International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
Impact Factor (2020): 6.733

Web-Application Security and Approach to Find Vulnerabilities into the Era of Web 2.0

Henil Sanjaykumar Gandhi

Computer Engineering Department, L. D. College of Engineering, Gujarat Technological University, Ahmedabad, Gujarat, India
Email: *hsghenil[at]gmail.com*

Abstract: ***Web application security is the process of defending websites and other online services from various security risks that take advantage of coding flaws in programs. This paper discusses popular web application threats such as injection attacks, XSS, and CSRF. We are particularly concerned about how these assaults may harm customer trust, data security, and brand reputation. To prevent these threats, a web application firewall (WAF) is recommended as a critical security solution. The WAF acts as a gatekeeper, screening and stopping malicious or suspicious requests. The research paper investigates the significance of proactive security techniques such as security vulnerability assessment and penetration testing. It emphasizes the importance of input validation, secure coding practices, and robust session management in preventing application vulnerabilities. The study also emphasizes the importance of the integration of automated techniques for the prompt discovery and remediation of vulnerabilities.***

Keywords: Web application Security, Web application Firewall, Approach towards vulnerability finding, Vulnerability scanners and tools, Vulnerability description

1. Importance of Web Application Security Testing

Hackers and other cybercriminals are continuously on the lookout for new vulnerabilities in online applications and ways to exploit them in order to obtain access or cause them to malfunction. Web security testing looks for security flaws in Web applications and their configuration. Testing the security of a Web application may involve providing multiple inputs to elicit mistakes and lead the system to act unpredictably[1].It's critical to understand that online security testing entails much more than the login and authorization techniques used in the program. Furthermore, ensuring the safe implementation of additional components (such as business logic, suitable input validation, and output encoding) is critical.

2. Recent Cyber Attacks

1) Twitter Data Breach:

Citing a technical flaw, Twitter acknowledged a data breach in July 2022 that exposed information about its users. The majority of the content consisted of information that was readily accessible to the public, including Twitter IDs, names, login names, locations, and verified statuses; however, there was also private information like phone numbers and email addresses. The attackers offered this material for sale in exchange for a large loss of organization [13].

2) Uber Data Breach:

Uber revealed in September 2022 that drivers' and passengers' private information had been stolen by hackers. After infecting the contractor's home computer with malware and exposing those credentials, the attacker purchased the contractor's Uber corporate password on the dark web. The attacker then attempted to log into the contractor's Uber account many times. Each time, the contractor was presented with a request for a two-factor login approval, which at first limited access [14]. The attacker was able to log on successfully after the contractor eventually accepted one.

3) Crypto.com Theft:

In January, Crypto.com, one of the most well-known cryptocurrency exchanges in the world, admitted that 483 of its users had been hacked earlier in the month which resulted in Unauthorized withdrawals totaling 4,836.26 ETH, 443.93 BTC, and roughly US$66,200 in other cryptocurrencies [15]. The platform claims that its risk monitoring systems "detected unauthorized activity on a small number of user-accounts where transactions were being approved without the 2FA authentication control being inputted by the user."

4) The Dragon force Malaysian group attacked several Indian websites:

According to reports, a hacker collective going by the name of "Dragon Force Malaysia" has attacked the websites of the Institute of Science in Nagpur, the National Institute of Agriculture Extension Management, the Delhi Public School, and the Indian embassy in Israel. It belonged to well-known public and private organizations across the nation, and they referred to it as "a special operation on the insult of our Prophet Mohammad by BJP spokesperson Nupur Sharma's comment [16]."

5) US airport's websites hacked by Russian Hackers:

On Monday morning, more than a dozen public-facing airport websites—including those for some of the biggest airports in the country—appeared to be inaccessible, and Russian-speaking hackers took responsibility. The hacker collective Kill-net named several US airports as targets. In response to Russia's invasion of Ukraine in February, it intensified its efforts to target organizations in NATO nations. Kill-net employed a "distributed denial of service" (DDoS) cyberattack, in which hackers bombard computer servers with fictitious web traffic in an effort to bring them offline [17].

Licensed Under Creative Commons Attribution CC BY

Role's Web Application Firewall in Web Application Security:

Injection attacks, XSS, and CSRF, to name a few, are just a few of the many attacks that can affect web applications. These attacks may have serious repercussions, including data theft, a loss of customer confidence, and harm to a company's reputation.

To defend web applications from these attacks, a security solution called a web application firewall (WAF) is used. It serves as a gatekeeper between the web application and the internet, inspecting incoming traffic and blocking any that it deems to be suspicious or malicious [2]. Incoming web traffic to a web application is monitored by a WAF, which identifies and blocks malicious requests before they can access the application itself. WAFs examine HTTP requests and responses for particular patterns and behaviors. Known attack signatures, unusual traffic patterns, or peculiar application behavior are a few examples of these patterns.

