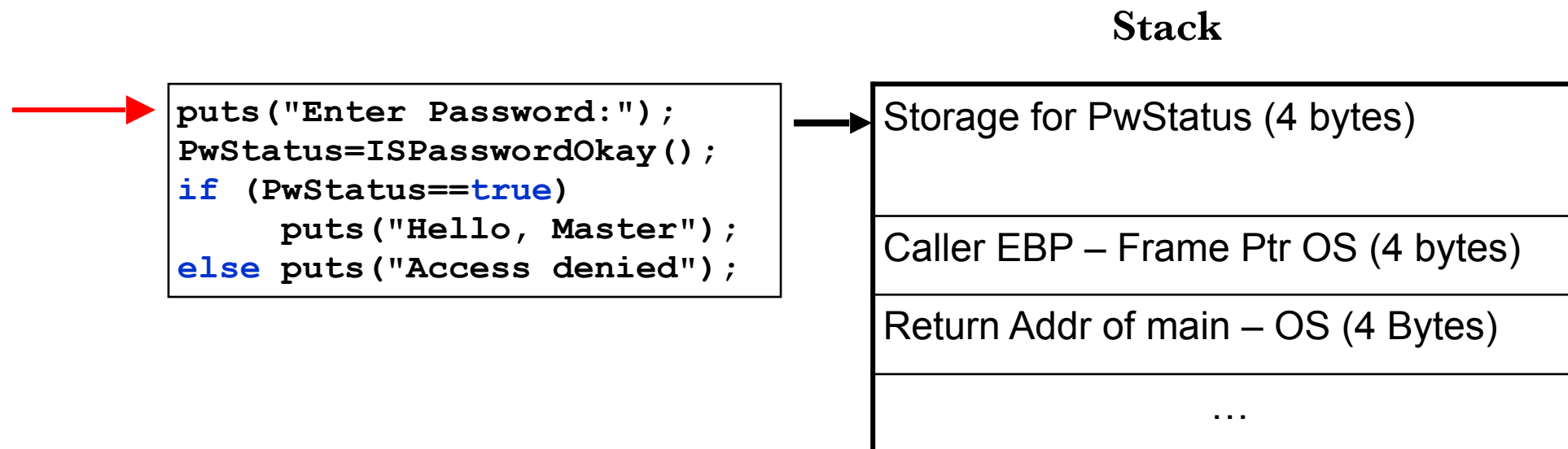
    puts("Enter password:");
    PwStatus = IsPasswordOkay();
    if (PwStatus == false){
        puts("Access denied");
        exit(-1);
    }
    else puts("Access granted");
}
```

```
C:\WINDOWS\System32\cmd.exe
C:\BufferOverflow\Release>BufferOverflow.exe
Enter Password:
badprog
Hello, Master
C:\BufferOverflow\Release>
```

```
C:\WINDOWS\System32\cmd.exe
C:\BufferOverflow\Release>BufferOverflow.exe
Enter Password:
badpas
Access denied
C:\BufferOverflow\Release>
```

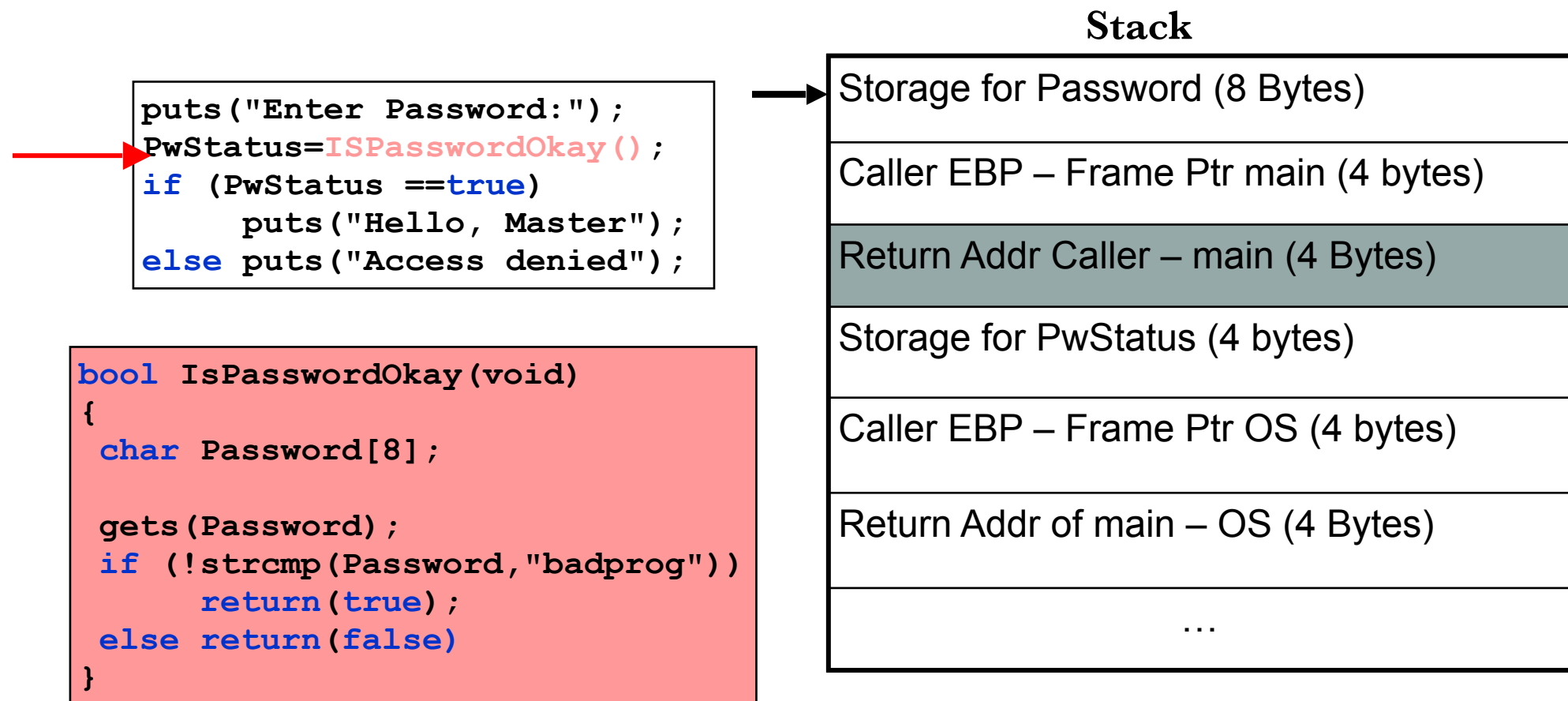
Buffer Overflow

Program stack before call to `IsPasswordOkay()`



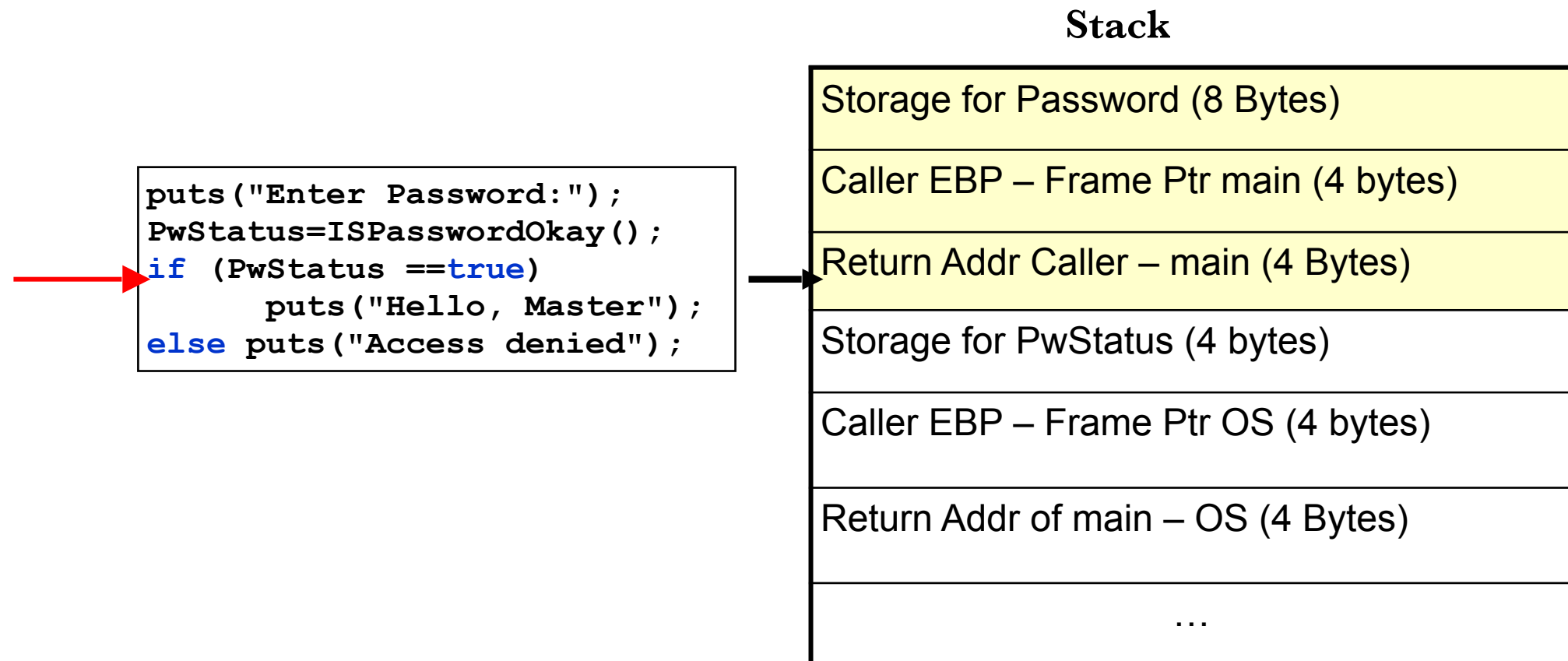
Buffer Overflow

Program stack during call to `IsPasswordOkay ()`



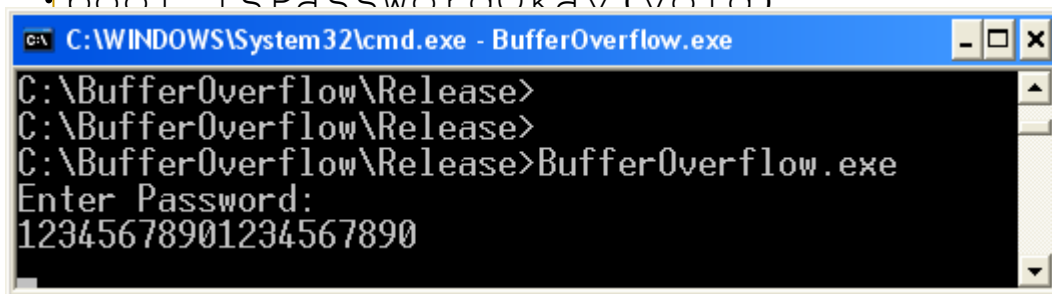
Buffer Overflow

Program stack after call to `IsPasswordOkay()`



Buffer Overflow

