

Web Application Pentest Methodology

DRAFT v0.5 – Authored by Joshua Wardle

Based on: https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/

Contents

- **INFO – Information Gathering**
 - INFO-01: Fingerprint Web Server
 - INFO-02: Enumerate Applications on Web Server
 - INFO-03: Identify Web Application Framework
 - INFO-04: Map Application
- **CONFIG – Configuration Testing**
 - CONFIG-01: Out-of-Date Server Software and Default Passwords
 - CONFIG-02: Web Application Framework Configuration
 - CONFIG-03: Weak Encryption
- **IDENTITY – Identity Management and Authorisation**
 - IDENTITY-01: Test User Roles and Privilege Escalation
 - IDENTITY-02: Account Registration and Provisioning
 - IDENTITY-03: User Account Enumeration
- **AUTHENTICATE – Authentication Testing**
 - AUTHENTICATE-01: Transfer of Sensitive Information in Plaintext
 - AUTHENTICATE-02: Password Policy Testing
- **SESSION – Session Management**
 - SESSION-01: Cookie Attributes
 - SESSION-02: Predictable Cookies
 - SESSION-03: Session Fixation
 - SESSION-04: Cross-Site Request Forgery
- **INPUT – Input Validation Testing**
 - INPUT-01: Reflected Cross-Site Scripting
 - INPUT-02: Stored Cross-Site Scripting
 - INPUT-03: DOM-Based Cross-Site Scripting
 - INPUT-04: SQL Injection
 - INPUT-05: OS Command Injection
 - INPUT-06: Local File Inclusion
 - INPUT-07: Remote File Inclusion
 - INPUT-08: Format String Injection
 - INPUT-09: Server-Side Template Injection
 - INPUT-10: Server-Side Request Forgery
 - INPUT-11: Others
- **ERROR – Testing for Error Handling**
 - ERROR-01: Stack Traces and Improper Error Handling
- **LOGIC – Business Logic Testing**
 - LOGIC-01: Circumvention of Business Logic
 - LOGIC-02: Insecure File Upload
- **CLIENT – Further Client-Side Testing**
 - CLIENT-01: Open Redirects
 - CLIENT-02: Clickjacking
 - CLIENT-03: Cross Origin Resource Sharing
- **ATTACK – Proof-of-Concept Attack Chains**
 - ATTACK-01: Demonstrate Viable Attacks Against Users
- **POST – Post-Engagement Clean-Up**
 - POST-01: Delete Deployed Payloads
 - POST-02: Notify Client of Dummy Accounts

INFO – Information Gathering

INFO-01: Fingerprint Web Server

[Link](#)

Use a tool such as [Nikto](#) or [Nmap](#) to identify the web server software.

INFO-02: Enumerate Applications on Web Server

[Link](#)

Using the aforementioned tools, enumerate further applications hosted on the web server.

INFO-03: Identify Web Application Framework

[Link](#)

Analyse cookies and HTTP response headers to identify web application framework.

INFO-04: Map Application

Browse through the application. For all possible links you identify, note:

- What the URL does and what kinds of requests you make (GET/PUT/POST)
- Any URL parameters, GET, or POST parameters
- The expected level of authentication/authorization (i.e., control flow)
- Areas where user-defined input is used on the page (stored or reflected)
- Any cookies
- Any preliminary thoughts, such as types of potentially relevant vulnerabilities

Use of tools such as OWASP Zap or Burp Suite can help with this.

A table is provided on the next page. Copy the table to add more URLs as needed. An example is given to aid understanding.

Key: S (SQL Injection), X (XSS), M (Business Logic Manipulation), ? (Feature Present), V (Vulnerable), E (Expected), A (Actual)

