# Detecting Vulnerabilities of Web Application Using Penetration Testing and Prevent Using Threat Modeling

**Sandip Sarkar**

**Abstract** The number of Web attacks is increasing gradually, mainly the popularity of Web application in organization, school, and colleges. For this reason, the security of their sensitive information against attacker becomes very important for all organization and companies. In this paper, we describe different type of Web application attack like SQL injection, XSS attack, CSRF attack, and Buffer overflow. Besides, we discuss about different types of penetration tools for Web applications. Penetration testing try to find the vulnerabilities of Web application so that we can build a defense mechanism to deal with Web attack. Finally, we build attack trees and defense trees to represent the attacks and to prevent those attack.

**Keywords** Web application · SQL injection · XSS attack · CSRF attack · Attack tree · Defense tree

## 1 Introduction

In the beginning of Web application, there were only static Web pages which contained static information. Now, the popularity of Web application gradually increases, and in the same time, the architecture of Web application become more complex. Web applications are used in organizations, bank, so companies concentrate to secure their sensitive data like username, password, bank card numbers, etc. An attacker can get sensitive information using malicious code. SQL injection and cross-site scripting (XSS) are the two most famous vulnerabilities in Web application. Detection or prevention of Web attack is a challenging issue. To detect the vulnerabilities of Web application, we have used different types of automatic vulnerabilities tools but none of them can guarantee to find the vulnerabilities of Web application. In our paper, Sect. 2 describes different types of Web attack. Section 3 gives some brief description of different types of Web application-based penetration tools. We build different types of Web attack trees and those are described in Sect. 4. Section 5

S. Sarkar (✉)
Hijli College, Kharagpur, India
e-mail: Sandipsarkar.ju@gmail.com

P. K. Mallick et al. (eds.), *Advances in Electronics, Communication and Computing*, Lecture Notes in Electrical Engineering 709,
https://doi.org/10.1007/978-981-15-8752-8_3

gives the information that how to prevent those attack using defense trees. Finally, Sect. 6 conclude the paper.

## 2 Different Types of Web Attack

In this paper, we discuss different types of vulnerabilities of Web applications. Besides, counter mechanism for those Web vulnerabilities is given.

### *2.1 SQL Injection*

SQL injection is one type of code injections in which malicious code injected into the SQL query so that attacker can direct access to the database and leak confidential, or even sensitive, information without proper authorization [1, 2]. The main reason of SQL injection is mainly because of insufficient validation of user input. Table 1 describes the statics of SQL injection from year 2012 to 2019.

(a) **Tautology**

In SQL tautology, attacker user injects malicious code into one or more conditional statements to bypass user authentication [3]. If a malicious user enters ‘*OR*’ 1 = 1- - instead of a legitimate username into username fields, then the SQL query looks as follows:

*select * from user where name = ‘Alice ‘OR’ 1=1- - ‘and password =”*

This statement is only checking the username field and successfully bypassing the authentication mechanism. Similarly, attacker injects *Alice*/* into the username field and */ into the password field to bypass user authentication. This malicious code.

*select * from user where name = ‘Alice’/* ‘and password =’*/*
*select * from user where name = Sandip/# and password =#/*

**Table 1** Statistics of SQL injection

| Year | Matches | Total | Percentage (%) |
|---|---|---|---|
| 2012 | 366 | 5288 | 6.92 |
| 2013 | 269 | 5187 | 5.19 |
| 2014 | 478 | 7937 | 6.02 |
| 2015 | 389 | 6487 | 6.00 |
| 2016 | 253 | 6447 | 3.92 |
| 2017 | 692 | 14,645 | 4.73 |
| 2018 | 732 | 16,512 | 4.43 |
| 2019 | 820 | 17,311 | 4.74 |

(b) **Union Query**

A common example of SQL injection to add the statement ‘union select’, along with an additional target dataset so that queries return the union of the intended database with the target database.

*Select * from users where username = ‘union select * from student -’ and pwd = ‘xyz’;*

The first SELECT query gives no result but the second query returns all information about student.

(c) **Piggy-Backed Query**

In this technique, malicious user supply relies on server configurations that allow several different queries within a single string of code. For example, an attacker can add a query delimiter such as ‘;’, use it in such as a way that can delete using drop table command.

