

Security Assessment of an Online Judge Web Application

Authors



Md Minhazul Islam (29)
Ruddro Mohammad Khorshad Aziz (32)
Sabuj Kumar Modak (08)

University of Information Technology & Sciences
Department of IT



Abstract

A website is a compilation of web pages and related material that is accessible through a single domain name and made available on at least one web server. Web sites have an identity called Domain name and a URL to find. Though the website is related to the collection of some web pages, there are many more in it. A website's structure is always changing as new technologies and features are added in order to improve the user experience and produce high-quality results. When an object is created there is always a risk of being destroyed or damaged. Web sites also have the risk of it. Attacks on websites happen frequently every day. A website vulnerability is a flaw or improper configuration in the code of a website or online application that enables an attacker to take some level of control over the website and maybe the hosting server. To prevent this here comes Web security. Web security consists of objectives that can help to ensure security for a web site. Vulnerability is a software problem, configuration error, or other weakness in the website/web application, its elements, or its procedures constitutes a website vulnerability. Attackers can get unauthorized access to an organization's systems, processes, and mission-critical assets thanks to web application flaws.

1. Introduction:

The internet is ever-expanding, and there are currently around 2 billion different websites in total. On average, just 5% of websites on the internet are active. The remaining 4–5 are inactive, which means they haven't had any updates or new postings in a while. The typical person spends a large portion of their day on websites. The typical internet user in the US visits more than 100 different websites per day.

According to estimates, cybercrime costs the American economy \$3.5 billion annually. According to some estimates, between 30 000 and 50 000 websites are hacked daily. Small firms are the target of 43% of cyber-attacks. Daily growth in both the population and the necessity of website security are both trends. Automated tools like vulnerability scanners and botnets are used to automatically exploit the majority of vulnerabilities. Cybercriminals use specialized programs that trawl the internet for certain Systems, such as Word Press or Joomla,

searching for widespread and well-known vulnerabilities. Once identified these flaws are subsequently used to gain access to vulnerable websites and steal data, send spam and other undesirable information, or deface them. Websites can be infected, data can be collected, and in certain situations, computer resources can even be taken over by malicious software. When an attacker gains access to a site, they can use it to reroute traffic and infect users with malicious software. This means that if your website is not secured, hackers may exploit it to infect users with malware.

Because of all of this, every day, the need for internet security grows, making it crucial to safeguard your website and the data it contains right away. Our work is to use those scanners on our targeted website and find the vulnerabilities of that website, then research how to prevent them to make our targeted website secure.

1. Objectives:

Every piece of work has goals. That refers to the actions we will do, the lessons we will acquire from doing this work, and the advantages of doing this work. This work's primary goals are also listed. In the security industry, "Vulnerability Assessment" is the term used to describe the work we undertake. An assessment of a network's vulnerabilities seeks to identify them and provide the best mitigation or remedy to lessen or eliminate the risks.

Utilizing automated network security scanning technique. An information system's security flaws are systematically examined during a vulnerability assessment. It determines whether the system is vulnerable to any known flaws, rates their seriousness, and, as necessary, makes remedy or mitigation recommendations.

3. System Model

Every website includes a blueprint for how the operation will be managed and the data flow diagram will develop. We can get a clear sense of how the process works on the website from the model, and we'll use that to determine the operating order. That is also present on our target website, [MBSTU Online Judge](#). That shows the process flow of that website.

Examples of threats that vulnerability assessment can include:

- Code injection attacks include SQL injection, XSS, and others.
- Increase in privileges as a result of inadequate authentication methods.
- Software that includes default settings that aren't secure, like admin passwords that are easy to guess.

Vulnerability assessment is to protect the system's and website's privacy. The owner of the system might not be aware of a vulnerability in it. These can be used by hackers to access the server and do damage. In order to prevent those vulnerabilities from appearing on our targeted website, we must first identify the high threat vulnerabilities, research how they operate, and identify their primary purposes.