- ```
#include <iostream>
bool IsPasswordOkay(void)
```



```
C:\WINDOWS\System32\cmd.exe - BufferOverflow.exe
C:\BufferOverflow\Release>
C:\BufferOverflow\Release>
C:\BufferOverflow\Release>BufferOverflow.exe
Enter Password:
12345678901234567890
```

```
 return(true);
 }
 else return(false);
}
void main()
{
 bool PwStatus;

 puts("Enter Password:");
 PwStatus = IsPasswordOkay();
 if (PwStatus == false){
 puts("Access denied");
 exit(-1);
 }
 else puts("Access granted");
}
```

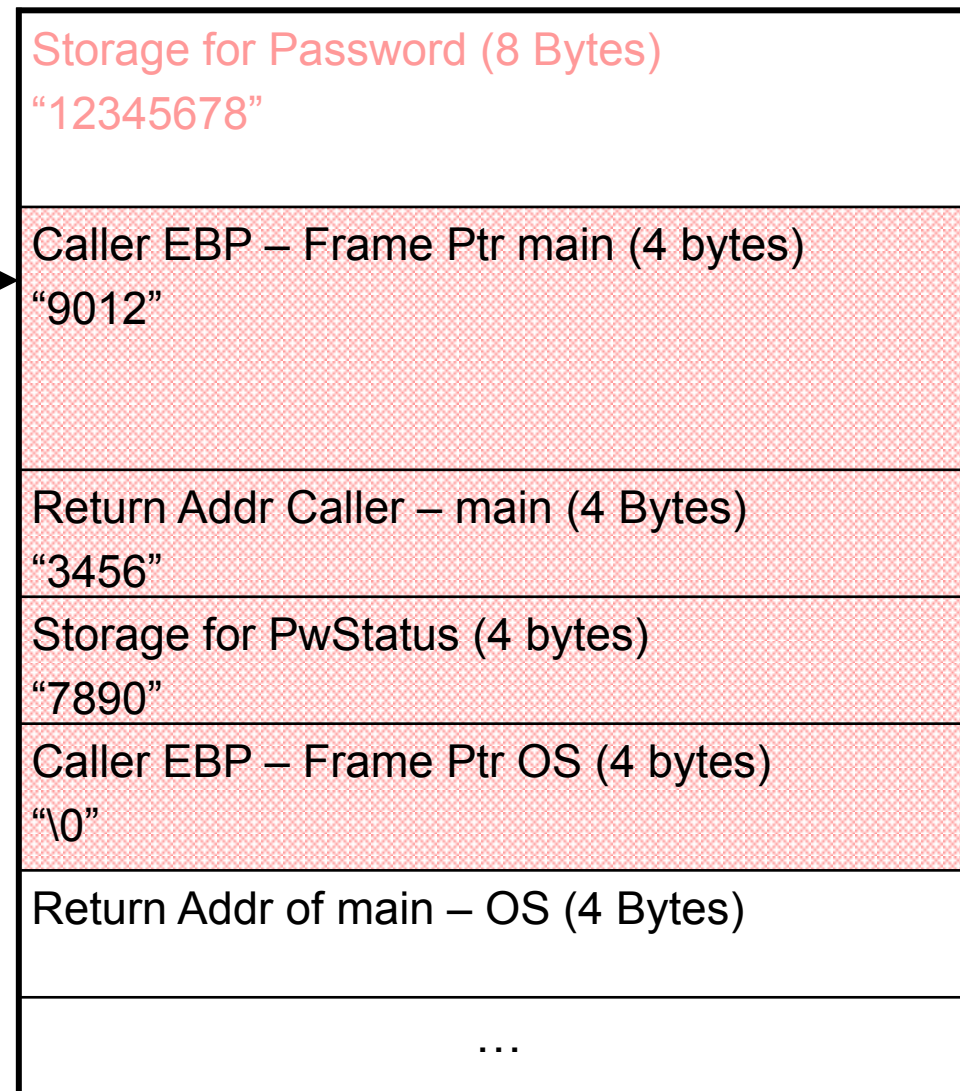


# Buffer Overflow

```
bool IsPasswordOkay(void)
{
 char Password[8];

