مجلة جامعة بني وليد للعلوم الإنسانية والتطبيقية
تصدر عن جامعة بني وليد – ليبيا
Website: https://jhas-bwu.com/index.php/bwjhas/index
المجلد التاسع، العدد الأول 2024

ثغرات تطبيقات الانترنت (تحليلها و طرق الوقاية منها)

طارق علي الشهيبية[1]، محسن إبراهيم محمد [2]، عبدالله محمد المهدي [3]
قسم علوم الحاسوب، كلية تقنية المعلومات، جامعة بنغازي، ليبيا [1]
قسم تقنيات الحاسوب ، المعهد العالي للتقنيات الهندسية ، بني وليد، ليبيا [3.2]
Tarig.elshheibia@uob.edu.ly

WEB Applications Vulnerability Analysis and prevention

Tarig Ali Elshheibia [1] , *Mohsen Ibrahim Mohamed* [2], *Abdullah Mohammed Almahdi* [3]
Department of Computer Science, Faculty of Information Technology, University of Benghazi, Libya [1]
Department of Computer Technologies, Higher Institute of Engineering Technologies, Bani Walid, Libya[3.2]

تاريخ الاستلام:16-02-2024 تاريخ القبول: 04-03-2024 تاريخ النشر : 17-03-2024

الملخص:

تطبيقات الويب عبارة عن حزمة برامج، والتي يمكن الوصول إليها من خلال اتصال الإنترنت عبر بروتوكول HTTP.، و يعمل تطبيق الويب عن طريق طلب واسترجاع المعلومات من خادم قاعدة البيانات ويقدم هذه المعلومات من خلال المتصفح. وبالرغم من مزايا التي تقدمها تطبيقات الويب، فقد أظهرت دراسة حديثة أن 75% من الهجمات الإلكترونية تقع على مستوى تطبيقات الويب. وفي هذه الورقة، سيتم عرض الخطوات المهنية لتحليل تطبيقات الويب لتحقيق مستوى عال من الأمان (آمن بما فيه الكفاية). الخطوات الثلاث هي العثور على الثغرات واستغلالها وإصلاحها. تم اختيار تطبيقي W-agora و Wordpress لإجراء هذا التحليل، وكلاهما تطبيق مفتوح المصدر، ويعمل كقناة اتصال بين المستخدمين. البيانات الرئيسية لهذين التطبيقين التي يجب حمايتها هي بيانات المستخدم (قاعدة البيانات، وكلمة مرور المستخدم ،هوية المستخدم). تهدف هذه الورقة إلى اتباع الخطوات الثلاث لتحليل تطبيقات الويب من خلال فحص التطبيقات واستغلال نقاط الضعف ومنع الهجمات. تم استخدام Acunetix Web Vulnerability Scanner (AWVS)والماسح الضوئي Netsparker لفحص التطبيقات من أجل العثور على نقاط الضعف. تم إجراء هذا المسح على نظام التشغيل Windows. بعد إجراء عمليات الفحص، تم اكتشاف العديد من نقاط الضعف في تطبيقي الويب وعلاوة على ذلك، قدمت الماسحات الضوئية هجومًا بسيطًا كمثال لكيفية استغلال كل ثغرة أمنية.

الكلمات الدالة: الثغرات، تطبيقات الويب، الحماية، ماسحات الويب ،أدوات البحث.

Abstract

Web based application is a software package, which is accessed through the internet connection via HTTP protocol. Therefore, web application operates by requesting and retrieving information from database server and presents this information through the browser. [1] Despite the advantages of web application, a recent study presents that 75% of the cyber-attacks accrue in web applications level. In this paper, the professional steps of web application analysis will be shown to achieve high level of security (secure enough). The three steps are finding, exploiting and fixing the vulnerabilities. W-agora and Wordpress applications were chosen to do this analysis, both of them are an open-source application, and a sort of forum acts as a communication channel between users. The main asset of these two applications needs to be protected are user's data (database), user's password (user's identity), and root's password. This paper aims to follow the three steps of web application analysis by scanning the applications, exploiting the vulnerabilities and preventing the attacks. Acunetix Web Vulnerability Scanner (AWVS) and Netsparker scanner was used to scan the applications in order to find vulnerabilities. This scanning was conducted on Windows operation system. After performing the scanners, several vulnerabilities in both web applications were detected. Furthermore, the scanners provided simple attack as an example of how each vulnerability can be exploited.