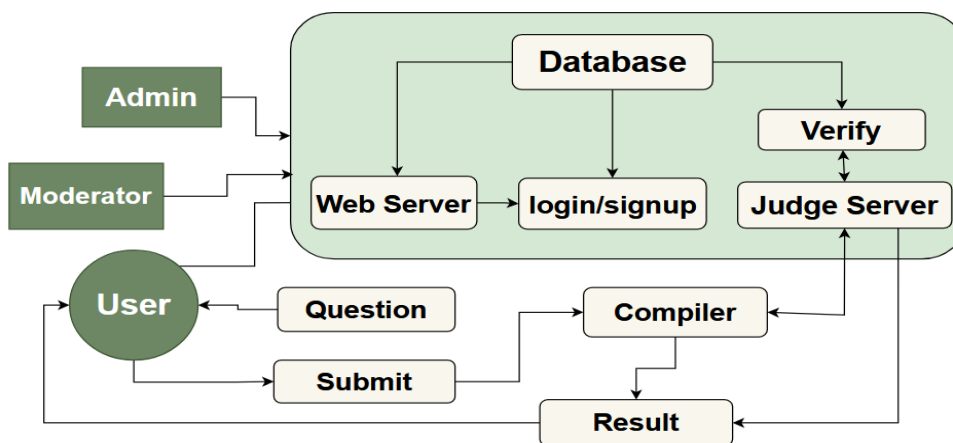
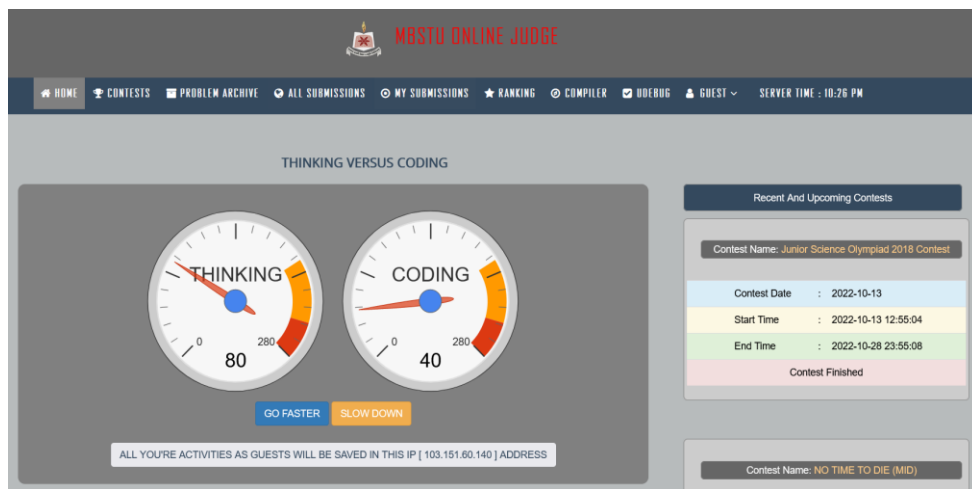


Fig: Website Front page & System model

4. Vulnerability Scanning

Vulnerability scans examine particular areas of your network for faults that threat actors are likely to use to gain access or conduct a recognized sort of cyber-attack. When utilized appropriately, they can add a crucial layer of protection to assist protect the sensitive data held by your firm. The purpose of external scanning is to determine what a hacker would see if he attempted to probe the website MBSTU Online Judge. Tool use is necessary for the vulnerability scan. The tool used to scan MBSTU Online Judge are: Nmap & Acunetix.

4.1 Nmap scanning

Network Mapper is referred to as Nmap. A network's IP addresses and ports can be scanned with this free and open-source Linux command-line tool in order to find installed programs. Network administrators can use Nmap to identify the devices that are connected to their network, find open ports and services, and find security holes.