*select name from student where password=‘Kharagpur’;drop table user;*
*update employee set position.id = ‘2456’ where id = ‘255’; delete from orders WHERE id = ‘C0201’;*

## 2.2 *Cross-Site Scripting*

A cross-site scripting attack (also known as XSS or CSS) occurs, due to poor security awareness of developers [4]. In this attack, the attacker executes malicious code on the victim’s machine for lack of input validation [5–7]. There are two type of XSS attack: (i) Reflected XSS attack and (ii) Stored XSS attacks. Figure 1 describes about

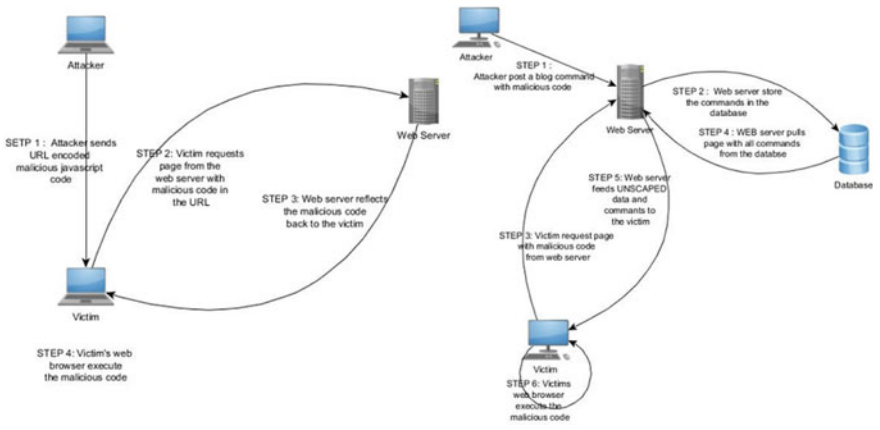
Attacker
SETP 1 : Attacker sends URL encoded malicious javascript code
Web Server
STEP 2: Victim requests page from the web server with malicious code in the URL
STEP 3: Web server reflects the malicious code back to the victim
Victim
STEP 4: Victim's web browser execute the malicious code
Attacker
STEP 1 : Attacker post a blog command with malicious code
Web Server
STEP 2 : Web server store the commands in the database
STEP 4 : WEB server pulls page with all commands from the databse
Database
STEP 5: Web server feeds UNSCAPED data and commants to the victim
STEP 3: Victim request page with malicious code from web server
Victim
STEP 6: Victims web browser execute the malicious code

**Fig. 1** Reflected and stored XSS attack

**Table 2** Statistics of cross-site scripting

| Year | Matches | Total | Percentage (%) |
|---|---|---|---|
| 2012 | 771 | 5288 | 14.58 |
| 2013 | 639 | 5187 | 12.32 |
| 2014 | 1092 | 7937 | 13.76 |
| 2015 | 780 | 6487 | 12.02 |
| 2016 | 409 | 6447 | 6.34 |
| 2017 | 943 | 14,645 | 6.44 |
| 2018 | 920 | 16,512 | 5.57 |
| 2019 | 856 | 17,311 | 4.94 |

reflected and stored XSS attack. Besides Table 2 shows the statistics of XSS attack form year 2012 to 2019.

(a) **Reflected XSS Attack**

In a reflected XSS attack, the actual malicious code is not stored on server but the malicious code are delivered to the victims via e-mail messages [8]. This type of attack mainly occurs when data submitted by the client is immediately processed by the server and send back the result to the client.

(b) **Stored XSS Attack**

In a stored XSS attack, the malicious code is permanently stored on the target server. The actual attack is occurred at later, when the client requests a dynamic page that is managed by this server. The user's Web browser executes the malicious code.

## 2.3 *Cross-Site Request Forgery*

Cross-Site Request Forgery is one type of Web attack where attack performs unauthorized activities using victims' authority and credentials [9, 10]. In this attack occurs while victim is currently logged into their account [11]. In the same time, victim's browser automatically sends request to the server without user's knowledge. In this attack, the server cannot understand which request is from the legitimate user. For example, a user sends a request to the server with its session ID while sending money to an account. An attacker can steal that user's session ID and send request to the server after modification of that request. An attacker can steal the session ID and send the request to the bank server and he can also put his account number. For this reason, the bank server deducts the money from the user and credit to the attacker account. Figure 2 describes the mechanism of this attack. The statistics of CSRF attack is given in Table 3.

