HACKTHEBOX

UPDATE: WEB FUZZING

CHEAT SHEET

Web Fuzzing Cheatsheet

What is Web Fuzzing?

`Web fuzzing` is a technique used to discover vulnerabilities, hidden resources, and security issues in web applications by automatically injecting a large set of input data into the application and analyzing its response. The goal is to identify unexpected behaviors or errors that could indicate potential security weaknesses or misconfigurations.

Fuzzing is commonly employed in security testing to find:

- Hidden directories and files
- Insecure APIs and endpoints
- SQL injection points
- Cross-site scripting (XSS) vulnerabilities
- Command injection flaws

Comparison: Brute-Forcing vs. Fuzzing

Criteria	Brute-Forcing	Fuzzing
Definition	Systematically trying all possible combinations of input data to guess a specific value.	Injecting unexpected or random data into an application to find vulnerabilities and hidden resources.
Purpose	Crack passwords, keys, or other access credentials.	Discover application vulnerabilities, hidden files, directories, and input validation issues.
Methodology	Exhaustive search over all possible input combinations.	Dynamic input injection to provoke unexpected application responses.

Criteria	Brute-Forcing	Fuzzing
Focus	Specific input or data, such as passwords or API keys.	General application behavior under various input conditions.
Efficiency	Time-consuming due to exhaustive nature; less efficient for large input spaces.	More efficient in identifying unexpected behaviors and vulnerabilities with varied input.
Tools Used	Password crackers, key recovery tools.	Web fuzzers, vulnerability scanners.
Output	Successful match of the correct input value.	Discovery of vulnerabilities, misconfigurations, and hidden resources.

Miscellaneous Commands

Below are some useful commands that can aid in various tasks related to web fuzzing and testing.

Command	Description
`sudo sh -c 'echo "SERVER_IP academy.htb" >> /etc/hosts'`	Add a DNS entry for a specific IP address to the `/etc/hosts` file. This helps resolve domain names locally.
`for i in $(seq 1 1000); do echo $i >> ids.txt; done`	Create a sequence wordlist from 1 to 1000. Useful for brute-forcing numerical IDs or similar patterns.
`curl http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'id=key' -H 'Content-Type: application/x-www-form-urlencoded'`	Use `curl` to send a POST request with specific data and headers, simulating form submissions or API calls.

HTB ACADEMY CHEATSHEET

Commonly Used SecLists Wordlists

SecLists is a collection of multiple types of wordlists used by security researchers and penetration testers. Below is a table of some commonly used wordlists from SecLists, which can be incredibly valuable during web fuzzing.

Wordlist	Description
`/usr/share/seclists/Discovery/Web-Content/common.txt`	**General-Purpose Wordlist**: Contains a broad range of common directory and file names on web servers. It's an excellent starting point for fuzzing and often yields valuable results.
`/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt`	**Directory-Focused Wordlist**: A more extensive wordlist specifically focused on directory names. It's a good choice when you need a deeper dive into potential directories.
`/usr/share/seclists/Discovery/Web-Content/raft-large-directories.txt`	**Large Directory Wordlist**: Boasts a massive collection of directory names compiled from various sources. It's a valuable resource for thorough fuzzing campaigns.
`/usr/share/seclists/Discovery/Web-Content/big.txt`	**Comprehensive Wordlist**: A massive wordlist containing both directory and file names. Useful when you want to cast a wide net and explore all possibilities.

Tips for Using Wordlists Effectively

Tip	Explanation
Choose the Right Wordlist	Select wordlists relevant to the target environment and technology stack for better results.
Combine Wordlists	Use multiple wordlists together to increase the breadth of your fuzzing efforts.
Customize Wordlists	Modify existing wordlists or create your own based on specific knowledge about the target.

Tip	Explanation
Monitor Performance	Large wordlists can be resource-intensive; monitor performance and adjust as needed.
Leverage Community Resources	Utilize community-maintained wordlists for the latest and most effective fuzzing strategies.

Tools for Web Fuzzing

ffuf (Fuzz Faster U Fool)

`ffuf` is a fast web fuzzer written in Go that allows you to discover directories and files on web servers.

