1.Weak Registration Implementation

we see login function in almost every website and when we signup then website sends verification link on our mail id and there is a link embedded in that mail so we have to copy that link and see if that is on http or on https. If then verification link is on http then it's a bug, but there is a condition in this.

Condition: when we open that link in our browser then account should be opened directly if we open that link and website ask credentials then company will not accept that bug.

So let's see how to create report for this bug...

Description:- The user registration process in the application is vulnerable due to weak implementation, which could allow an attacker to exploit the system by bypassing security mechanisms or manipulating user account creation. This can lead to unauthorized access, account takeover, or user data leakage.

Steps to reproduce:-

1. Open this url in browser — example.com/signup
2. An account verification link will be sent
3. Go to email inbox and open the email
4. Right-click on the link and copy the link
5. Paste the link in notepad/browser and check if it is on HTTP
6. Press enter and check if the account is opened or not.

Impact: Weak Registration Implementation could lead to data theft through the attacker's ability to manipulate data through their access to the application, and their ability to interact with other users, including performing other malicious attacks, which would appear to originate from a legitimate user.

2. Weak Password Reset Implementation

When we click on forgot password in website then it will send a link on our email id and if the link is on http then it is consider as bug, but there is a criteria for that..

Criteria: when we open that link in browser it must ask for new password directly.

There is one more feature on which we can test it on when we invite user then the invitation link that is sent over email must be on https it that link is on http then it's bug.

Now let's see how to create report for this bug...

Description:- When the password reset implementation is not implemented properly , the strength of the overall authentication process for the application is diminished. Tokens sent over HTTP, predictable reset tokens, and long expiry times create weak conditions for the password reset implementation.

Steps to reproduce:-

1. Go to forgot password page
2. Enter the registered email
3. Go to the email inbox
4. Right-click on the box and copy the link
5. Paste the link in the browser
6. Check if the link is on HTTP

Impact: This vulnerability could lead to data theft from the attacker's ability to manipulate data through their access to the application, and their ability to interact with other users.

3. Improper Cache-Control

So first of all what is cache-control and how it works. Fu*k it who cares this article is not about what, it is about how we can find it on a real website. If you are curious about what is cache-control then check this out.

How to find it: In website firstly login then go on a sensitive page like profile, password change etc. etc. and then logout from that page directly and after logout press back button in browser and if that sensitive page open without asking for credentials then it's a bug.

One more thing after pressing back button and if sensitive page opens directly and you can edit any data on that page then it's P3.

4. HTTP by default

If a domain doesn't have SSL certificate or SSL certificate expired or HSTS header not present and it's running on http and site is dynamic then it's a P4.

Condition: ***Only self hosted programs will accept this bug.***

Oh. you found this bug then it's time to make POC.

Description:-

The website is not fully protected by an SSL certificate. This could allow an attacker in a Man-in-the-Middle position to obtain usernames and passwords of users visiting the site.

Steps to reproduce:-

1. Copy the url http://example.com
2. Paste it in new tab and add a 's' in the domain
3. If it does not open on https, it is vulnerable

Impact:-

If a user were to visit this page from a public or shared network (eg, office, airport, library, etc.) and login into an account, a malicious user on the same network would be able to obtain that user's username and password by conducting a Man-in-the-Middle attack using Wireshark. This would allow the malicious user complete access to the user's account

5. Token is invalidated after use

There are various types of tokens that are used by a website such as reset password token, verification token, invite user token etc. if these types of token is not expired after use or you can use these tokens multiple times then it's a bug.

Criteria — after using verification token account should be directly opened without asking for credentials.

Time for creating POC for this..

Description:- Reset password token is not expired after single use.

Steps to reproduce:-

1. Open the URL https://site.com
2. Go to Forgot password page
3. Enter your email id and you will receive a reset link
4. Change the password multiple times using the same reset link
5. The password gets changed every time.