Keywords: *vulnerabilities,* prevention *,web applications, web scanners (search tools)*

Introduction:

This century, many individuals and institutions have relied on information technology in the conduct of their business, and communication networks have formed a medium in which data flows and information is stored through websites and thus these sites need protection that safeguards the integrity of their contents and ensures the continuity of their work, Due to the fact that many threats could face the integrity of the data or hacking attempts for the purpose of stealing information or sabotaging or modifying and tampering with it, comes the importance of protection and search for places of weakness and reasons that may allow the attacker to penetrate the site.

This study aimed to find some solutions to avoid many of the dangerous security flaws in Internet sites, which are due to poor security in the stage of designing the pages of the site and writing the code for the site, and with the large increase in the number of websites and the increase in the number of developers and designers for these sites It number Gaps the wish Which Highest detectable by attackers Much Than she was Previously, this means that Attacks Which Targeting Applications The web And exploit points Weakness On level The application Easier to be on the transport level or Server And in Time Himself increased risks And the impact of gaps the wish in a this is Applications Form Large.

Since many transactions are executed through these websites and although the user may be authenticated before providing access to the database, there may be a loophole through which the attacker can inject instructions and exploit this vulnerability within the web application, it is possible for the attacker to access These sites and violating the privacy of these sites and obtaining important information or modification in this information or tampering with it by unauthorized person, and most of the offensive injections are done through Cross Site Scripting and SQL Injection attacks.

1 Aim Of This Study

This study aims to describe how it is possible to analyze and discover the main weaknesses in some web applications, and the vulnerabilities that will focus on are SQL injection and Cross Site Scripting (XSS), to achieve this goal, scanners will be used , these scanners are designed for web application vulnerability analysis and vulnerability detection.

By conducting this analysis, will be able to discover most of the weaknesses in a large number of web applications and many websites that were designed without using web applications as well, and among the most common of these vulnerabilities in common web applications are SQL Injections and Cross Site Scripting XSS)) .

The study objectives can be summarized to achieve the following goals:

1. Identifying and analyzing a set of security vulnerabilities that threaten websites based on the Internet
2. Determine the necessary measures to address and repair the security holes that the Internet sites are exposed to.
3. Determine the most dangerous types of security vulnerabilities that threaten websites in the world.
4. Learn about the programs and tools used in the web application penetration testing process and protection from vulnerabilities.
5. Try to fix the vulnerability that were found

1.1 Definition of Vulnerability

In the programming world, a vulnerability is away that allows attackers to penetrate the system or gain illegally powers, and sometimes the presence of a vulnerability results in the complete destruction of the system, and the impact of the vulnerability depends on three elements that can be mentioned as follows:

- System sensitivity and defects.
- The attacker's access to the defect.
- The attacker's ability to exploit the flaw.

In order to exploit the vulnerabilities, the attacker must have at least one applicable tool or method in order to be able to exploit any vulnerability and violate the privacy of the system.

According to NIST SP 800-37, analyzing and assessing the vulnerability of these elements is important for every activity required within the NIST risk management. The RMF includes six steps, in each of which a vulnerability analysis and assessment are combined, as follows:

- Classification of information in terms of sensitivity.
- The methods used to control security to access the security system from the developer's point of view.
- Monitoring the methods used for security.
- Evaluation of the security support operations used.
- Control the powers within the system.
- Monitor safety controls.

1.2 How Vulnerability Assessment Tools Work

Vulnerability assessment tools usually work by attempting to complete steps that are often used to exploit vulnerabilities. They begin by conducting a footprint analysis to determine the services of the various networks and software (including versions and patch levels) that work on The target then the tools try to find indicators (patterns and features) in order to exploit the vulnerability that were previously found in those releases and programs used to report violations that may result from them, and care must be taken when operating some of these tools against direct targets (operational).. Due to the fact that harmful results may occur, for example, targeting a direct web application using SQL injection as limiting or modifying tables, and it can lead to loss of actual data and for this reason, some vulnerability assessment tools are completely passive and use a method known as negative analysis. And, passive checks, in which no data are entered by the tool, may only benefit from reading and collecting data, but the possibility of a weakness or loophole that can be discovered by negative analysis remains, such as the presence of a specific and simple defect during the process. Examination, and in general, negative analysis tools are often of limited use compared to non-negative tools because they can detect the presence of weaknesses based on specific evidence, instead of direct testing for these gaps, unlike effective tools capable of that.