For this website we tried Nmap host discovery and then Nmap Vulnerability scanner.

```
(namikaze@kali)-[~]
└─$ nmap 103.28.121.75
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-11 16:09 +06
Nmap scan report for 103.28.121.75
Host is up (0.017s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 5.12 seconds
```

Fig: Nmap Port Scan

```
(namikaze@kali)-[~/usr/share/nmap/scripts]
└─$ nmap -sV --script nmap-vulners/ 103.28.121.75
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-11 16:22 +06
Nmap scan report for 103.28.121.75
Host is up (0.016s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:7.6p1:
|     EXPLOITPACK:98FE96309F9524B8C84C508837551A19 5.8 https://vulners.com/exploitpack/EX
FE96309F9524B8C84C508837551A19 *EXPLOIT*
|     EXPLOITPACK:5330EA02EBDE345BFC9D6DDDD97F9E97 5.8 https://vulners.com/exploitpack/EX
30EA02EBDE345BFC9D6DDDD97F9E97 *EXPLOIT*
|     EDB-ID:46516 5.8 https://vulners.com/exploitdb/EDB-ID:46516 *EXPLOIT*
|     EDB-ID:46193 5.8 https://vulners.com/exploitdb/EDB-ID:46193 *EXPLOIT*
|     CVE-2019-6111 5.8 https://vulners.com/cve/CVE-2019-6111
|     1337DAY-ID-32328 5.8 https://vulners.com/zdt/1337DAY-ID-32328 *EXPLOIT*
|     1337DAY-ID-32009 5.8 https://vulners.com/zdt/1337DAY-ID-32009 *EXPLOIT*
|     SSH_ENUM 5.0 https://vulners.com/canvas/SSH_ENUM *EXPLOIT*
|     PACKETSTORM:150621 5.0 https://vulners.com/packetstorm/PACKETSTORM:150621 *E
|     EXPLOITPACK:F957D7E8A0CC1E23C3C649B764E13FB0 5.0 https://vulners.com/exploitpack/EX
57D7E8A0CC1E23C3C649B764E13FB0 *EXPLOIT*
|     EXPLOITPACK:EBDBC5685E3276D648B4D14B75563283 5.0 https://vulners.com/exploitpack/EX
DBC5685E3276D648B4D14B75563283 *EXPLOIT*
|     EDB-ID:45939 5.0 https://vulners.com/exploitdb/EDB-ID:45939 *EXPLOIT*
|     EDB-ID:45233 5.0 https://vulners.com/exploitdb/EDB-ID:45233 *EXPLOIT*
|     CVE-2018-15919 5.0 https://vulners.com/cve/CVE-2018-15919
|     CVE-2018-15473 5.0 https://vulners.com/cve/CVE-2018-15473
|     1337DAY-ID-31730 5.0 https://vulners.com/zdt/1337DAY-ID-31730 *EXPLOIT*
|     CVE-2021-41617 4.4 https://vulners.com/cve/CVE-2021-41617
|     CVE-2020-14145 4.3 https://vulners.com/cve/CVE-2020-14145
|     CVE-2019-6110 4.0 https://vulners.com/cve/CVE-2019-6110
```

Fig: Nmap Vuln Scan

Nmap port scan shows the report of which ports are active in the network or website. From the report we see that there are two ports open in that website, one is 22/TCP which state is OPEN and the service defined to SSH. Secure Shell (SSH) is a network protocol that allows users to access the server remotely. SSH protocol's default settings are to listen on TCP port 22 for connections.

Other one is 80/TCP which state is open and it accepts HTTP service. Hypertext Transfer Technology, a widely used internet communication protocol, is assigned port number 80. (HTTP). Unencrypted web pages are transmitted and received using this network port by default.

It indicates that utilizing this port, uuencoded data transmission occurs between the user's browser and the server. From this there is a security issue arrives which we indicate as Insecure Http login process. And we search for vulnerability using Nmap, and lot of report arrived, further we scan through Acunetix for a better and specified view.

4.1.1 Http Insecure login process

A crucial element discovered for a website in the modern era when scanning the MBSTU Online Judge with Nmap. This site uses http port, not an https one, as we can see.

Due to the wide range of attacks that can be used to extract a user's password from them, serving login forms over HTTP is particularly risky. By sniffing the network or altering the served page while it is being transmitted, network eavesdroppers could steal a user's password.

We know that with encryption and authentication, HTTPS equals HTTP.

The sole distinction between the two protocols is that HTTPS. Employs TLS (SSL) to encrypt and digitally sign requests and answers made using regular HTTP.