Command	Description
`ffuf -u http://example.com/FUZZ`	Basic fuzzing of a URL path.
`ffuf -u http://example.com/FUZZ -w wordlist.txt`	Fuzz with a specific wordlist.
`ffuf -u http://example.com/FUZZ -w wordlist.txt -ic`	Fuzz with a specific wordlist, automatically ignoring any comments in the wordlist.
`ffuf -u http://example.com/FUZZ -w wordlist.txt -c`	Colorize the output for better readability.
`ffuf -u http://example.com/FUZZ -w wordlist.txt -mc 200`	Filter results by status code (e.g., 200).
`ffuf -u http://example.com/FUZZ -w wordlist.txt -mr "Welcome"`	Filter results by matching a regex pattern.
`ffuf -u http://example.com/FUZZ -w wordlist.txt -e .php,.html`	Add extensions to each wordlist entry.
`ffuf -u http://example.com/FUZZ -w wordlist.txt -t 50`	Set the number of threads (e.g., 50) for faster fuzzing.
`ffuf -u http://example.com/FUZZ -w wordlist.txt -x http://127.0.0.1:8080`	Use a proxy for requests.

gobuster

`gobuster` is a tool used to brute-force URIs (directories and files) in web sites and DNS subdomains.

Command	Description
`gobuster dir -u http://example.com -w wordlist.txt`	Directory fuzzing using a wordlist.
`gobuster dir -u http://example.com -w wordlist.txt -x .php,.html`	Fuzz with specific extensions.
`gobuster dir -u http://example.com -w wordlist.txt -s 200`	Filter results by status code (e.g., 200).
`gobuster dir -u http://example.com -w wordlist.txt -t 50`	Set the number of concurrent threads (e.g., 50).
`gobuster dir -u http://example.com -w wordlist.txt -o results.txt`	Output results to a file.
`gobuster dns -d example.com -w subdomains.txt`	Fuzz DNS subdomains using a wordlist.
`gobuster dns -d example.com -w subdomains.txt -i`	Show IP addresses of discovered subdomains.
`gobuster dns -d example.com -w subdomains.txt -z`	Silent mode; suppress output except for results.

wneum (Wfuzz Fork)

`wneum` is a fork of `wfuzz`, a versatile web application fuzzer for testing web security.

Command	Description
`wneum -c -z file,wordlist.txt --hc 404 http://example.com/FUZZ`	Basic fuzzing excluding 404 responses.
`wneum -c -z file,wordlist.txt -d 'username=FUZZ&password=secret'`	Fuzz POST data in a form.
`wneum -c -z file,wordlist.txt -b 'session=12345'`	Use a specific cookie for requests.

Command	Description
`wneum -c -z file,wordlist.txt -H 'User-Agent: Wneum'`	Add a custom header to requests.
`wneum -c -z file,wordlist.txt -t 50`	Set the number of threads (e.g., 50) for faster fuzzing.
`wneum -c -z file,wordlist.txt -u http://example.com/FUZZ -X PUT`	Fuzz using a specific HTTP method (e.g., PUT).
`wneum -c -z file,wordlist.txt --hl 50`	Filter responses by content length (e.g., 50 bytes).

feroxbuster

`feroxbuster` is a tool designed for recursive content discovery and web fuzzing.

Command	Description
`feroxbuster -u http://example.com -w wordlist.txt`	Basic URL fuzzing with a wordlist.
`feroxbuster -u http://example.com -w wordlist.txt -e`	Include specified file extensions in fuzzing.
`feroxbuster -u http://example.com -w wordlist.txt -x 404`	Exclude responses with status code 404.
`feroxbuster -u http://example.com -w wordlist.txt -t 50`	Set the number of concurrent threads (e.g., 50).
`feroxbuster -u http://example.com -w wordlist.txt --depth 3`	Set maximum recursion depth (e.g., 3 levels deep).
`feroxbuster -u http://example.com -w wordlist.txt -o results.txt`	Save output to a file.
`feroxbuster -u http://example.com -w wordlist.txt --no-recursion`	Disable recursion into discovered directories.
`feroxbuster -u http://example.com -w wordlist.txt --url-redirect`	Follow redirects automatically.

Tips for Effective Web Fuzzing

Tip	Explanation
Use Comprehensive Wordlists	The quality of your wordlist can significantly impact results; choose or create wordlists relevant to the target.
Filter Unwanted Responses	Use status codes or response size filtering to focus on meaningful results and reduce noise.
Adjust Thread Count	Increase thread count for faster fuzzing, but be mindful of server capabilities to avoid overloading.
Monitor Server Responses	Pay attention to anomalies or unexpected behavior in server responses, indicating potential vulnerabilities.
Fuzz with Various HTTP Methods	Test different HTTP methods (GET, POST, PUT, DELETE) to uncover potential vulnerabilities in all endpoints.

Web APIs: REST, SOAP, and GraphQL

What is a Web API?

A **Web API** (Application Programming Interface) is a set of rules and protocols for building and interacting with software applications. APIs allow different applications to communicate with each other over the internet, enabling the integration of various services and data exchange.

