Chapter 14

Hacking Web Applications

Theory

Ethical Hacking

Web Application

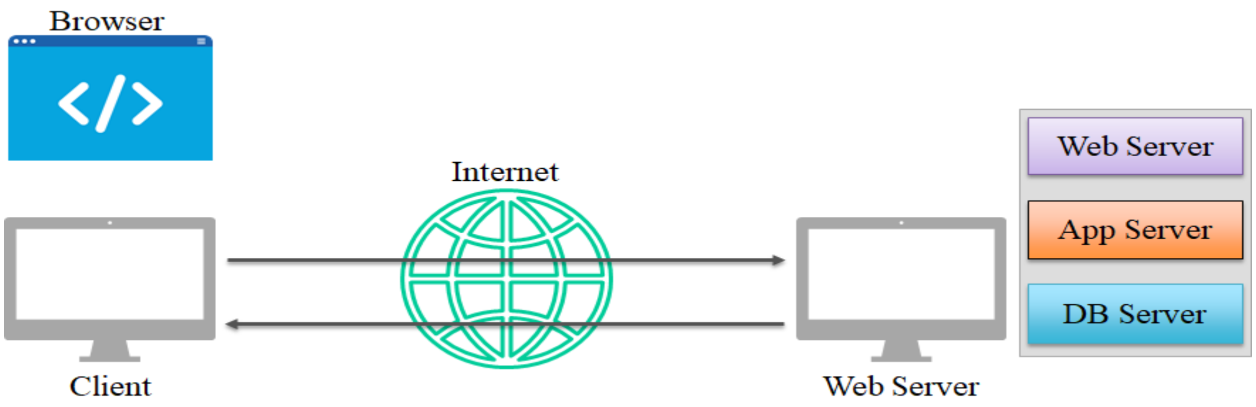
A Web Application is a program that is accessed over a network connection using HTTP or HTTPS existing in the web server. The web application is a client-server application which client run in web browser. The web application contains a set of web pages, scripts, images, etc. Web applications help organizations to grow their business.

Types of websites

Static Website: A static website contains web pages with fixed content. A static site can be built using HTML and hosted on a Web server.

Dynamic Website: The information on dynamic website changes based on user interaction, the time zone, the viewer's native language, and other factors. These pages include Web scripting code, such as PHP or ASP. When a dynamic page is accessed, the code within the page is parsed on the Web server, and the resulting HTML is sent to the client web browser. Dynamic websites can interact with the user, capable of access information stored on the database. Dynamic web pages are also known as database-driven websites.

How A Web Application Works



- The user sends a request to the web server over the internet through the web browser or the application interface in the form of URL or an HTML form
- Web server forward these requests to the web application server
- Web application server queries the database and generates the results for the requested task
- Web servers respond back to the client with the requested information.

OWASP

The Open Web Application Security Project (OWASP), an online community, produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security. It aims to raise awareness about application security by identifying some of the most critical risks that organizations are facing.

OWASP Top 10 Web Application Security Risks

A1:2017 - Injection
A2:2017 - Broken Authentication
A3:2017 - Sensitive Data Exposure
A4:2017 - XML External Entities (XXE)
A5:2017 - Broken Access Control
A6:2017 - Security Misconfiguration
A7:2017 - Cross-Site Scripting (XSS)
A8:2017 - Insecure Deserialization
A9:2017 - Using Components with Known Vulnerabilities
A10:2017 - Insufficient Logging & Monitoring

Parameter Tampering

This attack involves the manipulation of parameters exchanged between client and server to modify application data such as user credentials, permissions, price, the quantity of products. Establishing a proxy can make the process of tampering simple if the web application fails in proper session management.

Directory Traversal

- Directory Traversal or Path Traversal is an attack on HTTP which allows attackers to access restricted directories outside of the web server root location.
- Attackers try to access restricted directories that contain sensitive information like server configuration files, application source code, etc.
- Attackers can manage to access files located outside the web root because of this vulnerability.

Example: http://www.example.com/abc.php=../../../../etc/passwd

Cross-site scripting attack

- XSS attack takes advantage of dynamically generated web content based on user input provided on web pages.

- An attacker tries to inject commands in input fields provided on web pages.
- If the web server is unable to validate input fields on a web page properly, then it will execute the command provided by the attacker and unknowingly reveal information related to the client.

Types of XSS: Reflected XSS, Stored XSS, DOM Based XSS

Cross-site request forgery

Any request sent to the server are not validated (No server-side validation). The server processes the request without verifying whether the user made the request or not. Because of poor validation, requests can be forged and sent to users to force them to do things that they are not intended to do. By clicking some links, users may unknowingly change account passwords.

Command Injection

- Command injection vulnerability in a web application allows attackers to inject untrusted data and execute it as part of regular command or query.
- Attackers use specially crafted malicious commands or queries to exploit these vulnerabilities which may result in data loss.
- Injection attacks are possible because of poor web development capabilities.

File Inclusion

Local File Inclusion - Allows an attacker to gain access to any file on the server computer. An attacker can even access a file located apart from the web-root folder.
Remote File Inclusion - Allows an attacker to gain access to any file from any server. We can execute file located on a remote server on the vulnerable server.

Countermeasures

- Define access rights to private folders on the web server.
- Validate user input length, perform bound checking
- Use language-specific libraries for the programming language.
- Use the more secure HTTPS protocol instead of HTTP if available.
- Log out from online user accounts instead of directly closing the browser, to properly end sessions.
- Take careful note of security warnings from the web browser
- Avoid clicking links to sensitive portals, such as for e-banking, instead enter the URL of the website manually.
- Keep web application building software (frameworks) up to date to protect websites from application-based attacks.

References:

1. The Tech Terms Computer Dictionary. (n.d.). Retrieved from https://techterms.com/
2. OWASP. (2018, June 26). Retrieved from https://en.wikipedia.org/wiki/OWASP
3. Top 10-2017 Top 10. (n.d.). Retrieved from https://www.owasp.org/index.php/Top_10-2017_Top_10