DOI: http://dx.doi.org/10.26483/ijarcs.v13i4.6890

Volume 13, No. 4, July-August 2022

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

IMPLEMENTATION OF CYBER SECURITY ATTACKS AND STRATEGIC MITIGATION MECHANISMS

Rushabh Kela
School of Computer Science and Engineering
Vellore Institute of Technology
Vellore, India

Abhinav Chawla
School of Computer Science and Engineering
Vellore Institute of Technology
Vellore, India

Pratishtha Gaur
School of Computer Science and Engineering
Vellore Institute of Technology
Vellore, India

Dr. Manikandan K
School of Computer Science and Engineering
Vellore Institute of Technology
Vellore, India

Abstract: Cyber threats have increased drastically in the recent years and the most common targets are organisation applications or systems for data theft, disrupting the operations or any other malicious use. Incorporating website security prevents these sorts of attacks on the system. It is the act/practice of protecting websites from unauthorized access, use, modification, destruction, or disruption. A web application will be created and tested on various attacks such as Brute Force Dictionary attack, Denial-of-Service attacks, Cross Site Scripting (XSS) attack, NoSQL injections and WebSocket attacks. The vulnerabilities will be analysed, and resolved to ensure that the confidentiality, integrity, and authenticity of the user data is not compromised. To improve the website security and privacy, measures will be taken to add security features and the code of the website will be modified.

Keywords: Cyber Attacks, Cyber Security, Operation Security, Denial of Service, Cryptography, NoSQL Injection, Man-in-the-middle attack.

I. INTRODUCTION

Because of the global nature of the Internet, web applications, which are a critical component of any online-based business, are vulnerable to attacks from all over the world and at varying levels of scale and sophistication. Web application security refers to the security, integrity, confidentiality and availability of websites, web applications, and web services such as APIs deployed around the world. A vulnerability is a defect in a computer system that thieves can use to gain unauthorized access to it. After exploiting a vulnerability, a cyberattack can run malicious code, install malware, and even steal critical data. There are particular cyber security flaws that are more frequently targeted by attackers.

Vulnerabilities in websites and applications, as well as attacks launched by malicious actor attack, are examples of web security threats. Web security threats are intended to breach an organization's security barriers, allowing hackers and cybercriminals to take control of systems, gain access to data, and steal valuable assets. Some such attacks are Brute-force attack involves locating a hidden web page by trial and error. Brute force attacks can just be a prelude to further attacks such as malware/ransomware attacks. If a malicious hacker has direct access to devices or network, it makes it much easier to drop any kind of ransomware or malware on the network, further compromising the organization’s security. Cross Site Scripting (XSS) attack injects a malicious script in the form of a browser-side script to user end. This way the attacker performs operation on behalf of the authentic user. Using Denial of service attack attacker floods user PC with traffic which triggers a crash. NoSQLi is done to take advantage of flaws in the way a database handles search request. NoSQLi is used by attackers to obtain unauthorized access to data, modify or create new user permissions, or otherwise manipulate or destroy sensitive data. Effective website security requires design effort across the whole of the website: in the web application, the configuration of the web server, policies for creating and renewing passwords, and the client-side code. If a server-side web framework is used, it will almost certainly enable "by default" robust and well-thought-out defense mechanisms against a number of the more common attacks. Other attacks can be mitigated through the web server configuration, for example by enabling HTTPS. Finally, there are publicly available vulnerability scanner tools that can help to find out if there are any vulnerabilities in the system.

II. LITERATURE SURVEY

[1] This Paper proposes a Cyber Threat Intelligence (CTI) model, that enables cyber defenders to investigate threat intelligence capabilities and their security level against the cyber threat scenario. The model is used to evaluate several taxonomies, sharing standards and relevant frameworks and structures. A drawback in the model is that it must be validated by a recognized security organization, for acceptable use.

[2] identifies important security vulnerabilities and performs a detailed analysis on the targeted / victimized applications, migration techniques and infrastructures to present a better insight.

[3] A framework to facilitate the cyber – attack response process. The developed security state estimation

model has been applied to functional impact analysis and validated by demonstrating it on a case study. However, it has a limitation, for cyber – attacks that exploit unknown vulnerabilities, security state estimation performance may be weakened.

[4] introduces a real-time security assessment framework that has a nested security architecture, secure registration protocol and risk – based multi factor authentication to ensure secure binding and communication, thus solving the problem of insufficient authentication of interconnected components in a system. The authors must also take latency parameter into consideration.

[5] proposes and justifies a processing algorithm of computer security incidents based on the author's signatures of cyberattacks, along with inspection of the traffic of such application layer protocols as HTTP, DCE/RPC, FTP, SMB, SMTP and POP3.

[6] proposes intrusion detection and prevention system uses a rule set of agent and sensors to protect software and application on the private cloud. The system can be further extended to efficiently analyze multimedia packets.

[7] presented a new tool for conducting backup system evaluations during information security risk assessments that addresses common security issues with the potential to inhibit the ability to combat and recover from a ransomware attack.

[8] details about methods to combat cyber – attacks in critical areas like Connected and Autonomous vehicles, Khan, Shiwakoti, and Chen propose an integrated CAV framework that focuses on the mitigation strategies and its impact, vulnerabilities by proximity access to cyber-attacks and their resolutions to create a comprehensive CAV – cybersecurity framework. The proposed framework must be validated and certified by a recognized security organization.

