

# Anatomy of an Attack

---

Vitaly Shmatikov

# Typical Attack Steps

---

- ◆ Get your foot in the door
  - Steal a password file and run dictionary attack
  - Sniff passwords off the network; social engineering
  - Use input vulnerability in a networked application
- ◆ Use partial access to gain root / admin access
- ◆ Set up some way to return
  - Install login program or Web server with back door
- ◆ Cover your tracks
  - Disable intrusion detection, virus protection, Tripwire, system routines that show list of running programs, etc.

# Getting Your Foot in the Door

---

## ◆ Port scan

- Poke open ports on the target network
  - TCP SYNs to see if an ACK comes back
  - UDP (ICMP message will come back if port unreachable)
  - Use FTP bounce and similar techniques to jump over firewall
- See what network services are supported
- Look for old versions of network programs with known vulnerabilities

## ◆ Exploit a software bug or vulnerability to get in

- DNS is a good target (runs with root privileges!)
- FTP, Mail, Web...

# Common Vulnerabilities

---

- ◆ Weak input checking
  - Buffer overflow is an example of this!
- ◆ Inappropriate logging
- ◆ Unintended functionality
- ◆ Inappropriate privilege
- ◆ Race conditions
- ◆ Misconfigured systems
- ◆ Lack of diversity

# Weak Input Checking

---

- ◆ Programs accept input from untrusted sources
  - User input
  - Function calls from other modules
  - Configuration files
  - Network packets
  - Websites are especially vulnerable
    - Web forms; scripting languages with string input
- ◆ Extensible systems are vulnerable, too
  - Module designer assumed calls will come from trusted code, but system has been extended so untrusted code can call trusted module

# Example: PHP passthru

---

◆ passthru("string") PHP executes "string" as a command on local system

- Scripts may construct "string" from user input
- Put ";" in user input to run your favorite command
  - Morris Internet worm did something similar using "|"

◆ Example

- passthru("find . -print | xargs cat | grep \$test");
- User input: ; ls /
- Executed command: find . -print | xargs cat | grep ; ls /

# Example: Cold Fusion CFEXECUTE

(2001)

From Hوجلund and McGraw,  
"Exploiting Software: How to Break Code"

## ◆ Example website code

```
<CFSET #STRING#='/c: "' & #form.text# & "'C:\inetput\wwwroot\*'>  
<CFEXECUTE NAME = 'c:\winnt\system32\findstr.exe'  
    ARGUMENTS=#STRING#  
    OUTPUTFILE="c:\inetpub\wwwroot\output.txt"  
    TIMEOUT="120">  
</CFEXECUTE>
```

## ◆ Displayed web page



Enter a string to search for in files on the disk

◆ User inputs `x" c:\winnt\repair\sam ... "`

◆ Executes findstr.exe ... `c:\winnt\repair\sam ...`  
possibly with admin privileges

# Example: Unicode

---

- ◆ Some web servers check string input
  - Forbid ../ or \ but what about unicode %c0%af for '/'
- ◆ IIS Example (exploited by Nimda worm)

```
http://victim.com/scripts/../../winnt/system32/cmd.exe?<some command>
```

- Executes <some command> as command string
  - scripts directory of IIS has execute permissions
- ◆ Input checking would prevent that, but not this

```
http://victim.com/scripts/..%c0%af..%c0%afwinnt/system32/...
```

- IIS first checks input, then expands unicode

See <http://www.sans.org/rr/whitepapers/threats/458.php>

# Example: Buffer Overflow

---

- ◆ MS-IIS indexing service (used by CodeRed worm)
  - telnet <site> 80
  - GET /somefile.idq?<long buffer>
  - With buffer over 240 bytes, can take over server
- ◆ TFTP server in Cisco IOS
  - Long filename vulnerability
- ◆ MS Xbox
  - James Bond 007 game has a save game option
  - Code to restore game has buffer overflow vulnerability
  - Boot Linux or run code using game as “boot” loader