2 Methodology/Approach

The methods to be followed in this analysis are:

- Preparing the work environment to be safe and capable of taking the required tests.
- Designing websites using internet applications.
- Finding vulnerabilities by using some tools to check applications.

- Understanding the behavior of the attacks by exploiting some of the weaknesses that are found.
- Treating weaknesses and preventing attacks.
- System characterization

The following steps explain the setting of the work environment:

2.1 Install a local server (XAMPP)

Xampp program, which is an open source program, which creates a virtual server on the computer, through which it is possible to preview the results of what has been programmed from sites or website applications before uploading it to the real server, and this program supports several languages and programming tools, the most famous of which are PHP, Perl and MySQL In addition to other programming languages, it also supports many applications such as Wordpress, Druple and many other programs that help in translating and executing the codes that are used in designing websites, and after installing the program, we enter the server's control panel and run the database server

2.2 Create databases inside the server

In order to be able to design a site using the w-agora application, must first create its own database, and this can be done through the following link:

http: // localhost / phpmyadmin /

And the name of the database that was created for the w-agora application is agora, Moreover the database name for Wordpress application is press.

2.3 Designing the pages to be searched analyzed

Designing by w-agora application new pages can be created

at*: http: //localhost/w-agora/create_site.php*

And to design by Wordpress application new pages can be created site

at http: *//localhost/wordpress/wordpress/wp-admin/install.php*

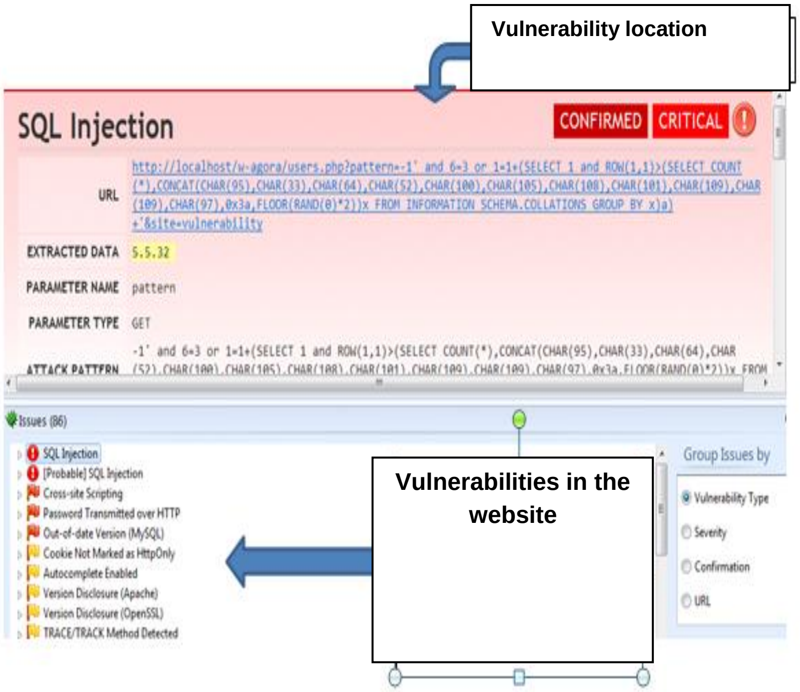
2.4 Install search tools

Netsparker and Acunetix scanners were downloaded and installing

3 Scanner result

In this study, a Netsparker and Acunetix scanners were used to analyze the sites that were created using the w-agora and wordpress applications, After searching the sites there are some vulnerabilities were found.

as an example of the result from Netsparker the following figure shows a simple sample of the search results



SQL Injection

CONFIRMED CRITICAL

URL: http://localhost/w-agora/users.php?pattern=-1' and 6=3 or 1=1+(SELECT 1 and ROW(1,1)>(SELECT COUNT(*),CONCAT(CHAR(95),CHAR(33),CHAR(64),CHAR(52),CHAR(100),CHAR(105),CHAR(108),CHAR(101),CHAR(109),CHAR(109),CHAR(97),0x3a,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.COLLATIONS GROUP BY x)a)+'&site=vulnerability

EXTRACTED DATA: 5.5.32

PARAMETER NAME: pattern

PARAMETER TYPE: GET

ATTACK PATTERN: -1' and 6=3 or 1=1+(SELECT 1 and ROW(1,1)>(SELECT COUNT(*),CONCAT(CHAR(95),CHAR(33),CHAR(64),CHAR(52),CHAR(100),CHAR(105),CHAR(108),CHAR(101),CHAR(109),CHAR(109),CHAR(97),0x3a,FLOOR(RAND(0)*2))x FROM