[9] Simulates a brute force and dictionary attack on passwords, and thereby got results that about over 95% of passwords could be cracked using a mid–range modern GPU. Further, use of graphical passwords, biometrics, bCrypt hash functions was discussed for higher level of security.

[10] identifies NoSQL injection threats in online applications, Eassa recommends an independent RESTful web service as part of a layered solution. It depends on comparing the generated patterns from NoSQL statement structure in static code state and dynamic code state. It consists of four components, the service interface, service controller, handler and detection and works as a filtering service to detect NoSQL injection requests.

[11] introduces a new method in the NoSQL design practice. It uses the asymmetric encryption algorithm RSA to generate key pairs during the design. The database server decrypts the legal queries before effecting any changes. The method was found to be safer than the traditional methods.

[12] presents various approaches that can be incorporated in web applications to detect and prevent NoSQL injection. The first approach uses machine learning to create a feature – based supervised learning model using dataset of benign and malicious MongoDB queries. The second approach uses automata, user input validation and parameterization-based model to check on the query and process it.

In [13] the authors built a prototype in JavaScript using end – to – end encryption that uses RSA Public Key encryption as well as the AES symmetric – key encryption algorithm. The prototype has not considered the scenario of regenerating the keys when a user loses his password.

[14] Cloud security has evolved continuously. Third-party data centres are used by the cloud platform. Lee, Dewi, and Wajdi use Heroku as a cloud platform, with data security provided by AES. AES cryptography can be employed for data security, according to the performance assessment.

The authors of [15] implement ECC algorithm to secure text message in messaging application of a smartphone. It proves that the ECC algorithm can be implemented and has competitive performance in both time and accuracy, there were no limitations in this work.

[16] overcomes the DoS attacks, Rengaraju and Ramanan propose a distributed Firewall with Intrusion Prevention System (IPS) for Software Defined Clouds. For several network settings, the proposed distributed security mechanism is explored for two DoS assaults, ICMP and SYN flooding attacks, and efficiently detects and prevents the Denial-of-Service Attack.

[17] Presents a comprehensive view on AI – driven cybersecurity that play an intelligent role in information security services and management. The study was quite in-depth and there were no limitations.

[18] Introduces five new cybersecurity challenges and opportunities to overcome the identified vulnerabilities. Creating a combination between the "edge computing + IoT" platform and developing blockchain and AI technologies could have a lot of positive outcomes.

[19] Implements and modifies Snort IDS by using prefix and random indexing method to all Snort rules to bring down the packet inspection time.

[20] proposes hybrid techniques of Beaufort and Vignere algorithms to optimize the security of text message encryption by modifying key generators. With a slight modification to the key build algorithm, this combination of methods succeeded in getting an avalanche effect value above 46%. This value is quite close to ideal and better than the previous method because the value of the avalanche effect has a stable value on the modification of keys and plain text.

[21] examined the modern real-time web ecosystem, offering an up-to-date picture of how WebSockets and other real-time protocols are currently being used. Various best practices were discussed that will create safer, more stable applications.

III. VULNERABLE WEBSITE ARCHITECTURE

To simulate the attacks, we have built a vulnerable website, analyzed its security flaws after a thorough penetration test and developed a secure web application to improve its security level.

To access the various features available on the website, the user must first register on the website, with all the necessary details. Once he is registered and logged in he can now chat with his friends, access the library which contains the URLs of various books or study materials, or upload and store his important documents at one place. PassportJS Cookie Strategy is used for the authentication process in the vulnerable website, whereas the secure website uses JSON Web Tokens (JWT) authentication technique.

Since this application is highly user-oriented and security is a major requirement, the application was tested under various attacks and the security issues were analyzed.

IV. Methodology

We first perform cyber security attacks on a vulnerable website whose architecture is discussed above and then suggest and implement preventive measures to mitigate those. The vulnerable website is deployed at https://oasis-cyber.herokuapp.com/.

A. *Performing the attacks*

1) *Brute – force attack*

Brute force attacks are performed to compromise the account or to take over it completely. The attack is also performed for Data exfiltration/Leak, accessing the website and malware/ransomware distribution. The attack was performed on the login page with various combinations of emails and passwords and stored the valid credentials in a file "results.txt".

Two vulnerabilities were identified after the attack was performed i.e. there is no limit on the failed attempts for users who are posting requests to the server. The user can post infinite requests to the server and can login and There is no two - factor verification (2FA) done via captcha, OTPs or any other source to find out if login requests are from genuine users or brute - force attacking programs.

In a MacBook Pro, with 8GB RAM, nearly 30 million attempts per second can be computed.

A python script is used to carry out the brute force attack which uses a dictionary containing combinations of letters, numbers and special characters, probable passwords saved in the password.txt file.

