# Web Threats Detection and Prevention for Vulnerabilities

[1]S. A. Gadhiya, [2]K. H. Wandra
[1]Research Scholar C. U. Shah University- Wadhwan City, [2]Principal, GMB Rajula.

*Abstract:* As the web and web application becomes increasingly complex, web applications become more sophisticated and dynamic with rapidly increasing speed of internet and computational power of computing devices. One of the most important challenges that web applications have become complex is in how they use input data. Web pages are no longer static, they contain dynamic content from sources that may be trusted, un trusted, or trusted but potentially buggy. There are many places that this type of data can come from user input, advertisements, or widgets, to name a few. These sources of data have led to a class of attacks know as content injection attacks. In these attacks, an attacker is able to place malicious content on a page and make it act as if it came from the developer. This can lead to cross-site scripting attacks, phishing, or malicious information. In order to counter these types of attacks, developers implement web application security technique that allowed and disallowed content on a web page. The security techniques have been implemented by use of sanitization functions, or content filters, that modify un trusted data to conform to a well-understood and safe set of behaviors. Many web applications available today make use of some way of session management to be able to couple state to a particular user. This state varies from the user's preferences to user authentication and private information. Unfortunately, it is possible for an attacker to exploit session management in order to impersonate another user at a web application. In this thesis we describe attacks that enable an attacker to impersonate a victim, and the ways in which they can be prevented. Different attacks abusing session management are known: session hijacking, wherein the attacker captures a victim's session identifier (or SID); session fixation, wherein the attacker imposes his own SID upon a victim's web browser; and cross site request forgery, wherein the attacker uses a victim's browser to issue requests as if they came from the victim. For all three attacks, different attack vectors exist, which allow an attacker to create complex attack scenarios which are difficult to prevent. In this paper we have discussed various tools available to detect Injection attack and various prevention measures for them.

***Index Terms* – Web Threats, Code Injection, Web Threats Detection, Web Threats Prevention.**

## I. INTRODUCTION

Currently all individual and large organizations are depends on web and web application. Each and everything is stored, available or traded on the web. Web applications can be personal websites, blogs, news, social networks, web mails, bank agencies, forums, e-commerce applications, etc. The Web and web application are essential for our life and in our economy. The security motivation of web and web application developers and administrators should reflect the importance and significance of the resources they are supposed to protect. Even though there is an increasing anxiety about security, there are major factors that make securing web applications a complicated task to complete. The web and web application demands and utilization are rising fast, ensuing in a huge production of web applications, based on different languages, frameworks, and protocols. Web and web applications are very much exposed to attacks from anywhere in the world, which can be conducted by using widely available and simple tools like a web browser. More than a few alternative solutions have been projected for various types of content. Increasingly, web frameworks are used to computerize the application of sanitizers to HTML content. For more dynamic and context specific content, such as advertisements, verifiers and dynamic enforcement mechanisms have been proposed. Alternatively, there have been a number of proposals, such as Content Security Policies [1][2] to make these HTML security policies explicit in web applications [3]. However, none of these approaches addresses the need to analyze what the security policies of web applications actually are. That is to say, all of these approaches apply mechanisms without asking what the policy is. Furthermore, many of these solutions have a variety of problems that prevent developers from fully embracing them. We propose that tools can be built to greatly improve the analysis [3][4] and understanding of web application security policies, as well as be used to better implement such policies, both on legacy and new applications.