Impact:-

The attacker can reuse the reset token of the user and update the password which would lead to an account takeover

6. Broken Link Hijacking

In this bug if some link is broken or available for takeover then it's consider as a vulnerability. Let me explain it in simple words, Like there is a website on which there is company's social media account links are available let's say company deleted it's twitter account for twitter but the link is still available on the webpage and it's pointing to the twitter account that company deleted, Now let's assume that username of company twitter account is xxyyzz and after deleted this username is now available for takeover and you as a hacker created new twitter account by username of xxyyzz.

Now when you visit company website and you click on twitter handle and boom you got redirected on your twitter account that you created by username of xxyyzz.

Criteria — ***Only those broken links are accepted in this bug which are managed by the organization***

Time for poc...

Description:- Broken link hijacking (BLH) is a type of web attack. It exploits external links that are no longer valid. If your website or web application uses resources loaded from external URLs or points to such resources and these resources are no longer there (for example due to an expired domain), attackers can exploit these links to perform defacement, impersonation, or even to launch cross-site scripting attacks.

Steps to reproduce:-

1. Open the link https://www.website.com
2. Click on the social media icons like — Twitter / Facebook / Instagram, etc.
3. If not the account will not be made, it will return — PAGE NOT FOUND or ACCOUNT NOT FOUND
4. The attacker can create an account by the company's name.

Impact:-

An attacker can create an account on the social media platform with that username and impersonate/misuse the company name

7.Clickjacking

I know you all know about this bug if not then you can check this out on google or on some other articles..

But this bug is mostly out of scope on platforms and if you want a bounty on this bug then do clickjacking on sensitive page like profile, account, setting etc. Use the exploit given below.

```
<!DOCTYPE html>
    <html lang="en">
        <head>
            <meta charset="UTF-8">
            <title>Testing Clickjacking</title>
        </head>
        <body>
            <p>Checking if the page is vulnerable!</p>
            <iframe src="https://paste-your-url-here"height="700px" width="1100px" frame
border="0"></iframe>
        </body>
    </html>
```

Paste your url in iframe tag of this exploit. Now it's time to create poc..

Description:- Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top-level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Steps to reproduce:-

1. Open the site.com and go to the profile/account/settings page
2. Copy the profile URL and paste it on the clickjacking exploit and save it
3. Open the clickjacking file and the target.com will be vulnerable to Clickjacking and loads successfully into the iframe of the attacker
4. The attacker can perform a sensitive action

Impact:-

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker

Add icon Add cover Add comment

8.Delete Account Without Password

This bug is very easy as we can understand it by it's name. So, when we delete our account form a website and it asks password of the account but in some website when we click on delete account, our account is directly deleted without password then it is considered as a bug.

Now, I think you found this bug so time to create it's report...

Description:- The removal of an account is one of the sensitive parts of a web application that needs to protect, therefore deleting an account should validate the authenticity of the user.

Steps to reproduce:-

1. Visit the website and login into your account.
2. Go to the profile/settings section.
3. A delete account button will be displayed.
4. Click on delete button and your account is successfully deleted.

Impact:-

The target doesn't verify the request with a Valid OTP or password before triggering Right to Access/Deletion & allows an attacker to delete User Accounts without user interaction.

9.SPF and DMARC Record

So, let's talk about the bug here, we usually found company email on their website and if the SPF and DMARC record is not published for their mail id then it's vulnerable to email spoofing attacks.

How to check if SPF and DMARC record is published or not? Check it here

SPF record — https://www.kitterman.com/spf/validate.html

DMARC record — https://mxtoolbox.com/

Criteria: These bugs may be out of scope on platform so read full scope before submitting.

Time to create report....

For SPF....

Description:- The Sender Policy Framework (SPF) is an email authentication protocol and part of email cybersecurity used to stop phishing attacks.