The WAF can modify a request or completely reject it if it detects a potential attack and neutralizes it. For instance, the WAF might alter the request if a SQL injection attack is discovered in order to remove the SQL injection code before allowing it to reach the application. This aids in avoiding the attack's compromise of the application. Benefits of WAFs for web application security include protection from both known and undiscovered threats, improved speed, customization, and less chance of data breaches.

Approach in-order to find the vulnerabilities:

The main approach should be followed during testing the web application security testing is given below:

- Identify the target
- Initial recon/fuzzing process on the target
- Vulnerability scanning
- Penetration testing/ Manual testing
- Find vulnerabilities and Exploitation
- Prepare detailed report

Identify the target:

First security analystlooks for the target scope according to given information like some sort of domains of the particular target is given or the wildcard target is given as in scope. Then collect all the subdomains of the target which is given in scope. Subdomain enumeration can be done by many tools and it is more useful to get more subdomains of the target. SUBFINDER, SUBLISTER, AMASS, FINDOMAIN, CRT.SH etc tools are generally used for finding subdomains.

We can use different API keys like github for more deep and accurate scan for getting more subdomains.

Commands:

subfinder -d target.com
sublist3r -d target.com
amass enum -d target.com
findomain -t target.com

After getting subdomain, httpx or httprobe tool is used to remove dead subdomains from the list and it will give running subdomains.

Command:

cat subdomain.txt | httpx | tee running_subdomain.txt

Above command will be save all the running subdomains from subdomain.txt file into running_subdomain.txt file.

Initial recon/fuzzing process on the target:

Recon/Fuzzing refers to a set of processes and techniques, such as foot printing and enumeration, that are used to gather as much information as possible about a target system.

There are some sort of tools are generally used for the initial recon process as well as finding some sensitive information of the target domain which is mention below:

1) Burpsuite
2) Nmap
3) Metasploit
4) Shodan / Censys
5) Nessus & Acunetix
6) Hydra / John the ripper
7) Waybackurls &GF-Patterns
8) SQL Map
9) Nuclei
10) Wappalyzer

After using this sort of tools on the target, most of the required information is collected about the target.

The above-given tools are mainly used during the penetration testing of any web application by a security researcher/tester.

Uses:

1) Burp suite:

It is the most used tool by security researchers and the most important tool in web application testing. During manual testing, before each request is sent to the server it comes in the burp suite and security researchers analyze these requests to find the vulnerabilities like IDOR, Rate Limit, Broken Access Control, etc.

2) Nmap:

Nmap is mainly used to do port scanning on the target web application or list of web applications [1]. It gives all the information about the open ports and the services running on that port. Nmap is also used in the initial recon part for gathering information gathering part. It can also run its script to find some basic vulnerabilities like previous version CVE vulnerability in the web application.

3) Metasploit:

Metasploit is a whole framework that contains thousands of exploits of the vulnerabilities according to the version of the technologies. It includes modules for testing SQL injection, XSS, and file inclusion vulnerabilities. It also includes auxiliary modules for fingerprinting, scanning, and other tasks. It is also used to create payloads also for further exploitation.

4) Shodan & Censys

Shodan and Censys are mostly used to discover publicly accessible assets and services, such as web servers, databases, and other network devices which can lead to

Licensed Under Creative Commons Attribution CC BY

being critical vulnerabilities further. It obtains information about target organizations' infrastructure, such as IP addresses, open ports, SSL/TLS certificates, and other relevant details [1]. This information can be used to identify potential attack vectors, misconfigurations, or vulnerabilities that can be exploited.

5) Nessus & Acunetix

Nessus and Acunetix is a network and web application vulnerability scanner that can also check for vulnerabilities in web applications [1]. It can run several tests to find flaws like SQL injection, XSS, and others.

Nessus uses a sizable database of publicly known vulnerabilities to find potential problems. Acunetix has a user-friendly interface and can be used by both beginners and advanced users.

6) Hydra & John the ripper

Hydra is used for network login that supports numerous protocols, including HTTP, FTP, Telnet, and others. It is used to perform brute-force attacks and dictionary attacks on password-protected services.

Another tool for password cracking is John the Ripper, which supports hybrid, brute-force, and dictionary attacks as well as other methods. For password cracking on Windows, Unix, and other systems, it supports a wide range of platforms. John the Ripper has a reputation for being quick and adaptable.

7) Waybackurls & GF-Patterns

Waybackurls is very useful tool in the initial information gathering phase. It used to collect all the URLs of the target domain.

After collecting all the URL's, GF-Patterns is generally used to sort all the URL's according to vulnerabilities like XSS, SSRF, SQLI, SSTI, s3-bucket etc.

