

## Firewalk: Determining Firewall Rules

- Ø Find out firewall rules for new connections
- Ø We don't care about target machine, just about packet types that can get through the firewall
  - Ø Find out distance to firewall using traceroute
  - Ø Ping arbitrary destination setting TTL=distance+1
  - Ø If you receive ICMP\_Time\_Exceeded packet went through

1

## Defenses Against Firewalking

- Ø Filter out outgoing ICMP traffic
- Ø Use firewall proxies

2

## Vulnerability Scanning

- Ø The attacker knows OS and applications installed on live hosts
  - Ø He can now find for each combination
    - Ø Vulnerability exploits
    - Ø Common configuration errors
    - Ø Default configuration
- Ø Vulnerability scanning tool uses a database of known vulnerabilities to formulate packets and send them to hosts
- Ø Vulnerability scanning is also used for sysadmin

3

## Vulnerability Scanning Tools

- Ø SARA
  - Ø <http://www-arc.com/sara>
- Ø SAINT
  - Ø <http://www.wvdsi.com/saint>
- Ø VLAD
  - Ø <http://razor.bindview.com/tools>
- Ø Nessus
  - Ø <http://www.nessus.org>



4

## Defenses Against Vulnerability Scanning

- Ø Close your ports and keep systems patched
- Ø Find your vulnerabilities before the attackers do

5

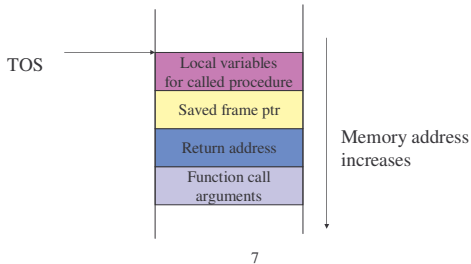
## Phase 3: Gaining Access

- Ø Exploit vulnerabilities
  - Ø Exploits for a specific vulnerability can be downloaded from hacker sites
  - Ø Skilled hackers write new exploits

6

## Stack-based Overflow Attacks

- ∅ Stack stores important data on procedure call



## Stack-based Overflow Attacks

- ∅ Consider a function

```
void sample_function(char* s)
{
  char buffer[10];
  strcpy(buffer, string);
  return;
}
```

- ∅ And a main program

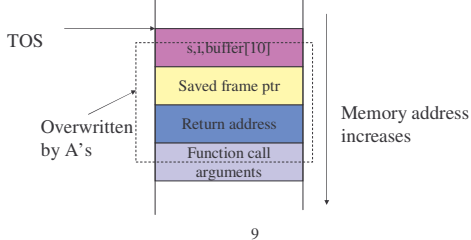
```
void main()
{
  int i;
  char buffer[200];
  for(i=0; i<200;i++) buffer[i]='A';
  sample_function(buffer);
  return;
}
```

Argument is larger than we expected

8

## Stack-based Overflow Attacks

- ∅ Large input will be stored on the stack, overwriting information



## Stack-based Overflow Attacks

- ∅ Attacker overwrites return address to point somewhere else
- ∅ 'Local variables' portion of the stack
- ∅ Places attack code in machine language at that portion
- ∅ Since it is difficult to know exact address of the portion, pads attack code with NOPS before and after

10

## Stack-based Overflow Attacks

- ∅ IDS could look for sequence of NOPS to spot buffer overflows
- ∅ Attacker uses polymorphism: he transforms the code so that NOP is changed into some other command that does the same thing, e.g. MV R1, R1
- ∅ Attacker XORs important commands with a key
- ∅ Attacker places XOR command and the key just before the encrypted attack code, for decryption
- ∅ XOR command is also obscured

11

## Stack-based Overflow Attacks

- ∅ What type of commands does the attacker execute?
- ∅ Commands that help him gain access to the machine
- ∅ Writes a string into `inetd.conf` file to start shell application listening on a port, then uses Netcat to make raw interactive connection to the port
- ∅ Starts TFTP to transfer Netcat onto the victim, then accesses it
- ∅ Starts Xterm

12

## Stack-based Overflow Attacks

- Ø How does an attacker discover stack-based overflow?
  - Ø Looks at the source code
  - Ø Runs application on his machine, tries to supply long inputs and looks at system registers
- Ø Read more at
  - Ø <http://packetstormsecurity.nl/docs/hack/smashstack.txt>