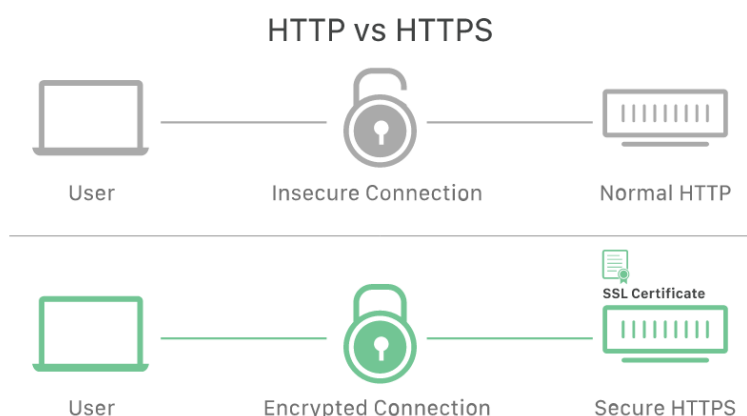
Public key cryptography is a technique used by TLS; it consists of two keys—a public key and a private key—with the public key being distributed to client devices via the server's SSL certificate. The public and private keys are used by the client and server to agree on new session keys to encrypt subsequent communications once the client and server establish a connection.

Then, using these session keys, all HTTP requests and responses are encrypted so that anyone intercepting communications can only see a random string of characters rather than the plaintext.

Since both the http request and response are in plain text, this is already known. In order to open a user account on this website, we enable a packet capturing session. Then, after examining the http packet, we discovered the plain-text versions of our username and password. Even though there is no vulnerability here, if someone attempts to capture a packet using this site while still on the same network, they will discover someone else's credentials. So, security is a concern here. This is the reason why our team included it in the vulnerability assessment.

4.1.2 How to fix this issue:

Install and set up an SSL/TLS certificate on your server to resolve this problem. Numerous businesses offer both free and paid certificates. It's possible that the cloud platform you're using has its own methods for turning on HTTPS.



No.	Time	Source	Destination	Protocol	Length	Info
10642	83.034326	192.168.31.80	103.28.121.75	HTTP	1375	POST /userSignup.do HTTP/1.1 (application/x-www-form-urle...
10649	83.045297	103.28.121.75	192.168.31.80	HTTP	735	HTTP/1.1 200 OK (text/html)
11425	88.663050	192.168.31.80	103.28.121.75	HTTP	585	GET /userLogin.do HTTP/1.1
11434	88.668834	103.28.121.75	192.168.31.80	HTTP	320	HTTP/1.1 200 OK (text/html)
13241	102.298795	192.168.31.80	103.28.121.75	HTTP	772	POST /userLogin.do HTTP/1.1 (application/x-www-form-urle...
13245	102.309559	103.28.121.75	192.168.31.80	HTTP	425	HTTP/1.1 200 OK (text/html)
13493	104.538662	192.168.31.80	103.28.121.75	HTTP	610	GET /userLogin.do HTTP/1.1
13496	104.544431	103.28.121.75	192.168.31.80	HTTP	319	HTTP/1.1 200 OK (text/html)
15033	117.949901	192.168.31.80	103.28.121.75	HTTP	774	POST /userLogin.do HTTP/1.1 (application/x-www-form-urle...
15044	117.959634	103.28.121.75	192.168.31.80	HTTP	60	HTTP/1.1 302 Found (text/html)
15049	117.966520	192.168.31.80	103.28.121.75	HTTP	612	GET /allContests.do HTTP/1.1
15053	117.973045	103.28.121.75	192.168.31.80	HTTP	615	HTTP/1.1 200 OK (text/html)
929	7.050157	fe80::8ede:f9f...	ff02::1	ICMPv6	86	Multicast Listener Query
1034	8.038653	fe80::31fb:237...	ff02::1:ff32:b...	ICMPv6	86	Multicast Listener Report
1104	8.538106	fe80::31fb:237...	ff02::1:3	ICMPv6	86	Multicast Listener Report

> Ethernet II, Src: HewlettP_06:c3:cd (c4:65:16:06:c3:cd), Dst: BeijingX_d5:84:da (8c:de:f9:d5:84:da)
 > Internet Protocol Version 4, Src: 192.168.31.80, Dst: 103.28.121.75
 > Transmission Control Protocol, Src Port: 57908, Dst Port: 80, Seq: 1, Ack: 1, Len: 1321
 > Hypertext Transfer Protocol
 > HTML Form URL Encoded: application/x-www-form-urlencoded