8) SQL Map

The best automation tool for locating SQL injection vulnerabilities in web applications is SQLMAP. It entails using customization to varying degrees of risk. Additionally, it is utilized to recognize time-based SQL injection as well as blind SQL injection. Attackers can also use it to take advantage of SQL injection flaws and obtain unauthorized access to private information kept in databases.

9) Nuclei

The open-source software called Nuclei is used to find security flaws. It sends the request to the server primarily in nuclei using a template of various vulnerabilities, CVEs, and technology stack. Numerous thousands of templates have previously been created and are used to find vulnerabilities. The tool is incredibly effective for customized functioning in accordance with the demands of the target domain because we are able to add our own custom templates to it.

10) Wappalyzer

The most popular extension used by security researchers to determine which services and technologies are utilized by the target domain is Wappalyzer [1]. Finding the weaknesses of that certain technology version is quite simple, according to the technologies.

a) Vulnerability scanning:

When automated scanners are used to find the system's vulnerabilities, vulnerability scanning is a component of testing. Scripts customized for each vulnerability are built into vulnerability scanners. Scripts are executed on the target, and after the scan is finished, a list of vulnerabilities discovered in the target is displayed.

Some vulnerability scanner tools used in automation scripts to identify vulnerabilities include Nessus, Acuntix, Nikto, and Openvas [1].Not all vulnerabilities will necessarily be discovered while utilizing vulnerability scanners. This makes manual testing the primary and most effective way for identifying vulnerabilities.

Penetration testing/ Manual testing:

The main component of web application penetration testing is manual testing. Each vulnerability in the target system is manually discovered by security testers. Therefore, it will be more beneficial to target the weakness with greater accuracy.

All vulnerabilities that are missed by automatic scanner tools are tested for manually. It is crucial because the manual technique relies on actual circumstances involving the specific vulnerability.

Security testers used recon data such as the type of technology being used, the open ports, and any sensitive information that is available to the general public [1].

In particular, file inclusion vulnerabilities (LFI, RFI), cross-site scripting (XSS), SSRF, SSTI, RCE, and injection vulnerabilities (SQLI, Graphql injection) such vulnerabilities that are listed in the most severe and high priority vulnerabilities can be find through manual testing.

b) Find vulnerabilities and Exploitation:

After performing manual testing on the system, a security tester will discover any weak points or defects, which are referred to as vulnerabilities. When a security tester identifies a vulnerability, they attempt to exploit it in order to get more advantages over the system, such as unauthorized access or other harmful acts. It will make the specific vulnerability that was discovered more severe.

Prepare detailed report:

After find the vulnerability and its exploitation, complete report is being prepared by the security tester which includes,

- Which vulnerability is found?
- Where the vulnerability is exists? (Location of the bug)
- Description of the vulnerability
- Steps to reproduce
- Proof of concept (POC)
- Impact of the vulnerability
- Mitigations/ Remediation

Licensed Under Creative Commons Attribution CC BY

Web Application Security Vulnerabilites:

1) Remote Code Execution (RCE):
RCE is the most severe vulnerability in web application security. RCE give whole remote control of the server to the attacker which can be very much crucial. RCE allows to execute any arbitrary code to the server. This vulnerability is mostly occurred when user input data is not properly handled by the server and give the details what the user ask for[1].

The reason for occurring RCE vulnerabilities:
Insecure Input Validation: Failure to properly validate user input can result in code injection attacks, in which an attacker injects malicious code that the system executes.

Inadequate handling of external input, such as command-line arguments or network data, might allow attackers to inject arbitrary code into the susceptible application, which is then executed.

Insecure Deserialization: Flaws in the deserialization process can allow an attacker to deliver a specifically constructed serialized object that, upon deserialization, executes arbitrary code.

Attacking Scenario with Example:

I) File upload vulnerability to RCE
One website has file upload functionality for the users. Where users can upload the images, document for save the file on the server. Here, attacker use one PHP file which contains the malicious code which give the full remote access of the server machine to attacker. Attacker use some tricks like double extension to bypass the filter of the upload functionality and upload that PHP shell file to the server. It executes on the server and attacker will get the full access of the system files.

II) Server-side template injection (SSTI) leads to RCE
For SSTI, Attacker manually try all the parameters like search boxes, form fields, and URL parameters, where user-supplied data may be reflected back in the application's response. In these types of fields, mostly attacker try {{7*7}} types of payloads to identify this vulnerability. If output got reflected as a 49 it means input parameter is executed on the server side and confirms the vulnerability. After confirming the SSTI vulnerability, attacker try to execute the malicious code of script to get the access for the server-side files of sensitive information which give the access of server side and RCE will be executed.