13

## Defenses Against Stack-based Overflow

- Ø For system administrators:
  - Ø Apply patches, keep systems up-to-date
  - Ø Disable execution from the stack
  - Ø Monitor writes on the stack
  - Ø Store return address somewhere else
  - Ø Monitor outgoing traffic
- Ø For software designers
  - Ø Apply checks for buffer overflows
  - Ø Use safe functions

14

## Password Attacks

- Ø Attacker attempts to login with some known username, and to guess a password
  - Ø Trying dictionary words
  - Ø Trying combinations of dictionary words
  - Ø Performing brute-force search
- Ø Attacker steals encrypted or hashed password file and tries to decrypt it

15

## Defenses Against Password Attacks

- Ø Make strong passwords
  - Ø Think of a phrase, take first letters, mix big caps and special characters
  - Ø Use password filtering software
  - Ø Use strong encryption/hash techniques

16

## Web Application Attacks

- Ø Account harvesting
  - Ø Gather usernames by observing error messages, then try to guess passwords
  - Ø Defense: use same error messages for everything
- Ø Hijack a session ID
  - Ø Observe session ID and how it changes between sessions
  - Ø Change your session ID to another one
  - Ø Defense: digitally sign or hash session ID, make them long enough and apply timestamps

17

## Web Application Attacks

- Ø SQL Piggybacking
  - Ø Malformed input into Web form may trigger informative message from an SQL server  
Input: 111111111'  
Error in SQL syntax near 111111111' at line 1  
SELECT \* FROM account WHERE (userid='10001' and number='111111111')
  - Ø Attacker then adds SQL commands into input  
Input: 111111111'+or+userid%3d'10002  
SELECT \* FROM account WHERE (userid='10001' and number='111111111' or userid='10002')
  - Ø Defense: filter user input

18

### Gaining Access Using Network Attacks

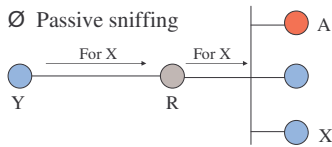
- Ø Sniffing for passwords and usernames
- Ø Spoofing addresses
- Ø Hijacking a session

### Sniffing

- Ø Looking at raw packet information on the wire
- Ø Some media is more prone to sniffing – Ethernet
- Ø Some network topologies are more prone to sniffing – hub vs. switch

### Sniffing on a Hub

- Ø Ethernet is a broadcast media – every machine connected to it can hear all the information

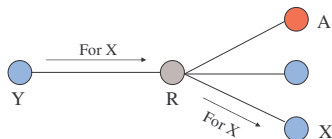


### Sniffing on a Hub

- Ø Attacker can get anything that is not encrypted and is sent to LAN
- Ø Defense: encrypt all sensitive traffic
- Ø Tcpcat
  - Ø <http://www.tcpcat.org>
- Ø Windump
  - Ø <http://netgroup-serv.polito.it/windump>
- Ø Snort
  - Ø <http://www.snort.org>
- Ø Ethereal
  - Ø <http://www.ethereal.com>

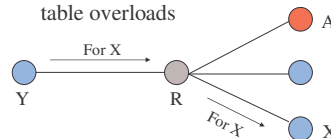
### Sniffing on a Switch

- Ø Switch is connected by a separate physical line to every machine and it chooses only one line to send the message



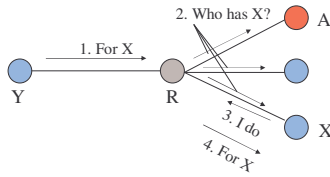
### Sniffing on a Switch – Take 1

- Ø Attacker sends a lot of ARP messages for fake addresses to R
- Ø Some switches send on all interfaces when their table overloads



### Sniffing on a Switch – Take 2

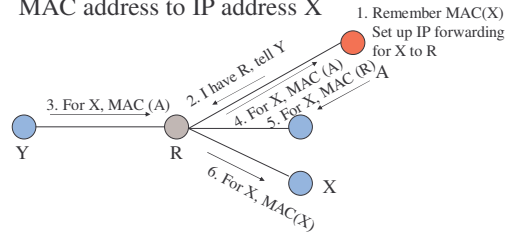
- Ø Address Resolution Protocol (ARP) maps IP addresses with MAC addresses