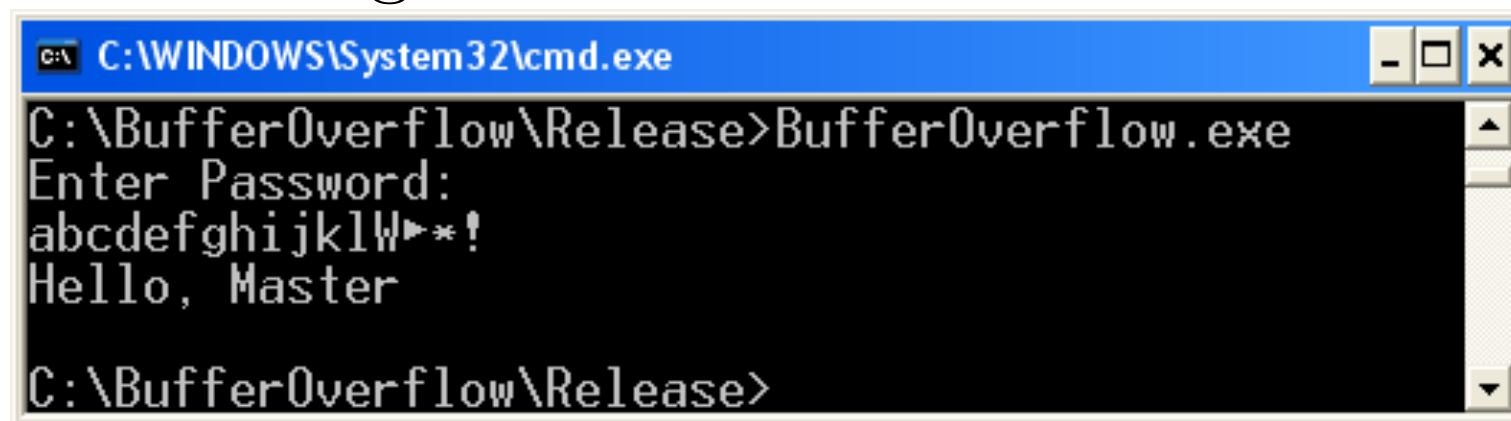
 gets(Password);
 if (!strcmp(Password, "badprog"))
 return(true);
 else return(false)
}
```

The return address and other data on the stack is over written because the memory space allocated for the password can only hold a maximum 7 character plus the NULL terminator.



# Buffer Overflow

- A specially crafted string “abcdefghijklW▶\*!” produced the following result:



```
C:\WINDOWS\System32\cmd.exe
C:\BufferOverflow\Release>BufferOverflow.exe
Enter Password:
abcdefghijklW▶*!
Hello, Master
C:\BufferOverflow\Release>
```

# Buffer Overflow

The string "abcdefghijkW▶\*!"  
overwrote 9 extra bytes of memory  
on the stack changing the callers  
return address thus skipping the  
execution of line 3

| Line | Statement                                |
|------|------------------------------------------|
| 1    | <code>puts("Enter Password:");</code>    |
| 2    | <code>PwStatus=ISPasswordOkay();</code>  |
| 3    | <code>if (PwStatus ==true)</code>        |
| 4    | <code>puts("Hello, Master");</code>      |
| 5    | <code>else puts("Access denied");</code> |

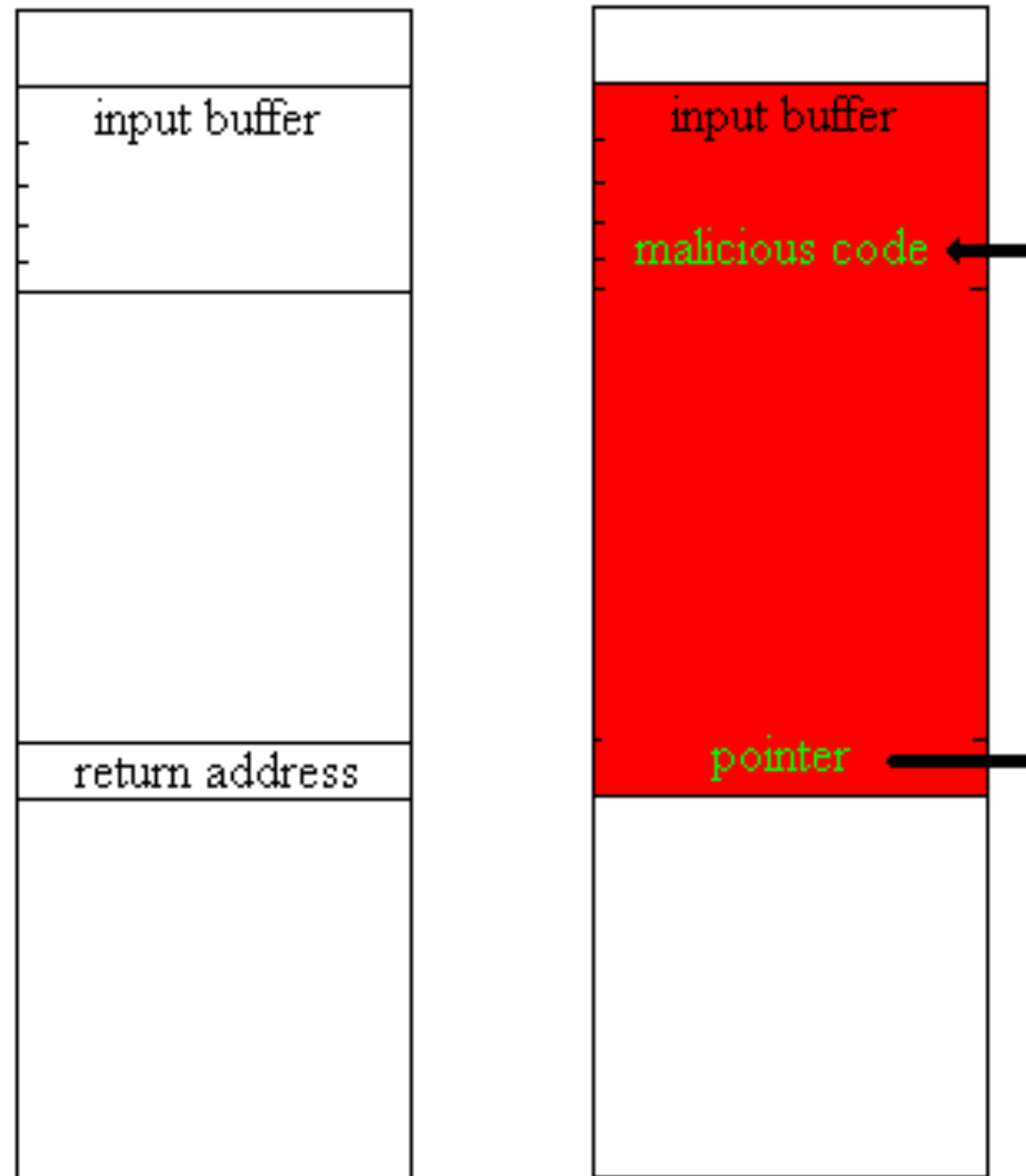
## Stack

|                                                                             |
|-----------------------------------------------------------------------------|
| Storage for Password (8 Bytes)<br>"abcdefgh"                                |
| Caller EBP – Frame Ptr main (4 bytes)<br>"ijkl"                             |
| Return Addr Caller – main (4 Bytes)<br>"W▶*!" (return to line 4 was line 3) |
| Storage for PwStatus (4 bytes)<br>"/0"                                      |
| Caller EBP – Frame Ptr OS (4 bytes)                                         |
| Return Addr of main – OS (4 Bytes)                                          |

# Buffer Overflow

- Buffer overflows can be exploited by
  - Changing the return address in order to change the program flow
    - Called “Arc injection”
  - Change the return address in order to start executing in a memory zone controlled by the attacker
    - Called “Code injection”

# Buffer Overflow



# Buffer Overflow

- The get password program can be exploited to execute arbitrary code by providing the following binary data file as input:

```
000 31 32 33 34 35 36 37 38-39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34-35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0-0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF-BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F-62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

- This exploit is specific to Red Hat Linux 9.0 and GCC



# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

- The first 16 bytes of binary data fill the allocated storage space for the password.
- **NOTE:** Even though the program only allocated 12 bytes for the password, the version of the gcc compiler used allocates stack data in multiples of 16 bytes

# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

The next 12 bytes of binary data fill the extra storage space that was created by the compiler to keep the stack aligned on a 16-byte boundary.

# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

The next 4 bytes overwrite the return address.

The new return address is 0X BF FF F9 E0 (little-endian)

# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

- The malicious code.
  - Purpose of malicious code is to call `execve` with a user provided set of parameters.
  - In this program, instead of spawning a shell, we just call the linux calculator program.

# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

- The malicious code:
  - `xor %eax,%eax #set eax to zero`
  - `mov %eax,0xbffff9ff #set to NULL word`
- Create a zero value and use it to NULL terminate the argument list.
- This is necessary to terminate the argument list.

# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

- The malicious code:
  - `xor %eax,%eax #set eax to zero`
  - `mov %eax,0xbffff9ff #set to NULL word`
  - `mov $0xb,%al #set code for execve`
- Set the value of register al to 0xb. This value indicates a system call to execve.

# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

- The malicious code:
  - **mov \$0xb,%a1 #set code for execve**
  - **mov \$0xbffffa03,%ebx #ptr to arg 1**
  - **mov \$0xbffff9fb,%ecx #ptr to arg 2**
  - **mov 0xbffff9ff,%edx #ptr to arg 3**
- This puts the pointers to the arguments into ebx, ecx, and edx registers.
-

# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

- The malicious code:
  - `mov $0xbffffa03,%ebx #ptr to arg 1`
  - `mov $0xbfff9fb,%ecx #ptr to arg 2`
  - `mov 0xbfff9ff,%edx #ptr to arg 3`
  - `int $80 # make system call to execve`
- Now make the system call to execve. The arguments are in the registers.
-



# Buffer Overflow

```
000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 "1234567890123456"
010 37 38 39 30 31 32 33 34 35 36 37 38 E0 F9 FF BF "789012345678a· +"
020 31 C0 A3 FF F9 FF BF B0 0B BB 03 FA FF BF B9 FB "1+ú · +|+· +|v"
030 F9 FF BF 8B 15 FF F9 FF BF CD 80 FF F9 FF BF 31 "· +i$ · +-Ç · +1"
040 31 31 31 2F 75 73 72 2F 62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

- The malicious code:
  - Last part are the arguments

■

# Buffer Overflow

- `./BufferOverflow < exploit.bin` now executes `/usr/bin/cal\0`.

# Buffer Overflow

- Buffer Overflow
  - on the stack
    - Can be prevented: stack guard, canaries, no-execute permission
  - on the heap
    - Much harder to prevent

# Buffer Overflow

- Standard bugs
  - Pointer mistakes
  - Dynamic Memory Management
  - Program Flow Mistakes
    - Integer overflow
  - Formatted Output Vulnerability
    - Only in C, culprit is a printf kludge
  - Concurrency Issues
    - Intervening event between check and access

# Rootkits



- Are Trojans that change the OS
  - Offer a backdoor for communication
    - Usually a command shell
- Protect against system audits
  - E.g.: A trojan installs a version of `ls` that hides some directories used by the adversary



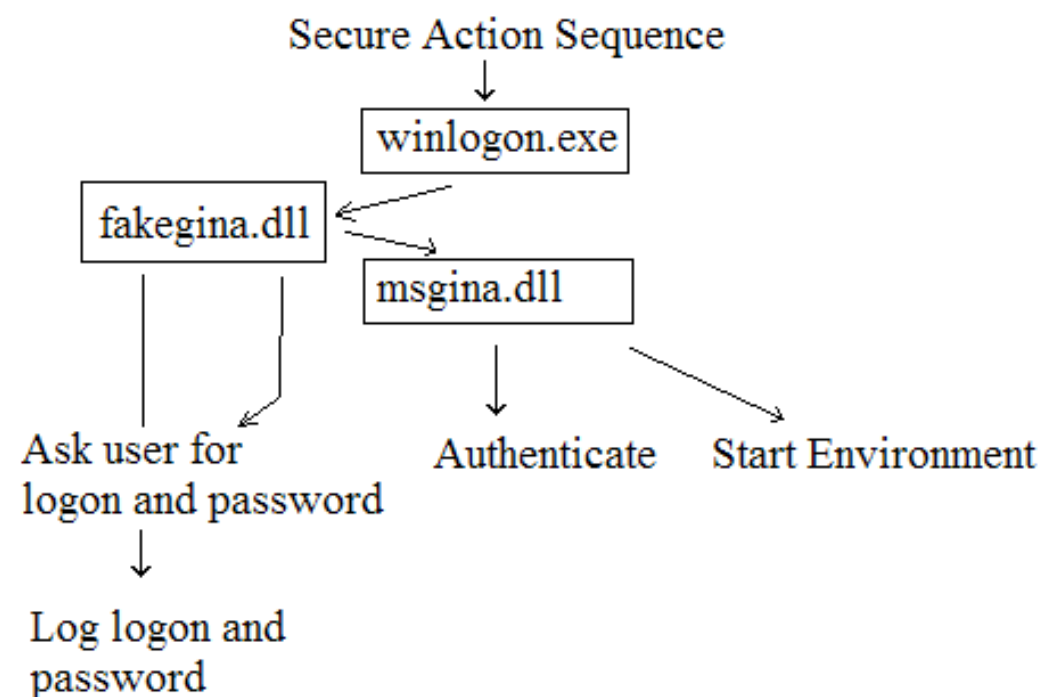
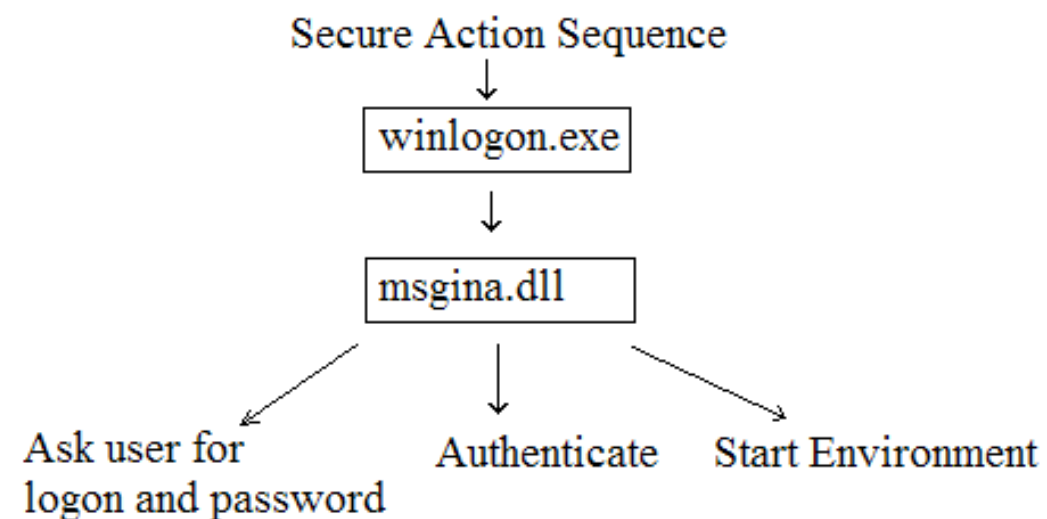
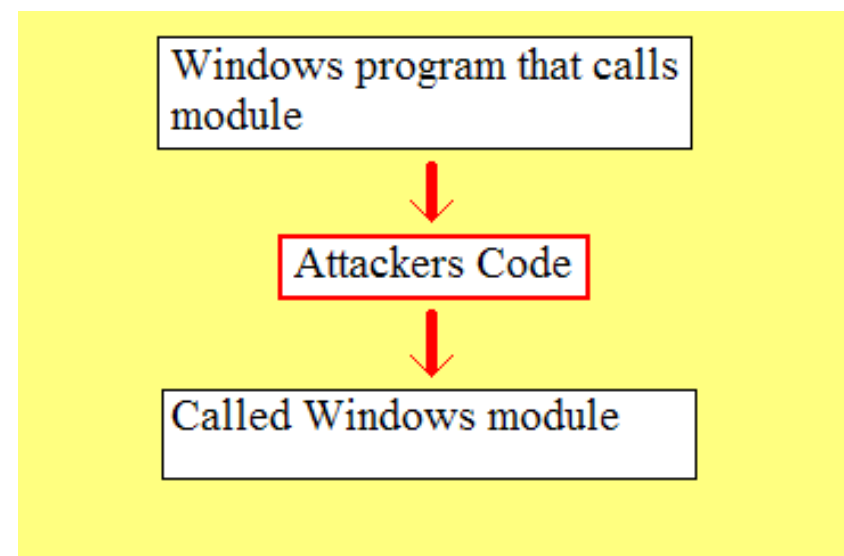
# Rootkits

- Old example: Windows user-level rootkit

- FakeGina
- Uses the winlogon.exe interface

- Graphical Identification and Authentication (Gina)

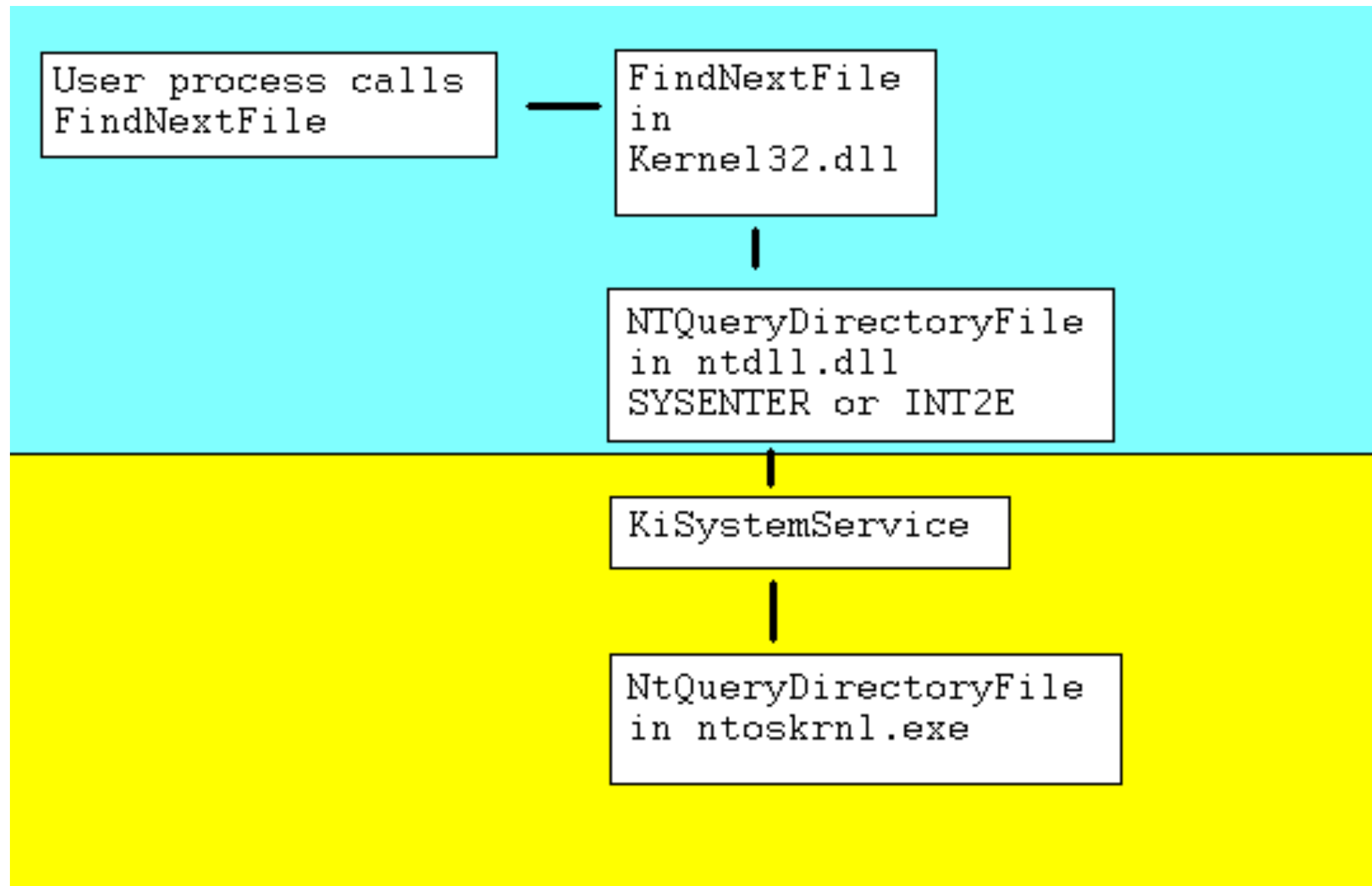
- Windows XP allows other GINA tools
- Adversary changes register contents to install fake gina
  - `HKEY_LOCAL_MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\Winlogon`