III) Local file inclusion (LFI) to RCE
While looking for LFI vulnerability, for example URL will be like https://www.example.com/?file=/img/3022.jpeg
for the following URL here attacker try to find LFI vulnerability and put the value for eg. as /../../../etc/passwd in file parameter [4]. If attacker got the password list in response it means this parameter is vulnerable to LFI vulnerability.

After getting LFI, attacker try to exploit LFI into RCE and try to execute the payload in order to get the access of sensitive files which give the remote access of the server side.

Impact:
The RCE vulnerability has a significant impact because it can result in the entire compromise of a system, allowing an attacker complete access and control. The RCE vulnerability can be exploited by a hacker to steal confidential data, edit or remove data, or even use the compromised system as a launch pad for assaults on other systems.

Remediation:
To solve Remote Code Execution (RCE) vulnerabilities, user input must be verified and sanitized to prevent malicious input from being executed. Implementing secure coding techniques such as input validation, output encoding, and correctly establishing server permissions can also help to limit RCE risk. Software upgrades and the application of security patches can also help to avoid the exploitation of known vulnerabilities.

2) Injection Vulnerability:
Injection vulnerability like SQL injection, command injection is also very critical vulnerability. It can lead to full access of database or other sensitive file exposure. Attackers can execute arbitrary commands or queries against the interpreter by taking advantage of injection vulnerabilities, which may allow them to compromise the entire system or gain unauthorized access to confidential information.

(i) SQL Injection:

Attacking Scenario:
Malicious SQL queries are typically inserted by an attacker into an input field like a search bar or login form, which the application will process and send to the database.

Attacking scenario,
Attacker will enter the SQL query into a login form. While entering the username attacker will enter malicious SQL query like ' or 1=1-- which will try to bypass the login credential and last (--) in SQL payload comment out the password field so that unauthorized login will be possible into the account [4]. This query will trick the application into thinking that the attacker has already authenticated and then the attacker take advantage of it.

Once login is completed, attacker will try to escalate the severity of the vulnerability and try to get the details of the database like below:
https://www.example.com/?query= SELECT * FROM user

When an attacker enters erroneous SQL queries into user-entry fields, the application's database executes these commands.This is known as classic SQL Injection. As a result, the attacker could have the ability to change the database, obtain sensitive information, edit data, or carry out other illicit actions.

The application does not reveal database errors or the outcomes of the injected SQL queries [4] in this type of blind SQL injection attack. In order to get information or perform unauthorized activities, the attacker uses conditional

Licensed Under Creative Commons Attribution CC BY

queries or time delays to infer information from the application's response.

Blind SQL injection attack that infers information from time delays in the application's response is known as "time-based blind SQLI." Attacker injects SQL queries that cause the application to delay in responding if the injected condition is true [4].

Error-Based SQL Injection: This attack uses database error messages to gather details about the database's structure or data [4]. Error-Based SQL Injection. The attacker deliberately inserts SQL queries that result in problems, then analyses the error reports to extract crucial data.

Union-Based SQL Injection: In this attack, using the UNION SQL operator, the outcomes of two or more database searches are merged into a single result set. [4]. By introducing a false UNION statement, the attacker can obtain information from other database tables or infer information.

Depending on the need, SQLMAP has degrees of risk ranging from 1-3 as well as levels from 1-5 for deeper scans.

SQL Payloads:

- ' or 1=1--
- '; drop table users;--
- '; select sleep(10);--
- ' union select 1,2,3--
- ' union select null,2,3--
- ' and sleep(5)--
- ' or sleep(5)-- [4]

Impact:
An attacker may be able to extract, modify, delete, or even seize control of the entire database server thanks to SQL injection's serious consequences. Sensitive information, including user credentials, private information, financial information, or intellectual property, may be exposed as a result of SQL injection. In some circumstances, SQL injection can also result in denial-of-service attacks or system failures.

Remediation:
Mitigating SQL injection vulnerabilities requires a comprehensive strategy to ensure strong protection against these types of attacks. Firstly, it is essential to employ thorough input validation and utilize parameterized queries to effectively filter and sanitize user input. This ensures that any malicious SQL commands are prevented from executing.

(ii) Cross-Site Scripting:
The attacker can inject malicious code into a victim's browser, typically in the form of a script or HTML. The code will run in the victim's browser, allowing the attacker to steal user passwords and hijack user sessions. XSS attacks often occur when an application fails to validate or sanitize user input adequately. There are three types of XSS attacks:

Reflected XSS:
It occurs when malicious code is injected into an application and then reflected to the user's browser. It will utilize this to steal the user's cookie, session id, and so on.

