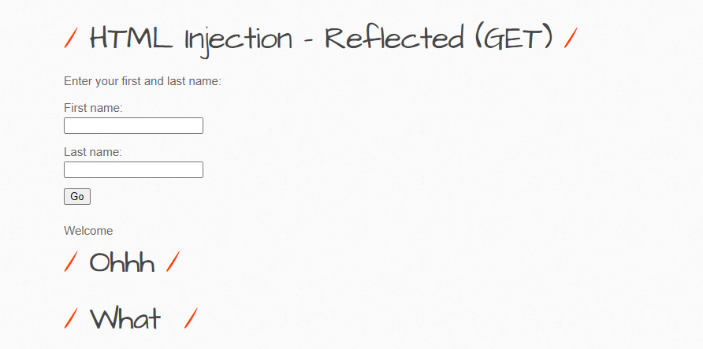
1. **HTML Injection Reflected Get:**

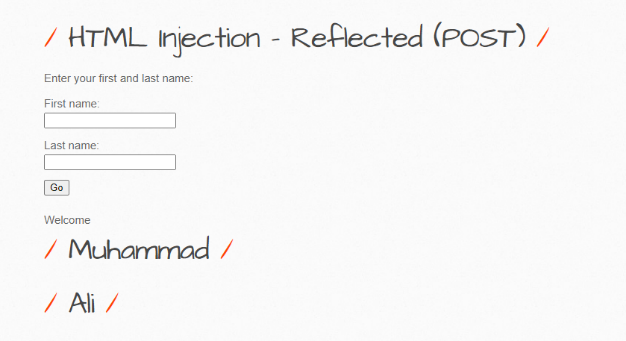
HTML injection is a type of injection issue that occurs when a user is able to control an input point and is able to inject arbitrary HTML code into a vulnerable web page.  
This vulnerability can have many consequences, like disclosure of a user’s session cookies that could be used to impersonate the victim, or, more generally, it can allow the attacker to modify the page content seen by the victims.

****

****

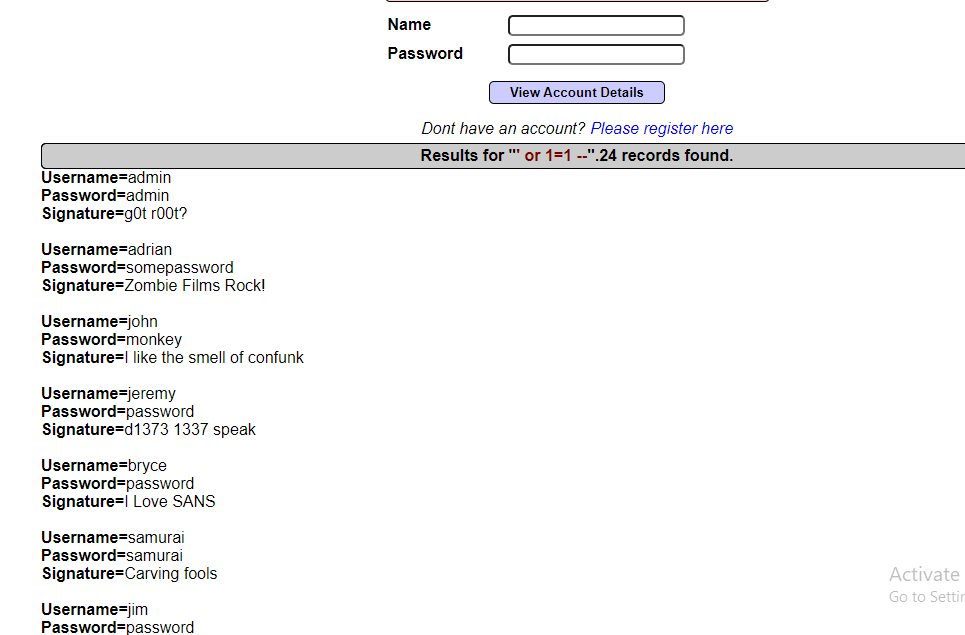
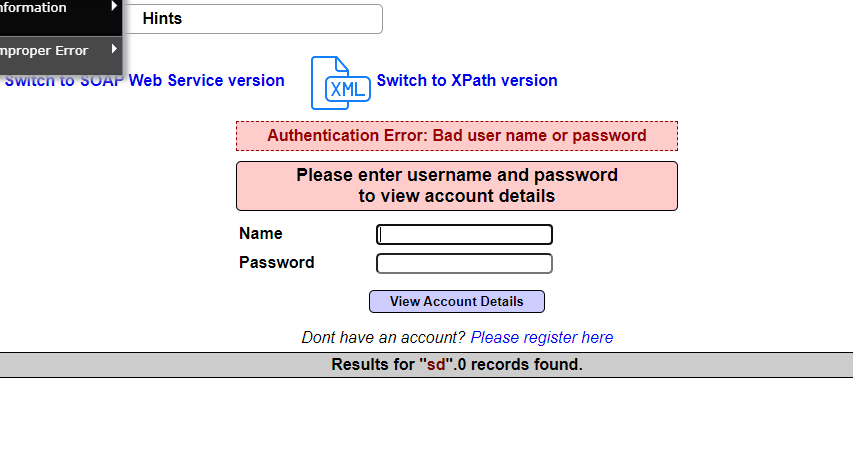
1. **Reflected POST HTML Injection**:

Reflected POST HTML Injection is a little bit more difficult. It occurs when a malicious **HTML** code is being sent instead of correct **POST** method parameters. For Example, we have a login form, which is vulnerable to **HTML** attack. Data typed in the login form is being sent with **POST** method.



1. **SQL injection**:

SQL injection also known as **SQLI**, is a common attack vector that uses malicious **SQL** code for backend database manipulation to access **information** that was not intended to be displayed. This **information** may include any number of items, including sensitive company **data**, **user** lists or private **customer details**.



**Injection Mitigation:**

Making use of Prepared Statements with Parameterized queries.

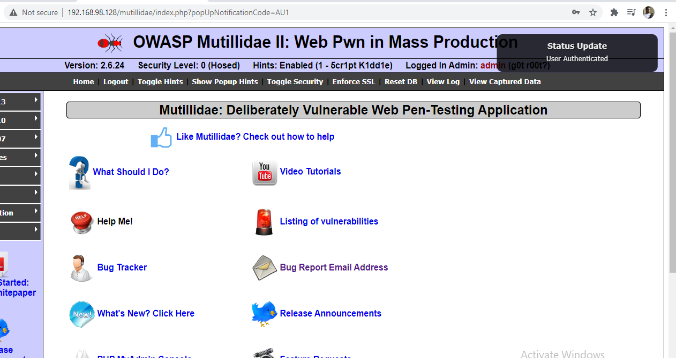
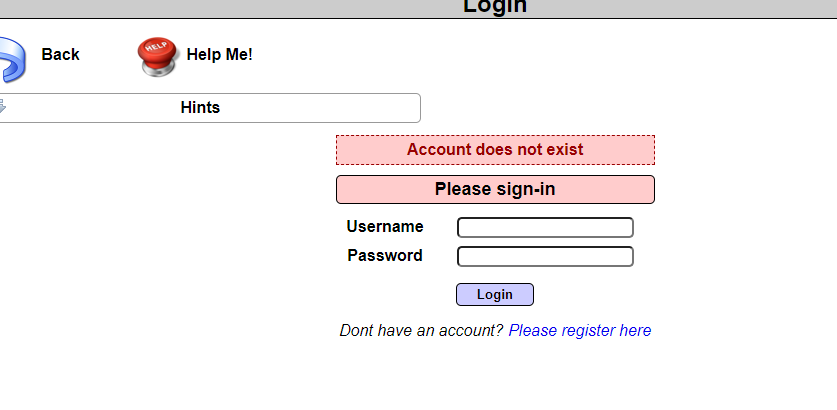
Making use of Stored Procedures.

Implement input validation and sanitization.

Make sure you are escaping all user-supplied input.

1. **Bypassing Authentication:**

An attacker gains access to application, service, or device with the privileges of an authorized or privileged user by evading or **circumventing** an **authentication** mechanism. The attacker is therefore able to access protected data without **authentication** ever having taken place.

****

**Broken Authentication Mitigation:**

Making use of captcha.

Reduce the number of tries for a particular user based on the session ID or the IP.