Fig: Captured packet list using Wireshark Packet Capture

After selecting and following this login ok 200 http stream, we can see the actual stream that passed through the network to the server.

```

POST /userLogin.do HTTP/1.1
Host: 103.28.121.75
Connection: keep-alive
Content-Length: 64
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://103.28.121.75
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://103.28.121.75/userLogin.do
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=fds3gvdr6k7a1rilffn09dff00

username=RMKA28&password=cdns12345&uri=%2FallContests.do&submit=HTTP/1.1 302 Found
  
```

Fig: Credentials in Plain text

We found that without encoded HTTP request, that passes to the website. Here the username and password is visible. If people from the same network capture and monitors the packets, the credential will be visible.

4.2 Acunetix scanning

Acunetix is an automated tool for assessing the security of web applications, auditing your web apps for exploitable flaws like SQL Injection and Cross-Site Scripting. In order to counter the growth in threats at the web application layer, it is an automated web application security testing. Through a series of attacks,

Acunetix WVS evaluates the security of a website. Then, it offers succinct descriptions of any vulnerabilities it discovered along with advice on how to remedy them.

After performing a scan on our targeted website, we have found some vulnerabilities that are high and low.

Threat level

Acunetix Threat Level 3

One or more high-severity type vulnerabilities have been discovered by the scanner. A malicious user can exploit these vulnerabilities and compromise the backend database and/or deface your website.

Alerts distribution

Total alerts found	71
High	37
Medium	11
Low	9
Informational	14

Fig: Web scan result of Host 103.28.121.75

Executive summary

Alert group	Severity	Alert count
SQL injection	High	22
Cross site scripting	High	15
HTTP parameter pollution	Medium	3
Vulnerable JavaScript libraries	Medium	2
Application error messages	Medium	1
Development configuration files	Medium	1
Directory listings	Medium	1
Unencrypted connection	Medium	1
URL redirection	Medium	1
User credentials are sent in clear text	Medium	1
Session token in URL	Low	2
Clickjacking: CSP frame-ancestors missing	Low	1

Fig: Integrated scan results

4.2.1 Vulnerability Found

- SQL injection
- Cross-Site Scripting (XSS)
- Vulnerable Java-script Library
- Clickjacking X-frame options header








 High	SQL injection	http://103.28.121.75/contestStanding.do	id
 High	SQL injection	http://103.28.121.75/details.do	name
 High	SQL injection	http://103.28.121.75/contestSubmission.do	show
 High	SQL injection	http://103.28.121.75/details.do	cod
 High	Cross site scripting	http://103.28.121.75/allsubmission.php	name
 High	Cross site scripting	http://103.28.121.75/submitCode.do	id
 High	Cross site scripting	http://103.28.121.75/contestSubmission.do	show

Fig: High Vulnerability







<input type="checkbox"/>	 Medium	User credentials are sent in clear text	http://103.28.121.75/
<input type="checkbox"/>	 Medium	Vulnerable JavaScript libraries	http://103.28.121.75/
<input type="checkbox"/>	 Medium	Vulnerable JavaScript libraries	http://103.28.121.75/
<input type="checkbox"/>	 Low	Clickjacking: CSP frame-ancestors missing	http://103.28.121.75/
<input type="checkbox"/>	 Low	Clickjacking: X-Frame-Options header	http://103.28.121.75/
<input type="checkbox"/>	 Low	Cookies with missing, inconsistent or contradictory properties	http://103.28.121.75/

Fig: Medium and Low Vulnerability

- We have found 2 high level vulnerability and many medium and low level vulnerability. Those medium and Low level vulnerabilities are not so harmful for a website like the High level vulnerabilities. So in this assessment, we will discuss only about the High level vulnerabilities.

4.3 SQL Injection

SQL injection attacks, also called SQLi attacks, are a type of vulnerability in the code of websites and web apps that allows attackers to hijack back-end processes and access, extract, and delete confidential information from databases.