Scenario for attack:
https://example.com/search?url=abcde

In this case, the attacker replaces ‘abcde’ with an XSS payload, and after the request is submitted, if this page displays the alert-box, the reflected XSS vulnerability is validated. The user's cookie and session id are then stolen, among other things.

Stored XSS:
In this case, the attacker put the malicious code in server side website and website saved on the application's server and served to all users that visit the affected page [5]. The XSS popup is not reflected on the browser or user screen in this case but XSS will be triggered.

DOM-based XSS:
In a DOM-based XSS attack, malicious code is injected into the affected page's Document Object Model (DOM) and executed when the page is loaded.

The victim visits a URL that the attacker has created with malicious JS code, and he receives a response free of harmful content. The malicious code is executed at the client side, and the attacker has access to the client side's sensitive data. [5].

This allows an attacker to install malicious code into the victim's web browser, potentially stealing personal information or performing unauthorized actions.

Impact:
XSS vulnerabilities can cause extensive and detrimental consequences. Through the injection of malicious scripts into web pages, adversaries can manipulate user sessions, extract confidential data, or deface websites. Exploiting XSS vulnerabilities allows unauthorized access to user accounts, jeopardizing sensitive information like passwords, financial details, or private communications.

Remediation:
To address XSS vulnerabilities, various actions can be taken. Firstly, it is essential to implement input validation to ensure that user-provided data is properly sanitized and free from any malicious scripts. Secondly, applying output encoding is crucial when presenting user input or dynamic content to prevent it from being treated as executable code. Additionally, the adoption of a Content Security Policy (CSP) can help restrict the execution of scripts and minimize the impact of XSS attacks

3) Server Side Request Forgery (SSRF):
A specific kind of web vulnerability called Server-Side Request Forgery (SSRF) enables an attacker to send requests on behalf of the server or application that is being targeted. In an SSRF attack, the attacker can force the application to send HTTP requests to a pre-specified, arbitrary domain or IP address. As a result, the attacker may be able to get

Licensed Under Creative Commons Attribution CC BY

around network limitations and access systems or sensitive data that shouldn't be accessible via the open internet.

Steps to Reproduce:

a) First, we have to identify the parameters on which interaction from the server is possible. Burp-collaborator or interests is mostly use to identify the interaction with the server.
b) Once the interaction is complete with the server, in burp collaborator HTTP request is appeared which discloses the IP address of the server system.
c) If the system is using some cloud service like AWS or google cloud then metadata of that cloud service can also be disclosed using this vulnerability.

Scenarios:

1) SSRF leads to AWS metadata expose:

An attacker tried to exploit a Server-Side Request Forgery (SSRF) vulnerability. The attacker creates a malicious URL that connects to the AWS metadata service endpoint, a well-known URL for retrieving metadata about an AWS instance http://169.254.169.254/latest/meta-data/.

The IP address 169.254.169.254 is reserved for link-local addressing and is used by AWS as the metadata service IP address [7]. An attacker can gain access to many types of sensitive information by appending other paths or parameters to this basic URL.

Well-known Sensitive Information directory path:
http://169.254.169.254/latest/meta-data/iam/security-credentials/

Unaware of the potential security dangers, the server-side code performs an HTTP request to the supplied URL. Because the malicious URL points to the AWS metadata service endpoint, the server-side malware collects the AWS instance's sensitive metadata, such as access keys, security group information, and other potentially sensitive information using above URLs.

2) SSRF leads to internal port scan disclosed:

The attacker uses the SSRF vulnerability to scan and enumerate open ports within the internal network in the scenario of SSRF leading to internal port scanning. To identify accessible services, the attacker redirects the SSRF request to internal IP addresses and specified port numbers rather than the AWS metadata service endpoint.

The attacker takes advantage of this by sending a malicious URL including an internal IP address and a port number, such as:
http://site.com/endpoint?url=http://192.168.0.1:8080

The attacker is attempting to scan the internal IP address 192.168.0.1 on port 8080 in this scenario. Because of the SSRF vulnerability, the susceptible application will make a request to the provided URL.

If the target internal system has open port 8080, the vulnerable application will connect successfully, showing that the port is open [7]. The attacker can then deduce that a service is running on that port and continue scanning other ports or performing more reconnaissance to learn more about the internal network.

By exploiting the SSRF vulnerability in this manner, the attacker is able to do internal port scanning, locate exposed services, and perhaps discover misconfigurations or vulnerabilities that can be used to gain unauthorized access or launch additional attacks within the internal network [7].

Impact:

The impact of an SSRF attack can be severe, depending on the capabilities of the vulnerable server and the data it has access to. An attacker can potentially access internal systems, read confidential data, and launch attacks against other systems on the network.

Remediation:

To prevent SSRF attacks, developers should validate all user-supplied URLs and ensure that they cannot be used to access sensitive internal resources or private IP addresses. Server administrators should also disable or restrict the use of protocols such as file:// or gopher:// that can be used in SSRF attacks.