URL & URL Parameters*	GET Parameters	POST Parameters	Authorisation	Features	Comments																																																																																																																																							
<div>/store/view?item=x [This is an example.]</div> <table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M													<table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td>item</td><td>Y</td><td>Y</td><td>N</td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M	item	Y	Y	N																																	<table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M																																					<table><tr><th>Role</th><th>E</th><th>A</th></tr><tr><td>Guest</td><td>N</td><td>Y</td></tr><tr><td>Customer</td><td>Y</td><td>Y</td></tr><tr><td>Admin</td><td>Y</td><td>Y</td></tr><tr><td></td><td></td><td></td></tr></table>	Role	E	A	Guest	N	Y	Customer	Y	Y	Admin	Y	Y				<table><tr><th>Feature</th><th>?</th><th>V</th></tr><tr><td>DB Read</td><td>Y</td><td>Y</td></tr><tr><td>DB Write</td><td>N</td><td>N</td></tr><tr><td>Logic</td><td>N</td><td>N</td></tr><tr><td>CSRF</td><td>N</td><td>N</td></tr><tr><td>Upload</td><td>N</td><td>N</td></tr><tr><td>Email</td><td>N</td><td>N</td></tr><tr><td>SetCookie</td><td>N</td><td>N</td></tr></table>	Feature	?	V	DB Read	Y	Y	DB Write	N	N	Logic	N	N	CSRF	N	N	Upload	N	N	Email	N	N	SetCookie	N	N	<div>item parameter is SQL-injectable if payload is URL encoded. Custom error message if item no found reflects item parameter. Items should not be visible to guests, but this is not the case.</div>
Parameter	S	X	M																																																																																																																																									
Parameter	S	X	M																																																																																																																																									
item	Y	Y	N																																																																																																																																									
Parameter	S	X	M																																																																																																																																									
Role	E	A																																																																																																																																										
Guest	N	Y																																																																																																																																										
Customer	Y	Y																																																																																																																																										
Admin	Y	Y																																																																																																																																										
Feature	?	V																																																																																																																																										
DB Read	Y	Y																																																																																																																																										
DB Write	N	N																																																																																																																																										
Logic	N	N																																																																																																																																										
CSRF	N	N																																																																																																																																										
Upload	N	N																																																																																																																																										
Email	N	N																																																																																																																																										
SetCookie	N	N																																																																																																																																										
<div>[URL goes here]</div> <table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M													<table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td>item</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M	item																																				<table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M																																					<table><tr><th>Role</th><th>E</th><th>A</th></tr><tr><td>Guest</td><td></td><td></td></tr><tr><td>Customer</td><td></td><td></td></tr><tr><td>Admin</td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>	Role	E	A	Guest			Customer			Admin						<table><tr><th>Feature</th><th>?</th><th>V</th></tr><tr><td>DB Read</td><td></td><td></td></tr><tr><td>DB Write</td><td></td><td></td></tr><tr><td>Logic</td><td></td><td></td></tr><tr><td>CSRF</td><td></td><td></td></tr><tr><td>Upload</td><td></td><td></td></tr><tr><td>Email</td><td></td><td></td></tr><tr><td>SetCookie</td><td></td><td></td></tr></table>	Feature	?	V	DB Read			DB Write			Logic			CSRF			Upload			Email			SetCookie			
Parameter	S	X	M																																																																																																																																									
Parameter	S	X	M																																																																																																																																									
item																																																																																																																																												
Parameter	S	X	M																																																																																																																																									
Role	E	A																																																																																																																																										
Guest																																																																																																																																												
Customer																																																																																																																																												
Admin																																																																																																																																												
Feature	?	V																																																																																																																																										
DB Read																																																																																																																																												
DB Write																																																																																																																																												
Logic																																																																																																																																												
CSRF																																																																																																																																												
Upload																																																																																																																																												
Email																																																																																																																																												
SetCookie																																																																																																																																												
<div>[URL goes here]</div> <table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M													<table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td>item</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M	item																																				<table><tr><th>Parameter</th><th>S</th><th>X</th><th>M</th></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Parameter	S	X	M																																					<table><tr><th>Role</th><th>E</th><th>A</th></tr><tr><td>Guest</td><td></td><td></td></tr><tr><td>Customer</td><td></td><td></td></tr><tr><td>Admin</td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>	Role	E	A	Guest			Customer			Admin						<table><tr><th>Feature</th><th>?</th><th>V</th></tr><tr><td>DB Read</td><td></td><td></td></tr><tr><td>DB Write</td><td></td><td></td></tr><tr><td>Logic</td><td></td><td></td></tr><tr><td>CSRF</td><td></td><td></td></tr><tr><td>Upload</td><td></td><td></td></tr><tr><td>Email</td><td></td><td></td></tr><tr><td>SetCookie</td><td></td><td></td></tr></table>	Feature	?	V	DB Read			DB Write			Logic			CSRF			Upload			Email			SetCookie			
Parameter	S	X	M																																																																																																																																									
Parameter	S	X	M																																																																																																																																									
item																																																																																																																																												
Parameter	S	X	M																																																																																																																																									
Role	E	A																																																																																																																																										
Guest																																																																																																																																												
Customer																																																																																																																																												
Admin																																																																																																																																												
Feature	?	V																																																																																																																																										
DB Read																																																																																																																																												
DB Write																																																																																																																																												
Logic																																																																																																																																												
CSRF																																																																																																																																												
Upload																																																																																																																																												
Email																																																																																																																																												
SetCookie																																																																																																																																												

*A URL parameter is one such as /view/post/<id>, whereas a GET parameter is a key-value pair such as /view/post?id=<x>.