A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

4.3.1 What Can Attackers Do With a SQL Injection Attack?

SQLi attacks make use of vulnerabilities in code at the point where it accesses a database. By hijacking this code, attackers are able to access, modify, and even delete secured data. When SQLi attacks are successful, attackers can:

- Log in to an app or a website front end without a password.
- Access, extract, and delete stored data from secured databases.
- Create their own database records or modify existing records, opening the door for further attacks.

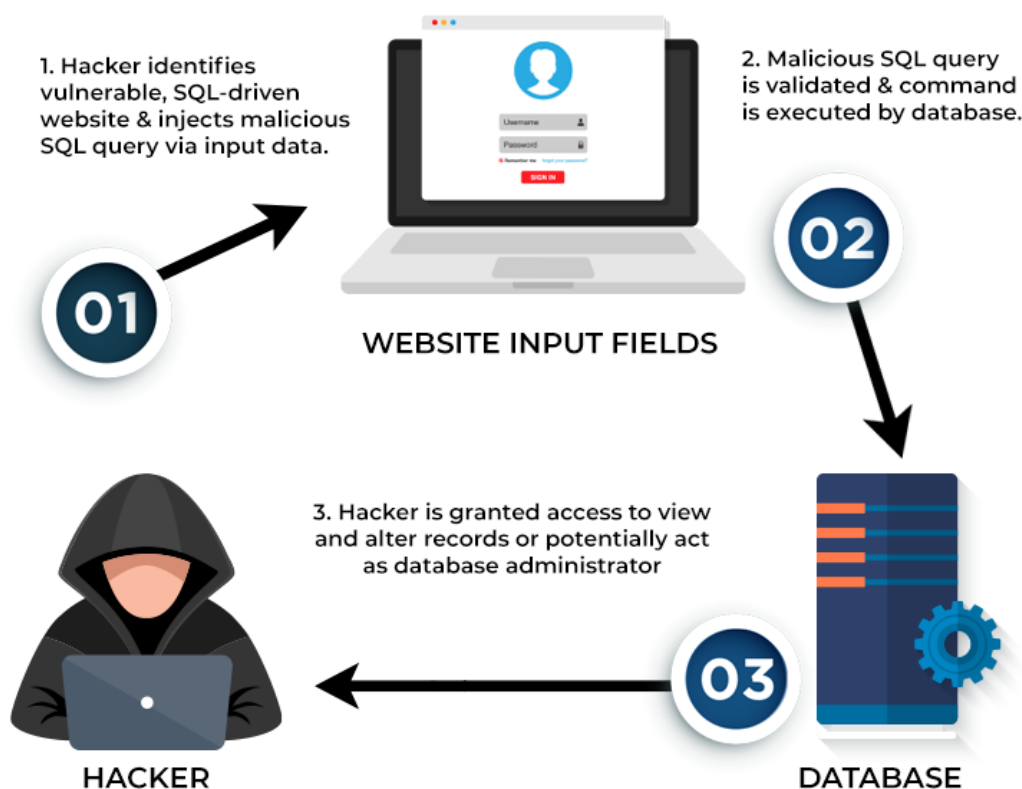


Fig: SQL Injection Demonstration

4.3.2 Examples of SQL:

In SQL: select id, firstname, lastname from authors. If one provided: Firstname: evil'ex and Lastname: Newman. The query string becomes:

```
Select id, firstname, lastname from authors
where firstname = 'evil'ex' and lastname
='newman'
```

Which the database attempts to run as: Incorrect syntax near il' as the database tried to execute evil. A safe version of the above SQL statement could be coded in Java as:

4.3.3 Preventing SQL Injection Attacks

Despite the significant dangers posed by SQLi attacks, they're easy to prevent once you learn some secure coding best practices that include foundational procedures:

- Discover vulnerabilities
- Repair vulnerabilities
- Remediate vulnerabilities
- Mitigate impact

Testing is the key to discovering vulnerabilities in code. Opt for robust tools like dynamic analysis (DAST) that looks at the app from the outside in as an attacker would, and static analysis tools (SAST) that looks for vulnerabilities at the code level. Look for areas where the app connects to a database and try to pass it unusual values.

For example, if you put in a value that contains a single quote, does the program treat that character as user data, or does it treat it as code?