Many many more examples

# Inappropriate Logging

---

- ◆ PDG's Web transaction processing system
  - Creates world-readable logfile `/cgi_bin/PDG_cart/order.log`
    - Contains mailing addresses, credit card numbers, ...
  - Could use Google to find sites that have this file
  - Patch: encrypt logfile, change protection domain
    - Many sites still vulnerable (admins don't install patches... why?)
- ◆ Cisco Resource Manager (CRM)
  - Admin tool, logs everything (incl. username/pwd)
  - World-readable file; anyone on system can read it
- ◆ Legato Networker, 2002
  - Logs usernames/passwords, logfile not protected

# Unintended Functionality

---

- ◆ Unintended consequences of useful features
- ◆ Example: Postscript and PDF readers
  - %pipe in Postscript allows Ghostview to read, delete files (attacker accesses machine through a bad PS file)
    - "ghostview -d SAFER" provides partial protection
  - Similar attack on some Unix, Linux PDF readers
    - Victim clicks on a hyperlink in malicious PDF file
    - Shell used to start external program to handle hyperlink
    - Attacker executes arbitrary command with privileges of victim
  - Macro languages (e.g., Word macros)
- ◆ Think about security implications of features!

# Unnecessary Privileges

---

## ◆ Principle of least privilege

- Application should only have minimal privilege needed to do its job

## ◆ Problems with setuid programs running as root

- Unix allows many programs to run as root - a bad idea!
  - In 1999, 50% of sendmail servers were vulnerable
  - Most DNS servers run bind, 60% of them vulnerable

# Race Conditions

---

## ◆ Example: Ghostscript temporary files

- Temporary file names in Unix often generated by `mktemp()`

```
name=mktemp("/tmp/gs_XXXXXXXX"); fp=fopen(name,"w")
```

- File names derived from process ID are predictable!

## ◆ Attack: at the right time, "re-route" filename

- Create symlink `/tmp/gs_12345A -> /etc/passwd`
  - This causes Ghostscript to rewrite `/etc/passwd`

- Solution: `mkstemp()` creates and opens a file atomically

## ◆ Think about concurrent execution of sequential programs!

# Misconfigured Systems

---

- ◆ Access control depends on human-created configuration files
- ◆ Example: `.rhosts` file in Unix
  - `rsh` daemon grants permission based on `.rhosts` file
  - What happens if `.rhosts` is not set up properly?
- ◆ Related attack: X window vulnerability
  - Xscan finds machines with X server port 6000 open
  - Tries to Xopen Display (will succeed if "xhosts +")
  - Dumps user keystrokes to file, can get user password
- ◆ Use Google to find Xscan, read source code

# Lack of Diversity

---

- ◆ Many systems run similar software
- ◆ Many commercial systems are built from public-domain software
- ◆ Same vulnerability may exist in many places
- ◆ ... but commonly attacked systems are not the only ones with bugs!

# Some UNIX Basics

---

◆ A user has username, group name, password

shmat, UID 13630      prof, GID 30      "WouldntchaLikeToKnow"



◆ Root is an administrator / superuser (UID 0)

- Can read and write any file or system resource (network, etc.)
- Can modify the operating system
- Can become any other user
  - Execute commands under any other user's ID
- Can the superuser read passwords?

