12.2.4 Web Application Hacking Facts

Web applications provide an interface between the website and the database. Hackers frequently target web applications because, typically, more effort is put into securing the web servers and the database, than the web applications. An attacker who is patient and strategic has a good chance of finding at least some of what he's looking for.

This lesson covers the following topics:

- Web application hacking methodology
- Web application hacking tools

Web Application Hacking Methodology

The web application hacking methodology breaks down web application hacking into 10 steps, each one targeting specific portions of the web application. It's important to note that depending on the scenario, not all steps are used. Also, the steps may not be used in the order present here. However, this methodology helps to provide a checklist to test various attacks.

1. The first step in the web application hacking methodology is footprinting the web infrastructure. This means gathering information about the system, its components, and how they work together. Whois Lookup can be used to gather information about the IP address and DNS names of the web server. DNS Interrogation can help to provide details about the type of servers you'll be working with and where they are located. Port scanning can be used to try to connect to TCP or UDP ports to determine the services that exist on the servers.

Footprinting Method	Description
Whois Lookup	Use Whois Lookup to gather information about the IP address of a web server and DNS names.
DNS interrogation	Collect information about the location and type of servers.
Port scanning	Attempt to connect to TCP or UDP ports to determine the services running on the server.
Service discovery	Scan the web server to find ports that the web server is using for various services. These services can be used as pathways for application hacking.
Banner grabbing	Analyze the header field in the server's response to determine the software and version type the web server is using.
Detect firewalls	Determine if the target is running a WAF (web application firewall). WAFs analyze HTTP traffic to prevent web application attacks. To determine if the target is running a WAF, the cookie responses can be checked. Most WAFs include their own cookies in the response.
Detect proxy servers	Determine whether a target is using a proxy by viewing the response header field for headers indicating a proxy server.

2. The next step is to analyze the web applications. First, use a vulnerability scanner to scan for known vulnerabilities. Second, target any newly discovered vulnerabilities with a web server attack. Third, launch a denial-of-service attack against the web server. Tools that can be used during this step include UrlScan, Nessus, and WebInspect.
3. The third step is to analyze the web applications. First, identify entry points for user input. Then, identify server-side functionality by observing the applications that are provided to the client. Identify the server-side technologies by examining the technologies that are active on the server.
4. The fourth step is to attack the authentication mechanism by targeting the implementation and misconfiguration errors in the web applications. This could include user names, cookies, session IDs, or passwords. Passwords can be cracked using brute force or simply by guessing weak passwords. Cookies can be targeted using sniffing tools or by cookie poisoning. Sessions can be attacked using prediction, brute force, or poisoning. The following table describes methods for gathering information to attack authentication mechanisms.

Method	Description
Identify entry points	Examine URL, POST data, cookies, and HTTP headers to identify user input fields. HTTP headers can contain hidden parameters such as user-agent, accept, host headers, and referrer. Tools used during this step include Burp Suite, HttPrint, and WebScarab.
Identify server-side functionality	Identify server-side structures and functionality using the applications that are shown to the client. Review URLs for file extensions or other information that could help determine the technologies being used on the server.
Identify server-side technologies	Identify active technologies on the server by using fingerprinting techniques such as HTTP fingerprinting.
Map the attack surface	Identify the attack surfaces and vulnerabilities associated with each.

5. The fifth step is to attack authorization schemes. Attackers modify input fields in HTTP requests to get around the application authorization schemes. Typically an attacker would access the web application using an account with fairly low privileges and then escalate the privileges to gain access to more secure resources. Sometimes the query string is readily visible in the address bar, making it easy for the attacker to change the string parameter. If the application is using an HTTP header for access control, it can also be modified to access secure functionalities. The following table identifies methods for gathering information to attack authorization schemes.

Method	Description
Username enumeration	Some applications generate usernames automatically based on sequence. An attacker may be able to determine the sequence and identify valid usernames.
Session attacks	Session IDs can be predicted or brute forced. During these attacks, the attacker collects valid session IDs using sniffing techniques. The attacker then analyzes these session IDs to help predict valid session IDs.
Cookie exploitation	Cookies can contain session IDs and even passwords. An attacker can use script injection and eavesdropping techniques to collect this data from the cookies. The attacker can use that same cookie to access web applications without additional authentication.
Password attacks	Passwords can be exploited by using the change password, forgot password, or remember me functions. In some instances, passwords can be guessed. A password list is a list of possible passwords by combining commonly used passwords with information gathered using fingerprinting and social engineering. This can be done manually or using brute force password cracking tools.