If we include a tautological test (like ' OR '1=1') in we input, are you able to gain access as though we entered a valid password?

Once you have discovered vulnerabilities, it's time to repair them. The best way to do this is by using parameters any time you need to make SQL queries to a database, entering placeholder values in your statements and then passing user-inputted values to the statements at the time of execution.

If your programming language does not support parameters, you can remediate your code by sanitizing or escaping input before passing it to a database. This lets your app know that user input is data rather than code it should execute.

Mitigation is also an important process to help reduce risk, but without addressing the underlying flaw. As an example, rather than looking to your app's code, you might mitigate a flaw by examining database accounts used by your app and making sure that they have the smallest amount of privileges needed to read or insert data to your database.

```
String firstname = req.getParameter("firstname");
String lastname = req.getParameter("lastname");
// FIXME: do your own validation to detect attacks
String query = "SELECT id, firstname, lastname FROM authors WHERE firstname = ? and
lastname = ?";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, firstname );
pstmt.setString( 2, lastname );
try
{
    ResultSet results = pstmt.execute( );
}
```

Fig: Example of SQL injection

4.4 Cross Site Scripting (XSS)

Cross site scripting (also known as XSS) is a common attack vector that injects malicious code into a vulnerable web application. XSS differs from other web attack vectors (e.g., SQL injections), in that it does not directly target the application itself. Instead, the users of the web application are the ones at risk.

A successful cross site scripting attack can have devastating consequences for an online business's reputation and its relationship with its clients.

Depending on the severity of the attack, user accounts may be compromised, Trojan horse programs activated and page content modified, misleading users into willingly surrendering their private data. Finally, session cookies could be revealed, enabling a perpetrator to impersonate valid users and abuse their private accounts.

4.4.1 How Does Cross Site Scripting Work?

XSS is an injection attack that exploits the fact that browsers cannot differentiate between valid scripts and attacker-controlled scripts. XSS attacks bypass the same-origin policy, which is designed to prevent scripts that originate in one website from interacting with other scripts from different websites.

When the same-origin policy is not properly enforced, attackers can inject a script that modifies the web page. For example, the script can allow an attacker to impersonate a pre-authenticated user. It also allows attackers to input malicious code, which is then executed by the browser, or execute JavaScript that modifies content on the page.

XSS can cause serious issues. Attackers often leverage XSS to steal session cookies and impersonate the user. Attackers can also use XSS to deface websites, spread malware, phish for user credentials, support social engineering techniques, and more.