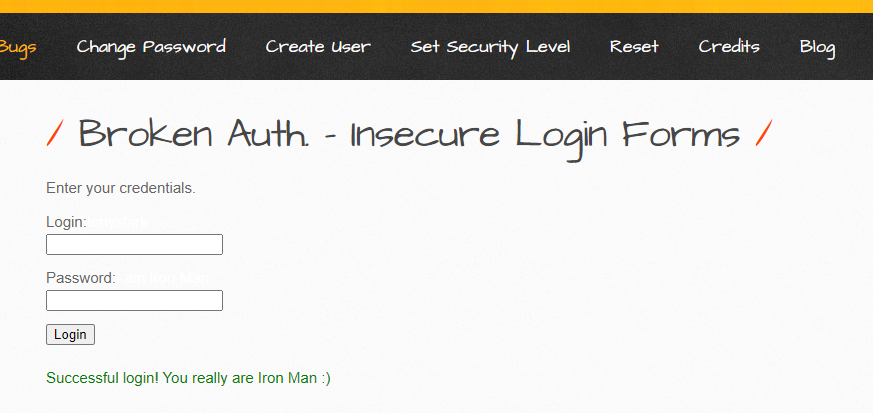
Blocking multiple requests coming from the same IP.

Making the admin login page inaccessible to the public.

Implement multi-factor authentication to prevent brute-forcing and credential theft.

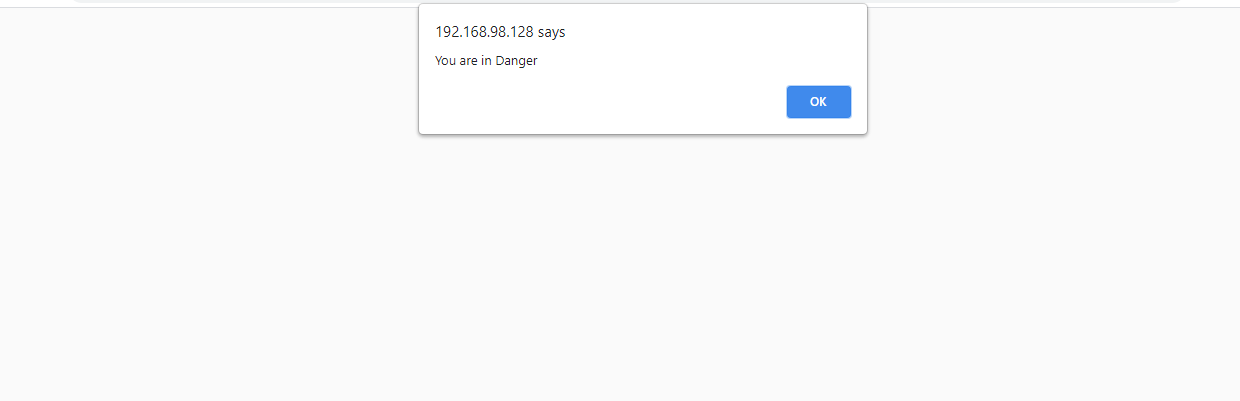
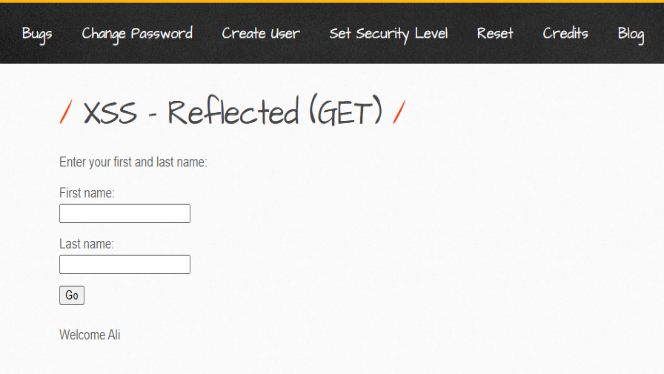
1. **Authentication is “broken”:**

**Authentication** is “**broken**” when attackers are able to compromise passwords, keys or session tokens, user account information, and other details to assume user identities. Due to poor design and implementation of identity and access controls, the prevalence of **broken authentication** is widespread.