25

### Sniffing on a Switch – Take 2

- Ø Attacker uses ARP poisoning to map his MAC address to IP address X



26

### Active Sniffing Tools

- Ø Dsniff
  - Ø <http://www.monkey.org/~dugsong/dsniff>
  - Ø Also parses application packets for a lot of applications
  - Ø Sniffs and spoofs DNS



27

### Spoofing DNS

- Ø Attacker sniffs DNS requests, replies with his own address faster than real server
- Ø When real reply arrives client ignores
- Ø This can be coupled with man-in-the-middle attack on HTTPS and SSH

28

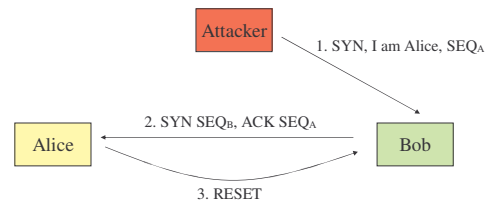
### Sniffing Defenses

- Ø Use end-to-end encryption
- Ø Use switches
  - Ø Statically configure MAC and IP bindings with ports
- Ø Don't accept suspicious certificates

29

### IP Address Spoofing

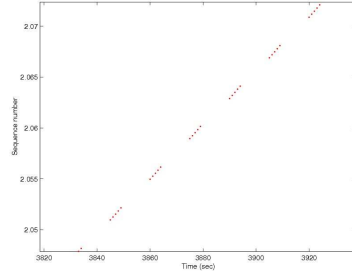
- Ø Attacker cannot see reply packets



30

### Guessing a Sequence Number

- ∅ It used to be  $ISN=f(\text{Time})$ , still is in Windows



31

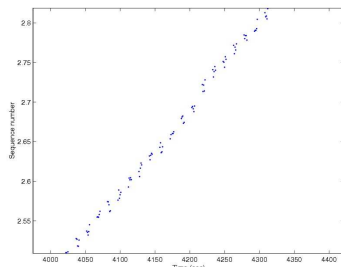
### Guessing a Sequence Number

- ∅ Attacker pretends to be Alice
  - ∅ He establishes many connections to Bob trying to figure out regularity in sequence numbers
  - ∅ He disables Alice (DDoS, ARP spoofing)
  - ∅ He sends SYN to Bob, Bob replies to Alice, attacker uses guessed value of  $SYN_B$  to complete connection
- ∅ If Bob and Alice have trust relationship (*/etc/hosts.equiv* file in Linux) he has just gained access to Bob
- ∅ He can add his machine to */etc/hosts.equiv*

32

### Guessing a Sequence Number

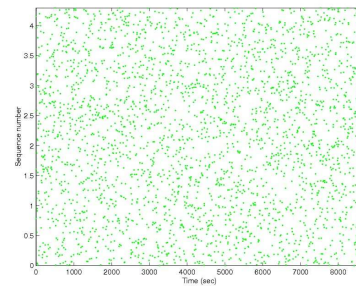
- ∅ On Linux  $ISN=f(\text{time})+\text{rand}$



33

### Guessing a Sequence Number

- ∅ On BSD  $ISN=\text{rand}$



34

### Spoofing with Source Routing

- ∅ Attacker uses *loose source routing* option to specify himself as a hop
- ∅ Spoofs Alice's address, sends packets to Bob
- ∅ Bob sends replies back on the same route

35

### Spoofing Defenses

- ∅ Ingress and egress filtering
- ∅ Prohibit source routing option
- ∅ Don't use trust models with IP addresses
- ∅ Randomize sequence numbers

36

## Netcat Tool

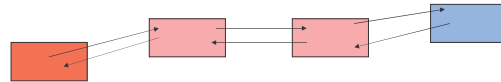
- Ø Similar to Linux *cat* command
  - Ø <http://netcat.sourceforge.net/>
  - Ø Server: Initiates connection to any port on remote machine
  - Ø Client: Listens on any port
  - Ø To transfer file
    - On source machine: `nc -l -p 1234 < file.txt`
    - On remote machine: `nc 123.32.34.54 1234 > file.txt`
    - or
    - On source machine: `nc 44.22.123.212 1234 < file.txt`
    - On remote machine: `nc -l -p 1234 > file.txt`



