Hacking web servers

Web server

- System used for storing, processing, and delivering websites
- Hosts web applications, allowing clients to access those applications
- Implements client-server model architecture where client can be e.g. a browser or an API.

Web server concepts

- **Document root**
  - Root directory of servable documents
  - E.g. HTML, JS, image files...
- **Server root**
  - Root directory of all the code that implements the server.
  - Stores configuration, log, and executable files
- **Virtual document tree**
  - Used when the original disk becomes full
  - Located sometimes on different disk, possibly on a different machine
- **Virtual hosting** is multiple sites on a single server
- **Web proxy**
  - Also known as **HTTP proxy**
  - Server placed between the client and server
  - All requests coming from the client go through the proxy to the server
    - instead of directly going to the server
- **Open-source Web Server Architecture**
  - Typically uses
    - **Linux** as an OS
    - **Apache** as a web server
    - **MySQL** as a database
    - **PHP (LAMP)** as principal components.
- **Internet Information Service (IIS)**
  - Web server application developed for Windows Server

Web server hacking methodology

1. **Information gathering** e.g.:
   - Acquiring `robots.txt` to see directories/files that are hidden from web crawlers.
   - Internet searches, WHOIS
   - 📝 Testing HTTP methods
     - Checks for `GET`, `HEAD`, `POST`, `OPTIONS`, `DELETE`, `PUT`, `CONNECT`, `TRACE`

        - Risky methods are `DELETE`, `PUT`, `CONNECT`, `TRACE` and should be disabled
        - `nmap --script http-methods <target>`
2. **Footprinting**
    - E.g.
        - List email addresses: `nmap --script http-google-email`
        - Enumerate common web apps `nmap --script http-enum -p80`
    - Tools: Netcraft, HTTPRecon, ID Serve, HTTPrint, Nmap
    - See also Banner grabbing
3. **Mirror** the target website to browse it offline
    - Tools: • Wget • BlackWidow • HTTrack • WebCopier Pro • Web Ripper • SurfOffline
4. Discover vulnerabilities using e.g.:
    - 📝 **Nikto**
        - Web-server scanner focusing on • misconfigurations • outdated/insecure files
        - Read more in Vulnerability analysis | Nikto
    - **Metasploit**
        - Find, exploit and validate vulnerabilities
5. Perform **session hijacking** and **password cracking** attacks

Web server hacking tools

- Wfetch: Microsoft tool to customize and send HTTP requests
- THC Hydra: login cracker which supports numerous protocols to attack
- HULK DoS: DoSer
- w3af: web application security scanner
- Metasploit: Penetration testing suit

Web server hacking countermeasures

- Patch and update server regularly
- Encrypt the traffic.
    - See also Encrypting communication | Cryptography
- Enforce code access security policy
- Monitor logs
- Use website change detection system
    - Check server files with hash comparison and alert if any modifications has happened.
- Filter traffic to SSH server
- Default passwords and unused default accounts should be changed and disabled respectively.

Place web servers securely

- Place web servers in separate secure server security segment on network
- Recommend to have three layered web application network: Internet, DMZ, internal
    - See also Multi-tier architecture | Hacking web applications
- Place web servers in DMZ zone isolated from public network as well as internal network.

    - See also Network security zoning | Network security
- Each layer should have its own firewalls
    - See also • Zone-based firewall | Firewall • Screened subnet firewalls | Firewall

Securing ports

- Audit the ports regularly
- Disabling insecure and unnecessary ports.
- Use Port 443 HTTPS over port 80 HTTP.

Using certificates

- Ensure validity of certificate data ranges and certificate's public key
- See also Digital certificate | Cryptography

Securing IIS

- Securing your web server | Microsoft docs
    - `Machine.config`
        - Disable tracing (`<trace enable="false"/>`)
        - Turn off debug compiles.
    - 📝 Remove unnecessary ISAPI extensions and filters
        - Allows custom Web Request handling
        - Exploited heavily by attackers in the past.