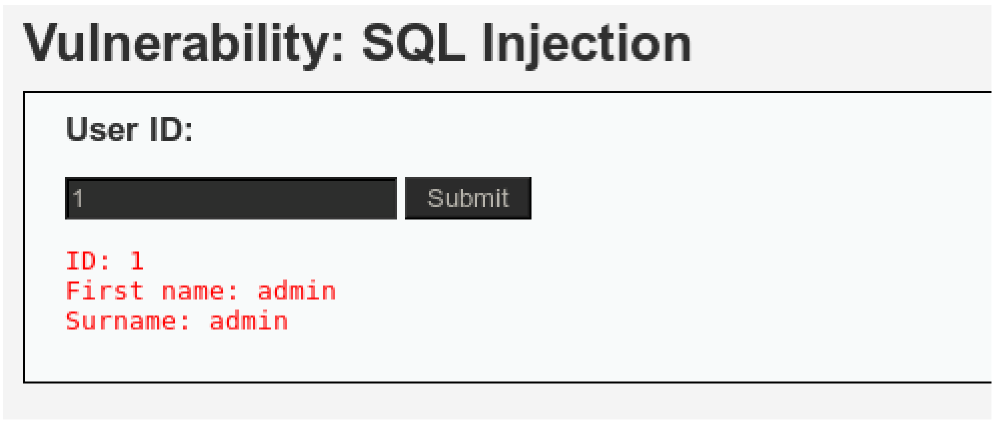
## II. TOOL USED FOR DETECTION OF INJECTION ATTACK

There are various tools available for detecting of Injection attack like SQL Injection [4][5][6] and Cross site scripting attack[5][6]. We have analyze tools those are free and available in Kali Linux 2.0

### 1. Damn Vulnerable Web Application (DVWA)

Damn Vulnerable Web App (DVWA) [6][7] is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

If we enter the number 1 and we click on the submit button we will notice that it will return the first name and the surname of the user with ID=1.



Vulnerability: SQL Injection

User ID:

```
ID: 1
First name: admin
Surname: admin
```

This means that the query that was executed back in the database as
" SELECT First_Name,Last_Name FROM users WHERE ID=' 1' ;"

We can try to change the ID number on the URL to other values (i.e 2,3,4 etc.) in order to find the first names and surnames of all the users.For example we have discovered the following:
id=2 —> First Name: Gordon Surname: Brown
id=3 —> First Name: Hack Surname: Me
id=4 —> First Name: Pablo Surname: Picasso
id=5 —> First Name: Bob Surname: Smith

## 2. SQLMAP

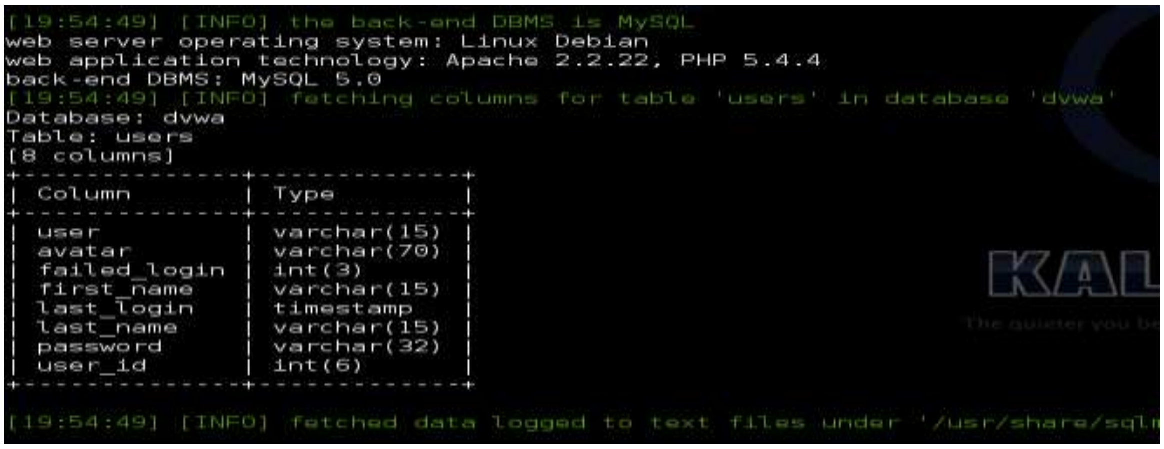
Sqlmap [8] is an open source penetration testing tool that automates the process of detecting and exploiting [10][11] SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections. Features of SQLMAP given below.

Full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB and HSQLDB database management systems [12][13].

Support to directly connect to the database without passing via a SQL injection, by providing DBMS credentials, IP address, port and database name [14].

Support to enumerate users, password hashes, privileges, roles, databases, tables and columns.
Automatic recognition of password hash formats and support for cracking them using a dictionary-based attack [15].