The script runs multiple combinations for the email and passwords provided and gives output for each query. The input field names of email and password are given as input to the python script. Python module requests is used to send POST requests to the domain and get the output. The logic behind obtaining a valid credential is that if the returned webpage has a Logout button in it, it means that the email and password combinations were valid and logged into the system. On successful attempts it stores the result in the result.txt file and the hacker can easily login to the system using different valid user credentials.

$$\mathit{Maximum\ Time} = \frac{\mathit{Number\ of\ possible\ characters}^{\mathit{Password\ Length}}}{\mathit{Attempts\ per\ second}}$$

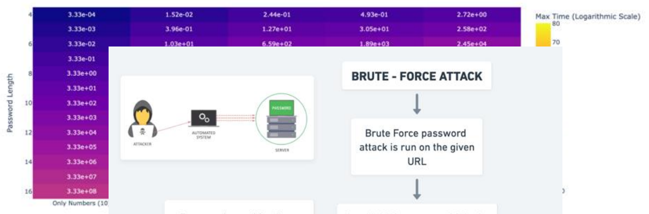


Fig 1: brute force heatmap on scale

Analysis of attack and logarithmic

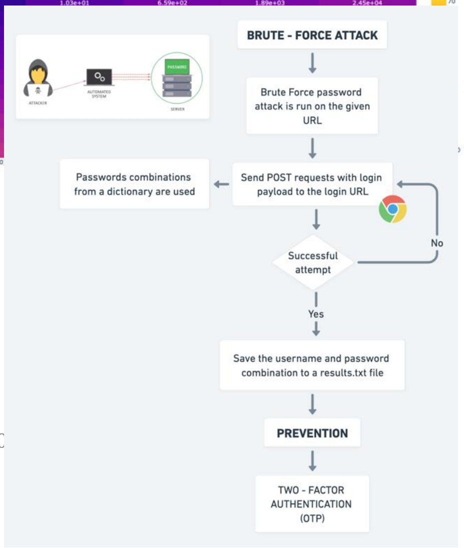

BRUTE - FORCE ATTACK
ATTACKER
AUTOMATED SYSTEM
SERVER
Brute Force password attack is run on the given URL
Passwords combinations from a dictionary are used
Send POST requests with login payload to the login URL
Successful attempt
No
Yes
Save the username and password combination to a results.txt file
PREVENTION
TWO - FACTOR AUTHENTICATION (OTP)

Fig 2: Brute force attack architecture

2) *Cross Site Scripting (XSS)*

Several vulnerabilities were identified for this type of attack such as in the books section of the web application, the users can add links of the various books to contribute to the library. However, a hacker / attacker can add a malicious link to any book to steal the user's data whenever a user who might want that book clicks on it. Another vulnerability is that in the vulnerable website, the user authentication is done through the use of cookies. Once the user is logged in, a cookie "token" is generated and saved in the browser until he logs out.

Following steps are followed while performing the attacks. First the attacker creates an account on the website then the attacker adds a link as a new book on the website. The URL entered in the users' browser will send the user's cookie to the hacker's server. The hacker now has access to the user's cookie - token and he can access the user's account using the cookie.

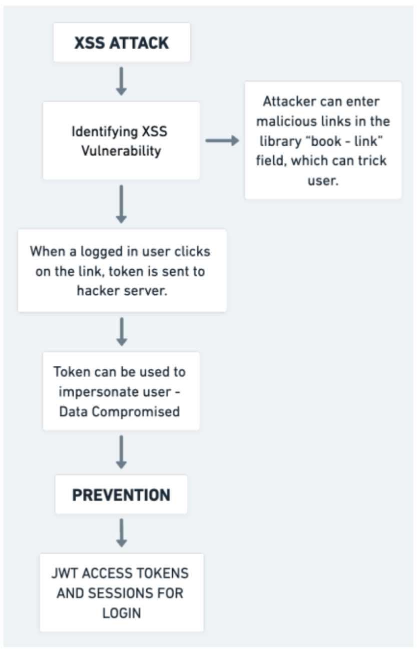

XSS ATTACK
Identifying XSS Vulnerability
Attacker can enter malicious links in the library "book - link" field, which can trick user.
When a logged in user clicks on the link, token is sent to hacker server.
Token can be used to impersonate user - Data Compromised
PREVENTION
JWT ACCESS TOKENS AND SESSIONS FOR LOGIN

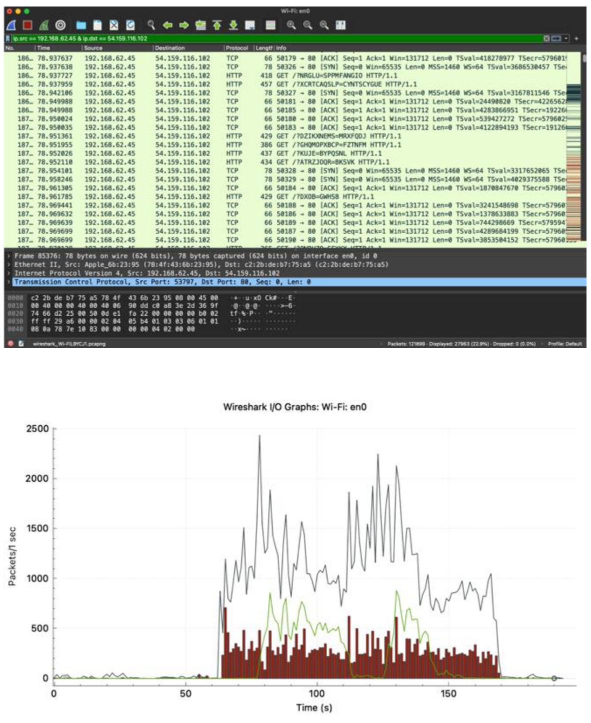
Fig 3: XSS attack architecture

3) *Denial – of – Service attack*

The hacker gets many benefits from the spread of hosts that constitutes a DDoS: It is possible to use a larger number of machines to carry out a very disruptive attack. It is difficult to find out the exact location of the attack as the attacking machines are randomly distributed over an area. Shutting down numerous machines is more difficult than shutting down one. The genuine attackers are difficult to spot since they hide behind a slew of (usually compromised) systems.

