**Key Vulnerabilities in Websites and their Detection & Prevention: A Review**

**Abstract-** From an elementary school to the University and from very small company to Multi National Corporation maintains their information online and hence websites are playing key role in this present age. Besides the advanced features Security is a key concern in the websites. Besides several Continuous Improvements There are a number of Vulnerabilities in the present age of websites. This paper presents a systematic review on the key vulnerabilities in the websites and the detection mechanism together with the Preventions methods.

**Introduction**

General Introduction to be written here

**Following are the most frequently reported Key vulnerabilities till date:**

1. **SQL Injection**
2. **Proper Authentication and Session Management**
3. **Indirect Object References**
4. **Security Misconfiguration**
5. **Sensitive Data Exposure**
6. **Missing Function Level Access Control**
7. **Known Vulnerable Components**
8. **Invalidated Redirects and Forward**



**1. INTRODUCTION TO SQL INJECTION**

SQL is defined as Structured Query Language. SQL injection is simply like an information getting technique or can say an attack. SQL is code injection techniques that used to attack data driven applications in which insert the malicious statement into login field. SQL injection works on vulnerable websites, which website affected its. Actually we want to see the whole info about web panel and just achieve data from website. It is a query on web by using some logical steps. Another simple way to find vulnerable websites by using an automated tools named wise like: **- BSQL Hacker, The Mole, Pangolin, SQL map, Havij, EnemaSQLi, SQL ninja, SQL sus, Safe 3SQL injector, SQL poison[1].**

These are all tools penetration testing tools and the purpose is that detecting and exploiting SQL injection on vulnerable websites. It’s based on Boolean query based tools. These use for professionally penetration testing and also known as auto-hacking software. SQL poison includes php, asp and etc [5, 7]. SQL injection can read sensitive data from database, and modify data (insert/delete/update). SQL injection has five sub category likewise: - Database management system specific SQLi, Classic SQLi, Blind or inference SQL, Compound SQLi. In Compound SQLi are four category: - SQL injection And Insufficient Authentication, SQLi and DDoS Attacks, SQLi and DNS Hijecking, SQLi and XSS [2].

Storm worm is one characterization of compound SQLi. It is backdoor trojan horse that affects computers by using Microsoft operating system [2].

In general the way SQL statements including SQL syntax written by the programmers with user-supplied data. Example:-

Select title or headline, where id=$id

In the example above the $id contains user supplied data. The example below decode or illustrates the user-supplied data “11 or 5=5”, changing the concept of the SQL statement, modify “or 5=5”. Select headline, where id=11 or 5=5. Which is structured or designed to show all records from the table for a username and password supplied by a user [4].

5' or '5' = '5

5' or '5' = '5

Result of query:-

Select\*From User Where Username=’5’or ‘5’=’5’ and Password=’5’or’5’=’5’.The condition '5' = '5' is always true, so this SQL query will always passed so and authentication process being bypassed.

Report: - This is reveal that more than 4000 websites have been hacked by criminal till 2012.

**PROCEDURE STEP BY STEP**

**FLOW CHART FOR APPLYING SQL INJECTION:-**

1. Log on website with ?id = some name and send query through the URL
2. Find number of Columns by using “ORDER BY” and “--“.
3. Find vulnerable column by using “UNION SELECT ALL” command.
4. Find Database version by using “@@version” or “version ()”.
5. Find table name in which data store
6. Find column name
7. Fetching data from column
8. Got password and username.

**STEP1:- LOG ON WEBSITES:-**

Firstly logon the website like http://www.asdfghj.com/news/php?id=123 Click here and open this website. Send query through the URL (Uniform Resource Locator) for getting password and user name that are stored in column’s and tables [2].

**STEP2:- FIND NUMBER OF COLUMNS:-**

By using ORDER BY we can find the total numbers of column in web. http://www.asdfghj.com/news/php?id=123 order by 11-- and click on it [5]. If it show error that means the columns are less than 11 and further try for 10, if it show again an error it means the columns are less than 10 and last we have 7 and do not give any error that means the columns are 7. ORDER BY is used to sort the columns [3].

**STEP3:- FIND VULNERABLE COLUMN:-**

By using UNION SELECT ALL we can find the most vulnerable column in website. http://www.asdfghj.com/news/php?id = - 123 union select all 1, 2, 3, 4, 5, 6, and 7 -- and

Click on it. Remember the – (dash) to put before 123. And after it we will get more numbers on screen and in the bold one is most vulnerable column. Like here 3 is most vulnerable column [2, 12].