**Test case :** Sqlmap –u "url" –cookie"cookievalue" –D dvwa -T users --columns

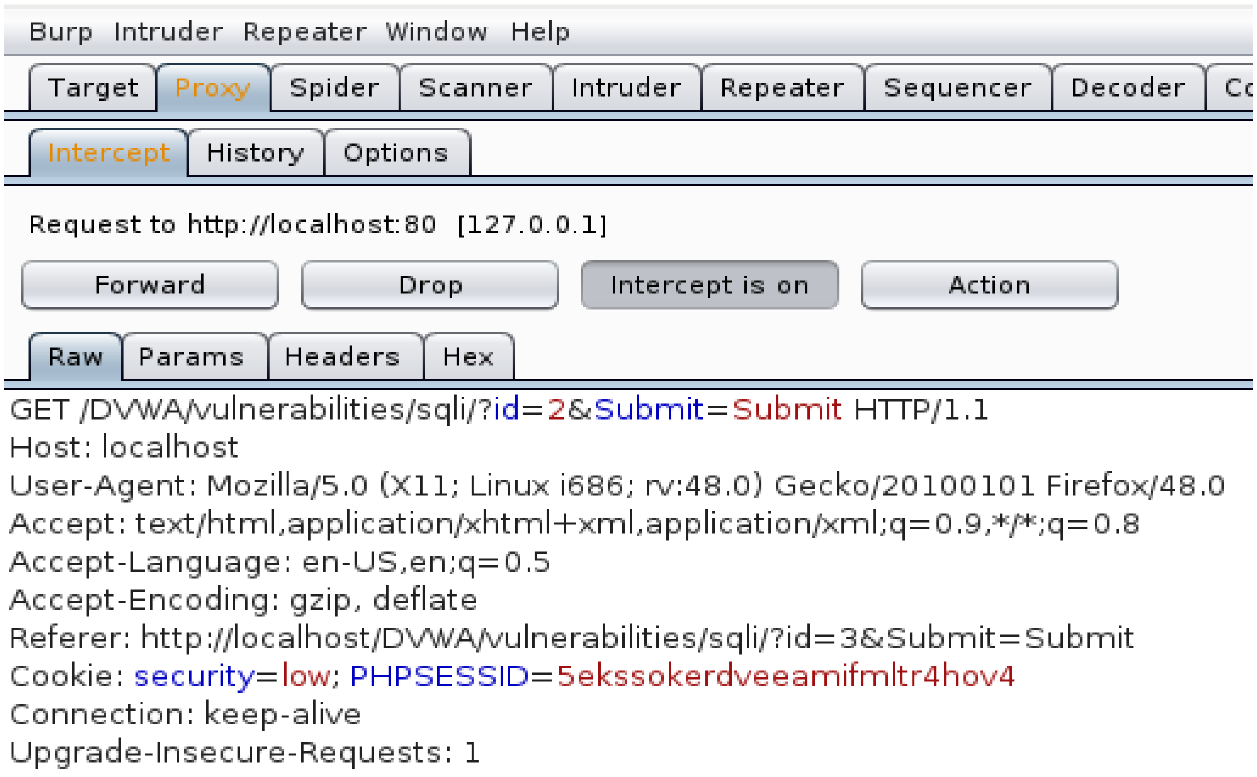


```
[19:54:49] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.2.22, PHP 5.4.4
back-end DBMS: MySQL 5.0
[19:54:49] [INFO] fetching columns for table 'users' in database 'dvwa'
Database: dvwa
Table: users
[8 columns]
+--------------+-------------+
| Column       | Type        |
+--------------+-------------+
| user         | varchar(15) |
| avatar       | varchar(70) |
| failed_login | int(3)      |
| first_name   | varchar(15) |
| last_login   | timestamp   |
| last_name    | varchar(15) |
| password     | varchar(32) |
| user_id      | int(6)      |
+--------------+-------------+

[19:54:49] [INFO] fetched data logged to text files under '/usr/share/sqlm
```