The vulnerability identified for this attack in the website is large levels of traffic overload the attacked system, which the server is unable to handle. The system gradually comes to a halt. A Python script is used to carry out the Denial of Service (DoS) attack on the website. Python modules such as urllib2 (for URL requests), threading (to carry out the request operations simultaneously using multi-threading) are used. The principle behind the flood being sent is that a unique pattern is generated at each and every request, with the intention of increasing the load on the servers as well as evading any intrusion detection and prevention systems.

The python script obfuscates the source client, with a list of known user agents. So, every time a request is constructed, it has a random user agent and referrer value. The no-cache header is used because a server that is not behind a dedicated caching service will present a unique page. Each URL also has a unique transformation to eliminate caching and other optimization tools, crafting custom parameter names and values and they are randomized and attached to each request, rendering it to be unique, enable the request to bypass many CDN systems.


```
ip.src == 192.168.62.45 & ip.dst == 54.159.116.102
No.   Time       Source         Destination     Protocol Length Info
186… 78.937637  192.168.62.45  54.159.116.102  TCP       66 50179 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=418278977 TSecr=579601
186… 78.937638  192.168.62.45  54.159.116.102  TCP       78 50326 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3686530457 TSe
186… 78.937727  192.168.62.45  54.159.116.102  HTTP     418 GET /?NRGLU=SPPMFANGIO HTTP/1.1
186… 78.937959  192.168.62.45  54.159.116.102  HTTP     457 GET /?XCRTCAQSLP=CYNTSCYGUE HTTP/1.1
186… 78.942106  192.168.62.45  54.159.116.102  TCP       78 50327 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3167811546 TSe
186… 78.949988  192.168.62.45  54.159.116.102  TCP       66 50181 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=24490820 TSecr=4226562
186… 78.949988  192.168.62.45  54.159.116.102  TCP       66 50185 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=4283866951 TSecr=19226
187… 78.950024  192.168.62.45  54.159.116.102  TCP       66 50180 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=539427272 TSecr=579602
187… 78.950035  192.168.62.45  54.159.116.102  TCP       66 50183 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=4122894193 TSecr=19126
187… 78.951361  192.168.62.45  54.159.116.102  HTTP     429 GET /?DZIKXNEMS=MRXFQDJ HTTP/1.1
187… 78.951955  192.168.62.45  54.159.116.102  HTTP     386 GET /?GHQMOPXBCP=FZTNFM HTTP/1.1
187… 78.952026  192.168.62.45  54.159.116.102  HTTP     437 GET /?KUJE=BYPQSNL HTTP/1.1
187… 78.952110  192.168.62.45  54.159.116.102  HTTP     434 GET /?ATRZJOQR=BKSVK HTTP/1.1
187… 78.954101  192.168.62.45  54.159.116.102  TCP       78 50328 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3317652065 TSe
187… 78.958246  192.168.62.45  54.159.116.102  TCP       78 50329 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=4029375588 TSe
187… 78.961305  192.168.62.45  54.159.116.102  TCP       66 50184 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=1870847670 TSecr=57960
187… 78.961785  192.168.62.45  54.159.116.102  HTTP     429 GET /?DXOB=GWHSB HTTP/1.1
187… 78.969441  192.168.62.45  54.159.116.102  TCP       66 50188 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=3241548698 TSecr=57960
187… 78.969632  192.168.62.45  54.159.116.102  TCP       66 50186 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=1378633883 TSecr=57960
187… 78.969639  192.168.62.45  54.159.116.102  TCP       66 50189 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=744298669 TSecr=579594
187… 78.969699  192.168.62.45  54.159.116.102  TCP       66 50187 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=4289684199 TSecr=57960
187… 78.969699  192.168.62.45  54.159.116.102  TCP       66 50190 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=3853504152 TSecr=57960

> Frame 85376: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface en0, id 0
> Ethernet II, Src: Apple_6b:23:95 (78:4f:43:6b:23:95), Dst: c2:2b:de:b7:75:a5 (c2:2b:de:b7:75:a5)
> Internet Protocol Version 4, Src: 192.168.62.45, Dst: 54.159.116.102
> Transmission Control Protocol, Src Port: 53797, Dst Port: 80, Seq: 0, Len: 0

0000  c2 2b de b7 75 a5 78 4f  43 6b 23 95 08 00 45 00
0010  00 40 00 00 40 00 40 06  90 dd c0 a8 3e 2d 36 9f
0020  74 66 d2 25 00 50 0d e1  fa 22 00 00 00 00 b0 02
0030  ff ff 29 a6 00 00 02 04  05 b4 01 03 03 06 01 01
0040  08 0a 78 7e 10 83 00 00  00 00 04 02 00 00

wireshark_Wi-FiLBYCJ1.pcapng    Packets: 121899 · Displayed: 27963 (22.9%) · Dropped: 0 (0.0%)    Profile: Default
```