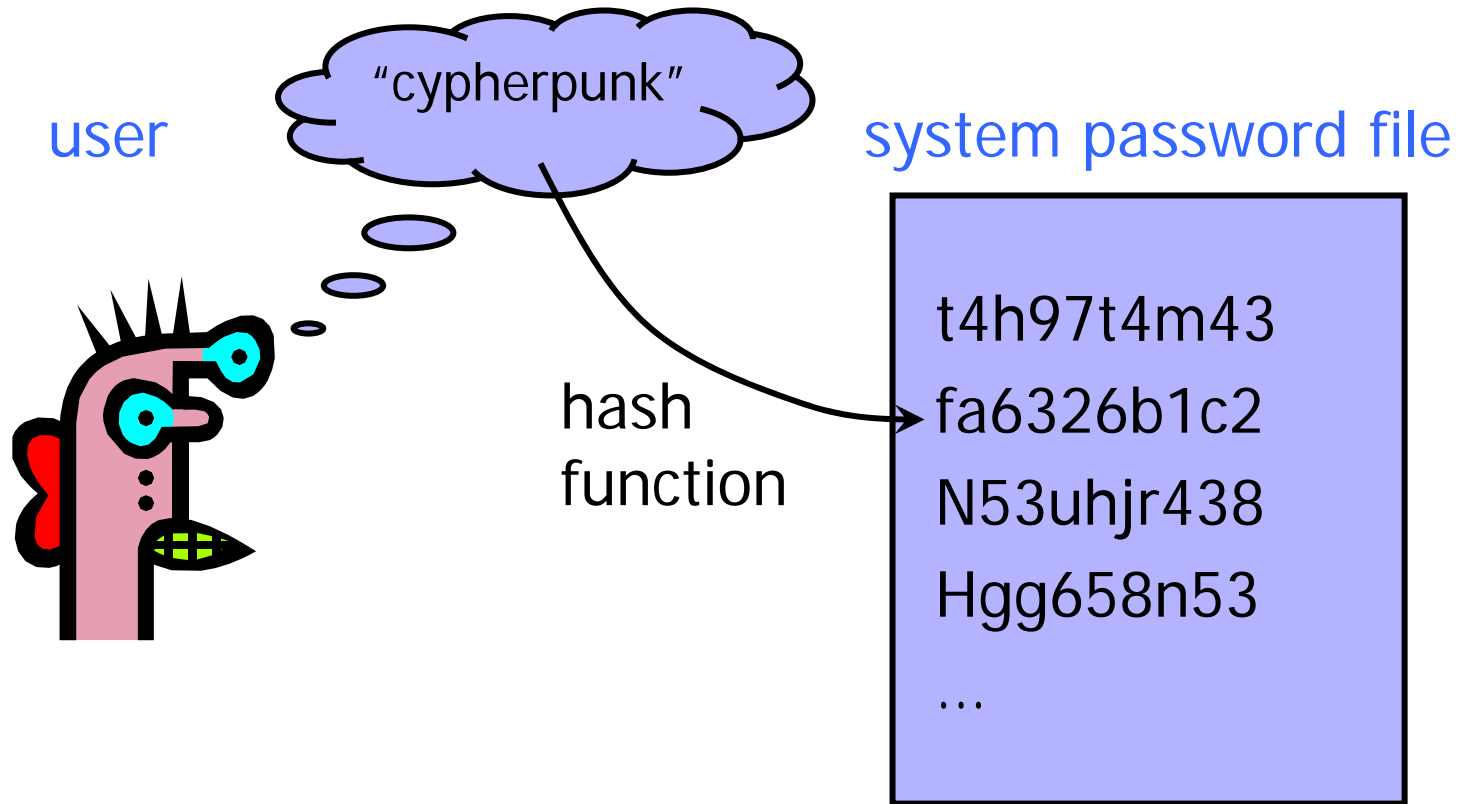
# Password-Based Authentication

---

- ◆ User has a secret password.  
System checks it to authenticate the user.
  - Vulnerable to eavesdropping when password is communicated from user to system
- ◆ How is the password stored?
- ◆ How does the system check the password?
- ◆ How easy is it to guess the password?
  - Easy-to-remember passwords tend to be easy to guess
  - Password file is difficult to keep secret