**STEP4:- FIND DATABASE VERSION:-**

Instead of vulnerable column just put @@version or version (). http://www.asdfghj.com/news/php?id=-123 union select all 1, 2, @@version, 4, 5, 6, and 7 -- and click on it. Here we got the version on the screen and write down it on paper [6].

**STEP5:- FIND TABLE NAME:-**

Replace vulnerable column with table\_name and by using “from information\_schema.tables where table\_schema= database ()”. http://www.asdfghj.com/news/php?id=-123 union select all 1, 2, table\_name, 4, 5, 6, 7 from information\_schema.tables where table\_schema=database ()--. Here we will get first table on the screen for all tables [7, 12].

http://www.asdfghj.com/news/php?id=-123 union select all 1, 2, group\_concat (table\_name), 4, 5, 6, 7 from information\_schema.tables where table\_schema=database () --. Click on it [9]. It will give us the table name for fetching data.

**STEP5:- FIND COLUMN NAME:-**

Just replace table\_name with column\_name.

http://www.asdfghj.com/news/php?id=-123 union select all 1, 2, groupconcat (column\_name), 4, 5, 6, 7 from information\_schema.columns where table\_schema=database () --. Click on it.

It will give us column name for fetching data.

**STEP 6:- FETCHING THE DATA:-**

Consider the column is 2f34x in hex [9]. These column are in first table that is wer\_admin.

http://www.asdfghj.com/news/php?id=-123 union select all 1, 2, group\_concat (username, 2f34x, password) 4, 5, 6, 7 from wer\_admin--. Click on it.

Simply we got the password and username.

**HOW CAN PREVENT?**

**1. Do not trust anyone: -** Suppose all user submitted data is defect or evil and validate everything.

**2. Do not use the dynamic SQL when it can be avoided:** - used parameterized queries and prepared statements or stored procedure [10].

**3. Make in mind that shred, patch and update:** - Vulnerabilities in applications and database that hackers can search and exploit daily discovered by use of SQL injection, so it is important to apply updates and patches.

**4. Firewall: -** To help filter out of malicious data. A WAF can be used to protect of some security against new vulnerabilities before a patch is available [5].

**5. Always use best or suitable perquisites and privileges:** - Do not connect to your database using an account with admin level privileges unless there is some compelling reason to do so [8].

**2. INTRODUCTION TO PROPER AUTHENTICATION AND SESSION MANAGEMENT**

Proper authentication is very important to secure web and proper and good session management is also required to web security. Defects and flaws in authentication and session management make vulnerable of credentials data of web and session tokens. These types of flaws make the cause of hijacking of user account and accountability control and undermined authorization [1]. User ID’s and passwords involve on web for user authentication. Encrypted passwords and long password are more secure for users. Application functions related to user authentication and session management are not properly appliance. Web servers, web applications, and application server’s environments are sensitive to proper authentication and session managements. Admin credentials should be properly encrypted that benefits is that it will not easily decrypted. There are five impact on proper authentication and session management likewise 1. Brute Force 2. Replay Attack 3. Session Fixation 4. Session Hijacking 5. Session Time-Out.

**Brute Force: -** It is information gathering method based on error method used to obtain information about user such as user passwords and PIN (Personal Identification Number) etc. [7]

**Replay Attack: -** In this section the valid data will be in form of maliciously or fraudulently belated.

**Session Fixation: -** It is based on some vulnerable issues in system and web .In it one person has power to set another person’s session identifiers.

**Session Hijacking: -** It is try to obtain or accepted unauthorized access to information.

**Session Time-Out: -** If user will not refresh a page within time out period then the session will automatically ends [12].

There are some important aspects in proper authentication and session management is given below: -

1. Risk:-

- Identity theft

-Undermined accountability controls and less proper authentication

2. Account session tokens and credentials details are not properly protected:-

-A third party can access to anyone’s account.

-Attackers resolve keys, passwords and authentication gesture.

3. Use drawbacks and flaws in authentication mechanism method:-

-Timeout

-Secret questions and account update

-Logout

-Remember me

-Password Management

**DETECTION:-**

If any websites vulnerable with proper authentication and session management if: -

1. Session ID’s are not rotated after successful login.
2. Some session ID’s are vulnerable with session hijacking so be careful.
3. Defects in URL so session ID’s are not protected properly, it can easily access.
4. Storing and saving own information are not properly encrypted or hashed.
5. Credential details can easily guessed and easily put in the password section if your session timing are not good and unauthorized access [14].