Fig 4: Wireshark analysis of the DoS attack

4) *NoSQL injection*

Vulnerable apps can be executed with any dump data and these attacks can potentially allow unknown code execution. There are several vulnerabilities that are identified for this type of attack. The authentication input fields of login are not validated with the data that is entered in the email and passwords. MongoDB query is run with the malicious hacker input that tricks the database by giving a TRUE Boolean value, thus giving access to the hacker of the web portal by account of another authentic user. There is no verification for the login POST request source and its input data. Hackers can use postman tool to send a post request with any TRUE Value fields and can access the website.

An example of a vulnerable payload can be

```
{
    "email" : {"$gt":""},
    "password" : {"$gt":""}
}
```

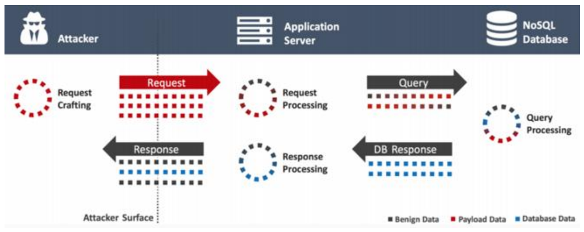


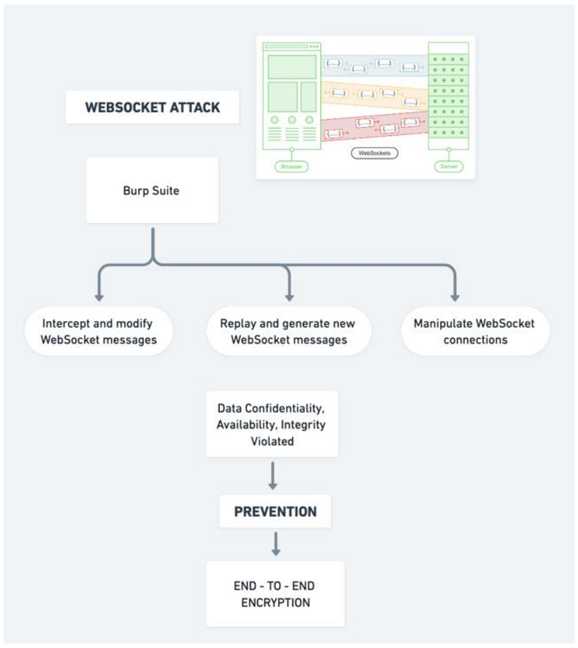
Fig 5: Flow of events in NoSQL injection

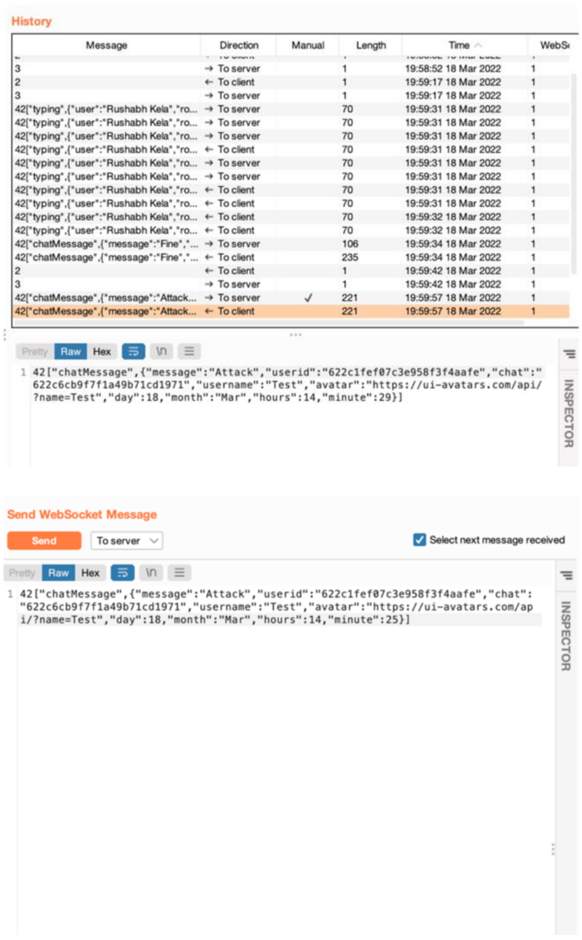
5) *Websocket Attack*

Sniffing / Man-in-the-Middle attacks can be used to read user messages, change them, and exploit other WebSocket flaws. Man-in-the-middle attacks take advantage of unencrypted connections by pretending to be an endpoint (since no one's verifying their identity anyway). In our website, the websocket man – in – the – middle attack is performed on the chat section, where the attacker is able to intercept the messages of users, thus compromising the confidentiality and integrity of the message transmission.

