ISSN: 2454-9940

INTERNATIONAL JOURNAL OF APPLIED
SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940

www.ijasem.org

https://zenodo.org/records/11632200

Vol 18, Issue 2, 2024

"Securing the Web: Mitigating Common Vulnerabilities and Enhancing Web Application Security"

Ibrahim Khaleel Khan (160520737002)[1], Md.Ibtesaam Ali Shah (160520737306) [2], Mohd Osama (160520737059) [3] Mr.Arshad Hussain [4]

Mail id : Ibrahimkhaleel2737@gmail.com, Mail Id : mohammedibtesaam101@gmail.com, Mail Id : Osamajamalpte@gmail.com,

[1,2,3] B.E Student, Dept. of Information Technology, ISL Engineering College

[4]Assistant Professor (PhD), Dept. of Information Technology, ISL Engineering College

ABSTRACT:

Web applications provide a plethora of services that are fundamental to modern society, ranging from online shopping to social networking. They are often the targets of cyber attacks because of the amount of sensitive information they handle and the complexity of the system. Web vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) are significant causes of the widespread problem of insecure online applications and user data. Provides an overview of the most common internet security flaws, their consequences, and the current approaches to preventing them. We discuss the value of secure coding techniques, regular security audits, and security frameworks and tools in protecting online applications from potential attacks. The promise of emerging technologies like machine learning and artificial intelligence to enhance the safety of web apps motivates us to learn more about them. Developers and organizations may protect themselves and their online applications from potential threats by being informed about web vulnerabilities and implementing appropriate security measures.

INTRODUCTION:

Web applications provide a plethora of services that are fundamental to modern society, ranging from online shopping to social networking. They are often the targets of cyber attacks because of the amount of sensitive information they handle and the complexity of the system. Web vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) are significant causes of the widespread problem of insecure online applications and user data. Provides an overview of the most common internet security flaws, their consequences, and the current approaches to preventing them. We discuss the value of secure coding techniques, regular security audits, and security frameworks and tools in protecting online applications from potential attacks. The promise of emerging technologies like machine learning and artificial intelligence to enhance the safety of web apps motivates us to learn more about them. Developers and organizations may protect themselves and their online applications from potential threats by being informed about web vulnerabilities and implementing appropriate security measures.

Literature Survey

Online application security is becoming an increasingly pressing issue for academics and practitioners, according to a literature review on online vulnerabilities. Web vulnerabilities are common and may compromise users' security and privacy, according to much research.

Attackers often take advantage of online application weaknesses to run malicious SQL queries, a technique known as SQL injection. The most prevalent vulnerability in online applications, impacting several websites, was determined to be SQL injection by Halfond et al. (2006).

Attackers may insert harmful code onto online pages seen by other users using cross-site scripting (XSS), another common web vulnerability. Researchers Huang et al. (2014) found that cross-site scripting (XSS) attacks are common and dangerous for online application security.

An attacker may use cross-site request forgery (CSRF) to cause users to do unauthorized activities even while they are authenticated on a website. Researchers Barth et al. (2008) found that cross-site request forgery (CSRF) vulnerabilities are common in online applications and might be used by malicious actors.
Scientists have come up with a number of ways to lessen the impact of these vulnerabilities. One way to make sure web apps aren't vulnerable is to adopt safe coding standards like encrypting output and validating input. Another strategy involves identifying and fixing vulnerabilities in preexisting web applications using automated technologies like static and dynamic analysis tools.
The literature review as a whole shows that security professionals are still quite worried about online vulnerabilities. To safeguard online applications against these dangers, researchers and industry professionals are collaborating to create mitigation measures.

EXISTING SYSTEM:

Standard procedures for handling online vulnerability management often include secure coding techniques, automated vulnerability assessment technologies, and human code reviews. Expert developers do manual code checks to find SQL injection, cross-site scripting, and cross-site request forgery issues. Manual reviews are successful, but they take a lot of time and could miss certain vulnerabilities, particularly in complicated and big online apps.In order to automatically find common vulnerabilities in online applications, automated vulnerability scanning technologies are used. In a short amount of time, these tools can check a web app for security flaws and provide a report detailing what needs fixing. Nevertheless, human oversight is necessary for validation and verification since automated systems might produce false positives or negatives. In order to avoid vulnerabilities in online applications, it is vital to use secure coding standards including input validation, parameterized queries, and output encoding. It is recommended that developers adhere to safe coding rules and best practices in order to minimize the chances of creating vulnerabilities when developing.

DRAW BACKS:

Manual Work: Due to their inefficiency and length, manual code reviews are not a viable option for web applications with a big user base or for projects with regular code updates.

Coverage Limitations: Some vulnerability, particularly those requiring complicated or context-dependent analysis, may evade automated vulnerability detection techniques.

PROPOSED SYSTEM:

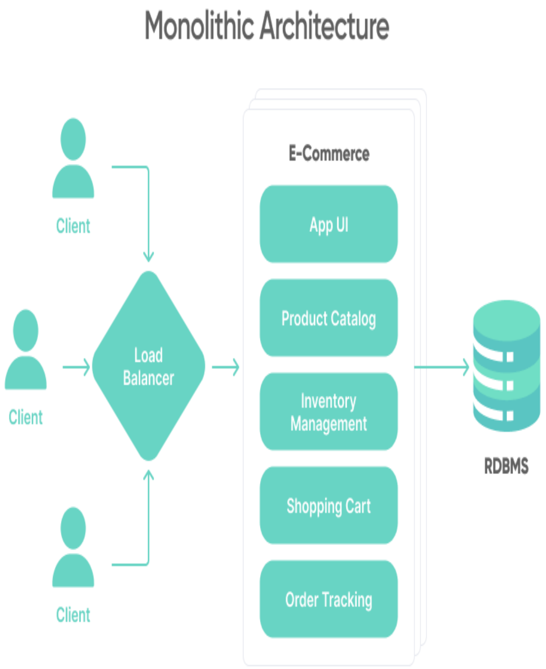
By using cutting-edge technology and methodologies, the suggested system for controlling online vulnerabilities seeks to rectify the shortcomings of the current system. Combining automated vulnerability detection tools with machine learning algorithms is a crucial part of the proposed approach. The technology is able to decrease the number of false positives and negatives and increase the accuracy of vulnerability detection thanks to machine learning. To improve their ability to detect possible vulnerabilities in web application code, machine learning models may be trained on big datasets of existing vulnerabilities. This allows them to discover patterns and anomalies more effectively. Continuous security testing and monitoring is another important part of the system that is being suggested. The technology keeps an eye on online apps for security flaws and notifies developers instantly, eliminating the need for periodic scans or manual assessments. This preventative method lessens the likelihood of exploitation by allowing vulnerabilities to be found and fixed quickly. Furthermore, developer training and safe coding standards are highlighted in the suggested system. Secure coding standards and automated code analysis tools are only two of the resources made available to developers to assist them in writing code that is free of vulnerabilities. Web security vulnerabilities and how to effectively protect against them are topics that are often covered in training sessions and seminars.