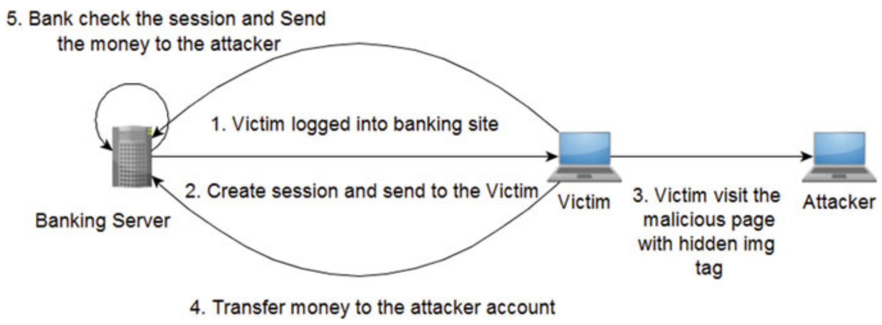
5. Bank check the session and Send the money to the attacker
1. Victim logged into banking site
2. Create session and send to the Victim
Banking Server
Victim
3. Victim visit the malicious page with hidden img tag
Attacker
4. Transfer money to the attacker account

**Fig. 2** CSRF attack

**Table 3** Statistics of cross-site request forgery

| Year | Matches | Total | Percentage (%) |
|---|---|---|---|
| 2012 | 165 | 5288 | 3.12 |
| 2013 | 120 | 5187 | 2.31 |
| 2014 | 247 | 7937 | 3.11 |
| 2015 | 246 | 6487 | 3.79 |
| 2016 | 73 | 6447 | 1.13 |
| 2017 | 190 | 14,645 | 1.30 |
| 2018 | 154 | 16,512 | 0.93 |
| 2019 | 226 | 17,311 | 1.31 |

## 2.4 *Broken Authentication and Session Management*

Broken authentication and session management attack is one of the most common application layer attack mechanism used by attacker [12, 13]. Developer uses different type of cryptographic algorithms and session management tokens, but it is still a major problem how to secure the authentication. Wireshark is well-known packet collector tool to perform this attack. The network packet may contain password, session ID, cookies. If the logging session of a user was not managed properly then after the user's logout, session may still reside in the Web application. Another reason of this attack is to use GET method. User's private credential may be visible if the developer of the Web application use GET method.

## 2.5 *Security Misconfiguration*

The most common Web vulnerability is security mis-configuration which can occur in any layer of Web application [14]. Most of the time, Apache HTTP server and MySQL database server are used in Web application. Normally users use those

Web application environment with default settings. The configuration of MySQL is controlled by my.conf file or using MySQL-specific directives in php.ini. For empty root password of MySQL causes command injection attacks or denial of service attacks. PHPSecInfo and PHP security edit are automatic tool to check security mis-configuration. But both automatic tools are only limited to PHP.

## 3 Web Application Penetration Tools

A Web Application Penetration Test tries to provide a clear idea of the system and also provide how to secure an organizations information from real world attacks. In this section, I discuss about well-known penetration tools which help to detect the vulnerabilities of Web applications.

### *3.1 AMNESIA*

AMNESIA is a well-known penetration tool to detect SQL injection. This tool consists of two parts: one is static analysis and another one is runtime monitoring. This technique finds malicious code before being executed on the database. It uses model-based approach. In static part, it analyzes the Web application code to build a model of the legitimate queries. In dynamic part, it checks the dynamically generated queries with the statically built model using run time monitoring. This model finds the malicious queries and prevents it to access the database.

### *3.2 Xsser*

Xsser is an automatic and open-source framework to find the vulnerabilities of Web application. This framework contains several mechanisms to break different filters and various special techniques of code injection.

### *3.3 Acunetix*

Acunetix Web vulnerability scanner is an automated tool to find the vulnerabilities of the Web application. Security analyst uses Acunetix to find the vulnerabilities such that SQL injection, cross-site scripting, and weak passwords.