Using the Burp Tool, we can intercept and modify the WebSocket messages. Attackers can generate new WebSocket messages and manipulate the existing web socket connections. Since the messages are being transferred between the clients without being encrypted, this situation is vulnerable to a man-in-the-middle attack where the messages are visible as it is to the attacker.

The Burp Suite tool and its browser is used to carry out the attack. The website is launched using the Burp Proxy Web Browser. Once the website is open, the tool can be used to view the websockets history and the data being passed through the sockets from server to client and vice versa. The data shows the messages, user ID and chat ID. Thus, a man-in-the-middle attack is simulated. This socket URL can also be sent to the intruder and repeater to resend those socket messages either as it is or by manipulating them. The confidentiality and integrity of the user chat is compromised by this simulation. The tool can be used to Intercept and modify WebSocket messages, Replay and generate new WebSocket messages and Manipulate WebSocket connections.


WEBSOCKET ATTACK
Burp Suite
Intercept and modify WebSocket messages
Replay and generate new WebSocket messages
Manipulate WebSocket connections
Data Confidentiality, Availability, Integrity Violated
PREVENTION
END - TO - END ENCRYPTION
Browser
WebSockets
Server



History

Message	Direction	Manual	Length	Time	WebS
3	→ To server		1	19:58:52 18 Mar 2022	1
2	← To client		1	19:59:17 18 Mar 2022	1
3	→ To server		1	19:59:17 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	→ To server		70	19:59:31 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	→ To server		70	19:59:31 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	→ To server		70	19:59:31 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	← To client		70	19:59:31 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	→ To server		70	19:59:31 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	→ To server		70	19:59:31 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	← To client		70	19:59:31 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	← To client		70	19:59:31 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	← To client		70	19:59:32 18 Mar 2022	1
42["typing",{"user":"Rushabh Kela","ro...	← To client		70	19:59:32 18 Mar 2022	1
42["chatMessage",{"message":"Fine","...	→ To server		106	19:59:34 18 Mar 2022	1
42["chatMessage",{"message":"Fine","...	← To client		235	19:59:34 18 Mar 2022	1
2	← To client		1	19:59:42 18 Mar 2022	1
3	→ To server		1	19:59:42 18 Mar 2022	1
42["chatMessage",{"message":"Attack...	→ To server	✓	221	19:59:57 18 Mar 2022	1
42["chatMessage",{"message":"Attack...	← To client		221	19:59:57 18 Mar 2022	1

```
42["chatMessage",{"message":"Attack","userid":"622c1fef07c3e958f3f4aafe","chat":"
622c6cb9f7f1a49b71cd1971","username":"Test","avatar":"https://ui-avatars.com/api/
?name=Test","day":18,"month":"Mar","hours":14,"minute":29}]
```

Send WebSocket Message

To server

```
42["chatMessage",{"message":"Attack","userid":"622c1fef07c3e958f3f4aafe","chat":
"622c6cb9f7f1a49b71cd1971","username":"Test","avatar":"https://ui-avatars.com/ap
i/?name=Test","day":18,"month":"Mar","hours":14,"minute":25}]
```

Fig 6: Websocket attack architecture and analysis using Burp Suite Community Edition

B. *Mitigation Strategies*

All the above vulnerabilities were identified and techniques to prevent them were devised, to make the system more secure and robust. The secure web application is deployed at https://oasis-secure.herokuapp.com/.

1) *Prevention of brute – force attack*

For the brute force attack we identified that we need to add two-factor authentication parameters, where we are able to give user an OTP verification with login and registration. We have implemented Twilio SendGrid Email API (cloud -based email delivery) services to provide a real-time OTP verification service. We made a gmail service provider and used email services to provide OTP services verification.

2) *Prevention of XSS attack*

The XSS attack was possible due to the vulnerability that we were allowing to users upload links (and hence attackers were able to inject malicious links) and the links were sending the user's cookie to the hacker server. Hence, we have eliminated both of these vulnerabilities. The authentication strategy was changed from a normal cookie authentication to JSON Web Tokens (JWT Auth). JSON Web Token (JWT) is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. A JWT token is signed digitally. The signature can be using a secret (with HMAC algorithm) or a public/private key using RSA algorithm. Signed tokens can verify the integrity of the claims contained within it, while encrypted tokens hide those claims from other parties. When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it.

Once the user is logged in, each subsequent request will also include the JWT, and based on the token permissions, the user can access routes, services, and resources. It has a very small overhead and its ability to be easily used across different domains. JWT are a good way of securely sending information between parties. Both the parties are also verifiable since JWTs can be signed using public/private key pairs. Since the signature is calculated using the header and the payload of the request, tampering of the content can be checked.

We identified that user can input malicious links in the input panel of Books and Documents and that can cause a header link that can cause a cookie sharing on the hacker's log. In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

- Filter input on arrival i.e. When user input is received at the server, it must be filtered as strictly as possible based on the expected or valid input.
- Encode data on output i.e. If the data related to the user in transit is output in HTTP responses, it must be encoded to differentiate it from active content.
- Introduce a proper reliable Database. Here we have used AWS Bucket S3 storage for Books and Documents upload.