ADVANTAGES:

The suggested solution may reduce the number of false positives and negatives and increase the accuracy of vulnerability identification when compared to more conventional automated techniques by making use of machine learning.

Quick Detection and Resolution of Security Incidents: The suggested approach provides real-time vulnerability monitoring of online applications.

Reducing the Risk of Exploitation: By continuously testing and monitoring security, a proactive strategy may be taken to manage online vulnerabilities.



SYSTEM REQUIREMENTS

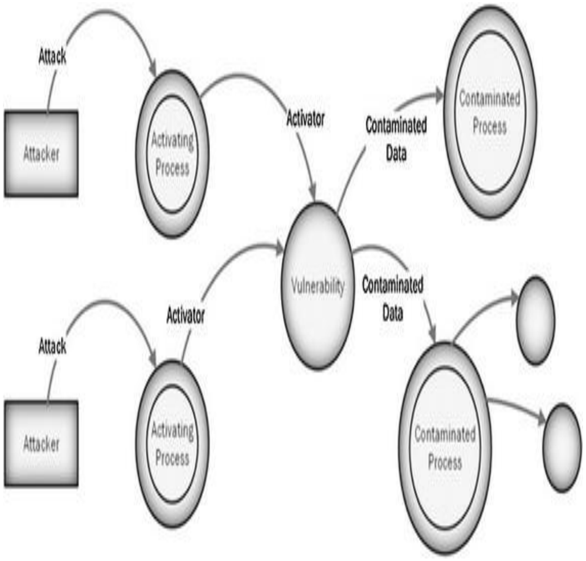
- **H/W System Configuration:-**

Processor	-	Pentium –IV
RAM	-	4 GB (min)
Hard Disk	-	20 GB
Key Board	-	Standard Windows Keyboard
Mouse	-	Two or Three Button Mouse
Monitor	-	SVGA

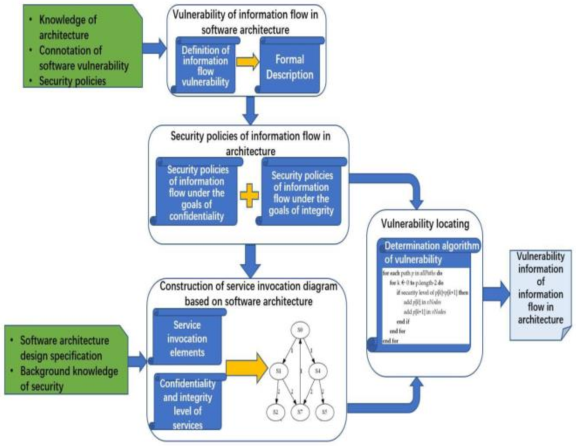
SOFTWARE REQUIREMENTS:

Operating system	:	Windows 7 Ultimate.
Coding Language	:	Python.
Front-End	:	Python.
Back-End	:	Django-ORM
Designing	:	Html, css, javascript.
Data Base	:	MySQL (WAMP Server).

DATA FLOW



FLOW CHAT:


Knowledge of architecture
Connotation of software vulnerability
Security policies
Vulnerability of information flow in software architecture
Definition of information flow vulnerability
Formal Description
Security policies of information flow in architecture
Security policies of information flow under the goals of confidentiality
Security policies of information flow under the goals of integrity
Vulnerability locating
Determination algorithm of vulnerability
Vulnerability information of information flow in architecture
Construction of service invocation diagram based on software architecture
Software architecture design specification
Background knowledge of security
Service invocation elements
Confidentiality and integrity level of services
S0
S1
S4
S2
S7
S5

```
for each path p in allPaths do
  for k ← 0 to p.length-2 do
    if security level of p[k]>p[k+1] then
      add p[k] in cNodes
      add p[k+1] in cNodes
    end if
  end for
end for
```

MODULES:

This project consists of two modules

Module for System Administrators: Username and password for system administrators are admin and admin. A new user account may be activated by the admin after login. Admins have full access to the CSRF list, user information, and POST and GET request details.

Users may create an account from inside the app. In order to scan URLs for vulnerabilities; users must first activate their accounts. After that, they may connect in to the system and input any URL along with a depth number. The system will then apply the Mitch procedure to determine whether the URL is susceptible or not. The user is able to train ML algorithms using various ways and thereafter determine measures such as accuracy.

You need to install python 3.7.2 and all the packages listed in requirements.txt before you can start the project. To build a database in MySQL, first install the database software. Then, enter the MySQL console and copy and paste the contents of the "database.txt" file.

BLOCK CHAIN:

You may have heard the phrase "block chain technology" tossed about often in recent years, most likely in reference to digital currencies like Bit coin. Actually, "what is block chain technology?" can be a question you're asking. Block chain seems to be a cliché, but only in a theoretical sense, as it lacks a clear and simple definition for the average person. A thorough explanation of "what is block chain technology?"—including its applications, inner workings, and growing importance in the digital realm—is essential.

You should educate yourself on this developing technology in order to be future-proof, as block chain is only going to become better and more accessible. This is the best place for someone new to block chain to learn the basics. "What is block chain technology?" is a question that this article teaches you how to answer. The course will also teach you the ins and outs of block chain technology, its significance, and how to build a career in this area.

Results

Here, we provide a strategy for finding security holes in web applications by use of Machine Learning (ML). Web applications are notoriously difficult to analyze because of how diverse they are and how commonplace bespoke programming approaches are. ML's ability to use manually labeled data to incorporate human knowledge of web application semantics into automated analysis tools makes it a valuable tool for web application security. We developed Mitch, the first machine learning (ML) solution for black-box CSRF vulnerability identification, using our technique. Thanks to Mitch, we were able to find three new CSRFs in production software and thirty-five new CSRFs on twenty big websites.