### *3.4 Sqlmap and Havij*

Sqlmap Havij are both automated SQL injection tool that help the developers to check SQL injection vulnerabilities of Web application. Attacker can retrieve username and password from login database using those automated tools. Sqlmap is developed using Python language and for this, it is independent of operating system.

### *3.5 Netsparker*

Netsparker is a very powerful Web application security scanner and it can find most of the vulnerabilities of the Web application. This penetration tool is platform independent. It is very useful for security analyst to build a secure Web application.

## 4 Threat Modeling Using Attack Tree

In the previous section, we discuss different types of threats of Web application. These threats can come inside the application or outside the application. For this reason, threat modeling is very essential to prevent sensitive information which are stored in the database. Threat modeling is mainly built to find problems before designing of a system. Threat modeling can be achieved by different mechanism. Nowadays, attack tree is very popular for the designing of threat modeling.

Attack trees describe a graphical representation of attacks which are performed by the attacker. The root of this tree is the main goal of attacker. Each node of the attack tree contains an action. There are two types of relationship (i.e., OR and AND relationship) which connect the child node. For OR relation between child node, if any of the child node is executed by the attacker, then attacker can access parent node. Likewise, for AND relation between child node, if all child nodes are executed

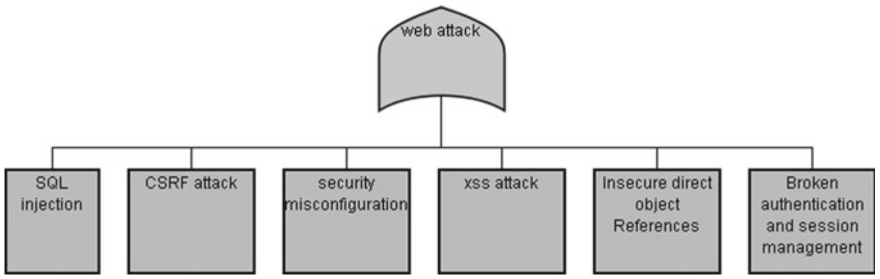
web attack
SQL injection
CSRF attack
security misconfiguration
xss attack
Insecure direct object References
Broken authentication and session management

**Fig. 3** Attack tree of web attack

by the attacker then attacker can access parent node. Figure 3 shows the attack tree of web attack.

We are trying to build an attack tree of Web attach which is shown in Fig. 4. It describes the general view of Web attacks is divided into six main categories.

Those are XSS attack, Insecure data object, Broken authentication and session management, CSRF attack, SQL injection, and security miss-configuration. Figure 5 describes the attack tree of SQL injection. SQL injection can be performed using steal system information or using attack against database. If any of the them is successful, then the parent node means SQL injection is successful. Similarly, other nodes of