Issues (86)

- SQL Injection
- [Probable] SQL Injection
- Cross-site Scripting
- Password Transmitted over HTTP
- Out-of-date Version (MySQL)
- Cookie Not Marked as HttpOnly
- Autocomplete Enabled
- Version Disclosure (Apache)
- Version Disclosure (OpenSSL)
- TRACE/TRACK Method Detected

Group Issues by

● Vulnerability Type
○ Severity
○ Confirmation
○ URL

Figure 1Search results on the Agora website

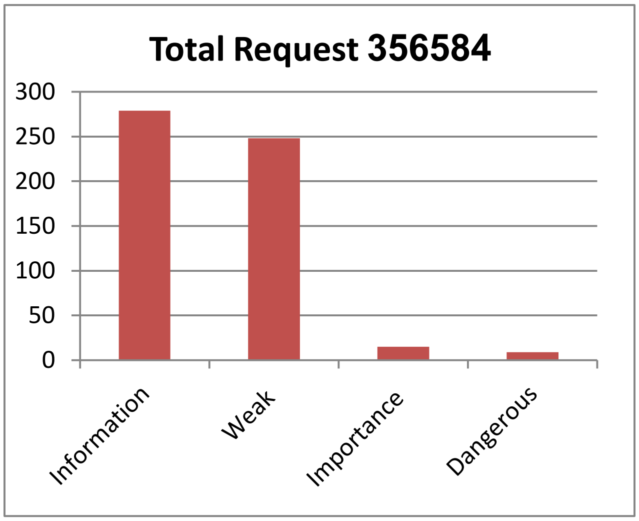
4 The analysis of the vulnerabilities:

After the process of the research there are many vulnerabilities were found in both applications w–agora and wordpress as well, a And the following table shows the classification of vulnerabilities and the impact that may result from these vulnerabilities [2].

Severity level	The effect that may result from the presence of the vulnerability
Critica	The attacker can access the database
Important	An attacker could access users' information
low	The attacker could use the website to make users send their sensitive data .
Information	The attacker could see the system information which could help in finding a new vulnerability

After performing the scanner on both web applications, W-agora and WordPress, the found results are shown in the following table, moreover there are some statistics and some results obtained to assess the severity of the vulnerabilities, and the following figure shows the number of links that were visited to evaluate the application and the number of links that were classified as dangerous or its queries

Level of security	Web application	
	w-agora	Wordpress
Dangerous	9	0
importance	62	3
Weak	9	9
information	24	158



Since the w-agora application contains more serious vulnerabilities, the focus has been on it and attempts to present the seriousness of these vulnerabilities and the level of their classification, and the following table shows the classification of vulnerabilities and the impact that may result from these vulnerabilities.

5 W-agora Attacking Tree:

Attacking tree is a tree that shows different ways of attacking W-agora. The attack is represented by a tree where the attack goal is the get user information for an example and the leaf nodes are several techniques to achieve it . [3]

As showing in the Attacking tree the most dangerous vulnerabilities W-agora has SQL Injection and Cross-site Scripting, by using the SQL Injection vulnerability an attacker will be able to read and modify sensitive data from a database as well as executing administration operations. Attacker can do that by injecting SQL query via input in the vulnerable parameters. Moreover by using the Cross-site Scripting vulnerability an attacker will be able attacker to insert his own JavaScript into the vulnerable parameters and as soon as the victim visit that site, the JavaScript will be executed. [2]

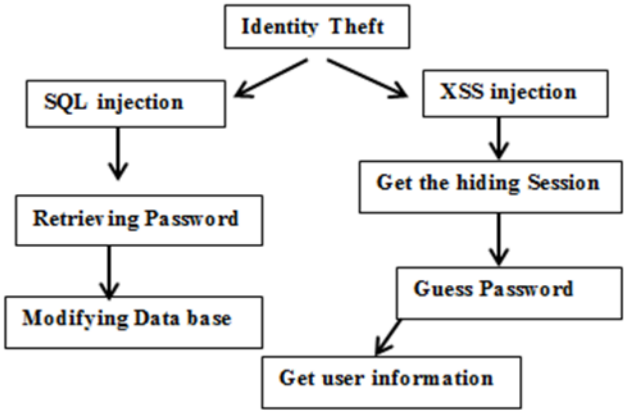


Figure 2VI. W-AGORA ATTACKING TREE

6 Preforming the attack

This is the second step of this analysis, which is exploiting vulnerabilities. The Cross-site scripting and SQL injection vulnerabilities will be exploited.