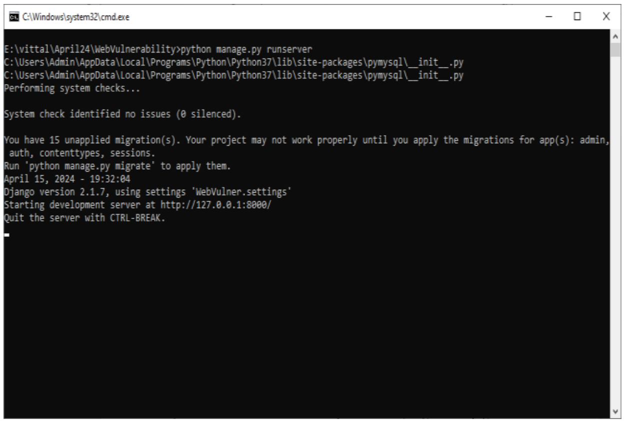
Two distinct parts make up this undertaking. Module for System Administrators: Username and password for system administrators are admin and admin. A new user account may be activated by the admin after login. Admins have full access to the CSRF list, user information, and POST and GET request details.

Brand-New Individual Users have the option to enroll directly inside the app.

In order to scan URLs for vulnerabilities, users must first activate their accounts. After that, they may connect in to the system and input any URL along with a depth number. The system will then apply the Mitch procedure to determine whether the URL is susceptible or not. The user is able to train ML algorithms using various ways and thereafter determine measures such as accuracy. You need to install python 3.7.2 and all the packages listed in requirements.txt before you can start the project. To build a database in MySQL, first install the database software. Then, enter the MySQL console and copy and paste the contents of the "database.txt" file.

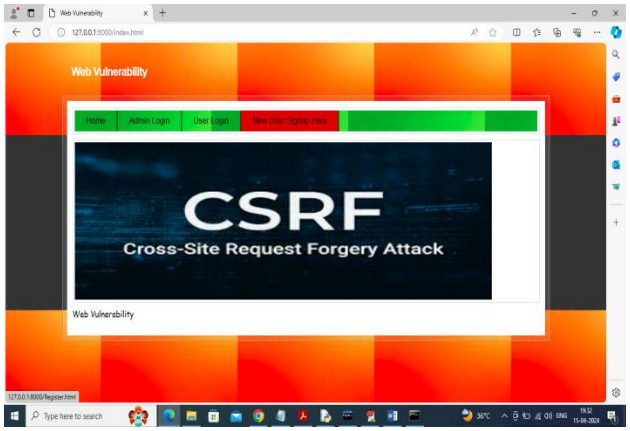
SCREEN SHOTS

To run project double click on 'run.bat file to start python server and get below page