- Fakegina harvests passwords and authentication credentials



# Rootkits

- To avoid detection, a rootlet has to change the behavior of OS calls used to audit systems
- Is done by installing *hooks*

# Rootkits







# Rootkits

- API Hooking
  - An application wants to install kernel32.dll in its private address space
  - Puts in space 0x00010000 y 0x7FFE0000
  - Rootkit overwrites all functions in kernel32.dll



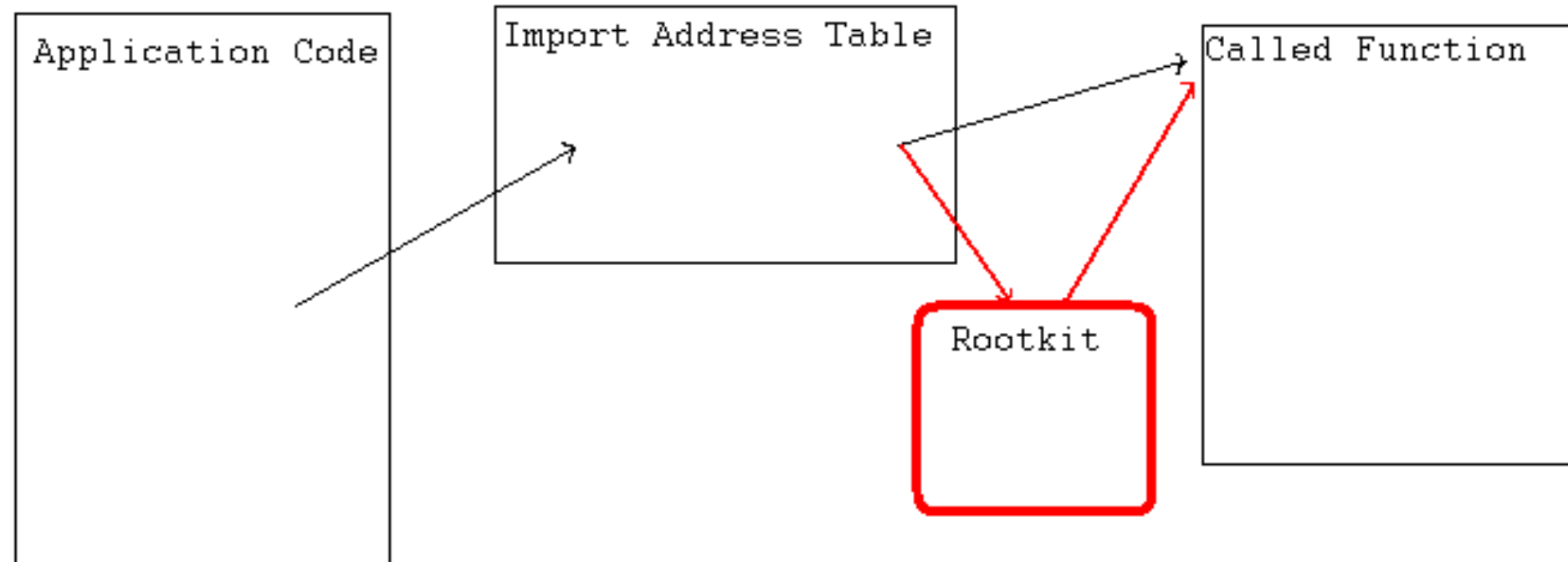
# Rootkits

- API Hooking Method 1:
  - Almost all applications using Win32API use an Import Address Table
  - Every DLL is in the application limit in a structure called `IMAGE_IMPORT_DESCRIPTOR`
  - When an OS loads an application, it uses `IMAGE_IMPORT_DESCRIPTOR` to place all necessary DLL in the memory of the application
  - When the DLL is mapped, the OS places all imported function into memory and places the address in an array.



# Rootkits

- Rootkit must be present in memory
- Overwrite some table entries in the import address tables so that rootkit runs





# Rootkits

- API Hooking Method 2:
  - Substitute the code of a DLL function
  - Substitutes the first five bytes of the function with a jump to the rootkit code
  - Overwritten bytes are stored in a trampoline
  - Rootkit executes the trampoline and later calls the original function
  - Original function returns to the rootkit
  - The rootkit edits the returned values



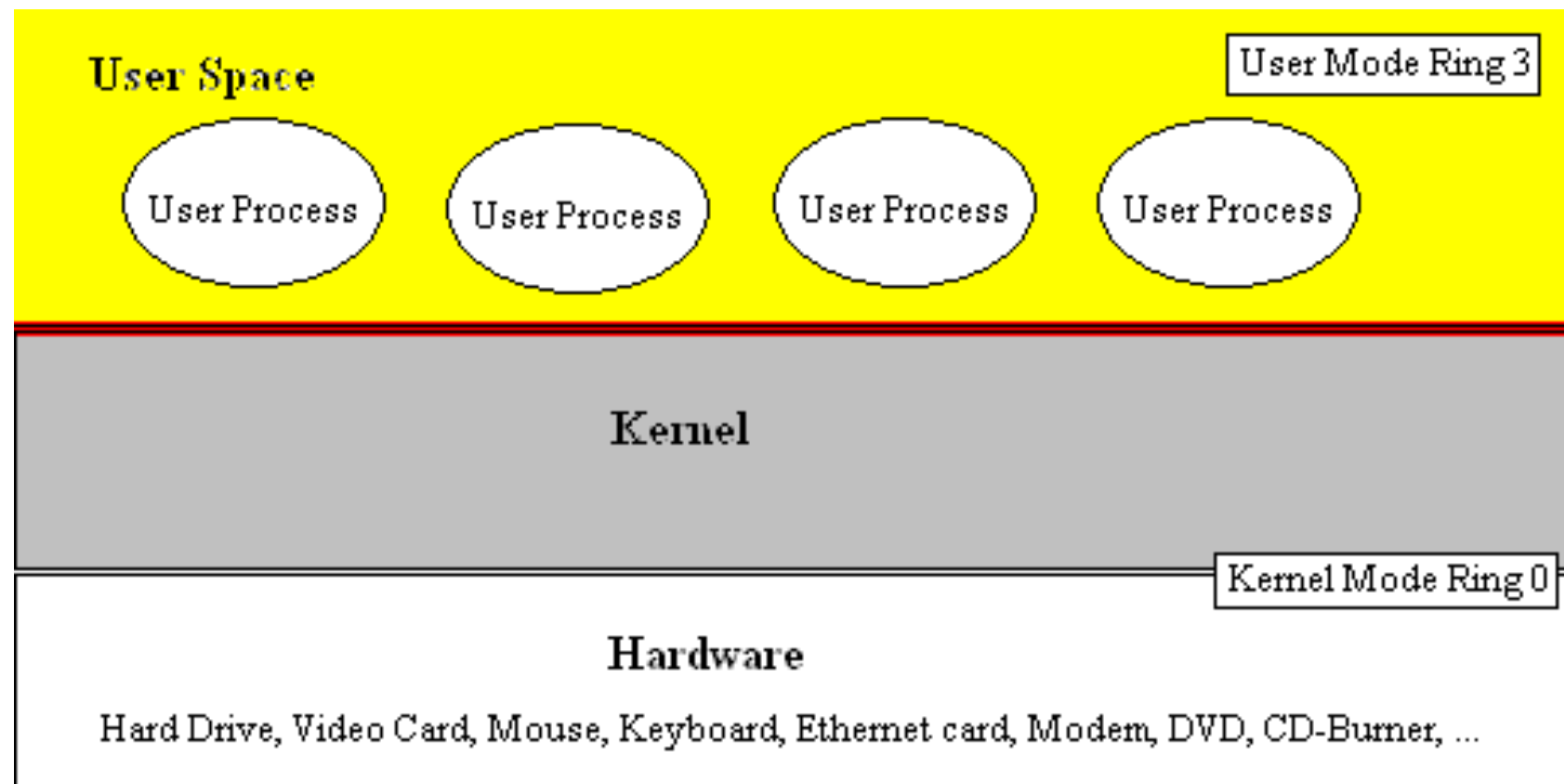
# Rootkits

- DLL injection
  - A method where the rootkit changes a DLL associated with an application
-



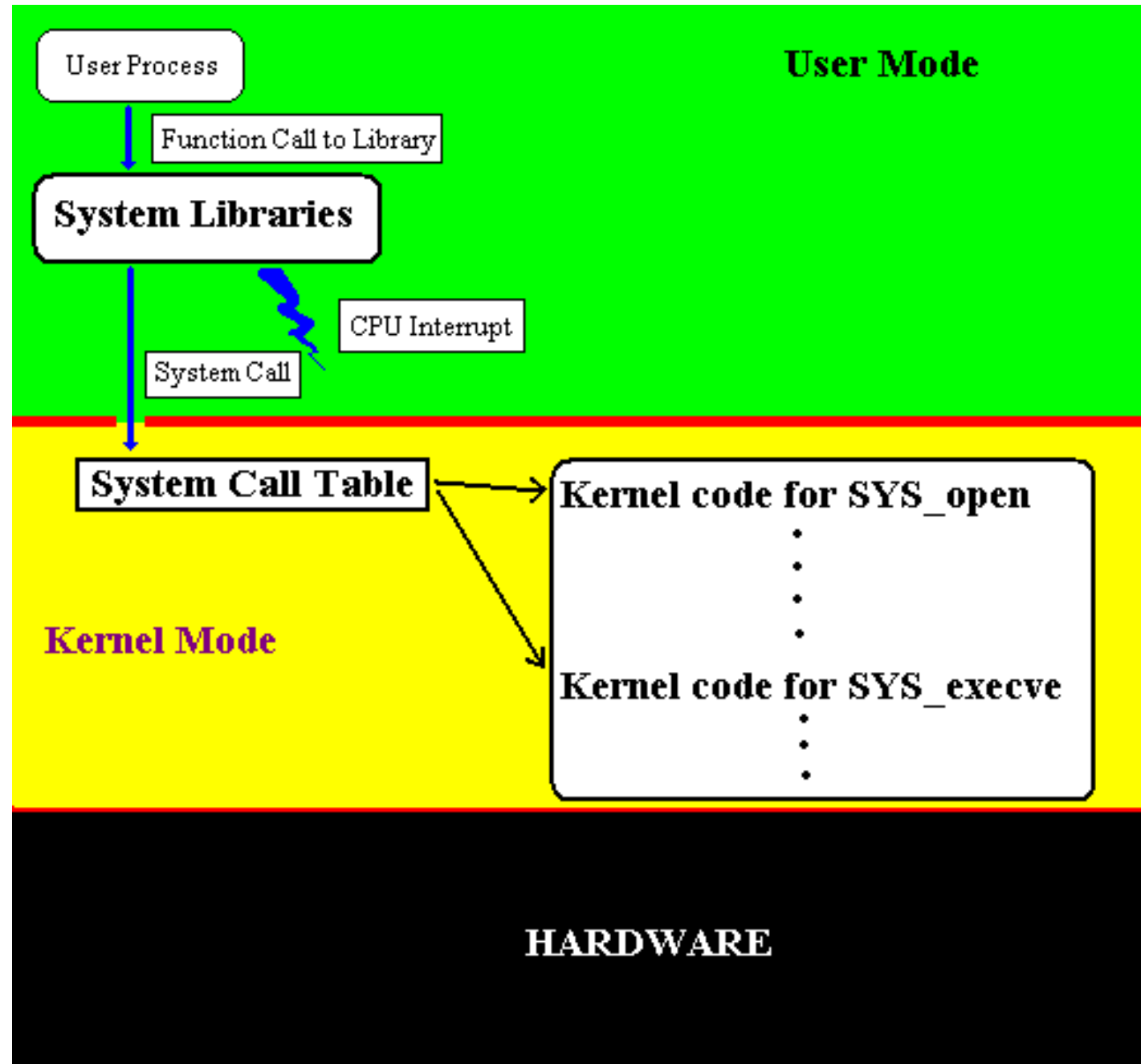
# Rootkits

- Kernel level rootkits
  - Change behavior of the kernel
    - And implicitly that of any application making kernel calls
    - Functionality is hiding of files or processes





# Rootkits



Unix rootkits can attack the System call Table



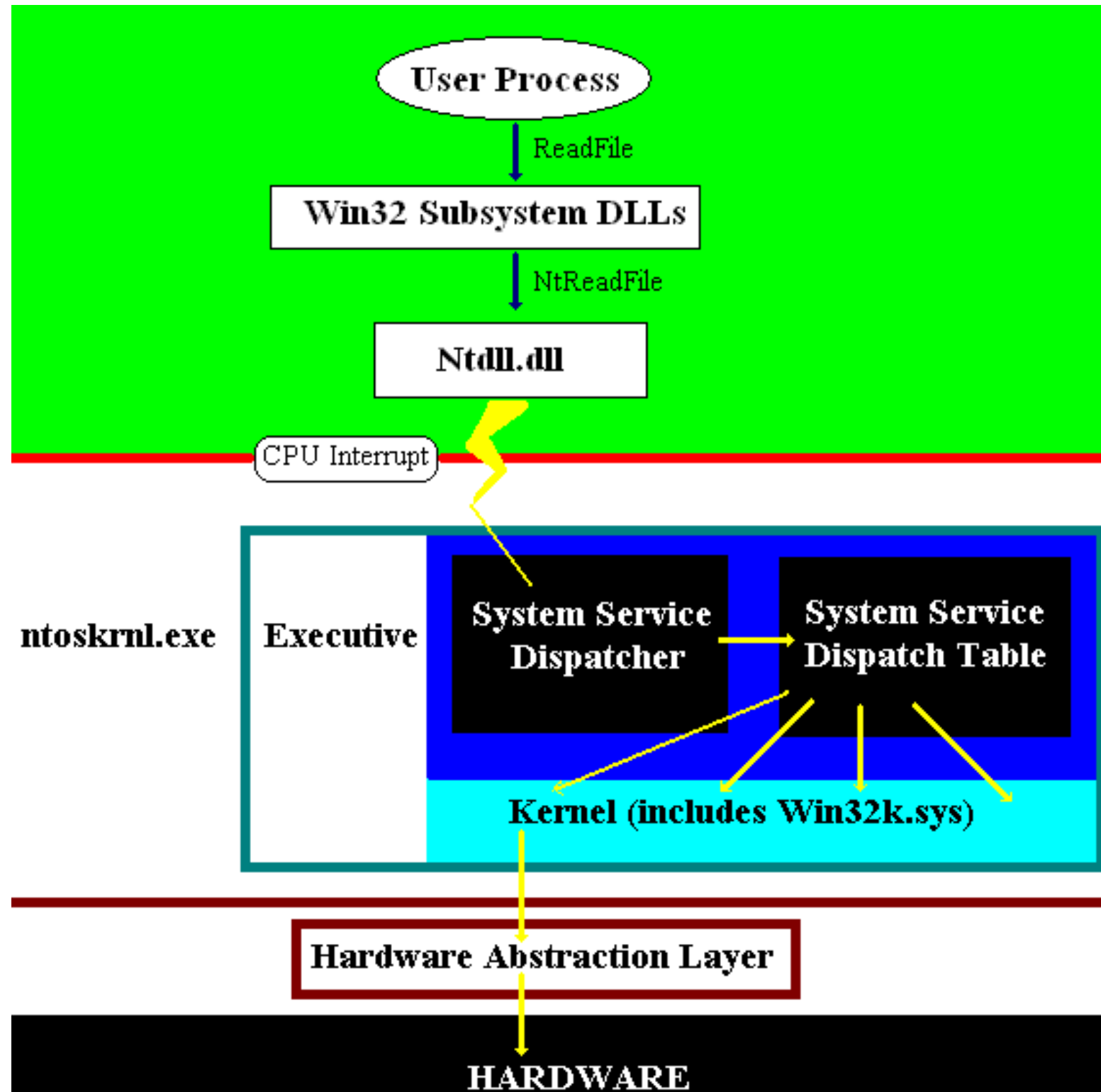


Window rootlets attack the Service Dispatch Table

Same techniques es in UNIX

1. Bad device driver
2. Change kernel in memory
3. Change the kernel image in the file
4. Put kernel in a virtual system
5. Execute user code at kernel level

# Rootkits







# **Social Engineering**

# Definitions

- The art of intentionally manipulating behavior using specialized techniques
- The art to use human behavior patterns to break security without the victim's recognition of manipulation

# Examples

- Using open questions instead of closed questions (that can be answered with “yes” and “no”)
  - “How many do you want to buy?” vs “Do you want to buy one?”
  - “How can I help you?” vs “Can I help you?”

# Examples

- RSA (2011)
  - RSA is a provider of two-factor authentication solutions (SecurID)