Steps to reproduce:-

1. Visit — https://www.kitterman.com/spf/validate.html
2. Enter the domain name — target.com and hit get SPF Record
3. The domain name will show No valid SPF record found

Impact:-

Spammers can forge the "From" address on email messages to make messages appear to come from someone in your domain. If spammers use your domain to send spam or junk email, your domain quality is negatively affected. People who get the forged emails can mark them as spam or junk, which can impact authentic messages sent from your domain.

For DMARC...

Description:- DMARC (Domain-based Message Authentication, Reporting, and Conformance) is an email authentication protocol. It is designed to give email domain owners the ability to protect their domain from unauthorized use, commonly known as email spoofing.

Steps to reproduce:-

1. Visit — https://mxtoolbox.com
2. Enter the domain name — target.com and hit go
3. The domain name will show No DMARC Record found

Impact:-

Spammers can forge the "From" address on email messages to make messages appear to come from someone in your domain. If spammers use your domain to send spam or junk email, your domain quality is negatively affected. People who get the forged emails can mark them as spam or junk, which can impact authentic messages sent from your domain.

10. Content Spoofing/Test Injection/External Authentication Injection

This bug is simple like on any auth pages when we make changes in url and it get reflects in client side then it a bug, for example there is a url like example.com/login.php?error=access-denied and you changed this error to example.com/login.php?error=you%20are%20hacked then it's a P4.

So, I think you got the point, great it's time to make poc..

Description:- Content Spoofing allows the end user of the vulnerable web application to spoof or modify the actual content on the web page. This presents the user with a modified page under the context of the trusted domain. This attack is typically used as, or in conjunction with, social engineering because the attack is exploiting a code-based vulnerability and a user's trust.

Steps to reproduce:-

1. Go to example.com
2. Then just change above url like this https://example.com/wp-login.php?error=access_denied to https://example.com/wp-login.php?error=you%20are%20hacked
3. Click enter and the massage got reflected on the page.

Impact:- The website it rendering the URL data to the client side of the website which can help to trick the user to input the data elsewhere.

11.Failure to invalidate session — On the password reset/change

To test this bug first of all open two browsers or one browser on incognito tab so there is no cookie exchange happens. Then login to you account in these tabs/browsers using same account. Now, change your account password or reset your account password and check second browser/tab if your second tab/browser account gets log out then then it's not bug but if you still log in to you account after password reset/change it means site's session is not terminating after password reset or change and it's a P4 bug.

Great, you got another p4 bug let's make poc...

Description:- Failure to invalidate a session after a password change is a vulnerability which allows an attacker to maintain access on a service. Most users have the expectation that when they reset their password, no one else can access their account.

Steps to reproduce:-

1. Create an account on example.com
2. Login using credentials in 2 browsers
3. Open the profile/settings.
4. Go to Change password and change the password in Browser 1
5. Visit Browser 2 and edit the profile data (name/contact no/profile picture) and click on save.
6. Refresh the page once and the data will be changed

Impact:- This vulnerability can lead to reputational damage and indirect financial loss to the company as customers may view the application as insecure.

12. No rate limit on report for other user comment

This bug is also consider as P4 bug. This bug is simple as you can understand this by it's name. Like lot's of website have comment section and they have a specific feature of report other user comment if you found that sensitive.

In order to exploit this feature let's say there is no rate limit on that and the comment got deleted after a specific number of reports to that comments, now you can capture the report comment request to Burpsuite and send it to intruder and start attack and after a number of request you will se that the comment got deleted.

POC Time... :)

Description:- The "report comment" functionality on the platform allows users to report comments that may be offensive or inappropriate. However, there is no rate limiting on the report request, which enables an attacker to automate the reporting process and reach the threshold number of reports required for comment removal.