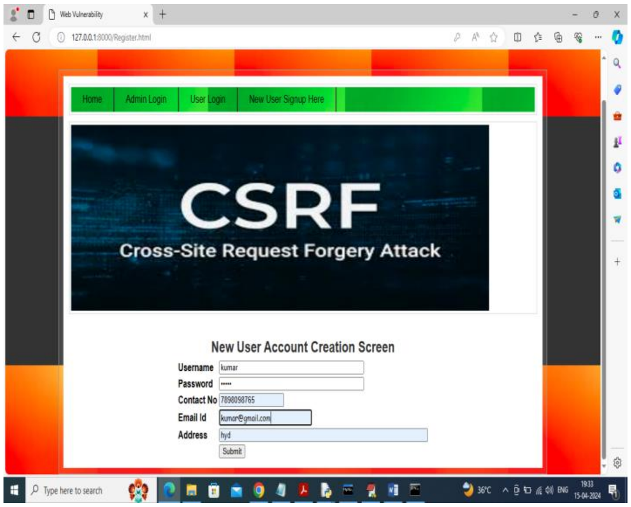


```
E:\vittal\April24\WebVulnerability>python manage.py runserver
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\pymysql\__init__.py
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\pymysql\__init__.py
Performing system checks...

System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
 auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
April 15, 2024 - 19:32:04
Django version 2.1.7, using settings 'WebVulner.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

In above screen python server started and now open browser and enter URL as



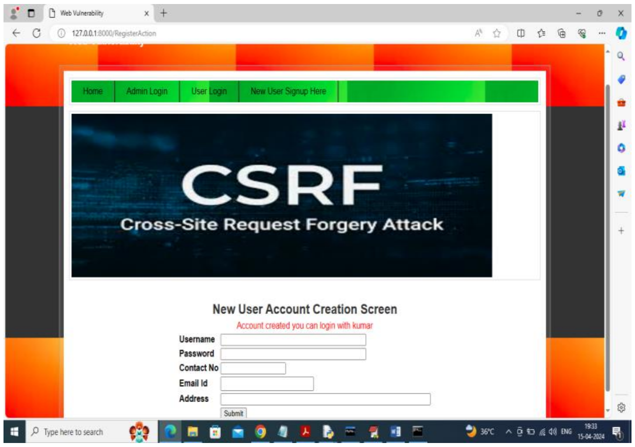
In above screen click on 'New User Sign up' link to get below page



New User Account Creation Screen

Username kumar
Password •••••
Contact No 7898098765
Email Id kumar@gmail.com
Address hyd

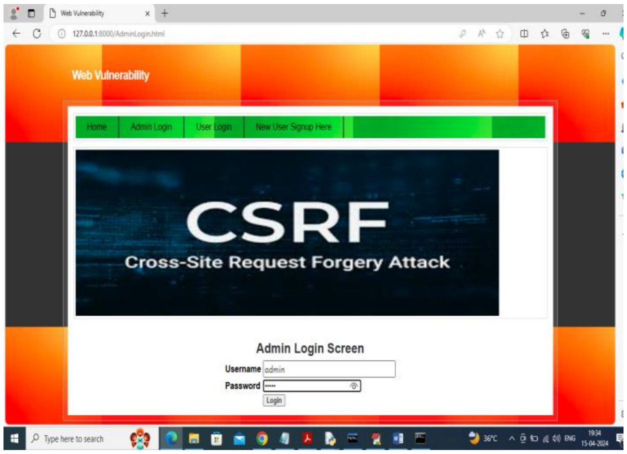
In above screen user is entering sign up data and then press button to get below page



New User Account Creation Screen

Account created you can login with kumar

In above screen user account created and now login as admin to activate user account

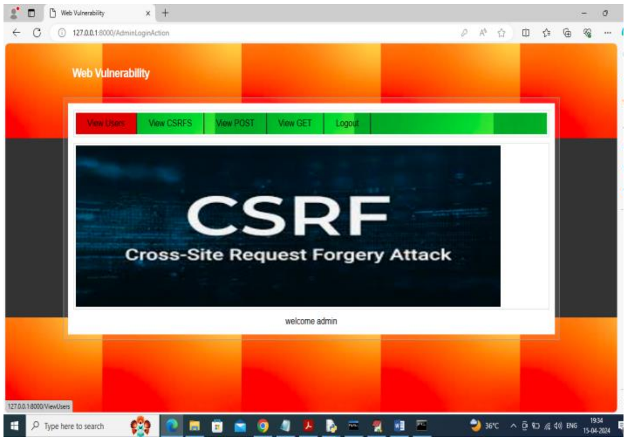


Admin Login Screen

Username admin

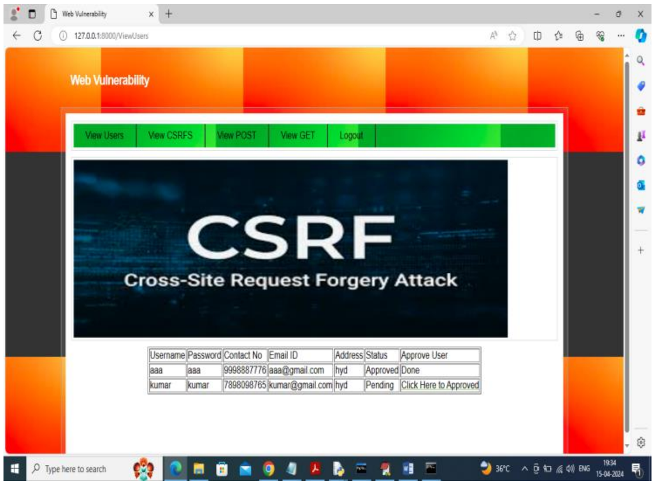
Password ••••

In above screen admin is login and after login will get below page