CONFIG – Configuration Management

CONFIG-01: Out-of-Date Server Software and Default Passwords

[Link](#)

Using the results of reconnaissance, determine if any server software is out-of-date, or if any default passwords are still in use.

CONFIG-02: Web Application Framework Configuration

[Link](#)

Check the web framework for any default/known files, any leftover debugging functionality, and whether ample logging of user actions occurs.

CONFIG-03: Weak Encryption

[Link \(tool\)](#)

Use a tool such as testssl.sh to analyse the TLS configuration of the server and report any known issues.

IDENTITY – Identity Management and Authorisation

IDENTITY-01: Test User Roles and Privilege Escalation

[Link 1 \(user roles\)](#) [Link 2 \(privilege escalation\)](#)

For each area of the application, test that only the expected user roles are able to use said areas (i.e., admin only URLs are only accessible by admins). If not, is privilege escalation possible through provisioning mechanisms (i.e., upgrade regular user to admin user)?

IDENTITY-02: Account Registration and Provisioning

[Link 1 \(registration\)](#) [Link 2 \(provisioning\)](#)

For this test, you want to verify that account registration and provisioning (if applicable) functionalities align with the client's business and security requirements. For example:

- Can anyone sign up? Are registered identities verified?
- Is there any verification, vetting and authorization of provisioning requests?

The above links provide a wider range of questions that are worth considering.

IDENTITY-03: User Account Enumeration

[Link](#)

If an application returns a different error message when login occurs due to an incorrect password versus a non-existent user account, then it is possible to automatically enumerate usernames using tools such as Burp.

AUTHENTICATE – Authentication Testing

AUTHENTICATE-01: Transfer of Sensitive Information in Plaintext

[Link](#)

Is there anywhere in the application where sensitive data is sent in an unencrypted form? This includes a lack of HTTPS should the application ideally have it enabled. A notable example is the transport of credential information in plaintext (e.g., session tokens, password reset tokens).

AUTHENTICATE-02: Password Policy Testing

[Link \(all bullet points\)](#)

Test all aspects of password policy, including:

- Use of default credentials
- Lock-out mechanism (is there one, and does it work as expected?)
- Means of bypassing login
- Vulnerable remember password functionality (i.e., test “Remember Me” functionality)
- Weak password requirements (length, uniqueness, and range of characters)
- Weak security question answer requirements (test complexity of questions and whether one can brute-force the answers)
- Weak password change / reset functionalities (can you reset the password to an account other than your own? How are new credentials generated?)

SESSION – Session Management

SESSION-01: Cookie Attributes

[Link](#)

Check cookies for Secure, HttpOnly, Domain, Path and Expires (where appropriate for non-persistent cookies) attributes.

SESSION-02: Predictable Cookies

[Link](#)

Are session cookies generated randomly, or not? If so, can an attacker hijack another users' session by guessing their cookie (e.g., cookie is MD5 hash of username)?

SESSION-03: Session Fixation

[Link 1 \(description\)](#) [Link 2 \(ways to exploit\)](#)

Does the website generate a cookie for a user before they log in (e.g., basket)? Does this cookie stay the same after login? If so, session fixation is possible if an attacker can induce their target to use a known cookie prior to authentication. This can be exploited by changing a cookie in transport (HSTS prevents this) or using XSS / custom malicious login page.

SESSION-04: Cross Site Request Forgery

[Link 1 \(description\)](#) [Link 2 \(payloads and flowchart to assist diagnosis\)](#)

CSRF is when an attacker can induce an authenticated user to perform an action without their knowledge. This typically occurs through state changing GET requests (e.g., embedded link in img tag), or by submitting a POST request (i.e., a form*) from an attacker-controlled domain. The latter can be quickly detected by checking for CSRF tokens in forms. For all state-changing actions (e.g., writing a post, changing user info, transferring money), check whether CSRF is possible. *CSRFER* can assist in writing payloads.