**PREVENTION: -**

All password should be saved in encrypted and store highly protected section. Encryption must be used when the plain text password is required. Decryption key should be very strongly protected. If the user wants to change own password then it should be required old and new passwords and security questions. If user forgotten password then there should be available an option that with the help of this option user can recover his password option such as e-mailed , phone number, and etc. Attempt should be limited for accessing passwords and user name. Otherwise authentication could be failed. So be careful. Information about user account should be updated within periodically and fast. Poor password should be not accepted [13, 16].

.

**3. INTRODUCTION TO INDIRECT OBJECT REFERENCES**

Insecure Direct Object References occur when an application provides direct access to objects based on user supplied input [1].

As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files.

Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object [1, 7].

**How to Test:-**

The way to find out if a web application is vulnerable to insecure direct object references is to clarify that all object references have proper defenses and protection [13, 15, and 16]. Some steps is given below, consider:-

1. If the reference is an indirect, does the mapping to the direct mapping to the direct reference fail to limit the values to authorize for the current user?

2. For direct reference to restricted resources, does the application fail to check the user is authorized to access the proper and exact resource they have requested?

And another else to test for this vulnerability the tester first needs to map out all locations in the application where user input is used to reference objects directly [12].

For example, locations where user input is used to access a database row, a file, application pages and more. Next the tester should modify the value of the parameter used to reference objects and assess whether it is possible to retrieve objects belonging to other users or otherwise bypass authorization [15].

Below are several typical scenarios for this vulnerability and the methods to test for each: The value of a parameter is used directly to retrieve a database record

Sample request:

http://www.djfhsmsncj.com?invoice=12345

In this case, the value of the invoice parameter is used as an index in an invoices table in the database.

The application takes the value of this parameter and uses it in a query to the database.

The application then returns the invoice information to the user. Since the value of invoice goes directly into the query, by modifying the value of the parameter it is possible to retrieve any invoice object, regardless of the user to whom the invoice belongs [11].

To test for this case the tester should obtain the identifier of an invoice and then check whether it is possible to access objects without authorization. The value of a parameter is used directly to perform an operation in the system.

Sample request:

http://www.djfgsmsncj.com?user=someuser

In this case, the value of the user parameter is used to tell the application for which user it should change the password [15].

In the first step the application will get a request stating for which user's password is to be changed, and in the next step the user will provide a new password (without asking for the current one).

The user parameter is used to directly reference the object of the user for whom the password change operation will be performed. To test for this case the tester should attempt to provide a different test username than the one currently logged in, and check whether it is possible to modify the password of another user.

The value of a parameter is used directly to retrieve a file system resource

Sample request:

http://www.djfgsmsncj.com?img=img00011

In this case, the value of the file parameter is used to tell the application what file the user intends to retrieve.

By providing the name or identifier of a different file (for example file=image00012.jpg) the attacker will be able to retrieve objects belonging to other users. To test for this case, the tester should obtain a reference the user is not supposed to be able to access and attempt to access it by using it as the value of file parameter [11, 15].

Sample request:

<http://www.djfgsmsncj.com?menuitem> = 12

In this case, the value of the menu item parameter is used to tell the application which menu item (and therefore which application functionality) the user is attempting to access.

Assume the user is supposed to be restricted and therefore has links available only to access to menu items 1, 2 and 3. By modifying the value of menu item parameter it is possible to bypass authorization and access additional application functionality [2, 5].

In the above examples the modification of a single parameter is sufficient. However, sometimes the object reference may be split between more than one parameter.

**PREVENT INSECURE DIRECT OBJECT REFERENCES:-**

1. Check Access: - Use of the direct object reference from an invalidated and untrusted source must involve an access control verify to ensure the user is authorized for the requested object.

2. Instead of using the resource’s database key, a drop down list of seven resources authorized for requires protection .current user could use the number of 1 to 7 to indicate which value the user selected. That is direct targeting unauthorized resources [12, 16]. So use the per user or session indirect object references.

**4. INTRODUCTION TO SECURITY MISCONFIGURATION**

Security misconfiguration can occurs at web servers, application servers, database, framework, custom code and at any level of application stack [1].

Administrators of system and developers must have to work together and with proper configuration to ensure that the entire stack is properly configured and protected [2, 6].

Misconfigurations, missing patches, use of default accounts, unnecessary services, etc. represent a lot of potential risk an attacker can use if he has appropriate and enough information on the environment. An automated scanners are useful for detecting these components [1].