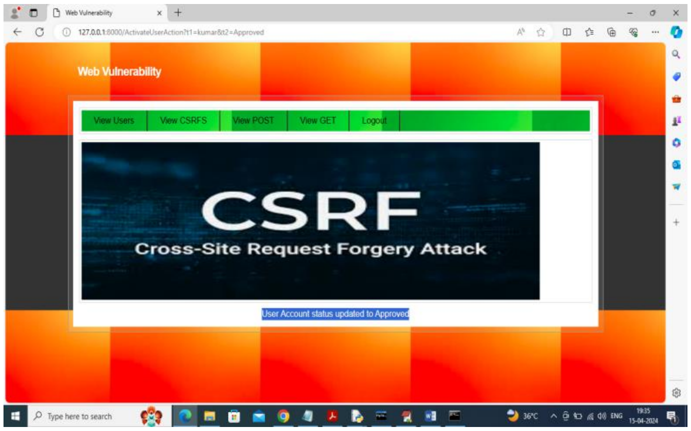
welcome admin

In above screen admin can click on 'View Users' link to get below page



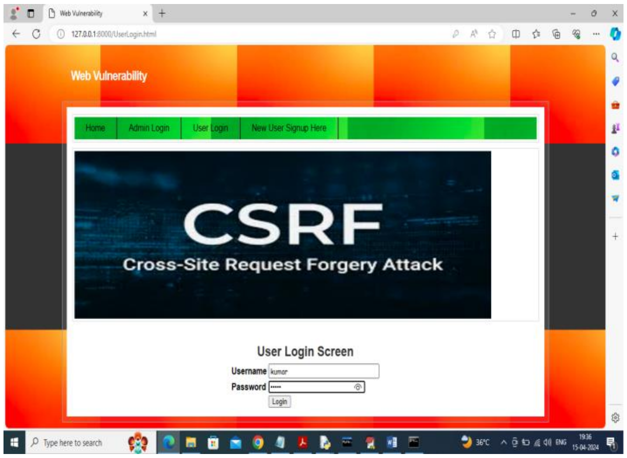
Username	Password	Contact No	Email ID	Address	Status	Approve User
aaa	aaa	9998887776	aaa@gmail.com	hyd	Approved	Done
kumar	kumar	7898098765	kumar@gmail.com	hyd	Pending	Click Here to Approved

In above screen admin can view list of accounts and can click on 'Click Here to Approved' link on pending accounts to approve users and get below page


Web Vulnerability
127.0.0.1:8000/ActivateUserAction?t1=kumar&t2=Approved
Web Vulnerability
View Users
View CSRFS
View POST
View GET
Logout
CSRF
Cross-Site Request Forgery Attack

User Account status updated to Approved

In above screen user account approved and now logout and login as user

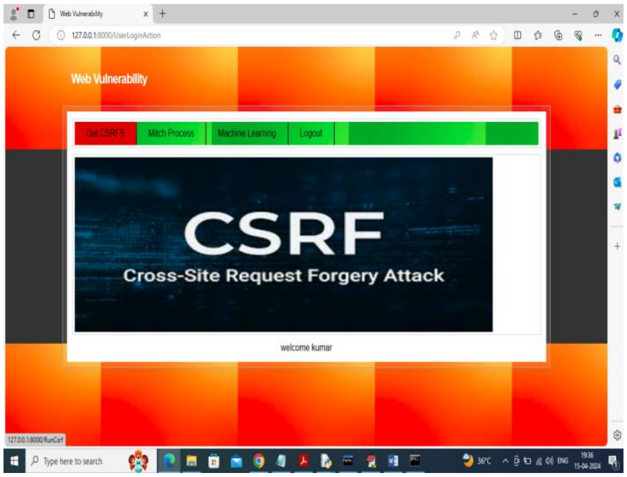


User Login Screen

Username kumar

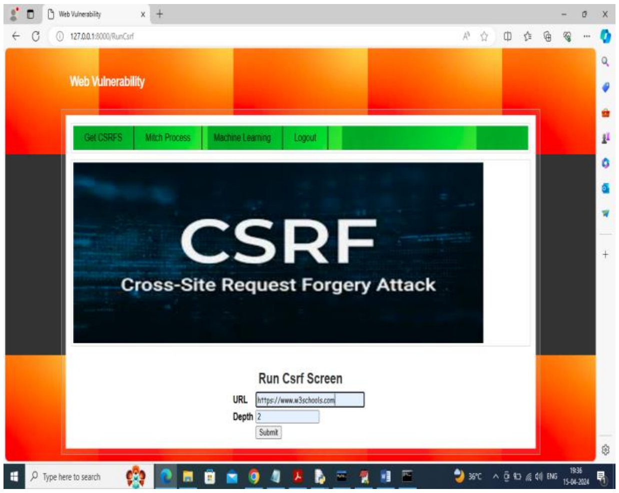
Password •••••

In above screen user is login and after login will get below page



welcome kumar

In above screen user can click on 'Get CSRFS' link to get below page

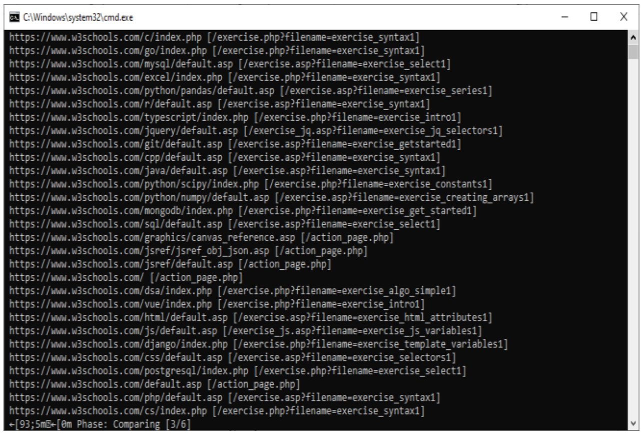


Run Csrf Screen

URL https://www.w3schools.com

Depth 2

In above screen enter URL and depth value and then click on button to get below URL scanning output