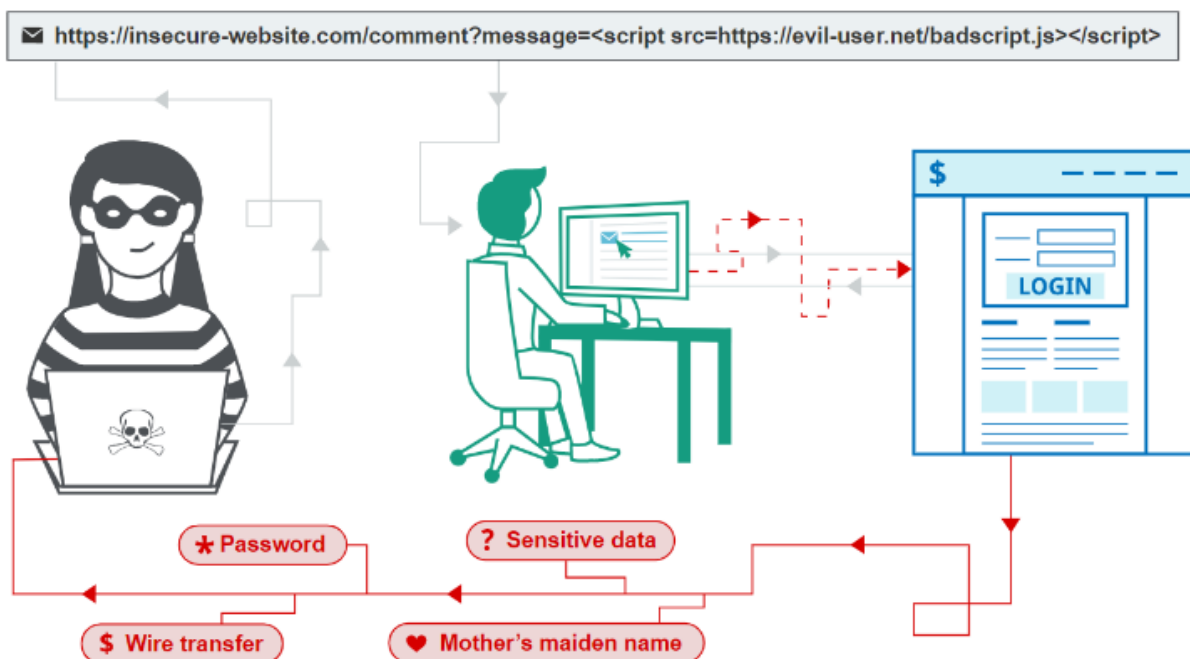


Fig: Working method of XSS

4.4.1 XSS Example with Code

Suppose there's a URL on Google's site, `http://www.google.com/search?q=flowers,which` returns HTML documents containing the fragment `<p>your search for 'flowers' returned the following results :</p>` i.e., the value of the query parameter `q` is inserted into the page returned by Google.

Suppose further that the data is not validated, filtered or escaped.

Evil.org could put up a page that causes the following URL to be loaded in the browser (e.g., in an invisible `<iframe>`):

```
http://www.google.com/search?q=flowers+%3Cscript%3Eevil_script()%3C/script%3E
```

When a victim loads this page from `www.evil.org`, the browser will load the `iframe` from the URL above. The document loaded into the `iframe` will now contain the fragment `<p>your search for 'flowers' returned the following results :</p>`

Loading this page will cause the browser to execute `evil_script()`. Furthermore, this script will execute in the context of a page loaded from `www.google.com`.

4.4.2 Impact of Cross Site Scripting (XSS)

When attackers succeed in exploiting XSS vulnerabilities, they can gain access to account credentials.

They can also spread web worms or access the user's computer and view the user's browser history or control the browser remotely.

After gaining control to the victim's system, attackers can also analyze and use other intranet applications. By exploiting XSS vulnerabilities, an attacker can perform malicious actions, such as:

- Hijack an account.
- Spread web worms.
- Access browser history and clipboard contents.
- Control the browser remotely.
- Scan and exploit intranet appliances and applications.

4.4.1 How to Prevent XSS attack

Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data.

In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

Filter input on arrival: At the point where user input is received, filter as strictly as possible based on what is expected or valid input.

Encode data on output: At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

Use appropriate response headers: To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the `Content-Type` and `X-Content-Type-Options` headers to ensure that browsers interpret the responses in the way you intend.

Content Security Policy: As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

5. Conclusion

To successfully manage the network security, regular vulnerability assessments are crucial. By demonstrating which portions of your network require patching and where to begin, they assist in reducing security breaches. In this article, we test a website for vulnerabilities. On this website, we ran a scanner to look for vulnerabilities and threat levels. We conducted study on high levels of vulnerabilities, discovered the reasons why they occur, and finally demonstrated the steps we should take to prevent those vulnerabilities.

5. References

- <http://103.28.121.75/index.do> [Access Date: 10 November 2022, 20:34 GMT+6]
- <https://www.contrastsecurity.com/glossary/cross-site-scripting-prevention> [Access Date: 11 November 2022, 13:54 GMT+6]
- <https://www.veracode.com/security/xss> [Access Date: 11 November, 2022, 17:20 GMT+6]
- <https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/> [Access Date: 12 November 2022, 20:30 GMT+6]
- <https://portswigger.net/web-security/cross-site-scripting> [Access Date: 14 November 2022, 10:20 GMT+6]
- <https://www.veracode.com/security/sql-injection> [Access Date: 14 November 2022, 9:20 GMT+6]
- https://owasp.org/www-community/attacks/SQL_Injection [Access Date: 15 November 2022, 10:00 GMT+6]
- <https://www.cloudflare.com/en-gb/learning/ssl/why-is-http-not-secure/> [Access Date: 15 November 2022, 5:00 GMT+6]