6.1 SQL injection:

It is away to inject or insert a partial or complete SQL command through data input or client browser to the web application. SQL injection represents a significant risk to the web application because it can read sensitive information from the application's database or even modifies it. Some exploitation techniques guarantee the success of the SQL injection attacks. In our project, Union exploitation technique was used. It is used to join the injected query with the original query that will allow the injected query to have result from other tables. [4]

In order to exploit these vulnerabilities, the example attack from the result of the Netsparker scanner is the starting point to perform this attack. The following link was taken from the scanner result and will be exploited

http://localhost/w-agora/index.php?site=vulnerability&cat=-1OR 17-7=10

After using the given link to exploit this vulnerability, the site changed. That means any input in "cat" variable is attached the original query directly

6.2 Cross-Site Scripting (XSS)

Cross-site scripting vulnerability is classified as a high-risk vulnerability. Netsparker scanner shows that the W-agora website has XSS vulnerability. Exploiting this vulnerability is the best way to understand the attack behavior. [3] Netsparker scanner provides an example of the attack, description and the vulnerable parameter The input for the vulnerable parameter is:

'"--></style></script><script>alert(0x000475)</script>

The pervious attack shows a pop up message which proves that the parameter “showuser” vulnerable to XSS vulnerability

7 How the search tool works

Many transactions of the type of GET and POST that are found in the pages of the site are collected and that may cause the existence of the vulnerability and the next part is an example showing the name of the page and the transactions on this page and the request that was used to find out the parameter that causes the vulnerability and the response that is Obtained from the server.

Page name (change_password.php) and the following table contains the parameters within this page and the

parameters that were used as inputs for this page.

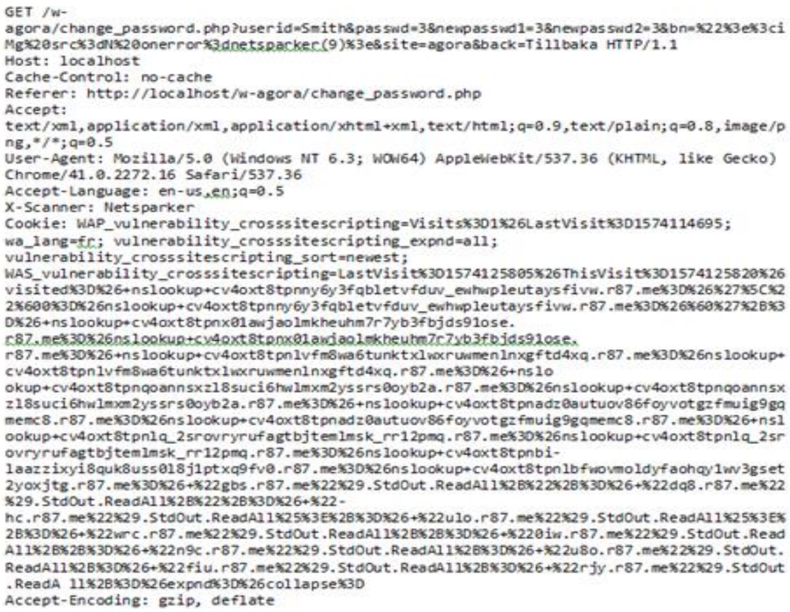
The parameter	**Type**	**The parameter value**
Userid	Get	Smith
Passwd	Get	3
newpasswd1	Get	3
Newpasswd2	Get	3
Bn	Get	"**><iMg src=N onerror=netsparker(9)>**"
Site	Get	Agora
Back	Get	Tillbaka

These parameters are used to verify the transactions that may cause a vulnerability through the following link :

http://localhost/w-agora/change_password.php?

userid=Smith&passwd=3&newpasswd1=3&newpasswd2=3&bn=%22%3e%3ciMg%20src%3dN%20onerror%3dalert(9)%3e&site=agora&back=Tillbaka