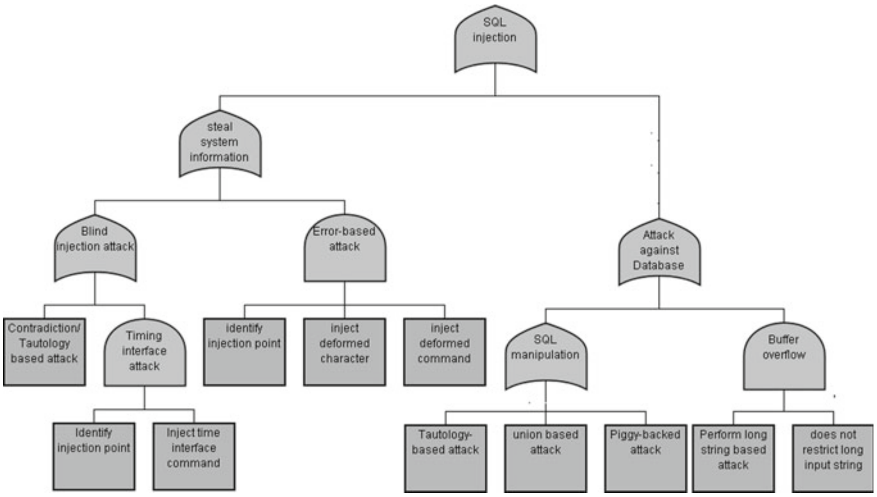

**Fig. 4** Attack tree of SQL injection

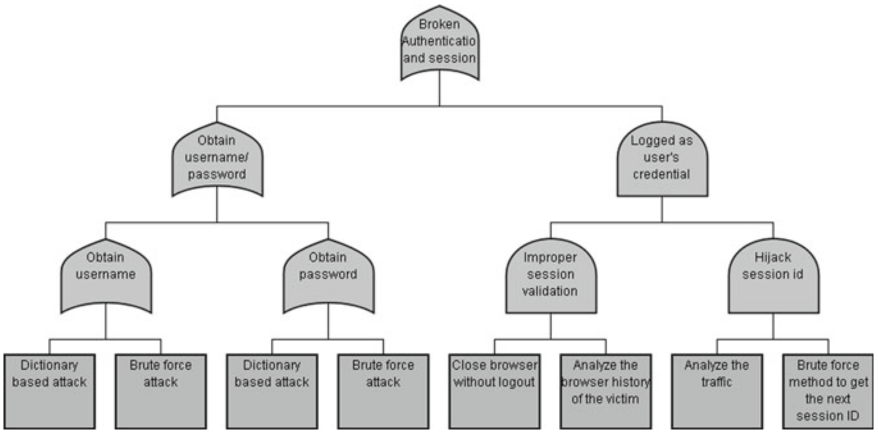
Broken Authenticatio and session
Obtain username/ password
Logged as user's credential
Obtain username
Obtain password
Improper session validation
Hijack session id
Dictionary based attack
Brute force attack
Dictionary based attack
Brute force attack
Close browser without logout
Analyze the browser history of the victim
Analyze the traffic
Brute force method to get the next session ID

**Fig. 5** Attack tree of broken authentication and session management

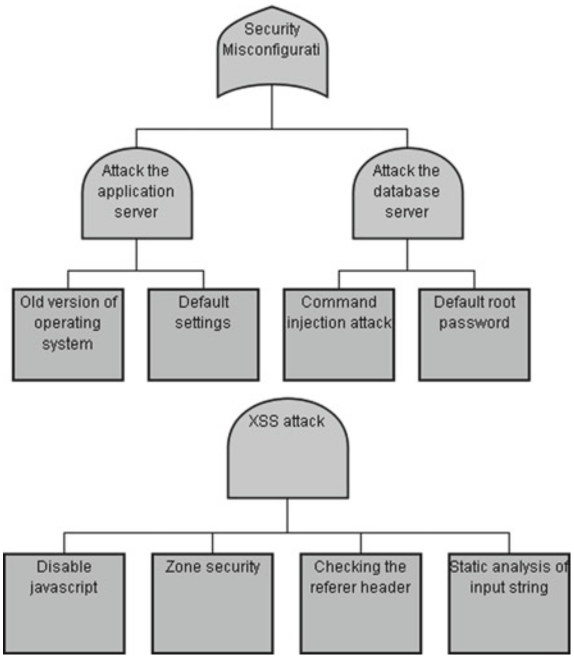
Security Misconfigurati
Attack the application server
Attack the database server
Old version of operating system
Default settings
Command injection attack
Default root password
XSS attack
Disable javascript
Zone security
Checking the referer header
Static analysis of input string

**Fig. 6** Attack tree of security mis-configuration and XSS attack

attack tree of SQL injection are executed using same scenario. Figure 5 shows that architecture of broken authentication and session attack. Attacker can obtain user name and password using brute-force attack and dictionary-based attack or user can enter into the system using user's session-id. Figure 6 described that xss attack is divided into three subdivisions. To perform persistence XSS attack, attacker first finds the injection point and then injects his code into the database. Here, the relation is AND operation, if the two-child node are successful then the Persistence XSS attack is successful. It is same for the reflected- and DOM-based XSS attack.

## 5 Defense Tree

Attack tree represents the attacking scenario based on attacker's point of view. We cannot secure our Web application using attack tree. For this reason, we need another type of mechanism (i.e., Defense tree). Defense tree represent the counter mechanism for different types of attack (i.e., described in attack tree). Figure 7 describes the defense tree of SQL injection. Database can be protected using encryption, minimum user privilege, and using prepare statement. In our previous section, we described about SQL injection. Similarly, Figs. 7, 8, and 9 describe the defense tree of broken authentication and session management, XSS attack, and security mis-configuration.