6. The sixth step is to attack the session management mechanisms. As we know, web applications use sessions to establish a connection and transfer sensitive information between a client and a server. Attacking an application's session management mechanisms can help to get around some of the authentication controls and will allow an attacker to use the permissions of more privileged application users. This can be accomplished several different ways including session token prediction, session hijacking, or man-in-the-middle attacks. The following table describes methods to hack session management mechanisms.

Method	Description
Query side tampering	If a query string is in the browser's address bar, an attacker can bypass authorization mechanisms by changing the string parameter.
HTTP headers	If an application is using the referrer header to make access control decisions, an attacker can modify it in order to access protected application functions.
Cookie parameter tampering	An attacker collects cookies and analyzes them to determine how the cookies are being generated. The attacker then uses tools to tamper with the parameters and replays them to the application.

7. The seventh step is to perform injection attacks. Injection attacks are used to target the input validation features of web applications. This attack involves the injection of malicious commands into an input string with the goal of modifying a database or altering a website. Injection attacks are quite common and can do serious damage to data, services, and overall system performance. Injection attacks fall into two broad categories, code injection and command injection. With code injection, the hacker inputs code that is executed by the application. With command injection, the attacker alters the functionality of the application's execution commands. The following table describes injection methods.

Method	Description
Web script injection	If user input is used in executed code, entries can be crafted in a way that ignores the original data and executes commands on the server.
LDAP injection	LDAP injection targets non-validated web application input vulnerabilities to get past LDAP filters and to gain access to the database.
OS command injection	OS command injection targets operating systems by putting malicious code into the input fields.
XPath injection	With XPath injection, altered strings are entered into input fields and are used to manipulate the XPath query in a way that interferes with the application's logic.
SMTP injection	SMTP commands can be injected into conversations between an application and an SMTP server to generate excessive amounts of spam email.
Buffer overflow	Buffer overflow injects large amounts of invalid data beyond the input field's capacity.
SQL injection	With SQL injection, a series of SQL queries are entered into the input fields to manipulate the database.

8. The eighth step is to attack the application logic flaws. Web applications are complex. They contain the many steps needed for the application to respond to input and to complete a particular action. These steps – or logic – are susceptible to flaws and hard to recognize loopholes; loopholes that a developer may not have time to discover because of a time crunch, but that attackers are more than happy to look for. These flaws could be found in user privileges, traffic filtering, cookie management, web parameters, directory access, business constraints, and others.

9. The ninth step is to attack the database connectivity. Database connection strings are used to make the connection. An attacker will inject parameters into these strings in an attempt to steal a password hash, connect to various ports, or to hijack web credentials. The following table describes various ways to hack database connectivity.

Method	Description
Connection string injection	When working in a delegated authentication setting, the attacker injects parameters into a connection string using a semicolon. When the connection string is populated, the encryption value will be added to the previously configured parameters.
Connection string parameter pollution	Connection string parameter pollution (CSPP) attacks overwrite the connection string's parameter values. This could be done using hash stealing, port scanning, or by hijacking web credentials.
Hash stealing	Hash stealing occurs when an attacker replaces a data source parameter value with a rogue SQL server that is running a sniffer. The attacker will use the sniffer to obtain password hashes when the application attempts to connect to the rogue server.
Port scanning	Port scanning occurs when an attacker attempts to connect to ports by changing the value and seeing the resulting error messages.
Hijacking web credentials	An attacker attempts to connect to the database using a web application system account instead of user-provided credentials.
Connection pool DoS	In a connection pool DoS, the attacker reviews the application's connection pool settings and creates a malicious SQL query before running multiple queries at the same time in an attempt to consume all connections in the connection pool. This results in failed queries for authorized users.

10. The final step in the methodology is to attack the web client application. Attackers use the client application to interact with the server in an attempt to access unauthorized data. This can be done using cross-site scripting, redirection attacks, HTTP header injection, frame injection, or session fixation attacks.

Web Application Hacking Tools

This table lists several web application hacking tools:

Tool	Description
Burp Suite	Burp Suite is a web spidering application that can be used as a local proxy. Once you have browsed every link and URL that you can find and completed every form and application available, Burp provides a site map and identifies hidden application content or functions that it can find. Browsing the application in various conditions (i.e. JavaScript enabled or disabled) can discover additional content or functionality.
CookieDigger	CookieDigger is a tool that identifies weak cookies and insecure session management by collecting and analyzing cookies issued to various users by a web application. The tool reports on the predictability and contents of the cookie values.
WebScarab	WebScarab is a tool for analyzing applications that use HTTP and HTTPS protocols. The attacker can use this tool to review and modify requests and responses sent between the browser and the server.

TestOut Corporation All rights reserved.