Steps to reproduce:-

1. Visit page https://example.com/blog/page4/report-comment?comment_id=33
2. Report this comment and capture request in Burpsuite
3. Send this request in burpsuite intruder and start attack
4. After few minutes the user comment got deleted

Impact:- The absence of rate limiting on the "report comment" feature allows attackers to automate reporting and remove legitimate comments without genuine user input. This can lead to unauthorized censorship, negatively affect user experience, and erode trust in the platform's moderation capabilities

Steps to Exploit:

1. **Identify the "Report Comment" Feature:**
   - Navigate to a page with a comment section, such as `https://example.com/blog/page4/report-comment?comment_id=33`.
   - Use the feature to report a comment and note the functionality.
2. **Capture the Request:**
   - Use **Burp Suite** to intercept the HTTP request generated when reporting a comment.
3. **Send Request to Intruder:**
   - Forward the captured request to Burp Suite's **Intruder** tool.
   - Configure Intruder to repeatedly send the same report request.
4. **Launch the Attack:**
   - Start the Intruder attack and monitor the server responses.
   - After sending a sufficient number of requests, you will notice the reported comment gets deleted.

13.Email Verification Bypass

Case 1:- Unprotected Account Activation URLs

Some websites use predictable URLs for email verification, like `example.com/verify?user_id=1234`. If these URLs do not include a sufficiently unique token or other secure identifier and simply rely on a user ID, an attacker could attempt to access `example.com/verify?user_id=1234` directly, and by this you are able to varify any account without email verification link.

Case 2:- Predictable Verification Tokens

If the website uses easily guessable or incremental tokens, like `example.com/verify?token=abcd1234` where the token pattern is predictable, you can force browse through possible tokens to activate other users' accounts. This case is rare to find as randomization of token is done by almost every website.

Case 3:- Bypassing Verification Status Checks

In some cases, you can directly access sections of the site meant for verified users if the site doesn't enforce verification checks on restricted actions. For example, if the site fails to check a user's verification status at each sensitive action, attackers could simply skip the verification step and proceed to use the account. There is also another way for this if the site is using "is_varified: false" then you can try bypassing that also just by "is_varified: true"

Case 4:- Direct Access to the Verified User Area

Sometimes when you signup to a website and it won't allow you to access any features or anything and asking for email verification then you can try some of predictable URL paths (e.g., `/dashboard` or `/profile`) that you can directly access by typing the URL in the browser. This might gives you the access of website without verification.

Case 5:- Verification email hijacking

Sometimes, when you change your email address on a website, it will send a new verification link to confirm the change. However, if you don't verify it right away and then change your email again to a different address you want to hijack, the previous verification link might still work. By clicking the first link, you might be able to verify the new email address without needing access to it. This allows you to hijack the target email on the account if the system hasn't invalidated the old verification link.

14. Exif Geo Location

This bug is consider as P4 but sometimes as per the impact, this can be consider as P3 vulnerability.

Let's dive into it's details like what this bug is and how to find it?

In almost every website, there is an upload image feature available, it may be profile picture or other type of upload. If website/webserver is not stripping exif metadata of an image uploaded by the user then it's a bug (P4). There are two different conditions in this like if the image is visible to more then one person then it's a P4 called Exif Geolocation Manual Enumeration. While if the image is visible publicly like Instagram profile picture is visible to everyone or the picture is uploaded by developer and visible to all then the impact of this vulnerability is consider as P3 (Exif Geolocation Automatic Enumeration).

How to check Metadata of Image : Go to this website (https://jimpl.com/) and paste the picture/URL.

As always: POC time.... :)

Description:- When a user uploads an image in example.com, the uploaded image's EXIF Geolocation Data does not get stripped. As a result, anyone can get sensitive information of example.com users like their Geolocation, their Device information like Device Name, Version, Software & Software version used, etc.

Steps to reproduce:-

1. Visit example.com 2. Go to the Upload option on the website 3. Upload the image with EXIF metadata. 4. Right click on the image and download it. 5. Visit https://jimpl.com 6. Upload the downloaded image and check for sensitive data.

Impact:- This vulnerability is CRITICAL and impacts all the example.com customer base. This vulnerability violates the privacy of a User and shares sensitive information of the user who uploads an image on example.com or any of the example.com instances.

15. Broken authentication — Failure to invalidate session on logout

This is also another simple vulnerability in which website fails to validate user's session even after they logout.