```
https://www.w3schools.com/c/index.php [/exercise.php?filename=exercise_syntax1]
https://www.w3schools.com/go/index.php [/exercise.php?filename=exercise_syntax1]
https://www.w3schools.com/mysql/default.asp [/exercise.asp?filename=exercise_select1]
https://www.w3schools.com/excel/index.php [/exercise.php?filename=exercise_syntax1]
https://www.w3schools.com/python/pandas/default.asp [/exercise.asp?filename=exercise_series1]
https://www.w3schools.com/r/default.asp [/exercise.asp?filename=exercise_syntax1]
https://www.w3schools.com/typescript/index.php [/exercise.php?filename=exercise_intro1]
https://www.w3schools.com/jquery/default.asp [/exercise_jq.asp?filename=exercise_jq_selectors1]
https://www.w3schools.com/git/default.asp [/exercise.asp?filename=exercise_getstarted1]
https://www.w3schools.com/cpp/default.asp [/exercise.asp?filename=exercise_syntax1]
https://www.w3schools.com/java/default.asp [/exercise.asp?filename=exercise_syntax1]
https://www.w3schools.com/python/scipy/index.php [/exercise.php?filename=exercise_constants1]
https://www.w3schools.com/python/numpy/default.asp [/exercise.asp?filename=exercise_creating_arrays1]
https://www.w3schools.com/mongodb/index.php [/exercise.php?filename=exercise_get_started1]
https://www.w3schools.com/sql/default.asp [/exercise.asp?filename=exercise_select1]
https://www.w3schools.com/graphics/canvas_reference.asp [/action_page.php]
https://www.w3schools.com/jsref/jsref_obj_json.asp [/action_page.php]
https://www.w3schools.com/jsref/default.asp [/action_page.php]
https://www.w3schools.com/ [/action_page.php]
https://www.w3schools.com/dsa/index.php [/exercise.php?filename=exercise_algo_simple1]
https://www.w3schools.com/vue/index.php [/exercise.php?filename=exercise_intro1]
https://www.w3schools.com/html/default.asp [/exercise.asp?filename=exercise_html_attributes1]
https://www.w3schools.com/js/default.asp [/exercise_js.asp?filename=exercise_js_variables1]
https://www.w3schools.com/django/index.php [/exercise.php?filename=exercise_template_variables1]
https://www.w3schools.com/css/default.asp [/exercise.asp?filename=exercise_selectors1]
https://www.w3schools.com/postgresql/index.php [/exercise.php?filename=exercise_select1]
https://www.w3schools.com/default.asp [/action_page.php]
https://www.w3schools.com/php/default.asp [/exercise.asp?filename=exercise_syntax1]
https://www.w3schools.com/cs/index.php [/exercise.php?filename=exercise_syntax1]
←[93;5m←[0m Phase: Comparing [3/6]
```

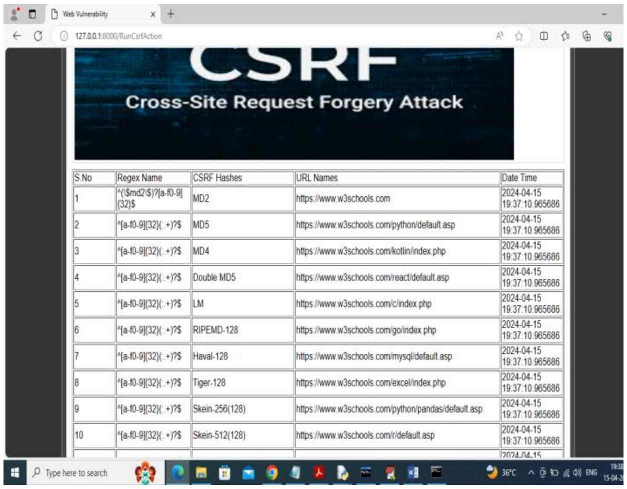
In above screen can see all scanned URL and then will get below page

ISSN2454-9940

www.ijasem.org

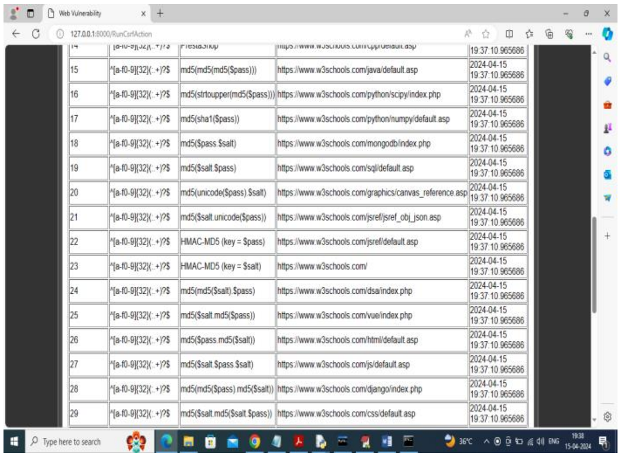
https://zenodo.org/records/11632200

Vol 18, Issue 2, 2024


Web Vulnerability
127.0.0.1:8000/RunCsrfAction
CSRF
Cross-Site Request Forgery Attack
Type here to search

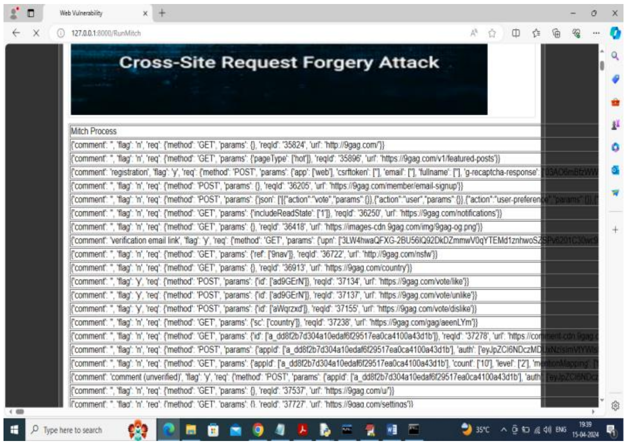
S.No	Regex Name	CSRF Hashes	URL Names	Date Time
1	^(\$md2\$)?[a-f0-9]{32}$	MD2	https://www.w3schools.com	2024-04-15 19:37:10.965686
2	^[a-f0-9]{32}(:.+)?$	MD5	https://www.w3schools.com/python/default.asp	2024-04-15 19:37:10.965686
3	^[a-f0-9]{32}(:.+)?$	MD4	https://www.w3schools.com/kotlin/index.php	2024-04-15 19:37:10.965686
4	^[a-f0-9]{32}(:.+)?$	Double MD5	https://www.w3schools.com/react/default.asp	2024-04-15 19:37:10.965686
5	^[a-f0-9]{32}(:.+)?$	LM	https://www.w3schools.com/c/index.php	2024-04-15 19:37:10.965686
6	^[a-f0-9]{32}(:.+)?$	RIPEMD-128	https://www.w3schools.com/go/index.php	2024-04-15 19:37:10.965686
7	^[a-f0-9]{32}(:.+)?$	Haval-128	https://www.w3schools.com/mysql/default.asp	2024-04-15 19:37:10.965686
8	^[a-f0-9]{32}(:.+)?$	Tiger-128	https://www.w3schools.com/excel/index.php	2024-04-15 19:37:10.965686
9	^[a-f0-9]{32}(:.+)?$	Skein-256(128)	https://www.w3schools.com/python/pandas/default.asp	2024-04-15 19:37:10.965686
10	^[a-f0-9]{32}(:.+)?$	Skein-512(128)	https://www.w3schools.com/r/default.asp	2024-04-15 19:37:10.965686

In above screen can see all CSRF list obtained from given URL and scroll down to view all details



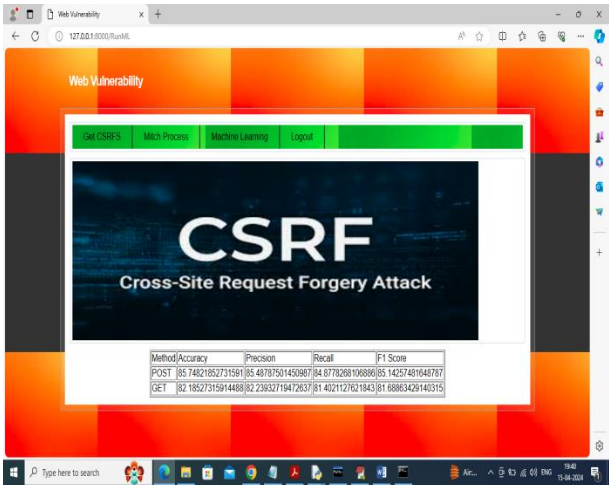
14	^[a-f0-9]{32}(:.+)?$	Prestashop	https://www.w3schools.com/cpp/default.asp	19:37:10.965686
15	^[a-f0-9]{32}(:.+)?$	md5(md5(md5($pass)))	https://www.w3schools.com/java/default.asp	2024-04-15 19:37:10.965686