### 3. Burf Suit

Burp Suite [16] is an integrated platform for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Co
Intercept History Options
Request to http://localhost:80 [127.0.0.1]
Forward Drop Intercept is on Action
Raw Params Headers Hex

```
GET /DVWA/vulnerabilities/sqli/?id=2&Submit=Submit HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:48.0) Gecko/20100101 Firefox/48.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/DVWA/vulnerabilities/sqli/?id=3&Submit=Submit
Cookie: security=low; PHPSESSID=5ekssokerdveeamifmltr4hov4
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

## III. Prevention for Web Threats

We have discussed various code injection approach in this section.

### 1. Prevent Cross Site Scripting Attack

XSS attacks, in essence, trick an application into sending malicious script [16][17] through the browser, which believes the script is coming from the trusted website. Each time an end user accesses the affected page, their browser will download and run the malicious script as if it was part of the page. In the majority of XSS attacks, the attacker will try to hijack the user's session by stealing their cookies and session tokens, or will use the opportunity to spread malware and malicious JavaScript.

XSS vulnerabilities are difficult to prevent simply because there are so many vectors where an XSS attack can be used in most applications. In addition, whereas other vulnerabilities, such as SQL injection or OS command injection, XSS only affects the user of the website, making them more difficult to catch and even harder to fix. Also unlike SQL injection [18][19], which can be eliminated with the proper use of prepared statements, there's no single standard or strategy to preventing cross-site scripting attacks.

There are two main types of cross-site scripting attacks: Stored (or persistent) XSS, which is when malicious script is injected directly into the vulnerable application, and reflected XSS, which involves 'reflecting' malicious script into a link on a page, which will activate the attack once the link has been clicked.

### 2. Prevent SQLI Attack

SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation [20] to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. The impact SQL injection can have on a business is far reaching. A successful attack may result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business. When calculating the potential cost of a SQLI, it's important to consider the loss of customer trust should personal information such as phone numbers, addresses and credit card details be stolen. While this vector can be used to attack any SQL database, websites are the most frequent targets.

There are several effective ways to prevent SQLI attacks [1][5][8] from taking place, as well as protecting against them, should they occur. The first step is input validation (sanitization), which is the practice of writing code that can identify illegitimate user inputs. While input validation should always be considered best practice, it is rarely a foolproof solution. The reality is that, in most cases, it is simply not feasible to map out all legal and illegal inputs—at least not without causing a large amount of false positives, which interfere with user experience and an application's functionality.

## IV. Conclusion

We have discussed various tools and technique for detection for code injection vulnerability. It is very challenging field to protect web and web application for new threats. However it is possible by updating current policy for securing.

## References

**[1].** D. Watson, "Web application attacks," Network Security, vol. 2007 issue 11, pp. 7-12, November 2007

**[2].** W. Kim, O. Jeong,C. Kim, and J. So, "The dark side of the internet:attacks, costs and responses", Information Systems, vol. 36, pp. 675-705, 2011

**[3].** http://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf

**[4].** https://www.owaspp.org/index.php/DOM_Based_XSS

**[5].** https://www.owaspp.org/index.php/Cross-site_Scripting_(XSS)

**[6].** en.wikipedia.org/wiki/SQL_injection

**[7].** code.tutplus.com

**[8].** en.wikipedia.org/wiki/Cross-site_scripting-15/12/2014

**[9].** www.excess-xss.com

**[10].** Omar Ismail Masashi Etoh Youki Kadobayashi, "A Proposal and Implementation of Automatic Detection/Collection System for Cross-Site Scripting Vulnerability", AINA-2004, pp 145-151.

**[11].** Abdul Razzaq, Ali Hur, Nasir Haider, Farooq Ahmad,"Multi-Layered Defense against Web Application Attacks", ITNG2009,pp 492-497.

**[12].** Adam Kie˙zun, Philip J. Guo, Karthick Jayaraman, Michael D. Ernst, "Automatic Creation of SQL Injection and Cross Site Scripting Attacks", ICSE-2009,pp 199-209.

**[13].** Rattipong Putthacharoen, Pratheep Bunyatnoparat,"Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique", ICACT-2011, pp 1090-1094.

**[14].** Yi Wang, Zhoujun Li, Tao Guo "Program Slicing Stored XSS Bugs in Web Application", TASE-2011, pp 191-194.

**[15].** Hossain Shahriar, Mohammad Zulkernine, "S2XS2: A Server Side Approach to Automatically Detect XSS Attacks", DASC-2011, pp 7-14.

**[16].** Hossain Shahriar, Mohammad Zulkernine"Injecting Comments to Detect JavaScript Code Injection Attacks",COMPSACW-2011, pp 104-109.

**[17].** L.K. Shar,H.B.K. Tan,"Auditing the XSS defence features implemented in web application programs",ITE-2012, pp 377-390.

**[18].** Indrani Balasundaram, E.Ramaraj "An Authentication Scheme for Preventing SQL Injection Attack Using Hybrid Encryption(PSQLIAHBE"(ISSN 1450-216X Vol.53 No.3(2011),pp.359-368)

**[19].** Pomeroy, A Qing Tan Sch. of Comput. & Inf. Syst., Athabasca Univ., Athabasca, AB, Canada " Effective SQL Injection Attack Reconstruction Using Network Recording" in Computer and Information Technology (CIT), 2011 IEEE 11th International onference Issue Date: Aug. 31 2011Sept. 2011 On page(s):552-556.