For Examples:-

1. Mitigations and peace:-

Components you have properly configure:-

-Portal (Life Ray).

-Third party frameworks [4, 6, 9]

-Web servers (HTTPD)

-Application server (Jetty, Tomcat, Glassfish)

-Java Runtime Environment (Sun, IBM)

-Operating System (Linux, Solaris, AIX, Windows)

- Relational Database Management System (RDBMS) (Oracle, SQL Server)

2. Stack Trace:-

-The stack trace showed or displayed in a web page. By using information, the attacker knows which portal solution is used and administration account was not deleted and the password was not updated or modified, then attacker can access and login as administrator and do whatever.

3. HTTP Header:-

The attacker knows which server you are using and find vulnerabilities by checking the server HTTP header.

There are two security misconfiguration.

**1. Application Misconfiguration:-**

Application based configuration files less protected or locked down may reveal clear text strings to the database [4].

Default string configuration files may conduct to unauthorized access to sensitive or may not have been set with security.

Application misconfiguration attacks make use of configuration weakness found in web applications.

Some aspects and features may provide a meaning for hacker to authentication methods and gain access to information and high privileges [4].

**2. Server Misconfiguration:-**

Sever misconfiguration attacks exploit or make use of configuration weakness found in web server and application servers.

Many server come with unnecessary sample files and default files, configuration files, scripts, including applications and web pages. They may have unnecessary enabled services [11, 16].

**How to Test:-**

Some procedures are given below [4, 8, 9, and 14]:-

1. Does your defect management and error handling reveal stack traces or other informative error shows to users?

2. Are passwords and their default accounts still enabled and not updated or unchanged?

3. Are any unnecessary features installed (pages, accounts, services, ports)?

4. Is any software out of date? This includes the DBMS, applications, App server, web server and OS and all code libraries.

5. Are the security configuration settings in frameworks and libraries not set to secure and properly values?

**How Do I Prevent Security Misconfiguration:-**

There are some precautions for security configuration:-

1. Strong application architecture must have that gives secure and effective separation between components.

2. Check running scans and consider doing audits periodically that help to detect future misconfigurations and missing patches [4, 7].

3. The process should be automated to minimize the effort required to setup a new secure environment.

**5**. **INTRODUCTION TO SENSITIVE DATA EXPOSURE**

Sensitive Data is information that can be used or manipulated for sneaky purposes to good effect, like that credit card numbers, tax id’s, and authentication credentials. Most generally and common flaws and defects are not simply encrypting sensitive data. If data is sensitive then protection of this data is required and to protect this data used to some technical steps, encryption, hashing, and etc. Internet burden is especially dangerous [1]. If any weak and old cryptographic algorithms are used then data can become sensitive and attackers can easily access your credentials [4, 7, and 9]. If cryptographic keys are weak or sleazy generated, or not proper key management then problem can occur and to solve this problem, crypto keys should be properly protected. Browser weakness and faults are very simple to find out but hard to exploit it [1].

**DETECTION: -**

There are given some steps to figure out sensitive data exposure vulnerability:-

1. If any cryptographic key is weak then it means credential details are in dangerous zone.
2. Credential data should not be simple and short term. It should be larger [11, 16].
3. If data transmitted in clear text or visible text then problem can occur.
4. Internet traffic can make a dangerous situation.

**PREVENTION**

The vast majority of the time, properly encrypting your sensitive data is sufficient to protect it. However, it must always be encrypted. This means, while it’s at rest in your data store, while it’s in transit on the wire, and while it is displayed in your customers’ browsers. This means encrypting sensitive data in storage, using TLS(Transport Layer Security)/SSL(Secure Sockets Layer) on pages displaying sensitive data and helping to protect against oblique attack vectors against your customers (such as via XSS) [4, 6, 11].

Steps:-

1. Do not save unnecessarily data. Discard remove it as soon as possible.
2. Checking the threats or bluster to protect this data from for example inside attacks and external attacks user, make sure you encrypt all sensitive data.
3. Strong encrypted algorithm should be used to protect sensitive data.
4. All internet burdens should be encrypted.
5. Proper browser and headers should be set to protect sensitive data exposure.
6. Use strong passwords and long term passwords with encryption.