3) *Prevention of DOS attack*

To come over with this case, We explored the npm packages that can prevent multiple request from the same IP address at a moment. Rate limiting stops the same IP from making too many requests, which helps us avoid brute force attacks. The npm package express-rate-limit is used to limit the number of requests a user may make. Rate limitation is a network traffic control strategy that limits the quantity of incoming and outgoing traffic. The channel of communication between a client (e.g., a web browser) and our server is referred to as the network in this context (e.g., API). As a result, it's a strategy that enables us to process user requests based on a set of constraints, such as: There is an improved data flow, There is a lower danger of attack, i.e., enhanced security, The server is never overwhelmed, Users can only do what the developer allows them to do.

For rate limiter to be implemented on the server, a constraint must be defined based on which the limit will be applied. Users can be rate limited based on their unique user identifiers. The constraint can be based on the geographical location of the user and the place from which the request was sent. Limiting specific IP addresses that are suspicious or sending a lot of requests to the endpoints continuously can also be done.

To store the constraints and entities on which the limiter is applied, in-memory store is used. That implies there's no need to set anything up, but it also means that several servers can't exchange state. We can configure express-rate-limit to use an external store like redis if we have numerous servers behind a load balancer and wish to share request rate data.

4) *Preventing NoSQL injection attacks*

We identified that the user can place TRUE Values in the Login field of Email and Password and can login randomly in any of the user's account. A successful MongoDB injection or other NoSQL injection attack can have far more catastrophic repercussions than a standard SQL injection attack. Attackers can not only extract data from the database, but they can also run code in the context of the application, for example, to launch denial-of-service attacks or compromise admin user accounts and take control of the server. Such attacks are particularly risky because NoSQL data stores are typically unfamiliar territory for developers who are only experienced with relational database systems, increasing the possibility of unsafe code. The simplest strategy to avoid NoSQL injection attacks is to avoid using unsanitised user inputs in application code, especially when creating database queries, as is typically the case in online application security. MongoDB, for example, includes built in functionality for constructing secure queries without the use of JavaScript. If you must use JavaScript in your queries, make sure to follow standard best practices such as validating and encoding all user inputs, following the principle of least privilege, and knowing your language to prevent employing susceptible constructions.

To rectify this, we observed that we need to sanitize the input and make sure we are receiving a string input a for the prevention of NoSQL Injection, for this I implemented the database fetch, read and write calls after sanitizing the input using the mongo-sanitize package.

5) *Prevention of Websocket attacks*

We identified that the chats are visible through the man in the middle attack and a man in the middle is able to view the messages from the Burp tool that we used for the identification of vulnerability. To prevent the man - in - the - middle attack, the chat module will be end - to - end encrypted. Two encryption techniques were taken and compared on the basis on their execution time. As seen from the graphs, AES takes considerably lesser time than RSA, both of which are quite popular for chat encryptions. AES encryption algorithm will be used. The man in the middle will only be able to read string hashes, not the actual communications.

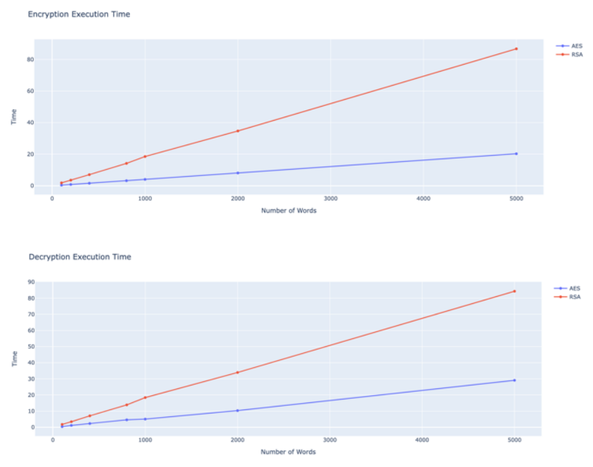


Fig 7: Comparison of AES and RSA algorithm for encryption and decryption against varying text data

V. Conclusion

Technology and Processes alone are not sufficient to keep a firm safe from data breaches. As a result, to integrate IT and human security, a true information security management system is required. We have hence tried to create a one stop portal to try and test different attacks and devise mechanisms to prevent each one of them.

We try to explore vulnerabilities in websites and applications by deploying a vulnerable website over the internet and performing cyber security attacks on it such as brute-force attack to guess passwords, denial-of-service attacks to deny website on time, XSS attack to inject malicious scripts into safe websites, NoSQL Injection to take advantage of flaws in the way database handles search requests and WebSocket attack to explore man in the middle type of attacks.

Finally, all these attacks are prevented one by one by methods such as two factor authentication to check on brute force attack, authorization using JWT to prevent XSS, implementing rate limiter to prevent DOS, implementing encryption to prevent man in the middle type of attacks and hence checking on web socket vulnerabilities.

Because information security is more than a technology issue, people and organisations must acknowledge that there are also human-related challenges that must be addressed if successful information security management is to be achieved.

VI. Acknowledgment

I would like to express my deep sense of gratitude to Prof. Manikandan K for giving me an opportunity to work on a project that helped me understand better and apply the concepts that me have been learning. It further helped me explore novel concepts beyond the traditional curriculum.

I would like to thank him for the continuous and constant support for helping me with the project throughout the semester.