37

## Netcat Tool

- Ø Used for
  - Ø Port scanning
  - Ø Passive backdoor
  - Ø Relaying the attack



38

## Phase 4: Maintaining Access

- Ø Attacker establishes a listening application on a port (*backdoor*) so he can log on any time with or without a password
- Ø Attackers frequently close security holes they find
- Ø Netcat as a backdoor
  - `nc -l -p 12345 -e /bin/sh`

39

## Trojans

- Ø Application that claims to do one thing (and looks like it) but it also does something malicious
- Ø Users download Trojans from Internet (thinking they are downloading a free game) or get them as greeting cards in E-mail, or as ActiveX controls when they visit a Web site
- Ø Trojans can scramble your machine
  - Ø They can also open a back`door on your system
- Ø They will also report successful infection to the attacker

40

## Back Orifice

- Ø Trojan application that can
  - Ø Log keystrokes
  - Ø Steal passwords
  - Ø Create dialog boxes
  - Ø Mess with files, processes or system (registry)
  - Ø Redirect packets
  - Ø Set up backdoors
  - Ø Take over screen and keyboard
  - Ø <http://www.bo2k.com/>



41

## Trojan Defenses

- Ø Antivirus software
- Ø Don't download suspicious software
- Ø Check MD5 sum on trusted software you download
- Ø Disable automatic execution of attachments

42

### Rootkits

- Ø Alter or replace system components (for instance DLLs)
- Ø For instance on Linux attacker replaces */bin/login* program
- Ø Rootkits frequently come together with sniffers:
  - Ø Capture a few characters of all sessions and write into a file to steal passwords
  - Ø Administrator would notice an interface in promiscuous mode
    - Ø Not if attacker modifies an application that shows interfaces

43

### Rootkits

- Ø Attacker will modify all key system applications that could reveal his presence
  - Ø List processes
  - Ø List files
  - Ø Show open ports
  - Ø Show system utilization
- Ø He will also substitute modification date with the one in the past

44

### Defenses Against Rootkits

- Ø Don't let attackers gain root access
- Ø Use integrity checking of files:
  - Ø Carry a floppy with md5sum, check hashes of system files against hashes advertised on vendor site or hashes you stored before
- Ø Use Tripwire
  - Ø Free integrity checker that saves md5 sums of all important files in a secure database (read only CD), then verifies them periodically
  - Ø <http://www.tripwire.org/>

45

### Kernel Rootkits

- Ø Replace system calls
  - Ø Intercept calls to open one application with calls to open another, of attacker's choosing
  - Ø Now even checksums don't help as attacker did not modify any system applications
  - Ø You won't even see attacker's files in file listing
  - Ø You won't see some processes or open ports
- Ø Usually installed as kernel modules
- Ø Defenses: detect some fingerprints, disable kernel modules and pray

46

### Phase 5: Covering Tracks

- Ø Rootkits
- Ø Alter logs
- Ø Create hard-to-spot files
- Ø Use covert channels

47

### Altering Logs

- Ø For binary logs:
  - Ø Stop logging services
  - Ø Load files into memory, change them
  - Ø Restart logging service
  - Ø Or use special tool
- Ø For text logs simply change file through scripts
- Ø Change login and event logs, command history file, last login data

48

## Defenses Against Altering Logs

- Ø Use separate log servers
  - Ø Machines will send their log messages to these servers
- Ø Encrypt log files
- Ø Make log files append only
- Ø Save logs on write-once media

49

## Creating Hard-to-Spot Files

- Ø Names could look like system file names, but slightly changed
  - Ø Start with .
  - Ø Start with . and add spaces
  - Ø Make files hidden
- Ø Defenses: intrusion detection systems and caution

50

## Covert Channels

- Ø Transfer data across the network in unsuspecting way
  - Ø Wrapping it up in ICMP packets
  - Ø Or in HTTP
    - Ø Server on infected machine goes to master “Web server” periodically
    - Ø If master has typed some commands, server executes them and pushes the result
    - Ø It appears as if machine is engaged in Web surfing
  - Ø Or in SMTP
  - Ø Or in TCP (SYN and ACK fields) and IP headers (ID field)

51

## Defenses Against Covert Channels

- Ø Detect malformed packets for certain protocols
- Ø Use port scan, detect unusual services

52