  - Target: Proprietary information on SecureID
  - Modus: Electronic email to all low level employees with an Excel spreadsheet titled “Recruitment Plan 2011”
    - Malware installed a backdoor

# Examples

- Daily Mirror reporter Ryan Parry responds to a job add with a faked CV
- Works for two months in Buckingham palace as a footman
- There was no verification of previous employment
- Served breakfast to the queen, took pictures, etc.



# Examples

- Financial Times (2011)
  - Syrian Electronic Army
    - Used spear-phishing to obtain access to Gmail and twitter accounts of a single employee
    - Used this account to send emails with a link to a malware site (clone of the start page for the email accounts at the Financial Times)
    - Financial Time reacted by sending a warning email to all its users with a link to a page for changing passwords
    - SEA cloned this messages, replacing the link with a link to a false password change page.

# Examples

- Microsoft XBox Life (2013)
  - Bought SSN of Microsoft employees working on XBox Life platform
  - Called phone company and used SSN to have calls redirected
  - Called XBox Life to reset passwords which succeeded because the verification call was made to a redirected phone
  - Attackers now had access to XBox Life



# People is the weak point of security

- Social engineering uses for information that does not seem to be security relevant
  - X is attending a conference
  - Which cleaning company is used
  - How to get into the building
  - When are receptionists on duty and what happens outside of duty hours
  - How are passwords reset

# People is the weak point of security

- Difficulties with security policies
  - Are too vague
  - Are too generic
  - Are too weak

# Techniques of Manipulation

- Pretexts
  - Plausible situation
  - Role playing
- Forging an identity
  - Dress
  - Presentation
  - Types of equipment
  - Accent and appearance
  - Speech pattern
  - Body languages

# Techniques of Manipulation

- Baiting
  - Give a victim an opportunity to profit
    - USB-pen drives left in a bathroom
    - Enticing adds on social media

# Techniques of Manipulation

- Scareware
  - Bombard victims with false alarms and fictitious threats
    - Offer a solution
  - Example:
    - Popup: “Your computer is infected”
    - Go to this website to download a disinfection agent

# Techniques of Manipulation

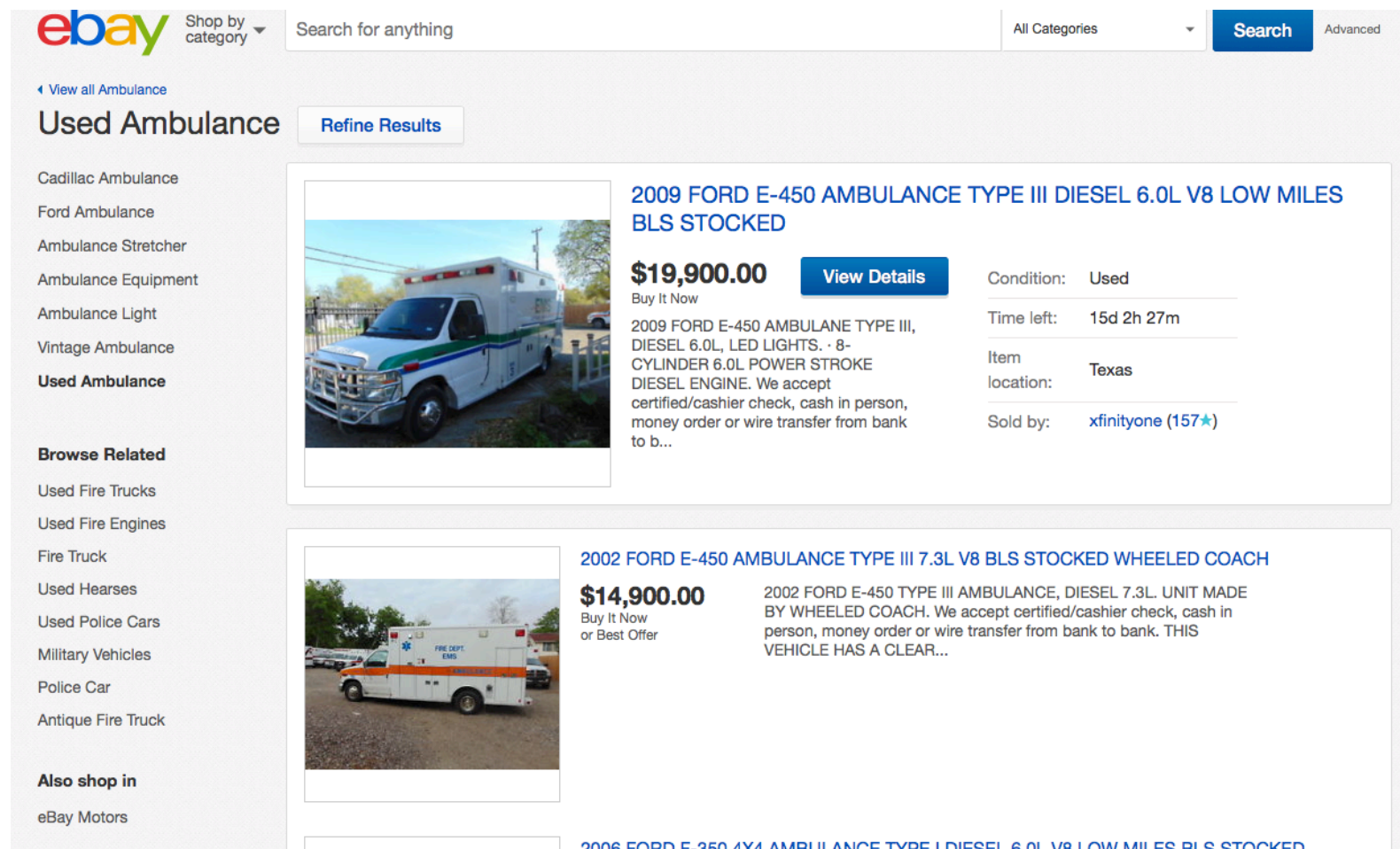
- Pressure and Solution
  - Create strong emotions in the victim that blind the victim
    - Fear
      - An important person needs immediate help
    - Anger - Indignation
      - Pretend that a computer was used to access bad sites

# Techniques of Manipulation

- Pretexting