And through the previous link, the request that the browser requests from the server is as follows :



```
GET /w-
agora/change_password.php?userid=Smith&passwd=3&newpasswd1=3&newpasswd2=3&bn=%22%3e%3ci
Mg%20src%3dN%20onerror%3dnetsparker(9)%3e&site=agora&back=Tillbaka HTTP/1.1
Host: localhost
Cache-Control: no-cache
Referer: http://localhost/w-agora/change_password.php
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/p
ng,*/*;q=0.5
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/41.0.2272.16 Safari/537.36
Accept-Language: en-us,en;q=0.5
X-Scanner: Netsparker
Cookie: WAP_vulnerability_crosssitescripting=Visits%3D1%26LastVisit%3D1574114695;
wa_lang=fr; vulnerability_crosssitescripting_expnd=all;
vulnerability_crosssitescripting_sort=newest;
WAS_vulnerability_crosssitescripting=LastVisit%3D1574125805%26ThisVisit%3D1574125820%26
visited%3D%26+nslookup+cv4oxt8tpnny6y3fqbletvfduv_ewhwpleutaysfivw.r87.me%3D%26%27%5C%2
2%600%3D%26nslookup+cv4oxt8tpnny6y3fqbletvfduv_ewhwpleutaysfivw.r87.me%3D%26%60%27%2B%3
D%26+nslookup+cv4oxt8tpnx01awjaolmkheuhm7r7yb3fbjds9lose.
r87.me%3D%26nslookup+cv4oxt8tpnx01awjaolmkheuhm7r7yb3fbjds9lose.
r87.me%3D%26+nslookup+cv4oxt8tpnlvfm8wa6tunktxlwxruwmenlnxgftd4xq.r87.me%3D%26nslookup+
cv4oxt8tpnlvfm8wa6tunktxlwxruwmenlnxgftd4xq.r87.me%3D%26+nslo
okup+cv4oxt8tpnqoannsxzl8suci6hwlmxm2yssrs0oyb2a.r87.me%3D%26nslookup+cv4oxt8tpnqoannsx
zl8suci6hwlmxm2yssrs0oyb2a.r87.me%3D%26+nslookup+cv4oxt8tpnadz0autuov86foyvotgzfmuig9gq
memc8.r87.me%3D%26nslookup+cv4oxt8tpnadz0autuov86foyvotgzfmuig9gqmemc8.r87.me%3D%26+nsl
ookup+cv4oxt8tpnlq_2srovryrufagtbjtemlmsk_rr12pmq.r87.me%3D%26nslookup+cv4oxt8tpnlq_2sr
ovryrufagtbjtemlmsk_rr12pmq.r87.me%3D%26nslookup+cv4oxt8tpnbi-
laazzixyi8quk8uss018j1ptxq9fv0.r87.me%3D%26nslookup+cv4oxt8tpnlbfwovmoldyfaohqy1wv3gset
2yoxjtg.r87.me%3D%26+%22gbs.r87.me%22%29.StdOut.ReadAll%2B%22%2B%3D%26+%22dq8.r87.me%22
%29.StdOut.ReadAll%2B%22%2B%3D%26+%22-
hc.r87.me%22%29.StdOut.ReadAll%25%3E%2B%3D%26+%22ulo.r87.me%22%29.StdOut.ReadAll%25%3E%
2B%3D%26+%22wrc.r87.me%22%29.StdOut.ReadAll%2B%2B%3D%26+%220iw.r87.me%22%29.StdOut.Read
All%2B%2B%3D%26+%22n9c.r87.me%22%29.StdOut.ReadAll%2B%3D%26+%22u8o.r87.me%22%29.StdOut.
ReadAll%2B%3D%26+%22fiu.r87.me%22%29.StdOut.ReadAll%2B%3D%26+%22rjy.r87.me%22%29.StdOut
.ReadA 11%2B%3D%26expnd%3D%26collapse%3D
Accept-Encoding: gzip, deflate
```

Figure 3, the request that the browser requests from the server

As a result of the request, the response from the server was as follows:

HTTP/1.1 200 OK

Server: Apache/2.4.4 (Win32) OpenSSL/0.9.8y PHP/5.4.19

X-Powered-By: PHP/5.4.19

Content-Length: 176

Content-Type: text/html

Date: Tue, 19 Nov 2019 01:10:46 GMT

<div align="center"><div class="msgwarning">Forum <b></b> does not exist<br>Could not access configuration file: <b>conf/"><iMg src=N onerror=netsparker(9)>.php</b></div></div>

Defenses that were used to address some of the vulnerabilities that were discovered:

First: SQL instructions injection vulnerabilities: -

After reviewing the search results and knowing the parameter that caused the SQL vulnerability, the search was done inside the page code and searched for the parameter named (cat), it became clear that there are many places where this parameter was used and the problem is that the parameter parameter is It was entered by the user and later this parameter is sent to the databases on the server that have prior contact with the site's pages without verifying the parameter parameter, and to prevent this attack and exploit this vulnerability on the site, two places in the code on the page have been modified. (dbaccess.php), which contains the parameter (cat) and a change in the part in which the parameter cat is located, and the cause of this vulnerability is through validation by adding the verification statement for the parameter (cat) as in the following code :

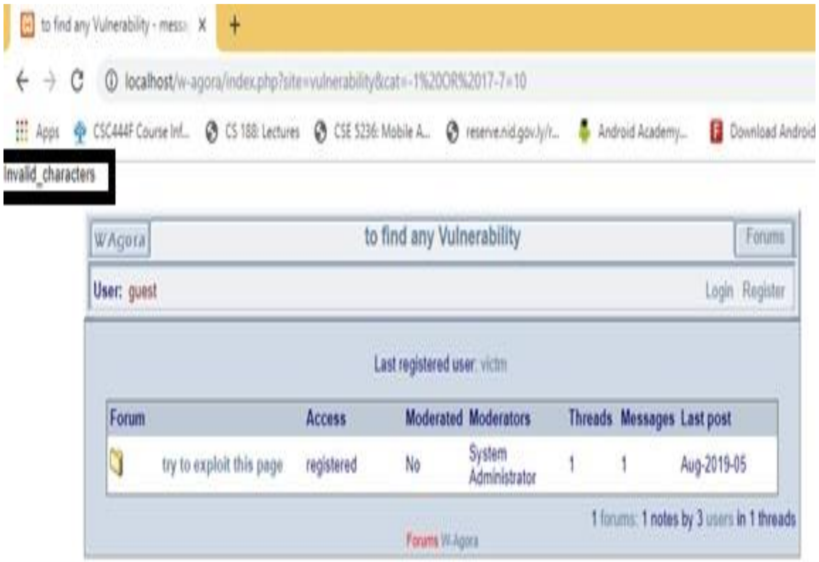
Before :

```
Line 921 if ($cat != '') {
Line 922     $query .= " AND S.parent=$cat؛"
Line 923    }
```

After :

```
Line 921  if ($cat != '') {
Line 922   if(preg_match("/[^0-9]/",$cat)){
Line 923   echo '<p>'.'Invalid_characters'.'</p>';
Line 924   $cat=0 ؛
Line 925   }
```

This technique will remove any character that is used to formulate code written in SQL language such as "(), =" because it will accept numbers only as a result, and to make sure that this vulnerability has been eliminated and then use the same link that was obtained from the search tool that contains an instruction (SQL) as input for the parameter (cat), and after changing the code, the result is as in the following figure



localhost/w-agora/index.php?site=vulnerability&cat=-1%20OR%2017-7=10

Invalid_characters

to find any Vulnerability

User: guest

Last registered user: victim

Forum	Access	Moderated	Moderators	Threads	Messages	Last post
try to exploit this page	registered	No	System Administrator	1	1	Aug-2019-05

1 forums: 1 notes by 3 users in 1 threads

Figure 4 result after changing the code

A. Second: - Site scripting vulnerability (XSS):

After reviewing the search results that were reached and knowing the transactions that caused the presence of XSS vulnerabilities, it was found that there are several parameters that caused this vulnerability and are present in several pages. The correctness of the user's input in terms of the type of the entered data. Input validation technique can be used in terms of type, and there are several functions that can be used, among which “htmlspecialchars ()” is a function used to replace special characters and convert them into non-harmful characters that cannot be Who can run code?