****

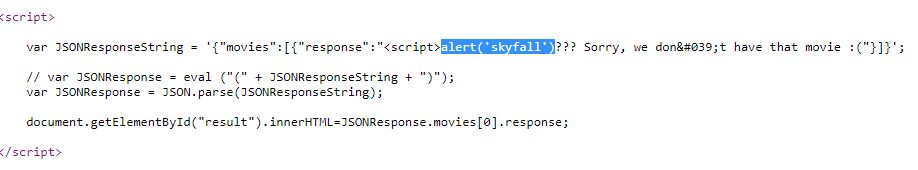
1. **XSS Reflected Get:**

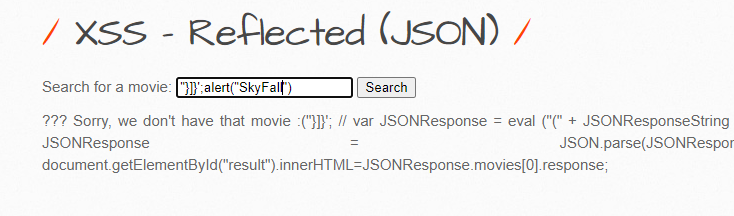
**Reflected XSS** attacks, also known as non-persistent attacks, occur when a malicious script is **reflected** off of a web application to the victim's browser. The script is activated through a link, which sends a request to a website with a vulnerability that enables execution of malicious scripts.

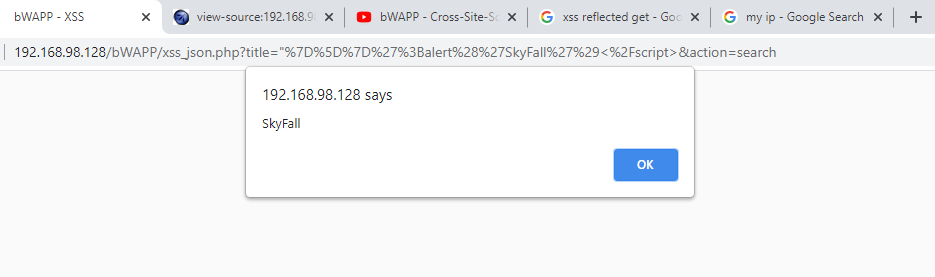
****

1. **XSS Reflected (JSON):**

**XSS** occurs when a user-manipulatable value is displayed on a web page without escaping it, allowing someone to inject Javascript or HTML into the page. ... Calls to Hash#to\_json can be used to trigger **XSS**.

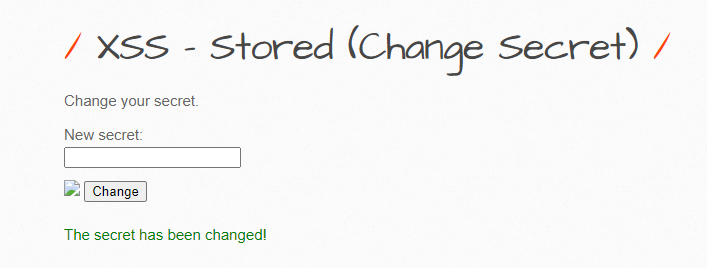
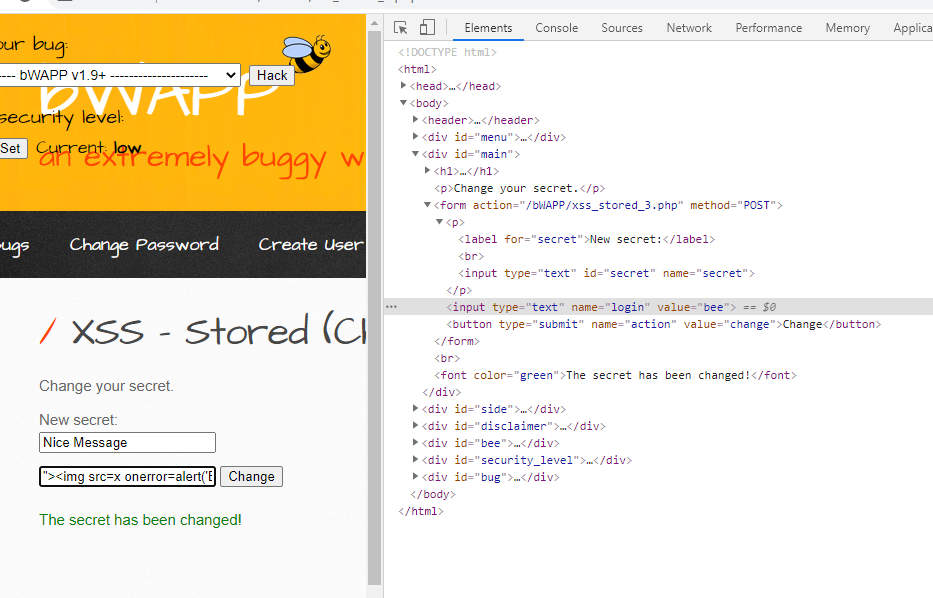
****

****

****

1. **XSS Stored Secret:**

**Cross site scripting** (**XSS**) is a common attack vector that injects malicious code into a vulnerable web application. ... **Stored XSS**, also known as persistent **XSS**, is the more damaging of the two. It occurs when a malicious script is injected directly into a vulnerable web application.