# UNIX-Style Passwords

---



# Password Hashing

---

- ◆ Instead of user password, store  $H(\text{password})$
- ◆ When user enters password, compute its hash and compare with entry in password file
  - System does not store actual passwords!
- ◆ Hash function  $H$  must have some properties
  - **One-way:** given  $H(\text{password})$ , hard to find password
    - No known algorithm better than trial and error
  - **Collision-resistant:** given  $H(\text{password1})$ , hard to find password2 such that  $H(\text{password1})=H(\text{password2})$ 
    - It should even be hard to find any pair  $p1, p2$  s.t.  $H(p1)=H(p2)$

# UNIX Password System

---

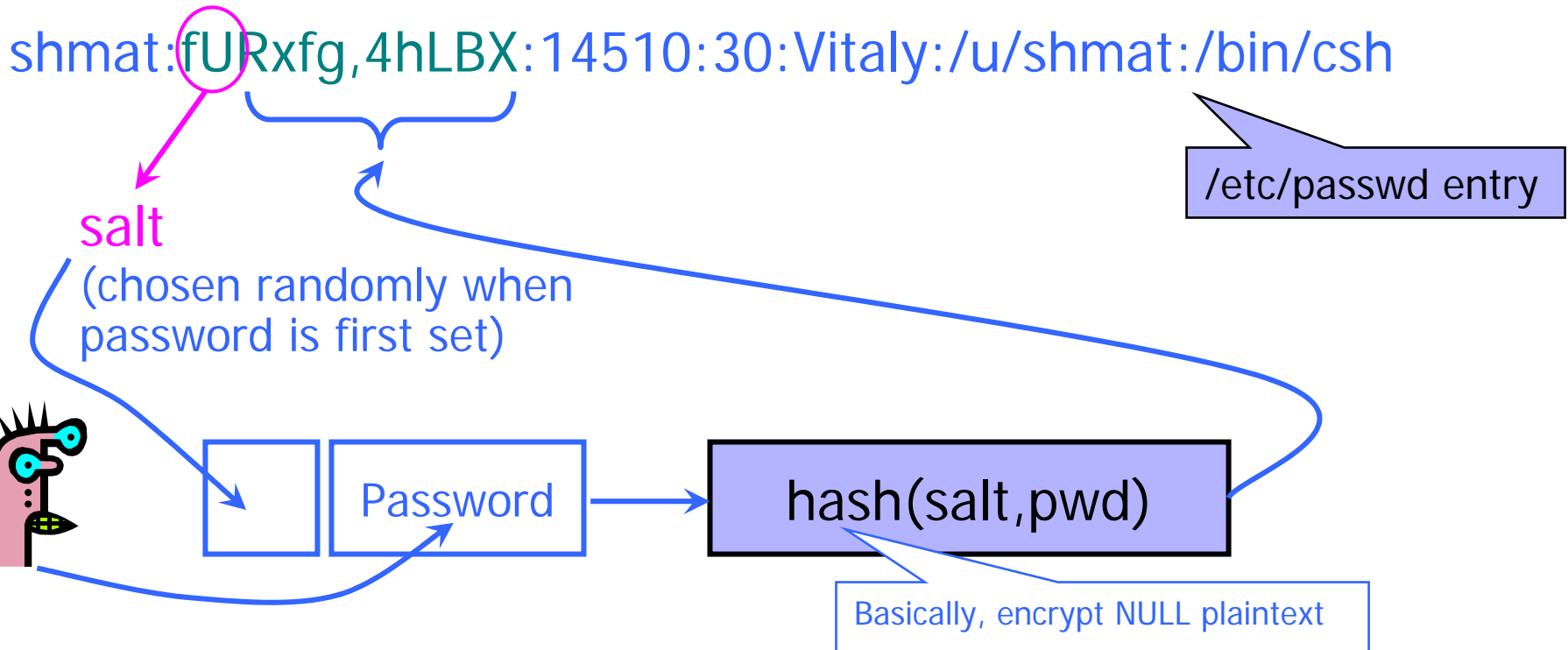
- ◆ Uses DES encryption as if it were a hash function
  - Encrypt NULL string using password as the key
    - Truncates passwords to 8 characters!
  - Artificial slowdown: run DES 25 times
  - Can instruct modern UNIXes to use MD5 hash function
- ◆ Problem: passwords are not truly random
  - With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are  $94^8 \approx 6$  quadrillion possible 8-character passwords
  - Humans like to use dictionary words, human and pet names  $\approx 1$  million common passwords