To find this vulnerability first login to your account in two different tabs of same browser. Then logout from one tab and if your account is automatically logout from the second tab of same browser then it's not a bug but if the session still persist in another tab even after you logout from the first tab then try to change some data in the account and if you successfully updated some data in your account then it a bug.

Criteria: ***Bugcrowd didn't consider this vulnerability as P4***.

Let's create POC :)

Description:- The application fails to invalidate a user's session on logout, leaving the account vulnerable to session hijacking. An attacker may compromise a user's session then be able to change the password of the account and lock out the legitimate user.

Steps to reproduce:-

1. Go to the URL — example.com
2. Open the same account on two different tabs on the same browser — Browser A
3. Click on the Logout from one tab — TAB A
4. Once the session is terminated, go to the second tab (TAB B) and update some data and save it
5. Post changing the data, click on the refresh button.
6. The data will be updated.

Impact:- This vulnerability can lead to reputational damage and indirect financial loss to the company as customers may view the application as insecure as the session is still running after logout.

16. HTML Email Injection

This is a common type of bug available in lot's of websites, i discovered this bug in lot's of programs even some of these are hosted on platforms like hackerone, bugcrowd etc.

When you signup in a website then it send a verification link on email address and in that email there is something like Hi <Yourname> , so to find HTML email injection you have to put HTML payload in First name and Last name while signing up in a website as only this data reflects in the email sent by the website.

So you have to put payload like this given below. (Make your own image payload), and this payload will fire up in your email and if website doesn't allows you to put HTML content in first and last name then try to bypass it using burpsuite if there is client side validation is placed in website.

```
CopyPayload- <img src="https://t.ly/DXvtn">
```

Now let's make POC :)

Description:- HTML injection is a vulnerability in which attacker provided input is rendered as HTML. HTML injection in emails can lead to attackers phishing users from a legitimate email address.

Steps to reproduce:-

1. Go to the URL https://site.com
2. Create an account with HTML payload in first name and last name.
3. Generate a reset password/verification email
4. The image will be executed in the verification/reset password email sent by the website.

Impact:- This vulnerability can lead to the reformatting/editing of emails from an official email address, which can be used in targeted phishing attacks. This could lead to users being tricked into giving logins away to malicious attackers.

17.Origin IP disclosure leads to WAF Bypass

This is an interesting and impactful bug. If the application is using some type of web application firewall only in that case this bug is applicable otherwise this is out of scope.

Sometimes companies use WAF but they misconfigured it or due to some other reasons the origin IP of the website is disclosed, so in order to find out the origin IP you have to remember three steps –

1. Check the application is using some short of WAF, check it via some tools like wappalyzer, wafWOOF, DNSlytics, etc.
2. Try to find origin IP on Shodan, Censys etc. or use tools available on Github.
3. When you found out the IP address then open the IP in browser and the content hosted on the IP should match to the content hosted on the domain like the page of IP and domain must match, the second criteria is that — do an IP lookup via https://whatismyipaddress.com/ip-lookup and the hostname of IP must match to domain name.

After all these steps you are good to go...

So, let's see the POC :)

Description:- By using these IP address as a resolver instead of the intended addresses I'm able to access the service without going through the WAF, thus I'm able to forward unfiltered payloads to the service, as well as avoiding the common protections offered by Cloudflare, also being able to perform crippling denial-of-service towards the origin.

Steps to reproduce:-

1. Enumerate the subdomains of https://target.com
2. Check the firewall used by the tool DNSlytics or WafW00f
3. To get origin IP — Use sites like : https://search.censys.io/ , https://www.shodan.io/
4. Do a IP lookup of the IP
5. Enter the IP on the URL and hit enter to check if the IP loads the subdomain name

Impact:- Cloudflare bypasses can have a significant impact, as any adversary is now able to communicate with the origin server directly, enabling them to perform unfiltered attacks (such as denial-of-service), and data retrieval