**6. INTRODUCTION TO MISSING FUNCTION LEVEL ACCESS CONTROL**

The functions are what your web app is capable of doing. The “access control” means making sure only the right people use your tools. And the “missing” means, well… you get the picture. If each functions are accessed Applications need to perform the similar access control figure out on the server when each function is accessed. Virtually all web applications calibrate function level access rights before making that functionality penetrable in the User Interface [1, 7, and 9]. A typical example of this vulnerability is when a user notices a URL parameter and manually changes it to access unintended information. For example, if after authenticating, I visited my secure party at www.party.example.com?user=josh. And then manually changed the URL to be www.party.example.com?user=yourName. And was able to read your private diary that would be missing function level access control [1].

**Risk Factors:** Anyone can send a request to web app. Question is, do they have access to private functions which are meant for registered users?

**Attack Vector:** Simply URL changing.

**Security Weakness:** Functions are not protected enough. Developers forget to include code check, if user is logged in, or do they have access. They assume that they have. Detecting is easy. Hardest part is to get to know URLs and functions which there are available to attack [6, 9].

**Technical impacts:** Unauthorized access to functions. Anonymous unnamed users can simply access your private or credential functions if it’s are no properly protected.

**DETECTION**

There are some points to give an idea about this vulnerability and the appropriate way to find out this vulnerability is given below:-

1. If checking on information about data transmission or provided by attacker is not properly from server side then your data, credential function and personal information will suffer in problem and attacker can find out your info [12, 16].
2. If any UI (User Interface) demonstrate will occur or navigation to unauthorized functions then this vulnerability can make sure.
3. If server side authentication is not good and server disable to proper authenticate of any function level then this vulnerability can occur.

**PREVENTION**

1. Authentication from server side should be proper and clearly [12, 15].
2. Verify every application function.
3. Automated tools are used to find out missing function level access control.
4. Application function should have simple utilization or analyzed authentication module.
5. Clearly check the conditions are in the proper state to allow access and make sure that access is utilized or not [11, 15].
6. Ensure your updates about function and processing should be good managed.

**7. KNOWN VULNERABLE COMPONENTS**

If sometimes developers and development team do not focus on ensuring application libraries and components are properly update then flaws can occur in application libraries. Sometimes developers do not aware all that vulnerable factors and never remind their versions so application features can easily defect [1, 5]. Some automated tools are used to help for finding these vulnerable components and some framework also use. Recognition of vulnerability many useful frameworks are used to help this vulnerability. By using scanning and simple analysis, Attackers can easily recognize a weak or not proper protected components and contain information about everything. The most vulnerable components are Apache CXF’s authentication by pass [1].

**DETECTION**

There are some steps to figure out this vulnerability such as: -

1. If you are using a vulnerable components, libraries, and frameworks but you do not aware about it then it means you are in dangerous zone [11, 14].
2. If components version do not specify or numbering it means there are a problem.
3. If all properties of components, libraries, and frameworks are not specified then it can vulnerable.

**PREVENTION**

There are some prevention steps used to protect from this vulnerability: -

1. Upgrading of new versions should have important aspects and focus on it.
2. In next version, the problems of old version should be solved and new version should have all good things that can be increased our system performance.
3. Vulnerability patches should be constructed for old version [10, 14].
4. Security policies should be established in components, software, and frameworks.
5. Every components should have approved license that can be security protected.
6. Control the security of these components in public databases and security mailing lists and keep them update.
7. Every components should have security wrappers and disable unsafe functionality.

**8. INTRODUCTION TO INVALIDATED REDIRECTS AND FORWARD**

Invalidated redirects and forward are occur when a web application uses or confess unreliable input data that could cause the web application to redirect the request to a URL implied within unreliable input data. By proper improvement in untrusted URL input to a malicious web sites, an attacker can proudly steal user credentials details and launch a phishing scam [1, 10, and 16].

Because the server name in the variation link is same to the real website, phishing attempts can have a more reliable appearance or present. Invalidated redirects and forward attacks may be used to maliciously fetch a URL that would pass the application’s access control find and then forward the attacker to privileged functions that they would normally not be able to access. If a user is redirected or forwarded to a page defined by an unverified entry, the target URL may be manipulated or handle by an attacker [1].

**How to TEST:-**

The proper way to figure out the problem if an application has any invalidated redirects or forwards is:-

1. If code is not available, secure and check out those parameters to figure out if they view as of a redirect or forward URL destination [12].

2. The website to see if it generates any redirects (HTTP response code 400-408, typically 403). Figure out the factors or parameters or argument supplied to their direct to look if they seem to be a target URL. If so, change the target and analyze whether site redirects to new right way or target [1, 5, 8, and 9].