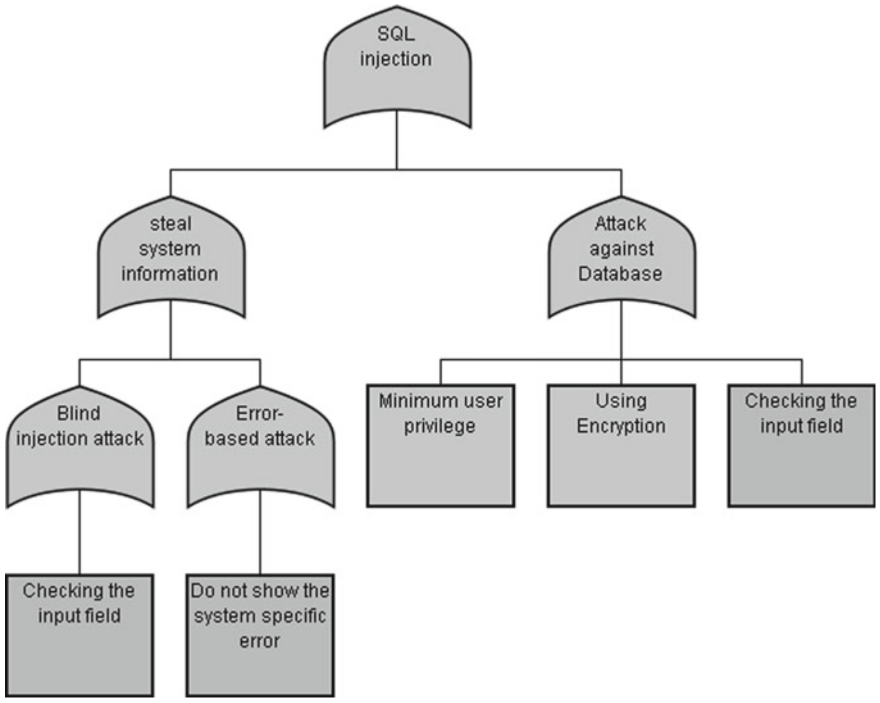

**Fig. 7** Defense tree of SQL injection

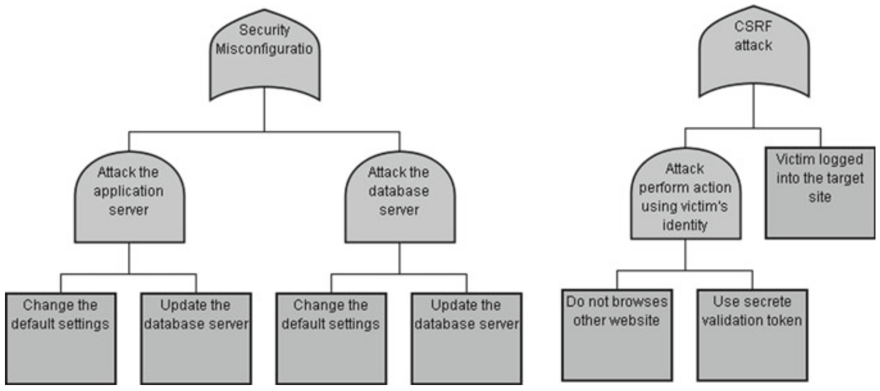
Security Misconfiguratio
Attack the application server
Attack the database server
Change the default settings
Update the database server
Change the default settings
Update the database server
CSRF attack
Attack perform action using victim's identity
Victim logged into the target site
Do not browses other website
Use secrete validation token

**Fig. 8** Defense tree of security mis-configuration and CSRF attack

## 6 Conclusion and Future Work

In this literature survey, different type of security flaws of Web applications is described. Web application vulnerabilities are mainly because of improper input