I would also like to thank Vellore Institution of Technology for giving me an opportunity to carry out this project. I hope that this research lives up to the professors' expectations.

VII. References

[1] Mavroeidis, V., & Bromander, S. (2017, September). Cyber threat intelligence model: an evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In 2017 European Intelligence and Security Informatics Conference (EISIC) (pp. 91-98). IEEE.

[2] Humayun, M., Niazi, M., Jhanjhi, N. Z., Alshayeb, M., & Mahmood, S. (2020). Cyber security threats and vulnerabilities: a systematic mapping study. Arabian Journal for Science and Engineering, 45(4), 3171-3189.

[3] Lee, C., Chae, Y. H., & Seong, P. H. (2021). Development of a method for estimating security state: Supporting integrated response to cyber-attacks in NPPs. Annals of Nuclear Energy, 158, 108287.

[4] Sani, A. S., Yuan, D., Yeoh, P. L., Qiu, J., Bao, W., Vucetic, B., & Dong, Z. Y. (2019, August). CyRA: A real-time risk-based security assessment framework for cyber attacks prevention in industrial control systems. In 2019 IEEE Power & Energy Society General Meeting (PESGM) (pp. 1-5). IEEE.

[5] Vorobiev, E. G., Petrenko, S. A., Kovaleva, I. V., & Abrosimov, I. K. (2017, May). Analysis of computer security incidents using fuzzy logic. In 2017 XX IEEE International Conference on Soft Computing and Measurements (SCM) (pp. 369-371). IEEE.

[6] Ali, Z. A., & Ameen, S. Y. (2018). Detection and prevention cyber-attacks for smart buildings via private cloud environment. International Journal of Computing and Network Technology, 6(01), 27-33.

[7] Thomas, J., & Galligher, G. (2018). Improving backup system evaluations in information security risk assessments to combat ransomware. Computer and Information Science, 11(1).

[8] Khan, S. K., Shiwakoti, N., Stasinopoulos, P., & Chen, Y. (2020). Cyber-attacks in the next-generation cars, mitigation techniques, anticipated readiness and future directions. Accident Analysis & Prevention, 148, 105837.

[9] Bošnjak, L., Sreš, J., & Brumen, B. (2018, May). Brute-force and dictionary attack on hashed real-world passwords. In 2018 41st international convention on information and communication technology, electronics and microelectronics (mipro) (pp. 1161-1166). IEEE.

[10] M Eassa, A. M., Elhoseny, M., El-Bakry, H. M., & Salama, A. S. (2018). NoSQL injection attack detection in web applications using RESTful service. Programming and Computer Software, 44(6), 435-444. 53

[11] Imam, A. A., Basri, S., González-Aparicio, M. T., Balogun, A. O., & Kumar, G. (2022, January). NoInjection: Preventing Unsafe Queries on NoSQLDocument-model Databases. In 2022 2nd International Conference on Computing and Information Technology (ICCIT) (pp. 243-247). IEEE.

[12] Shachi, M., Shourav, N. S., Ahmed, A. S. S., Brishty, A. A., & Sakib, N. A Survey on Detection and Prevention of SQL and NoSQL Injection Attack on Server-side Applications. International Journal of Computer Applications, 975, 8887.

[13] Schillinger, F., & Schindelhauer, C. (2019, July). End-to-end encryption schemes for online social networks. In International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage (pp. 133- 146). Springer, Cham.

[14] Lee, B. H., Dewi, E. K., & Wajdi, M. F. (2018, April). Data security in cloud computing using AES under HEROKU cloud. In 2018 27th wireless and optical communication conference (WOCC) (pp. 1-5). IEEE.

[15] Natanael, D., & Suryani, D. (2018). Text Encryption in Android Chat Applications using Elliptical Curve Cryptography (ECC). Procedia Computer Science, 135, 283-291.

[16] Rengaraju, P., Ramanan, V. R., & Lung, C. H. (2017, August). Detection and prevention of DoS attacks in Software-Defined Cloud networks. In 2017 IEEE Conference on Dependable and Secure Computing (pp. 217-223). IEEE.

[17] Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2021). AI-driven cybersecurity: an overview, security intelligence modeling and research directions. SN Computer Science, 2(3), 1-18.

[18] Pan, J., & Yang, Z. (2018, March). Cybersecurity challenges and opportunities in the new "edge computing+ IoT" world. In Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (pp. 29-32).

[19] Kabir, M. F., & Hartmann, S. (2018, May). Cyber security challenges: An efficient intrusion detection system design. In 2018 International Young Engineers Forum (YEF-ECE) (pp. 19-24). IEEE.

[20] Sugiarto, E., Setiadi, D. R. I. M., Fahmi, A., Rachmawanto, E. H., Sari, C. A., Sarker, M. K., & Widjajanto, B. (2020, July). Securing Text Messages using the Beaufort-Vigenere Hybrid Method. In Journal of Physics: Conference Series (Vol. 1577, No. 1, p. 012032).1OP Publishing.

[21] Murley, P., Ma, Z., Mason, J., Bailey, M., & Kharraz, A. (2021, April). Websocket adoption and the landscape of the real-time web. In Proceedings of the Web Conference 2021 (pp. 1192-1203).