* Note some forms may want JSON as an input; Link 2 can assist in writing such payloads. Remember, CSRF works because a browser will pass your session cookies to a site even if the HTTP request originated from a different site!

INPUT – Input Validation Testing

INPUT-01: Reflected Cross-Site Scripting

[Link 1 \(description\)](#) [Link 2 \(payloads\)](#)

This variant of XSS applies if the data within a single HTTP request can be used to insert new code onto a page, such as `http://www.example.com/message?txt=Hello%20There`. This occurs if the web server places untrusted user input dynamically onto the page. Examine any areas where user input is used on the page and test for reflected XSS, then see if any filters / WAF (web application firewall) stops you. The second link above can help find payloads to bypass some protection mechanisms (e.g., filtering `<script>` tags).

INPUT-02: Stored Cross-Site Scripting

[Link 1 \(description\)](#) [Link 2 \(payloads\)](#)

This variant of XSS applies to data stored within the application, such as a database. Otherwise, perform the same steps as before, checking for where stored user input is used dynamically on pages across the application.

INPUT-03: DOM-Based Cross-Site Scripting

[Link 1 \(description\)](#) [Link 2 \(common sources, sinks, and payloads\)](#) [Link 3 \(tooling\)](#)

The above two types of XSS are caused by the *server* inserting untrusted data onto the page. DOM XSS is different, in that it is caused by *client-side* code – JavaScript takes untrusted data from a *source* (e.g., `searchParams` to retrieve GET parameters) and sends to it a *sink* (e.g., `document.getElementById(x).innerHTML` to change content of a given element), which leads to the execution of arbitrary commands or insertion of new HTML elements.

Three major sources of DOM XSS are reflected data (example above), stored data (i.e., from a database), and web messages. A common example is JSON data containing user input being used as the input to `eval()`. You'll need to test for all three scenarios; tooling such as *DOM Invader* can help with finding these vulnerabilities – see link 3 above.

INPUT-04: SQL Injection

[Link 1 \(description\)](#) [Link 2 \(payloads cheat sheet\)](#)

For pages that interact with a database, is it possible to manipulate the backend query made to perform unintentional actions (i.e., UNION keyword, SLEEP(n), or stacked queries*)? This occurs when unsanitised user input is used *directly* in an SQL query string. Start with manual in-band and blind payloads on input fields before moving to *sqlmap*.

* Stacked queries (e.g., payload of “`‘; DROP TABLE users; ’`”) do not work in MySQL but will in other DBMSes – *sqlmap* can help identify the DBMS if you were unable to do so earlier. Parameterised queries are not vulnerable.

INPUT-05: OS Command Injection

[Link](#)

For areas of the application that you suspect directly influences an OS command call, try to modify said command (e.g., use `;` on Unix to start a new command). This occurs if untrusted user input is used *directly* in an OS command string.

INPUT-06: Local File Inclusion

[Link](#)

For URLs that define a file to load (e.g., `http://www.example.com/help?f=account.php`), is it possible to specify – and thus access – any local file on the web server (e.g., `/etc/passwd/`)? You may need to URL encode the target file once or twice to trigger LFI or use the null byte to bypass any file extensions (e.g., `.php`) the server adds to the parameter.

INPUT-07: Remote File Inclusion

[Link](#)

Same as the above, but can you specify *remote* resources (e.g., other websites)?

INPUT-08: Format String Injection

[Link](#)

Format strings contain identifiers (e.g., %s) that are substituted at runtime for their correct value (think printf in C). Select combination of format strings can cause undesirable effects such as printing memory contents. Try placing format strings into input fields and see if any unintended behaviour occurs.

INPUT-09: Server-Side Template Injection

[Link 1 \(description\)](#) [Link 2 \(payloads\)](#)

Template engines are designed to generate web pages by combining fixed templates with volatile data. SSTI can occur when user input is concatenated directly into a template, rather than passed in as data. This allows attackers to inject arbitrary template directives to manipulate the template engine, often enabling them to take complete control of the server.

If you suspect a templating engine such as Flask or Jinja2 is in use, try common SSTI payloads (e.g., `{{7 * 7}}`) in any input fields and see if unintended behaviour occurs. Tools such as *tplmap* can automate this, should manual identification fail.