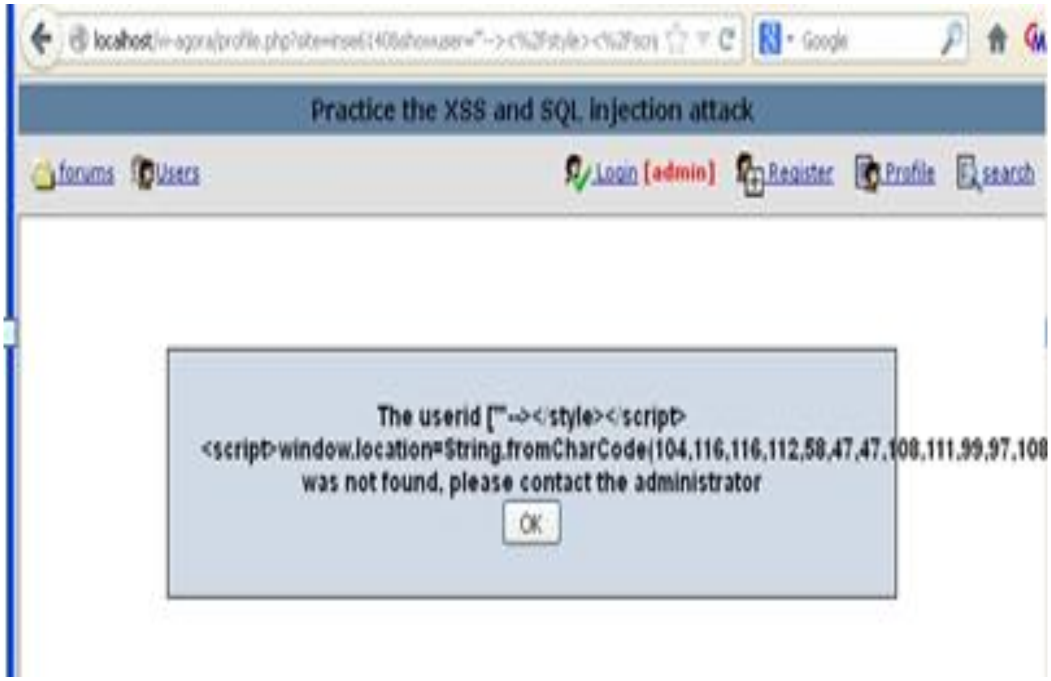
This function converts special characters into HTML entities, for example "<" (which means less than) will be converted to text character and "& which means and to lt for example;".

In order to address this vulnerability and eliminate it, the pages that contain the labs (showuser) were searched and an attempt was made to make the necessary adjustments. The following figure shows the part in which the code for the lab is located (showuser)

And after finding the parameter causing the vulnerability, the code was changed and the function for converting special characters into HTML entities for the (showuser) parameter

$showuser = htmlspecialchars($showuser

And when the attack is re-carried out and the link obtained from the search tool is visited and after the change in the source code, the code that was written using the JavaScript language is not executed and the alert message does not appear as it was in the past, but the website looks like the figure. next one :


localhost/w-agora/profile.php?site=[illegible]&showuser="--></%2Fstyle></%2Fscri
Google
forums
Users
Login [admin]
Register
Profile
search
OK

Practice the XSS and SQL injection attack

The userid [""--></style></script>
<script>window.location=String.fromCharCode(104,116,116,112,58,47,47,108,111,99,97,108
was not found, please contact the administrator

Figure 5alert message after the attack

8 Conclusion

There are many websites that are designed using web applications that may contain some security flaws, which may result in vulnerabilities in the site that was designed, and this study dealt with how to search for vulnerabilities and security lapses that may exist in the source code of the site, which may cause the site to be completely destroyed, it is important to inspect the site using one of the search tools and treat the vulnerabilities that are found. The whitelist technology can be used to prevent SQL Injection attack in one of the site's pages. Moreover, an XSS attack can be prevented on one of the pages. By using the input validation method, which converts any special codes into scripts from which no code can be written.

References

[1] Ajjarapu Kusuma Priyanka و Siddemsetty Sai Smruthi ،"Web Application Vulnerabilities: Exploitation and Prevention *2020* "، *International Conference on Electrotechnical Complexes and Systems (ICOECS)* ،Ufa, Russia .2020 ،

[2] N. Jovanovic, C. Kruegel, and E. Kirda ،"A static analysis tool for detecting web application vulnerabilities "، *IEEE Symposium on Security and Privacy* ،May 2006 ..

[3] Ankit Shrivastava ،Santosh Choudhary و Ashish Kumar ،"XSS vulnerability assessment and prevention in web application *2016* "، *nd International Conference on Next Generation Computing Technologies (NGCT)* ،Dehradun, India .2016 ،

[4] S. Akshay Kumar و Y. Usha Rani ،"Implementation and analysis of Web application security measures using OWASP Guidelines *2022* "، *International Conference on Recent Trends in Microelectronics, Automation, Computing and Communications Systems (ICMACC)* ، Hyderabad, India 30−28 ،December 2022 .