ALBUKHARY INTERNATIONAL UNIVERSITY

Client-side Penetration Testing

Brief Overview of Client-side Penetration Testing

Client-side penetration testing is a security assessment focused on identifying and exploiting vulnerabilities in the applications and environments that end-users interact with

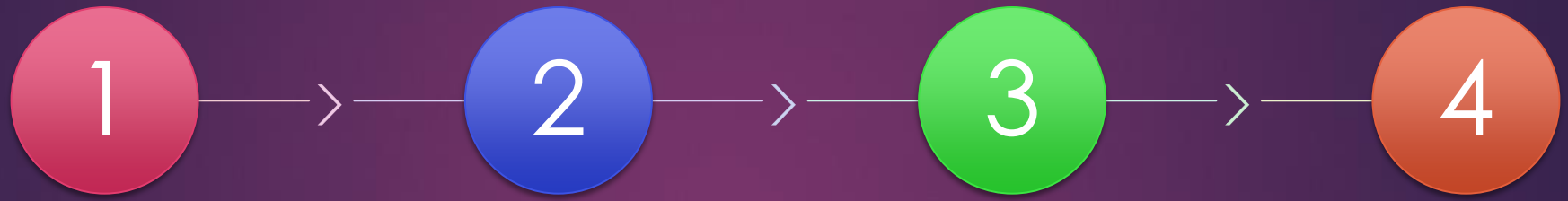
Key Components:

- User Interfaces: Client-side testing examines web browsers, email clients, document readers, and other applications that users interact with daily.
- Endpoints: It includes the security assessment of desktops, laptops, and mobile devices, aiming to discover vulnerabilities that could be exploited by malicious actors.
- Web Applications: Since many client-side applications are web-based, testing involves scrutinizing web browsers, plugins, and extensions for potential security weaknesses.

Importance of Securing Client-side Applications and Environments

- **User-Centric Focus**
- **Attack Surface**
- **Data Protection**
- **Comprehensive Security**
- **Regulatory Compliance**
- **Business Continuity**

Objectives



1. Identify Weaknesses: Discover vulnerabilities in applications accessed by end-users.
2. Understand Attack Vectors: Analyze potential routes attackers might take through client-side exploits.
3. Evaluate User Awareness: Assess the effectiveness of security awareness training among users.
4. Test Defensive Measures: Evaluate the resilience of client-side security controls.

Bypassing Filters with Metasploit Payloads

- Metasploit is an open-source penetration testing framework.
- Allows security professionals to test and exploit vulnerabilities.
- Supports a wide range of exploits, payloads, and auxiliary modules.
- Facilitates automated and manual exploitation of security issues.

Techniques for Bypassing Filters

- Encoded Payloads:
  - Encoding payloads to evade signature-based filters.
  - Example: Base64 encoding to obfuscate malicious code.

- Polymorphic Payloads:

  - Payloads that change their appearance to avoid detection.
  - Example: Using polymorphic engines to generate unique payloads.

- Evading IDS/IPS:
  - Adjusting payload characteristics to bypass intrusion detection/prevention systems.

Examples of Successful Bypasses

- Case 1: Firewall Evasion
  - Scenario: Bypassing a firewall using Metasploit.
  - Outcome: Successful penetration without detection.
- Case 2: Email Filter Bypass
  - Situation: Evading email content filters with Metasploit payloads.
  - Result: Infiltrating systems through email-based attacks.
- Case 3: Network-level Filtering
  - Challenge: Overcoming network-level filtering using Metasploit techniques.
  - Success: Gaining unauthorized access without triggering alarms.

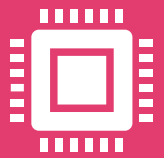
Best Practices for Filter Bypass

REGULARLY UPDATE SIGNATURES: STAY AHEAD OF SIGNATURE-BASED FILTERS.

EMPLOY STEALTHY TECHNIQUES: UTILIZE OBFUSCATION AND POLYMORPHISM.

TEST EFFECTIVENESS: VALIDATE FILTER BYPASS TECHNIQUES IN CONTROLLED ENVIRONMENTS.

Overview of Scanning All Ports



Port scanning is the process of actively probing a system or server for open ports.

To identify all available ports on a client-side system.

Importance of Identifying Open Ports on the Client-Side

Network Visibility: Knowing all open ports provides a comprehensive view of the client-side network.

Attack Surface: Identifying open ports helps understand potential entry points for attackers.

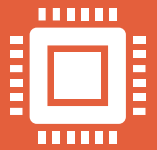
Service Enumeration: Open ports reveal running services, aiding in vulnerability assessment.

Risks Associated with Open Ports

- Unauthorized Access: Open ports may be exploited by attackers to gain unauthorized access.
- Malicious Services: Unnecessary open ports might run services with security vulnerabilities.
- Data Exposure: Open ports increase the risk of data exposure and unauthorized data transfers.