# Dictionary Attack

---

- ◆ Password file `/etc/passwd` is world-readable
  - Contains user IDs and group IDs which are used by many system programs
- ◆ **Dictionary attack** is possible because many passwords come from a small dictionary
  - Attacker can compute  $H(\text{word})$  for every word in the dictionary and see if the result is in the password file
  - With 1,000,000-word dictionary and assuming 10 guesses per second, brute-force online attack takes 50,000 seconds (14 hours) on average
    - This is very conservative. Offline attack is much faster!

# Salt



- Users with the same password have different entries in the password file
- Dictionary attack is still possible!

# Advantages of Salting

---

- ◆ Without salt, attacker can pre-compute hashes of all dictionary words once for all password entries
  - Same hash function on all UNIX machines
  - Identical passwords hash to identical values; one table of hash values can be used for all password files
- ◆ With salt, attacker must compute hashes of all dictionary words once for each possible salt value
  - With 12-bit random salt, same password can hash to  $2^{12}$  different hash values
  - Attacker must try all dictionary words for each salt value in the password file

# Shadow Passwords

---

shmat:x:14510:30:Vitaly:/u/shmat:/bin/csh



Hashed password is **not**  
stored in a world-readable file

/etc/passwd entry

- Store hashed passwords in `/etc/shadow` file which is only readable by system administrator (root)
- Add expiration dates for passwords
- Early Shadow implementations on Linux called the login program which had a buffer overflow!

# Access Control in UNIX

---

- ◆ Everything is a file
  - Files are laid out in a tree
  - Each file with associated with an **inode** data structure
- ◆ inode records OS management information about the file
  - UID and GID of the file owner
  - Type, size, location on disk
  - Time of last access (atime), last inode modification (ctime), last file contents modification (mtime)
  - **Permission bits**

# UNIX Permission Bits

---

`-rw-r--r-- 1 shmat prof 116 Sep 5 11:05 midterm.tex`

## File type

- regular file
- d directory
- b block file
- c character file
- l symbolic link
- p pipe
- s socket

Access rights of file owner

Access rights of everybody else

Access rights of group members

## Permission bits

- r read
- w write
- x execute (if directory, traverse it)
- s setuid, setgid (if directory, files have gid of dir owner)
- t sticky bit (if directory, append-only)

# Process IDs in UNIX

---

- ◆ Each process has a real UID (ruid), effective UID (euid), saved UID (suid); similar for GIDs
  - **Real:** ID of the user who started the process
  - **Effective:** ID that determines effective access rights of the process
  - **Saved:** used to swap IDs, gaining or losing privileges
- ◆ If an executable's setuid bit is set, it will run with effective privileges of its owner, not the user who started it
  - E.g., when I run lpr, real UID is shmat (13630), effective UID is root (0), saved UID is shmat (13630)

# Setting UIDs Inside Process

---

- ◆ `setuid(uid)` sets `ruid,euid,suid` to `uid`
  - Allowed only if `uid=euid` OR `euid=0`
    - Superuser (`euid=0`) can run as any uid!
- ◆ `seteuid(uid)` sets `euid` to `uid`
  - Allowed only if `uid=ruid` or `uid=suid` OR `euid=0`
- ◆ What if there is a buffer overflow and attacker injects code into a process running with `euid=0`?
- ◆ What if there is an exception and the process quits before resetting `uid` to `suid`?

# Reading Assignment

---

- ◆ Read “Improving the Security of Your Site by Breaking Into It”
  - [Linked from the course website](#)