16	^[a-f0-9]{32}(:.+)?$	md5(strtoupper(md5($pass)))	https://www.w3schools.com/python/scipy/index.php	2024-04-15 19:37:10.965686
17	^[a-f0-9]{32}(:.+)?$	md5(sha1($pass))	https://www.w3schools.com/python/numpy/default.asp	2024-04-15 19:37:10.965686
18	^[a-f0-9]{32}(:.+)?$	md5($pass.$salt)	https://www.w3schools.com/mongodb/index.php	2024-04-15 19:37:10.965686
19	^[a-f0-9]{32}(:.+)?$	md5($salt.$pass)	https://www.w3schools.com/sql/default.asp	2024-04-15 19:37:10.965686
20	^[a-f0-9]{32}(:.+)?$	md5(unicode($pass).$salt)	https://www.w3schools.com/graphics/canvas_reference.asp	2024-04-15 19:37:10.965686
21	^[a-f0-9]{32}(:.+)?$	md5($salt.unicode($pass))	https://www.w3schools.com/jsref/jsref_obj_json.asp	2024-04-15 19:37:10.965686
22	^[a-f0-9]{32}(:.+)?$	HMAC-MD5 (key = $pass)	https://www.w3schools.com/jsref/default.asp	2024-04-15 19:37:10.965686
23	^[a-f0-9]{32}(:.+)?$	HMAC-MD5 (key = $salt)	https://www.w3schools.com/	2024-04-15 19:37:10.965686
24	^[a-f0-9]{32}(:.+)?$	md5(md5($salt).$pass)	https://www.w3schools.com/dsa/index.php	2024-04-15 19:37:10.965686
25	^[a-f0-9]{32}(:.+)?$	md5($salt.md5($pass))	https://www.w3schools.com/vue/index.php	2024-04-15 19:37:10.965686
26	^[a-f0-9]{32}(:.+)?$	md5($pass.md5($salt))	https://www.w3schools.com/html/default.asp	2024-04-15 19:37:10.965686
27	^[a-f0-9]{32}(:.+)?$	md5($salt.$pass.$salt)	https://www.w3schools.com/js/default.asp	2024-04-15 19:37:10.965686
28	^[a-f0-9]{32}(:.+)?$	md5(md5($pass).md5($salt))	https://www.w3schools.com/django/index.php	2024-04-15 19:37:10.965686
29	^[a-f0-9]{32}(:.+)?$	md5($salt.md5($salt.$pass))	https://www.w3schools.com/css/default.asp	2024-04-15 19:37:10.965686

Now user can click on 'Mitch Process' link to run mitch and get below output



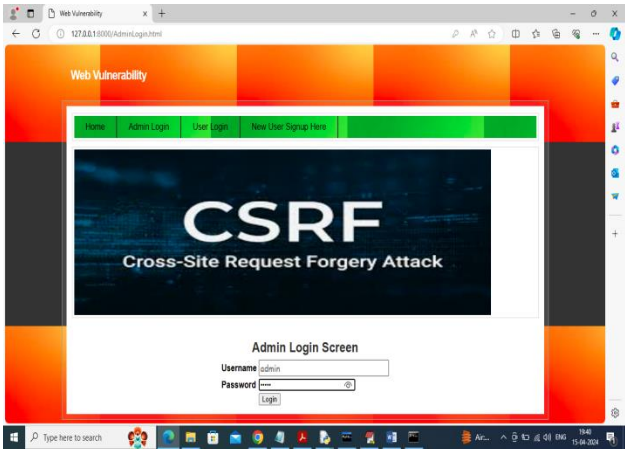
```
Mitch Process
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {}, 'reqId': '35824', 'url': 'http://9gag.com/'}}
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {'pageType': ['hot']}, 'reqId': '35896', 'url': 'https://9gag.com/v1/featured-posts'}}
{'comment': 'registration', 'flag': 'y', 'req': {'method': 'POST', 'params': {'app': ['web'], 'csrftoken': [''], 'email': [''], 'fullname': [''], 'g-recaptcha-response': ['03AO6mBfzWW
{'comment': '', 'flag': 'n', 'req': {'method': 'POST', 'params': {}, 'reqId': '36205', 'url': 'https://9gag.com/member/email-signup'}}
{'comment': '', 'flag': 'n', 'req': {'method': 'POST', 'params': {'json': ['[{"action":"vote","params":{}},{"action":"user","params":{}},{"action":"user-preference","params":{}},{
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {'includeReadState': ['1']}, 'reqId': '36250', 'url': 'https://9gag.com/notifications'}}
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {}, 'reqId': '36418', 'url': 'https://images-cdn.9gag.com/img/9gag-og.png'}}
{'comment': 'verification email link', 'flag': 'y', 'req': {'method': 'GET', 'params': {'upn': ['3LW4hwaQFXG-2BU56IQ92DkDZmmwV0qYTEMd1znhwoSZSPv6201C30w
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {'ref': ['9nav']}, 'reqId': '36722', 'url': 'http://9gag.com/nsfw'}}
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {}, 'reqId': '36913', 'url': 'https://9gag.com/country'}}
{'comment': '', 'flag': 'y', 'req': {'method': 'POST', 'params': {'id': ['ad9GErN']}, 'reqId': '37134', 'url': 'https://9gag.com/vote/like'}}
{'comment': '', 'flag': 'y', 'req': {'method': 'POST', 'params': {'id': ['ad9GErN']}, 'reqId': '37137', 'url': 'https://9gag.com/vote/unlike'}}
{'comment': '', 'flag': 'y', 'req': {'method': 'POST', 'params': {'id': ['aWqrzxd']}, 'reqId': '37155', 'url': 'https://9gag.com/vote/dislike'}}
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {'sc': ['country']}, 'reqId': '37238', 'url': 'https://9gag.com/gag/aeenLYm'}}
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {'id': ['a_dd8f2b7d304a10edaf6f29517ea0ca4100a43d1b']}, 'reqId': '37278', 'url': 'https://comment-cdn.9gag.c
{'comment': '', 'flag': 'n', 'req': {'method': 'POST', 'params': {'appId': ['a_dd8f2b7d304a10edaf6f29517ea0ca4100a43d1b'], 'auth': ['eyJpZCI6NDczMDUxNzIsImVIYWls
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {'appId': ['a_dd8f2b7d304a10edaf6f29517ea0ca4100a43d1b'], 'count': ['10'], 'level': ['2'], 'mentionMapping': ['