Web server threats and attacks

DoS/DDoS attacks

- Flooding server with large number of requests to prevent it from functioning properly.
- Services targeted include
  - network bandwidth
  - memory
  - application exception handling mechanism
  - database / hard disk space
  - CPU usage
- DDoS is distributed DoS (denial of service) attack using more than single machine (bots)

DNS server hijacking

- DNS (Domain Name System)
  - Resolves a domain name to its corresponding IP address when queried.
  - See also DNS | DNS enumeration
- Attacker compromises a DNS server and change its DNS server to redirect clients to a malicious website.
  - E.g. by configuring DNS server to redirect requests to a rogue DNS server.
- E.g. user types in legitimate URL in a browser but browser redirects to a fake banking site.
- See also DNS poisoning | Sniffing attacks

DNS server hijacking countermeasures

- Deploying DNSSEC (DNS authentication)
- Choosing a more secure DNS server
- Using a VPN

DNS amplification attack

- A DoS attack is to overwhelm the victims DNS server.
- Includes sending large number of requests to DNS server using target websites IP address.
- Query goes through DNS servers recursively until DNS server fails to find the mapping.

Recursive DNS lookup

- Client asks for a mapping DNS server does not have
- DNS server queries further the root DNS server and then caches the results

DNS amplification attack flow

1. Attacker spoofs and uses victims IP address as source IP
   - All replies from DNS server will be sent to the victims servers
2. Attacker finds internet domain with many DNS records
   - E.g. `subdomain1.cloudarchitecture.io`, `subdomain2.cloudarchitecture.io`, `subdomain3.cloudarchitecture.io`...
3. Attacker sends DNS queries (using bots) to get all records for the domain
   - Results in large replies (usually split over several packets)
   - The replies are sent to the victim.
     - Victims web server becomes too busy with receiving and merging packets and stops functions.
   - **Amplification**
     - For each short DNS query, DNS server replies with larger response
     - Sometimes up to x100 larger, e.g. 3 MBPS DNS queries is amplified as 3000 MBPS DNS replies to victim.

Directory traversal attacks

- Manipulating the target URL to gain access to restricted directories.
- Attackers may reach to restricted directories using `../`.
- Can be caused by e.g. code vulnerabilities (such as no proper validation), or poorly patched/configured web servers.

Man-in-the-middle (MITM) attacks

- Intercepting/altering the traffic between an end-user and web servers.
- Done by tricking client into thinking the attacker is the proxy.
- Allows attacker to steal sensitive information.
- This often uses IP spoofing to trick a victim into connecting to the attack.

Phishing attacks

- Attackers trick an user into visiting a malicious website.
- See also Phishing | Social Engineering Types

Website defacement

- Changing the visual appearance of the target website.
- Attacker can inject malicious code to show e.g. popups/images or change the whole website.
- Usually exposes visitors to some propaganda, or "hacked by ..." message.
- 📝 Can be done through e.g.
  - SQL injection
  - Cross-site scripting (XSS)
  - Malware infection
  - DNS cache poisoning

  - ...

Web server misconfiguration

- Configuration weaknesses that are exploited.
- E.g. directory traversal, server intrusion, and data theft.
- E.g. Capital One misconfiguration that led to compromising more than 100 million peoples personal data in 2019 (read more)
- "Keeping the server configuration secure requires vigilance"— OWASP

HTTP response splitting attack

- Occurs when HTTP headers are generated dynamically using user input.
  - exploits input validation vulnerability.
- Attacker passes malicious data to a vulnerable application
  - and the application includes the data in an HTTP response header.
- Exploited by injecting new lines into response headers e.g.:
  - Author variable is sent by user such as `Ulle Bulle`
  - Application sets `Set-Cookie author=Ulle Bulle` header in back-end.
  - If user sends following instead (with a newline) he can manipulate the website behavior:

    ```
    Ulle Bulle\r\n
    Content-Type: text/html\r\n
    <html>malicious content...</html>
    ```

- Permits variety of attacks such as
  - Cross-Site Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)
  - SQL Injection.