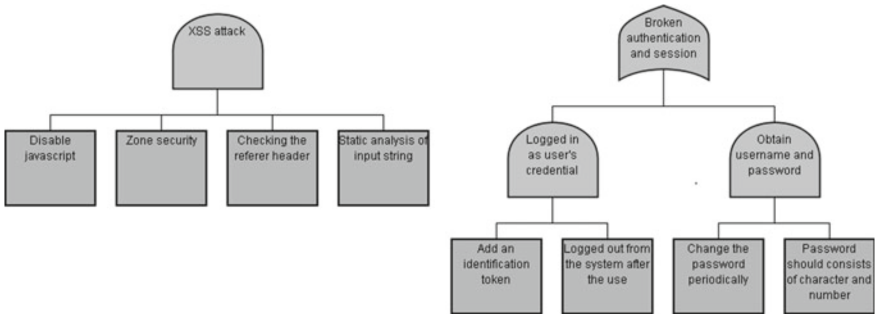
XSS attack
Disable javascript
Zone security
Checking the referer header
Static analysis of input string
Broken authentication and session
Logged in as user's credential
Obtain username and password
Add an identification token
Logged out from the system after the use
Change the password periodically
Password should consists of character and number

**Fig. 9** Defense tree of XSS and broken authentication and session management

validation and unawareness of security mechanism. In the same time, we also represent different Web attacks using attack tree. To prevent those Web attacks, we used different types of mechanism which are presented by defense tree. In the future, we plan to investigate new types of Web attacks which are top rank in upcoming years and want to investigate better counter mechanism to prevent those attack.

## References

1. Focardi, R., Luccio, F., & Squarcina, M. (2012). Fast sql blind injections in high latency networks. In *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)* (pp. 1–6), October 2012.
2. Benjamin, A. (2016). Search-based SQL injection attacks testing using genetic programming. In *Genetic Programming: 19th European Conference, EuroGP 2016*, Porto, Portugal (pp. 183–198), March 30–April 1, 2016.
3. Dharam, R., & Shiva, S. (2012). Runtime monitors for tautology based sql injection attacks. In *International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)* (pp. 253–258), June 2012.
4. Owasp top 10-2013. https://www.owasp.org/. The ten most critical web application security risks.
5. Zeng, H. (2013). Research on developing an attack and defense lab environment for cross site scripting education in higher vocational colleges. In *2013 Fifth International Conference on Computational and Information Sciences (ICCIS)* (pp. 1971–1974), June 2013.
6. Matsuda, T., Koizumi, D., & Sonoda, M. (2012). Cross site scripting attacks detection algorithm based on the appearance position of characters. In *2012 Mosharaka International Conference on Communications, Computers and Applications (MIC-CCA)* (pp. 65–70), October 2012.
7. Avancini, A., & Ceccato, M. (2013). Circe: A grammar-based oracle for testing cross-site scripting in web applications. In *2013 20 th Working Conference on Reverse Engineering (WCRE)* (pp. 262–271), October 2013.
8. Sun, Y., & He, D. (2012). Model checking for the defense against cross-site scripting attacks. In *2012 International Conference on Computer Science Service System (CSSS)* (pp. 2161–2164), August 2012.

9. Alexenko, T., Jenne, M., Roy, S., & Zeng, W. (2010). Site request forgery: Attack and defense. In *2010 7th IEEE Consumer Communications and Networking Conference (CCNC)* (pp. 1–2), January 2010.
10. Czeskis, A., Moshchuk, A., Kohno, T., & Wang, H. J. (2013). Server support for browser based csrf protection. In *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW 13 (pp. 273–284). Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee. [Online]. Available: https://dl.acm.org/citation.cfm?id=2488388.2488413.
11. Barth, A., Jackson, C., & Mitchell, J. C. (2008). Defenses for cross-site request forgery. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, ser. CCS 08 (pp. 75–88). New York, NY, USA: ACM. [Online]. Available: https://doi.acm.org/10.1145/1455770.1455782.
12. Dinesh Chandra Misra, P. A., & Srivastava, A. K. (2012). Web application using broken authentication and session management, cross site request forgery and scripting attacks and sql injection. In *2012 International Conference on VSRD International Journal of Computer Science and Information Technology* (Vol. 2, No. 4, pp. 356–364), January 2010.
13. Huluka, D., & Popov, O. (2012). Cause analysis of session management and broken authentication vulnerabilities. In *2012 World Congress on Internet Security (WorldCIS)* (pp. 82–86), June 2012.
14. Eshete, B., Villafiorita, A., & Weldemariam, K. (2011). Early detection of security misconfiguration vulnerabilities in web applications. In *2011 Sixth International Conference on Availability, Reliability and Security (ARES)* (pp. 169–174).
15. https://web.nvd.nist.gov/.