{'comment': 'comment (unverified)', 'flag': 'y', 'req': {'method': 'POST', 'params': {'appId': ['a_dd8f2b7d304a10edaf6f29517ea0ca4100a43d1b'], 'auth': ['eyJpZCI6NDcz
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {}, 'reqId': '37537', 'url': 'https://9gag.com/u/'}}
{'comment': '', 'flag': 'n', 'req': {'method': 'GET', 'params': {}, 'reqId': '37727', 'url': 'https://9gag.com/settings'}}
```

In above screen can see all MITCH process and now click on 'Machine Learning' link to train ML algorithm and get below output



Method	Accuracy	Precision	Recall	F1 Score
POST	85.74821852731591	85.48787501450987	84.8778268106886	85.14257481648787
GET	82.18527315914488	82.23932719472637	81.4021127621843	81.68863429140315

In above screen can see ML accuracy on both GET and POST methods and now logout and login as admin to view other process



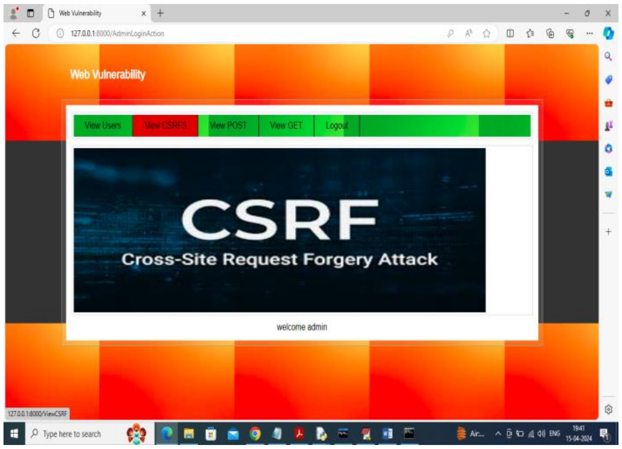
Admin Login Screen

Username admin

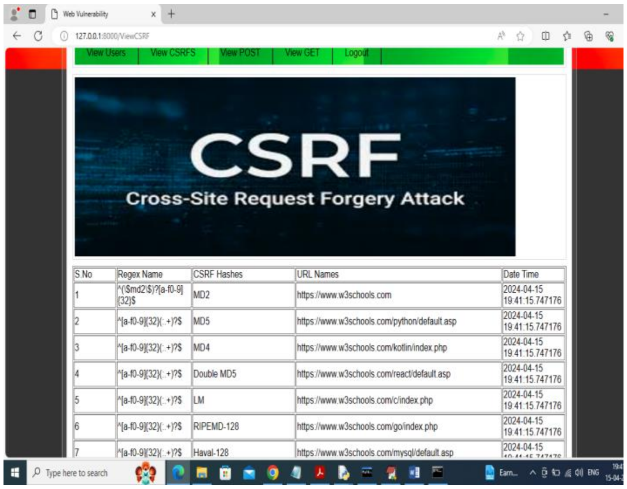
Password ••••

Login

In above screen admin is login and after login will get below page

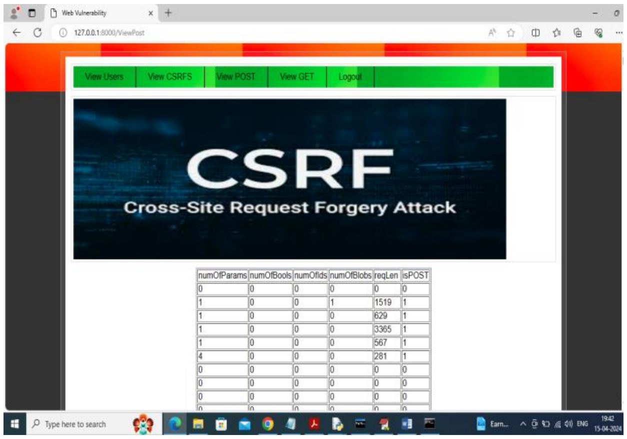


In above screen admin can click on 'View CSRFS' link to view all past CSRFS list and get below page



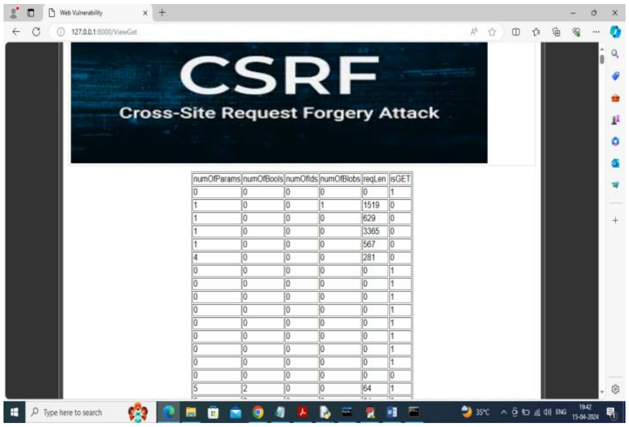
S.No	Regex Name	CSRF Hashes	URL Names	Date Time
1	^(\$md2\$)?[a-f0-9]{32}$	MD2	https://www.w3schools.com	2024-04-15 19:41:15.747176
2	^[a-f0-9]{32}(:.+)?$	MD5	https://www.w3schools.com/python/default.asp	2024-04-15 19:41:15.747176
3	^[a-f0-9]{32}(:.+)?$	MD4	https://www.w3schools.com/kotlin/index.php	2024-04-15 19:41:15.747176
4	^[a-f0-9]{32}(:.+)?$	Double MD5	https://www.w3schools.com/react/default.asp	2024-04-15 19:41:15.747176
5	^[a-f0-9]{32}(:.+)?$	LM	https://www.w3schools.com/c/index.php	2024-04-15 19:41:15.747176
6	^[a-f0-9]{32}(:.+)?$	RIPEMD-128	https://www.w3schools.com/go/index.php	2024-04-15 19:41:15.747176
7	^[a-f0-9]{32}(:.+)?$	Haval-128	https://www.w3schools.com/mysql/default.asp	2024-04-15

In above screen admin can view list of CSRF list and now click on 'View Post' link to view all post request



numOfParams	numOfBools	numOfIds	numOfBlobs	reqLen	isPOST
0	0	0	0	0	0
1	0	0	1	1519	1
1	0	0	0	629	1
1	0	0	0	3365	1
1	0	0	0	567	1
4	0	0	0	281	1
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

In above screen can see all POST request data and now click on 'View Get' link to get all get request data



numOfParams	numOfBools	numOfIds	numOfBlobs	reqLen	isGET
0	0	0	0	0	1
1	0	0	1	1519	0
1	0	0	0	629	0
1	0	0	0	3365	0
1	0	0	0	567	0
4	0	0	0	281	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	0
5	2	0	0	64	1

In above screen can see list of all GET request and similarly by following above screens you can run all modules of the project

CONLUSION :

To sum up, web vulnerabilities are a major risk to the safety and reliability of online applications. Attackers may compromise systems by taking advantage of common vulnerabilities like SQL injection, XSS, and CSRF to steal data, alter data, or run malicious code. Organizations may protect their web apps from these vulnerabilities by using safe coding techniques like input validation and output encoding. In addition, existing online applications may have their flaws found and fixed with the use of automated vulnerability detection technologies and routine security audits.An ever-changing threat environment makes it more important than ever for enterprises to be proactive and watchful when it comes to fixing online vulnerabilities. In order to safeguard their assets and reputation, enterprises should take proactive steps to secure their online applications. This will lower the danger of exploitation.

References:

1. Halfond, W. G., Orso, A., & Manolios, P. (2006). AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE'06).

2. Huang, Y., Huang, C., Su, S., & Lee, J. (2014). A Survey on Web Application Security. Journal of Software, 9(1), 218-228.

3. Barth, A., Jackson, C., & Mitchell, J. C. (2008). Robust Defenses for Cross-Site Request Forgery. In Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS '08).

4.These references provide insights into the prevalence of web vulnerabilities and effective mitigation strategies, offering valuable information for researchers and practitioners in the field of web application security.