4) IDOR (Insecure Direct Object Reference):

When a program fails to implement proper authorization or validation checks while using user input to access objects, it exposes an "insecure direct object reference" (IDOR) vulnerability. Exploiting this vulnerability, attackers can gain unauthorized access to sensitive information and features, compromising the security of the application [8].

Attack Scenario: An attacker can manipulate the ID value in the URL to bypass access restrictions and access unauthorized resources. This is especially prevalent in applications that utilize sequential numbers to identify user accounts, orders, or other resources. In some cases, attackers may employ brute-force techniques to discover valid object references.

For instance, let's consider an online marketplace with an "order" feature that allows users to view their previous orders. Each order is assigned a unique ID, which is included in the URL to fetch the order details. However, due to an IDOR vulnerability, an attacker can modify the ID in the URL to access other users' order details.

To exploit this vulnerability, the attacker first logs into their own account and identifies the ID of one of their own orders. They can then alter the ID in the URL to the ID of another user's order, granting them access to their personal and financial information.

For example, the URL for viewing an order might appear as follows: https:/abc.com/view_order?id=0123. By changing the ID to 3658, the attacker gains access to the details of a different order [8].

Impact:

An Insecure Direct Object Reference (IDOR) attack can have severe consequences as it grants unauthorized access to sensitive data or enables malicious actions [8]. An attacker exploiting an IDOR vulnerability may exploit it to gain

Licensed Under Creative Commons Attribution CC BY

unauthorized access to confidential user information, manipulate or delete user data, or even take control of user accounts.

Mitigation:
Access controls should be implemented based on user privileges and permissions to prevent IDOR vulnerabilities. It is crucial to ensure that users can only access resources for which they have explicit authorization. The application should employ randomized and unique identifiers that are difficult for attackers to predict or manipulate.

5) Broken Access Control (BAC):

Attacking scenario:
An attacker can identify broken access control vulnerabilities by examining the application's source code, APIs, or network traffic. Once a vulnerability is identified, an attacker can use various techniques to exploit it, such as brute-forcing passwords, guessing object references, manipulating input parameters, or using session fixation attacks.

The flaw was caused by a misconfiguration in the web application's access controls, especially the permissions and limitations on resource access. To get access to the sensitive data, the attacker exploited this misconfiguration and used a Server-Side Request Forgery (SSRF) vulnerability.

The attacker made fraudulent queries to the web application, fooling it into requesting metadata from the cloud hosting provider. The attacker was able to get temporary access credentials for the Capital One environment by altering the requests, which provided them enhanced rights [9].

The attacker used the stolen credentials to access and exfiltrate sensitive consumer information such as names, addresses, credit scores, and social security numbers. The event demonstrated the serious implications of a Broken Access Control vulnerability.

Impact:
It can allow an attacker to gain unauthorized access to sensitive data or functionality and it can lead to privilege escalation of that access to increase the severity of the vulnerability. Mainly the impact of a broken access control vulnerability depends on the specific application and the data it stores.

Remediation:
Implementing strategies like role-based access control, the least privileged approach, input validation and sanitization, access control testing, access control logging and monitoring, routine system updates and patches, and security awareness training are all part of fixing broken access control vulnerabilities. By utilizing these methods, businesses can lower the risk of unauthorized users accessing and manipulating their data and systems.

6) Cross-Site Request Forgery (CSRF):
The attacker creates a request that appears to be coming from a reliable source, like a user's browser, and sends it to the target website. When the victim accesses the website, their browser sends the malicious request unintentionally along with their login information, enabling the attacker to act on their behalf [10].

Attacking Scenario:
The attacker developed a malicious website or constructed a malicious link and lured the victim (a Facebook user) into visiting the website or clicking the link while logged into their Facebook account.

The malicious website or link, unbeknownst to the user, featured a hidden form that made a request to Facebook's email address change tool.

The hidden form filled in the victim's Facebook account information, such as their user ID or email address, as well as the attacker's desired email address.

When the victim visited the infected website or clicked the link, their browser submitted the hidden form, which triggered a request to Facebook's email address change tool.

Because the victim was already logged in to Facebook, the request seemed valid, and Facebook processed it.

Impact:
Using a CSRF attack, an attacker may do unauthorized activities on the victim's behalf without the victim's knowledge or agreement [10]. This category includes changing account information, adding or deleting products from a shopping cart,making fraudulent payments or contributions, manipulating data or settings, and other actions. The consequences might range from minor annoyances to major security breaches that sensitive data, cause monetary losses, or destroy a company's brand.