3. Review the code for all uses of redirect or forward. For each use, recognition if the target URL is conjoined in any parameter values. If so, if the target URL is not validated against a whitelist, then you are vulnerable.

**PREVENT:-**

1. Make clarify input by creating a list of reliable URL’s (lists of hosts).

2. Force all redirects to first go through a page notifying users that they are going off your website, and have them click a link for confirmation.

3. Simply avoid by use of redirects and forwards [12].

4. If user input the data cannot be excluded, secure or confirm that the feed or supplied value is authorized and valid, appropriate for the application, and is valid for the user.

5. There is a recommended that any of such destination input be mapped to a value, rather than the proper URL or part of the URL, and that server side code interpret this value to target URL.

6. If preferred, do not allow the URL as input for the destination. This may usually be done, should have a validated URL.

**CONCLUSION**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **NAME OF VULNERABILITY** | **FIRST APPLIED** | **WHO DISCOVER THIS** | **YEAR** | **EFFECTIVE RATE** | **APPLIED AREA** | **TYPE** | **AFFECTED PART IN WEBSITE** | **NOW CAPABILITY TO ATTACK** | **FLAWS** | **PRECAUTIONS** | **TOOL** | **REFERENCES** |
| SQL INJECTION | In February 2002, Jeremiah Jacks discovered that Guess.com was vulnerable to an SQL injection attack | Jeremiah Jacks | The first public discussions of SQL injection started around 1998. | No. 1 | No Proper coded Session login page | Attack | Login panel | Down Because of Advanced SQL | Less proper java script | Use proper java script | SQL injection me ,HAVIJ | [1, 6,9,11] |
| BROKEN AUTHENTICATION AND SESSION MANAGEMENT | ----- | ------ | ------ | No. 4 | Normally weak websites | Vulnerability | administrative accounts, undermine authorization and accountability controls | Stop because technology is best and good Coders | Less knowledge of developers | Use best Developers | Hack bar | [1, ,5, 8, 12] |
| INSECURE DIRECT OBJECT REFERENCE | This type of attack occurred to the Australian Taxation Office’s GST Start Up Assistance site. | hostile user | In 2000. | No. 5 | All web application frameworks are vulnerable to attacks on insecure direct object references. | Vulnerability | direct references to restricted resources of web sites | Normal | unauthorized resources and access | Authorized resources and session | Burp | [1, 3. 7, 9, 11] |
| SECURITY MISCONFIGURATION | ------- | ------ | ------ | No. 6 | web server, application server, database, framework, and custom code | Vulnerability | HTTP header and Stack Trace | Average Probable | the security settings in your development frameworks (e.g., Struts, spring, ASP.NET) and libraries not set to secure values | strong application architecture that provides effective, secure separation between components | WATOBO | [1, 4, 6, 9, 15] |
| SENSITIVE DATA EXPOSURE | ------ | ----- | ------ | No. 7 | unauthorized access | Vulnerability | unauthorized access | Stop | weak crypto keys generated | Ensure strong standard algorithms | QUALYS SSL server test | [1, 13, 15] |
| MISSING FUNCTION LEVEL ACCESS CONTROL | ------- | ------ | ------ | No. 8 | URL | Vulnerability | Unauthorized access to functions | N/A | server side authentication or authorization checks missing | Proper server side authentication | Open VAS | [1, 14, 16] |
| USING KNOWN VULNERABLE COMPONENTS | ------- | ----- | -------- | No. 9 | frameworks, and other software modules | Vulnerability | frameworks, and other software modules | N/A | Use of Unknown components | Use of proper Authorized components | Open VAS | [1, 11, 12, 16] |
| UNVALIDATED REDIRECTS ANDFORWARDS | ------- | ------ | ------ | No. 10 | the target URL can be manipulated by an attacker | Vulnerability | URL | N/A | the code for all uses of redirect or forward | Sanitize input by creating a list of trusted URL's | Watcher | [1, 11, 12] |

Table-1: Summary of the Key Vulnerabilities in the

**REFERENCES:-**

1. https://www.owasp.org

2. http://www.w3schools.com

3. http://technet.microsoft.com

4. http://php.net

5. https://www.acunetix.com

6. http://www.veracode.com

7. http://www.unixwiz.net

8. http://www.imperva.com

9. http://blog.credera.com

10. http://www.triadsquare.com

11. https://www.indusface.com

12. http://www.troyhunt.com

13. http://kb.entersoft.co.in

14. http://www.jtmelton.com

15. https://bounty.github.com/classifications/security-misconfiguration.html

16. http://blog.ippon.fr