Defend against HTTP response splitting attack

- **Server admin**
  - Use latest web server software.
  - Regularly update/patch OS and Webserver.
  - Run web vulnerability scanners.
- **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.
  - See also Proxy servers | Bypassing IDS and firewall

Web cache poisoning

- Replacing cached content with malicious one.
- Possible by
  - Response-caching on server side
  - HTTP response splitting

Web cache poisoning flow

1. Find the service code that vulnerable to filling the HTTP header field with many headers.
2. Force the cache server to flush its actual cache content
   - E.g. by sending `Pragma: no-cache` or `Cache-Control: no-cache` headers in a request.
3. Send a specially crafted request to store in cache
   - Inject malicious code using HTTP response splitting
   - E.g. GET request where encoded URI does the splitting

     ```
     GET http://cloudarchitecture.io/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>deface!</html> HTTP/1.1
     Host: cloudarchitecture.io
     ...
     ```
4. Send the next request.
   - Previously injected content will be the response as it's cached.

SSH brute force attacks

- Done by cracking SSH login and using SSH tunnels.
- **Flow**
  1. Acquire the SSH login credentials
     1. Port scan to find possible vulnerabilities (e.g. using `nmap`)
     2. Gain login credentials using brute force attack by e.g.
        - Using `metasploit` with `auxiliary/scanner/ssh/ssh_login`
        - Using `hydra` as `hydra -L users.txt -P passwords.txt ssh://172.16.1.102 -t 4`
        - Using custom script in `nmap` e.g. `nmap 172.16.1.102 -p 22 --script ssh-brute --script-args userdb=users.txt,passdb=passwords.txt`
  2. Create SSH tunnels between two hosts to transfer exploitations.
- See also SSH | Tunneling protocols

Web server password cracking attacks

- Attacker cracks the target server passwords and uses them to perform new attacks.

- Server password can belong to e.g. SMTP and FTP servers, Web shares, SSH tunnels...
- Attacking methods include social engineering, spoofing, phishing, using a trojan horse or virus, wiretapping, keystroke logging.
- See also Password attack types | Cracking passwords

Web server password cracking tools

- **Brutus**
  - Cracks online (remote) passwords including FTP, HTTP (form), SMB, Telnet..
- **THC Hydra**
  - Fast login cracker using many protocols
- See also Password cracking tools

Cain and Abel

- Also known as **Cain & Abel** or **Cain**
- 📝 Password recovery tool for Windows
- Website can be found from web archive as it's down since end of 2019.
- Methods include packet sniffing, dictionary, brute force and cryptanalysis attacks
- Has also modules for e.g. Cisco VPN Client Password Decoding
- See also • Cain and Abel | Wireless threats and attacks • Cain and Abel | Sniffing tools

Web application attacks

- Attacker exploits vulnerabilities in the application code.
- See also hacking web applications

Buffer overflow attacks

- Also known as **buffer overrun**
- Anomaly that happens when web server writes data to a buffer in memory that's bigger than buffer size.
  - May cause application crash or other vulnerable behavior.
- Exploited by attacker through sending data in big sizes to the server.
- 📝 Example attack against a local server:

```
#!/usr/bin/python
import socket

big_data = 5000 * 'A'
payload = "TRUN /.:/" + big_data
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("10.0.2.4", "9999")
s.send(payload)
print "[+] " + str(len(payload)) + " bytes sent"
s.close()
```

- Memory manipulation functions in C and C++ without bound checking are most vulnerable.

    - Higher level languages like Python, Java, and Swift do a better job of protecting against a buffer overflow, but they are still vulnerable.
- Dangerous SQL functions (without checking size of destination buffers) include • `gets()` • `strcpy()` • `stract()` • `printf()`

NOP Sled

- Oldest and most common way of exploiting stack buffer overflows
- Sends a large # of NOP instructions into buffer.
- Solves the problem of finding the right place for code execution, as NOPs does nothing and target area is big the execution will slide to no-ops where malicious code will be executed.
- Most IDS protect from this attack.