Mitigation Strategies For Open Ports

Port Filtering: Implementing firewalls and port filtering rules to allow only necessary traffic.



Regular Audits: Conduct periodic port scans to identify and close unnecessary open ports.

Patch Management: Keep software and systems updated to address known vulnerabilities.

HTTP Payloads

HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the World Wide Web.

In penetration testing, attackers often exploit vulnerabilities in the HTTP protocol to compromise systems.

HTTP Payloads Overview:

Payloads are scripts or pieces of code that can be injected into an HTTP request or response.

These payloads aim to exploit vulnerabilities in web applications, servers, or client-side components.

Common HTTP Payloads:

Cross-Site Scripting (XSS): Injecting malicious scripts into web pages viewed by other users.

SQL Injection: Exploiting vulnerabilities in database queries through manipulated HTTP requests.

Remote File Inclusion (RFI): Forcing a web application to include and execute external files.

HTTP Payload Delivery:

- Payloads can be delivered via various methods, such as form inputs, cookies, headers, and URL parameters.
- Tools like Burp Suite or OWASP ZAP are often used to manipulate HTTP requests and insert payloads.

Impact of Successful HTTP Payloads:

- Data theft, including login credentials and sensitive information.
- Session hijacking, allowing unauthorized access to user accounts.
- Defacement or manipulation of web content.

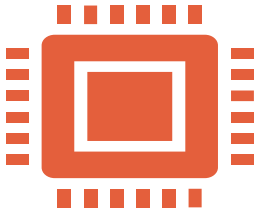
HTTPS Payloads

HTTPS (Hypertext Transfer Protocol Secure) is a secure version of HTTP that encrypts data in transit. While encryption enhances security, vulnerabilities may still exist in the application layer.

- HTTPS Payloads Overview:
  - Payloads targeting HTTPS often focus on exploiting weaknesses in the SSL/TLS protocols or vulnerabilities in the web application.
- Common HTTPS Payloads:
  - SSL Stripping: Downgrading a secure connection to an insecure one to capture sensitive information.
  - Man-in-the-Middle (MitM) Attacks: Intercepting communication between two parties to eavesdrop or manipulate data.

HTTPS Payload Delivery:

Payloads may be injected into encrypted traffic through advanced techniques, such as exploiting weaknesses in SSL/TLS protocols or leveraging vulnerabilities in web applications.



Mitigating HTTPS Payload Attacks:

Use strong encryption algorithms and keep SSL/TLS protocols up to date.

Regularly update and patch web applications to address known vulnerabilities.

Employ secure coding practices to prevent common web application vulnerabilities.

Client-side Attacks

Client-side attacks refer to a category of cyber attacks that target the client or end-user systems rather than the server or network infrastructure.

These attacks exploit vulnerabilities in software or applications running on the user's device, such as web browsers, email clients, or other software, to compromise the security of the system.

Common Types Of Client-side Attacks

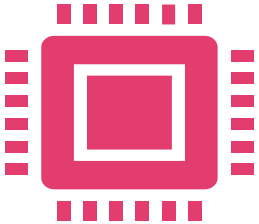
- Malicious Websites and Drive-By Downloads:
  - Phishing: Attackers create fake websites that mimic legitimate ones to trick users into entering sensitive information.
  - Drive-By Downloads: Malicious code is automatically downloaded and executed when a user visits a compromised or malicious website.
- Malvertising:
  - Malicious advertisements are used to spread malware. These ads can appear on legitimate websites and may contain hidden code that exploits vulnerabilities in the user's browser or plugins.

Watering Hole Attacks:

- Attackers compromise websites that are frequently visited by the target audience. When users visit these legitimate sites, they may unknowingly download malware or become victims of other types of attacks.

Cross-Site Scripting (XSS):

- Attackers inject malicious scripts into web pages that are viewed by other users. These scripts can steal sensitive information or perform actions on behalf of the user without their consent.



Cross-Site Request Forgery (CSRF):

Malicious websites trick users into performing actions on other websites where the user is authenticated, potentially leading to unauthorized actions.

Email-Based Attacks:

Malicious Attachments: Emails with infected attachments that, when opened, can compromise the user's system.

Malicious Links: Emails containing links to phishing websites or sites hosting malware.

- Browser Exploits:
  - Exploiting vulnerabilities in web browsers to execute malicious code on the user's system.
- File Format Exploits:
  - Malicious files, such as PDFs, Word documents, or Excel spreadsheets, that exploit vulnerabilities in the associated software to execute malicious code.
- Social Engineering:
  - Manipulating users into taking actions that could compromise their security, such as clicking on links, downloading malicious files, or revealing sensitive information.

Browser Exploitation

- Browser exploitation refers to the process of taking advantage of vulnerabilities or weaknesses in web browsers to compromise the security of a user's system.