Remediation:
Developers may adopt approaches including implementing a CSRF token, validating the referrer header, utilizing Same cookies, HTTP headers, restricting the scope of sensitive actions, and keeping the program updated [10]. Developers may limit the danger of CSRF attacks and safeguard their apps by applying these approaches.

7) Open Redirect
An open redirect vulnerability is a sort of security weakness that enables an attacker to reroute a victim to any website or page by using the redirect URL parameter of a susceptible website [11]. This error occurs when a website fails to validate user input or change URLs.

Attacking Scenario:
Let's imagine that abc.com’s login page at
http://abc.com/login.php?redirect=http://attacker.com
has an open redirect vulnerability. The victim or user will be sent to the login page of "example.com" when they click the link, but the redirect parameter will take them to the attacker's website, "attacker.com." The attacker can steal the victim's or user's login information and use it for their own objectives, which may be more detrimental to the victim or user [11].

Licensed Under Creative Commons Attribution CC BY

In this situation, an attacker can also reroute the user to their website in order to collect private data such as the victim's browser cookie and login.

XSS, SSRF, and account takeover vulnerabilities can also result from open redirect vulnerabilities.

Attacking Scenario for Open redirect to SSRF:
Attackers created a product listing on ABC.com with an external link and a URL parameter. The URL parameter was altered by the attacker, who made it go to a malicious URL under their control. They did present it as a legitimate internet link, though.

When users clicked on the product listing, which appeared genuine and reliable, they were directed to the modified URL.

Unbeknownst to the users, the modified URL started a server-side request within the architecture of ABC.com. An internal system or an API endpoint that was open to SSRF attacks received the request [7].

Due to the SSRF vulnerability, the attacker had access to private data on ABC.com's internal network and could send queries to other internal systems that were vulnerable.

Attacking Scenario for Open redirect to XSS:
On ABC.com, users have the option to include external links in their postings or profiles. These links are meant to direct visitors to other websites.

By changing the redirect parameter and including a script payload, an attacker creates a malicious link that appears to have come from a reputable source.

The attacker may present the malicious link as important information or as a helpful resource in an effort to trick the victim into clicking on it.

The user is sent to a website that appears legitimate although is really under the attacker's control when they click the link.

The attacker's script payload encoded in the URL parameter is performed as part of the redirect process on the safe abc.com website.

The malicious script can now change or steal the user's session cookies and other sensitive information because it has access to them [6].

Impact:
Attackers can create a link that appears legitimate but redirects any user to a malicious website or address. This can be applied to phishing attacks, in which a hacker poses as a reputable website to steal sensitive data or user credentials.

Remediation:
It is required to validate all the user inputs that could lead to a redirection. The URLs being redirected should be only from trusted and approved sources. In the redirect URL, stay away from using user-supplied parameters. whenever possible, switch to server-side redirection from client-side redirection.

8) Broken Authentication and Session Management
Broken authentication and session management vulnerabilities happen when the mechanisms for authentication or session management are weak, allowing attackers to steal user credentials and session tokens or get around the authentication process. Weak password policies, session hijacking and fixation, poor authentication procedures, inadequate encryption, and a lack of a multi-factor authentication system can all contribute to this issue. As a result, the attackers are able to get unauthorized access to the system and server information.

Attacking Scenario:
User2 frequently shops on the internet at the site "ABC.com" ABC.com 's authentication and session management system is flawed, putting consumers at risk for security breaches. User1 can use this vulnerability to access user accounts without authorization and carry out nefarious deeds [12].

Exploitation:
User1 learns that ABC.com's login process does not enforce account lockouts after numerous failed login attempts or necessitate stringent password restrictions.

Using a widely used password, User1 registers on ABC.com as a user and logs into her own account. User1 observes that ABC.com makes use of session IDs that are neither generated or managed securely. User1 develops a strategy to take advantage of this weakness. She wants to break into User2's account without authorization and steal her personal data.

User1 begins by impersonating a ABC.com promotional email in order to send User2 a phishing email. A link in the email sends User2 to a false login page that is disguised to look like ABC.com's actual login page.

User2 submits her username and password on the fake login page, thinking she is logging into her ABC.com account, not seeing that it is a phishing attempt.User2's login information is intercepted by the fake login page and sent to User1's server [12].
User1 attempts to log in using User2's login information on the genuine ABC.com website after receiving User2's login information.

Due to improper techniques for identifying suspicious login patterns, ABC.com's defective authentication system is unable to recognize or prevent User1's login attempts [12].User1 is able to successfully connect into User2's account and gain access to all of the personal data, order history, and payment information etc.

Impact
This is an extreme circumstance that could result in unauthorized access to sensitive information or features, such as user accounts, credit card data, or intellectual property. Attackers may use this vulnerability to hijack user sessions, evade authentication, steal passwords, or

Licensed Under Creative Commons Attribution CC BY

masquerade as other users, putting their own identities at risk of theft, monetary loss, or reputational damage.