Web APIs can be categorized into three main types:

1. **REST (Representational State Transfer)**
2. **SOAP (Simple Object Access Protocol)**
3. **GraphQL**

Each type has its own unique characteristics, advantages, and use cases.

REST (Representational State Transfer)

REST is an architectural style that uses standard HTTP methods to access and manipulate resources on a server. It is known for its simplicity, scalability, and statelessness.

Feature	Description
Protocol	Uses HTTP/HTTPS.

Feature	Description
Data Format	Typically JSON, but can also use XML, HTML, or plain text.
Stateless	Each request from a client to a server must contain all the information needed.
CRUD Operations	Uses HTTP methods: GET, POST, PUT, DELETE.
Scalability	Highly scalable due to its stateless nature.
Caching	Supports caching mechanisms to improve performance.
URL Structure	Uses endpoints that represent resources, e.g., `/api/users/{id}`.
Advantages	Simplicity, flexibility, scalability.
Disadvantages	Can lead to over-fetching or under-fetching data.

REST Fuzzing Tips:

Tip	Explanation
Test All HTTP Methods	Ensure all CRUD operations are tested, as vulnerabilities might exist in any of them.
Validate Input Fields	Fuzz input fields with unexpected data types and formats to uncover validation issues.
Examine Error Messages	Analyze error messages for information disclosure or unintended behavior.
Test Authentication Mechanisms	Check for improper authentication and authorization controls.
Explore API Rate Limits	Test rate limits and throttling controls to ensure the API handles requests properly.
Use Comprehensive Payloads	Leverage a variety of payloads (SQLi, XSS) to test for potential security flaws.

HTB ACADEMY CHEATSHEET

Tip	Explanation
Check Resource Representation	Test different resource representations (JSON, XML) for consistency and security flaws.

SOAP (Simple Object Access Protocol)

SOAP is a protocol for exchanging structured information in web services. It uses XML as its message format and can operate over various protocols like HTTP, SMTP, or TCP.

Feature	Description
Protocol	Protocol-independent but often used with HTTP/HTTPS.
Data Format	Exclusively XML.
Stateful/Stateless	Can be either stateful or stateless.
WS-Security	Built-in security features for message integrity and confidentiality.
Error Handling	Uses specific error codes and messages.
Complexity	More complex due to extensive standards and specifications.
Extensibility	Highly extensible via WS-* standards.
Advantages	Strong security, reliability, and extensibility.
Disadvantages	More complex and less flexible compared to REST.

SOAP Fuzzing Tips:

Tip	Explanation
Analyze WSDL Files	Use WSDL (Web Services Description Language) files to understand the service's operations and inputs.
Validate XML Schema	Test XML inputs against the schema to identify validation flaws.
Check for XML Injection	Fuzz XML data to test for injection vulnerabilities.

HTB ACADEMY CHEATSHEET

Tip	Explanation
Test SOAP Headers	Fuzz SOAP headers to find potential security issues or misconfigurations.
Evaluate WS-Security Implementations	Ensure security implementations are robust and correctly configured.
Test Transport Security	Verify that transport-level security (e.g., HTTPS) is enforced and properly implemented.
Examine SOAP Faults	Analyze SOAP fault messages for potential information leakage.

GraphQL

GraphQL is a query language and runtime for APIs that allows clients to request specific data and define the structure of the response.

Feature	Description
Protocol	Uses HTTP/HTTPS, typically over POST requests.
Data Format	JSON-based queries and responses.
Stateful/Stateless	Stateless architecture.
Query Flexibility	Clients can request exactly what they need, minimizing over-fetching and under-fetching.
Single Endpoint	Typically uses a single endpoint for all operations.
Introspection	Allows clients to query the API schema for available operations and data types.
Advantages	Efficiency, flexibility, and powerful developer tooling.
Disadvantages	Potential for complex queries leading to performance issues if not properly managed.

GraphQL Fuzzing Tips:

Tip	Explanation
Test Query Depth and Complexity	Evaluate the server's handling of deeply nested or complex queries to avoid performance bottlenecks.
Validate Input Types and Arguments	Fuzz input arguments with unexpected values and data types to uncover validation flaws.
Examine Query Aliasing and Batching	Test the server's response to aliased queries and batching for potential information leakage.
Check for Introspection Misuse	Ensure introspection is not exposing sensitive information or internal schema details.
Assess Authorization Controls	Verify that access controls are properly enforced for different queries and operations.
Evaluate Rate Limiting	Test rate limits to ensure the API can handle excessive or malicious requests appropriately.
Fuzz Mutations	Mutations can alter data; test for security issues and improper input validation.

HTB ACADEMY CHEATSHEET