  - Impersonate coworkers, police, bank, tax-officials
  - Often use bits of information to appear legitimate
  - Ask questions that establish the victims identity
  - Gather personal information such as SSN, phone numbers, ...

# Techniques of Manipulation

- Pretend to be an authority
  - Operation Truck
    - Terrorists bought emergency vehicles on e-bay



The screenshot shows an eBay search results page for "Used Ambulance". The page features a search bar at the top with the text "Search for anything" and a "Search" button. Below the search bar, there are navigation links for "View all Ambulance" and "Used Ambulance". A sidebar on the left lists various vehicle categories, including "Cadillac Ambulance", "Ford Ambulance", "Ambulance Stretcher", "Ambulance Equipment", "Ambulance Light", "Vintage Ambulance", "Used Ambulance", "Browse Related", "Used Fire Trucks", "Used Fire Engines", "Fire Truck", "Used Hearses", "Used Police Cars", "Military Vehicles", "Police Car", "Antique Fire Truck", and "Also shop in eBay Motors".

The main content area displays three search results for ambulances:

- 2009 FORD E-450 AMBULANCE TYPE III DIESEL 6.0L V8 LOW MILES BLS STOCKED**  
Price: **\$19,900.00**  
Buy It Now  
View Details  
Condition: Used  
Time left: 15d 2h 27m  
Item location: Texas  
Sold by: xfinityone (157★)  
Description: 2009 FORD E-450 AMBULANE TYPE III, DIESEL 6.0L, LED LIGHTS. - 8-CYLINDER 6.0L POWER STROKE DIESEL ENGINE. We accept certified/cashier check, cash in person, money order or wire transfer from bank to b...
- 2002 FORD E-450 AMBULANCE TYPE III 7.3L V8 BLS STOCKED WHEELED COACH**  
Price: **\$14,900.00**  
Buy It Now or Best Offer  
Description: 2002 FORD E-450 TYPE III AMBULANCE, DIESEL 7.3L. UNIT MADE BY WHEELED COACH. We accept certified/cashier check, cash in person, money order or wire transfer from bank to bank. THIS VEHICLE HAS A CLEAR...
- 2006 FORD F-350 4X4 AMBULANCE TYPE I DIESEL 6.0L V8 LOW MILES BLS STOCKED**



# Techniques of Manipulation

- Reverse Engineering
  - Pretend to be manipulated by the victim
    - Example:
      - Pretend to be a member of IT support
      - Victim usually has problems to solve
      - Pretend to be talked into helping solve them
  - Introduction (to gain confidence of victim)
  - Sabotage (in order to cause a problem for the victim)
  - Help (to fix the problem)

# Techniques of Manipulation

- Chain of authentication
  - Generate or orchestrate a situation in which the victim assumes that the social engineer is legit
  - Example:
    - Call a receptionist pretending to be a client, create illusion of authenticity, then ask for additional information