Remediation

- Implement the policies for secure passwords.
- Using features for multi-factor authentication
- After the user logs out, the running session should time out promptly.
- Use HTTP-only and secure flags for sending sensitive data, such as cookies.

2. Conclusion

Finally, the topic of web application security is crucial in securing sensitive data and retaining user confidence. Finding vulnerabilities necessitates a combination of proactive steps and rigorous testing. This includes doing rigorous security assessments, such as human code reviews, penetration testing, and the use of automated technologies. Developers may minimize typical vulnerabilities such as injection attacks, cross-site scripting, and cross-site request forgery by using security best practices such as input validation, safe code, and session management. Continuous monitoring and patch management are critical for dealing with evolving threats. Using web application firewalls as a defense mechanism can aid in the prevention of attacks and reduce the danger of data breaches. Collaboration between security specialists, developers, and system administrators is critical in putting in place effective security policies.

References

[1] Nagendran, K., A. Adithyan, R. Chethana, P. Camillus, and KB Bala Sri Varshini. "Web application penetration testing." *Int. J. Innov. Technol. Explor. Eng* 8, no. 10 (2019): 1029-1035.

[2] Clincy, Victor, and Hossain Shahriar. "Web application firewall: Network security models and configuration." In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 835-836. IEEE, 2018.

[3] Mohammad, S., and Soulmaz Pourdavar. "Penetration test: A case study on remote command execution security hole." In *2010 Fifth International Conference on Digital Information Management (ICDIM)*, pp. 412-416. IEEE, 2010.

[4] Begum, Afsana, Md Maruf Hassan, Touhid Bhuiyan, and Md Hasan Sharif. "RFI and SQLi based local file inclusion vulnerabilities in web applications of Bangladesh." In *2016 International Workshop on Computational Intelligence (IWCI)*, pp. 21-25. IEEE, 2016.

[5] Ma, Limei, Dongmei Zhao, Yijun Gao, and Chen Zhao. "Research on SQL injection attack and prevention technology based on web." In *2019 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, pp. 176-179. IEEE, 2019.

[6] Liu, Miao, Boyu Zhang, Wenbin Chen, and Xunlai Zhang. "A survey of exploitation and detection methods of XSS vulnerabilities." *IEEE access* 7 (2019): 182004-182016.

[7] Luo, Haibo. "Ssrf vulnerability attack and prevention based on php." In *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pp. 469-472. IEEE, 2019.

[8] KumarShrestha, Ajay, Pradip Singh Maharjan, and Santosh Paudel. "Identification and illustration of insecure direct object references and their countermeasures." *International Journal of Computer Applications* 114, no. 18 (2015): 39-44.

[9] Hassan, M., M. Ali, T. Bhuiyan, M. Sharif, and S. Biswas. "Quantitative assessment on broken access control vulnerability in web applications." In *International Conference on Cyber Security and Computer Science 2018*. 2018.

[10] Kombade, Rupali D., and B. B. Meshram. "CSRF vulnerabilities and defensive techniques." *International Journal of Computer Network and Information Security* 4, no. 1 (2012): 31.

[11] Wang, Xianbo, Wing Cheong Lau, Ronghai Yang, and Shangcheng Shi. "Make redirection evil again: Url parser issues in oauth." *Black Hat Asia* (2019).

[12] Hassan, Md Maruf, Shamima Sultana Nipa, Marjan Akter, Rafita Haque, Fabiha Nawar Deepa, Mostafijur Rahman, Md Asif Siddiqui, and Md Hasan Sharif. "Broken authentication and session management vulnerability: a case study of web application." *Int. J. Simul. Syst. Sci. Technol* 19, no. 2 (2018): 1-11.

[13] Twitter Official, "An incident impacting some accounts and private information on Twitter", https://privacy.twitter.com/en/blog/2022/an-issue-affecting-some-anonymous-accounts

[14] Uber security Officials, "Uber data breach" https://www.uber.com/newsroom/security-update/

[15] Crypto.com security reports, "unauthorized crypto withdrawals approximately US$66,200 in other cryptocurrencies."https://crypto.com/product-news/crypto-com-security-report-next-steps

[16] "Insult on Prophet Muhammad: Malaysian hackers attack over 70 Indian government websites" https://sea.mashable.com/tech/20569/insult-on-prophet-muhammad-malaysian-hackers-attack-over-70-indian-government-websites

[17] Lori Aratani, "Hackers knock some U.S. airport websites offline" https://www.washingtonpost.com/transportation/2022/10/10/hackers-cyber-attack-airport-website

Licensed Under Creative Commons Attribution CC BY