Common Techniques of Browser Exploitation

- Code Execution: Attackers may exploit vulnerabilities in a browser to execute arbitrary code on the user's system. This code could be used to install malware, steal sensitive information, or carry out other malicious activities.
- Cross-Site Scripting (XSS): In an XSS attack, malicious scripts are injected into web pages that are then viewed by other users. These scripts can steal user information, manipulate content, or perform actions on behalf of the user without their consent.
- Drive-By Downloads: Malicious code is automatically downloaded and executed on a user's system when they visit a compromised or malicious website. This can lead to the installation of malware without the user's knowledge.
- Browser Plugin Exploits: Vulnerabilities in browser plugins, extensions, or add-ons can be exploited to compromise the security of the browser. Attackers may target popular plugins that have a wide user base.

- Clickjacking: This involves tricking users into clicking on something different from what they perceive, potentially leading them to perform unintended actions. Clickjacking attacks can be used to trick users into unknowingly interacting with malicious content.

- Browser Redirection: Attackers may exploit vulnerabilities to redirect users to malicious websites, phishing pages, or other destinations without their knowledge.

- Browser Fingerprinting: This technique involves collecting unique information about a user's browser and device to create a unique identifier (fingerprint). While not always malicious, this information can be exploited for tracking or targeted attacks.

PDF Exploitation

PDF exploitation refers to the abuse of vulnerabilities or security weaknesses in PDF (Portable Document Format) files to compromise the security of a computer or network.

Common Aspects Of PDF Exploitation

- Malicious PDF Documents
- JavaScript Exploitation
- Embedded Malware
- Social Engineering Attacks
- Zero-Day Exploits

Java Exploitation

Java exploitation refers to the exploitation of vulnerabilities or weaknesses in the Java programming language or the Java Runtime Environment (JRE) to compromise the security of a computer system.

Common Aspects of Java Exploitation

- Java Applet Vulnerabilities
- Java Web Start Exploits
- Remote Code Execution
- Network-Based Attacks
- Privilege Escalation
- Denial-of-Service (DoS) Attacks

Browser_autopwn

A term associated with the Metasploit Framework, a widely used penetration testing and exploitation tool.

The term "Browser_autopwn" specifically refers to a module or feature in Metasploit that automates the process of exploiting web browsers.

It typically involves the creation and delivery of malicious payloads through web browser vulnerabilities, allowing the penetration tester or security professional to test the security of a system.

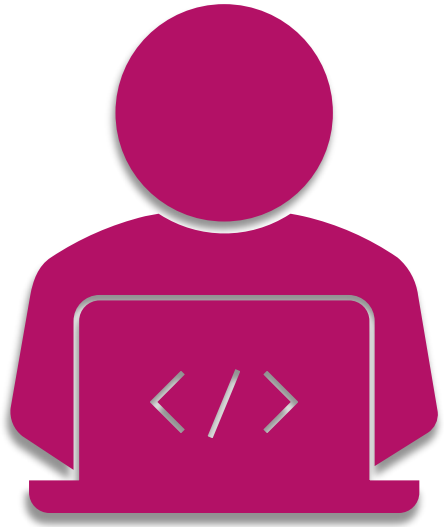
Key Points Related to Browser_autopwn

- Automation: Browser_autopwn is designed to automate the exploitation of web browser vulnerabilities. It streamlines the process of identifying and exploiting browser-based security issues.

- Payload Delivery: The module is used for delivering various types of payloads to the target system. These payloads could include malicious code or software that, when executed, provides the tester with access to the target system.

Browser Exploitation: Browser_autopwn focuses on exploiting vulnerabilities within web browsers. These vulnerabilities may include issues with browser plugins, JavaScript engines, or other components that could be leveraged for unauthorized access.

Metasploit Framework: Metasploit is a comprehensive framework that includes a wide range of modules for different types of security testing. Browser_autopwn is just one of the many modules available within Metasploit.



- **Ethical Hacking:** Metasploit is often used by security professionals and ethical hackers to identify and fix vulnerabilities in systems. It is crucial to use such tools responsibly and legally with proper authorization.

Winamp

- Winamp is a media player application that was popular in the late 1990s and early 2000s for playing audio and video files on Windows-based systems.
- In the context of client-side penetration testing, Winamp might be used as a vector for exploitation if a system has a vulnerable version installed.

Points Related to Winamp

Outdated Software: If a system is running an outdated version of Winamp with known vulnerabilities, it could be exploited as part of a client-side attack.

Malicious Media Files: Attackers might attempt to craft malicious media files (audio or video) that exploit vulnerabilities in Winamp when played.

Exploitation Frameworks: Some penetration testing frameworks may include modules or tools designed to exploit vulnerabilities in specific software applications like Winamp.

Security Patching: Regularly updating and patching software, including media players like Winamp, is crucial for minimizing the risk of exploitation.

ALBUKHARY INTERNATIONAL UNIVERSITY

Thank You