    - Receptionist passes on the task to an engineer, telling her that an important client is on the phone

# Techniques of Manipulation

- Chain of authentication
  - Call receptionist pretending to be an engineer working in the office
  - “A colleague from XYZ will come by in 20 minutes. Please send him to my office.”
  - Receptionist assumes that the engineer from XYZ has been authenticated by the local engineer.

# Techniques of Manipulation

- Gaining credibility with bits of information
  - Used almost universally
    - Use known tidbits of information to appear to be an insider
      - “Hello Andrew, are you still in project XYZ or can you help me for two seconds?”
      - “Hello, can I talk to Steve. It is urgent!” (Knowing that Steve is this week in another country). “Oh, but maybe you can help me.”

# Techniques of Manipulation

- Priming / loading
  - A psychological effect where a victim exposed to certain words and ideas is more likely to act in a certain manner

# Techniques of Manipulation

- Social Influence
  - A sign: “Many past visitors have removed petrified wood from the park, destroying the natural state of the petrified forest” increases the number of thefts of petrified wood.

# Techniques of Manipulation

- Social Influence

All,  
We're trying to push our social media presence. Unfortunately, the vast majority of staff haven't liked our corporate page. Please could you follow the link to remedy this.

<http://www.somesocialmediawebsite.com/>

IT Support

# Techniques of Manipulation

- Change the presentation of information
  - There is a 35% chance of winning
  - There is only a 65% chance of not winning



# Techniques of Manipulation

- Generate emotional state such as fear or regret
  - “Oanh, there was a break-in into the MSCS office. There is broken glass and Jackie’s computer is missing. Security is on its way. What is the code to turn off the alarm?”

# Techniques of Manipulation

- Emotional state
  - Kindness
    - A small favor makes people less strict

# Techniques of Manipulation

- Using selective attention
  - Psychological experiment
    - Ask a group of experimental subjects to observe a group dressed in white with a ball and count the number of times in which the ball is passed from one person to another. In the middle of the exercise, appears a person dressed as a gorilla, walks through the group, and salutes with the hand. The majority of the group will not notice.

# Techniques of Manipulation

- Affinity of personality types
  - Can be used to lower suspicion

# Techniques of Manipulation

- Body Language
  - Used to support social engineering
    - Smile
    - Posture and presence
    - Eye contact
    - Feet and arms