INPUT-10: Server-Side Request Forgery

[Link](#)

If the web server itself makes a request on behalf of users, can said requests be manipulated? A classical example of SSRF is manipulating a server-side request to access internal resources, as the web server is often treated as a trusted host.

INPUT-11: Others

[Link](#)

Review the OWASP Web Security Testing Guide for any other relevant input validations. This depends on the application itself.

ERROR – Testing for Error Handling

ERROR-01: Stack Traces and Improper Error Handling

[Link](#)

Try and cause errors in the application, such as:

- Removing data from HTTP requests
- Supplying invalid data types (requires fuzzing)
- Invalid SQL statements through SQL injection
- Using invalid HTTP verbs (e.g., not GET/PUT/POST) or otherwise sending a malformed HTTP request
- Accessing invalid pages (i.e., a 404)

Observe the result of such tests. Does the application return a custom error page, or does the default server / web framework error page display? Is a stack trace returned to the user?

--

LOGIC – Business Logic Testing

LOGIC-01: Circumvention of Business Logic

[Link 1 \(OWASP advice\)](#) [Link 2 \(definition\)](#)

Business logic refers to application interactions between a web app and backend infrastructure which encodes the real-world business rules that determine how data can be created, stored, and changed. For this, one must check for issues such as:

- If a part of the application requires the user to follow a series of steps (“work flows”), is it possible to skip steps or perform them out-of-order?
- Can requests be forged to access functionality a user isn’t intended to see (e.g., a hidden form field)?
- If a piece of functionality is supposed to be used a certain number of times, can this be bypassed?
- Does the time a process take to execute leak unintentional information?

LOGIC-02: Insecure File Upload

[Link 1 \(unexpected files\)](#) [Link 2 \(malicious files\)](#)

For areas of the application where you can upload files, is it possible to upload files of the wrong type, or upload malicious files? Start by manipulating POST requests (file extension and Content-Type) to try and upload PHP files before moving to using exiftool to embed commands in metadata* of legitimate files (e.g., an *actual* image or video).

* If a file contains a PHP command (directly or in metadata) but doesn't end in .php, it can't be executed by simply browsing to it; you'll likely need to use a file inclusion vulnerability to force the application to pass the file to the PHP interpreter.

CLIENT – Further Client-Side Testing

DOM XSS is covered in the INPUT section, as it is easier to test for DOM XSS at the same time as regular XSS.

CLIENT-01: Open Redirects

[Link](#)

This vulnerability arises in URLs where the user can specify a link to redirect to, such as `http://www.target.site?#redirect=www.fake-target.site`. For any URLs which redirect, can the user specify *any* URL? Attacks leveraging open redirects usually include phishing.

CLIENT-02: Clickjacking

[Link](#)

Clickjacking is when a user is deceived into interacting (in most cases by clicking) with something other than what the user believes they are interacting with. This occurs by placing the target site inside an invisible inline frame (*iframe*) and putting a fake one on top of it. Any clicks on the fake site are registered on the *iframe*, causing the user to perform unintended actions. A site is vulnerable to clickjacking if it can be placed in an *iframe*, or if any *frame-busting* techniques do not work as intended.

CLIENT-03: Cross Origin Resource Sharing

[Link](#)

The XMLHttpRequest L2 API can be used to perform cross-domain requests. Note that L1 only allowed requests to be sent with the same origin (as it was restricted by the same-origin policy). The Access-Control-Allow-Origin header specifies which domains can read a given HTTP response (i.e., which domains can make a CORS request). You want to test this header for insecure configurations, such as * (which allows any domain to read the request).

ATTACK – Proof-of-Concept Attack Chains

ATTACK-01: Demonstrate Viable Attacks Against Users

[Link 1 \(exploiting XSS\)](#) [Link 2 \(BeEF tool\)](#) [Link 3 \(exploiting file upload vulnerabilities\)](#)

[Link 4 \(generate phishing pages\)](#) [Link 5 \(use of Burp Clickbandit\)](#)

Combine multiple relevant vulnerabilities to produce larger scale attacks which represent a *risk* to the client.

POST – Post-Engagement Clean-Up

POST-01: Deleted Deployed Payloads

Delete/remove any payloads used during the pentest.

POST-02: Notify Client of Dummy Accounts

Make note of any accounts you create, so the client can remove them.