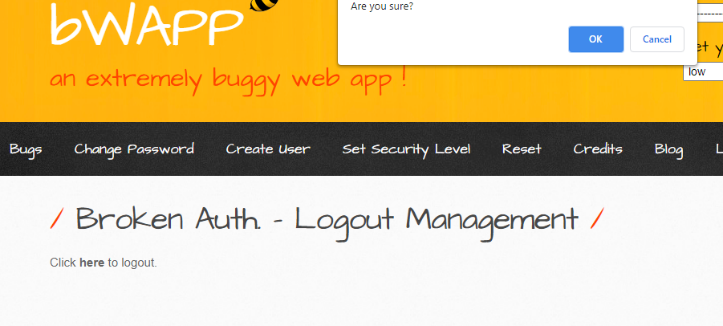
****

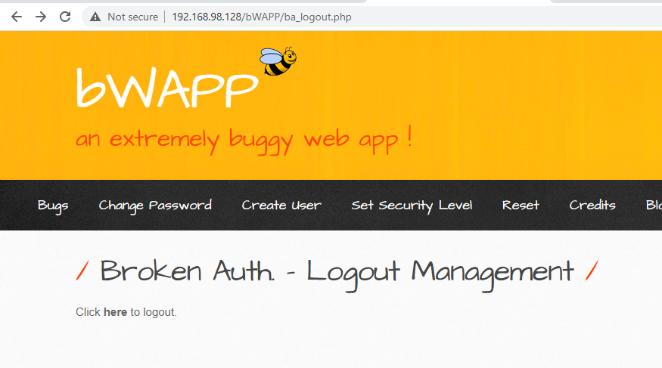
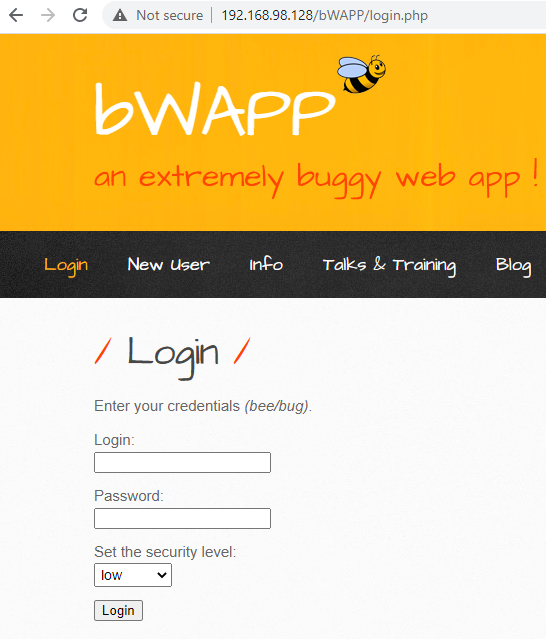
**Cross Site Scripting Mitigation:**

* We can encode the following characters with HTML entity encoding to prevent any execution of any form.
* & –> & amp;
* < –> & lt;
* ->& gt;
* ” –> & quot;
* ‘ –> &# x27;
* CSS encode and make sure it’s validated before Inputting untrusted data into HTML Style Property Values.
* Using frameworks like Ruby on Rails and React JS that escape XSS with ease.
* JavaScript encode Before Inputting untrusted data into JavaScript data values.
* HTML encode JSON values in an HTML context and read the data with JSON.parse.
* URL encode Before Inputting Untrusted Data into HTML URL Parameter Values.
* Implement Content Security Policy.
* Use HTTPOnly cookie flag.

1. **Broken Auth. - Logout Management:**

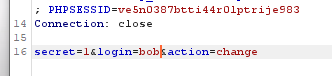
According to OWASP, **Broken Authentication** and Session **Management** was defined as 'Application functions related to **authentication** and session **management** are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users

****

****

1. **Insecure DOR (Change Secret):**

Simply a text box, asking for the new **secret** key. Now if we know the username of any other user then we can modify the request to make **changes** in someone else account whose account access we don't have

****