

## Web Vulnerability Through Cross Site Scripting (XSS) Detection with OWASP Security Shepherd

R M Wibowo<sup>1</sup>, Sulaksono<sup>2</sup>

<sup>1</sup> Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia

<sup>2</sup> Faculty of Environmental Design, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia

Email: rjonowibowo@stu.kau.edu.sa<sup>1</sup>, ahadiwisastrosulaksono@stu.kau.edu.sa<sup>2</sup>

Submitted: 31 January 2021, revised: 23 February 2021, accepted: 25 February 2021

**Abstrak.** Aplikasi web dibutuhkan sebagai solusi pemanfaatan teknologi internet yang mudah digunakan dan mampu menampilkan informasi yang kaya konten, hemat biaya, dan mudah diakses. Pada kuartal kedua tahun 2020, Wearesocial merilis informasi terkait pengguna internet di dunia sekitar 4,54 miliar dengan penetrasi 59%. Orang menjadi sangat tergantung pada internet dan juga teknologi. Kondisi ini juga dipicu akibat pandemi Covid-19. Satu hal yang menjadi masalah pada keamanan aplikasi website adalah serangan internet pada platform website dan adanya kerentanan. Salah satu jenis serangan atau ancaman keamanan yang sering muncul dan sering terjadi adalah Cross Site Scripting (XSS). XSS adalah salah satu dari daftar 10 *Open Web Application Security Projects* (OWASP) teratas. Ada beberapa alternatif yang bisa digunakan untuk mencegah serangan *cyber* seperti OWASP Security Shepherd. OWASP Shepherd memungkinkan pengguna untuk mengembangkan keterampilan pengujian penetrasi manual mereka. Dalam penelitian ini terdapat beberapa contoh kasus atau tantangan yang dapat digunakan sebagai simulasi peran OWASP Security Shepherd untuk mendeteksi XSS ini. Tujuan dari penelitian ini untuk melakukan tinjauan singkat dan jelas tentang teknologi pada OWASP Security Shepherd. Teknologi ini dipilih sebagai alternatif yang tepat dan murah bagi pengguna untuk menangkal serangan XSS.

**Kata kunci:** *Cross Site Scripting (XSS)*, keamanan aplikasi web, *OWASP Security Shepherd*

**Abstract.** Web applications are needed as a solution to the use of internet technology that is easy to use and capable of displaying information that is rich in content, cost-effective, and accessible. In the second quarter of 2020, Wearesocial released information that around 4.54 billion people in the world used the internet with 59% penetration. People become very dependent on the internet and technology. This condition was also triggered by the Covid-19 pandemic. One thing that becomes an issue on website application security is internet attacks on website platforms and the vulnerability. One type of attack or security threat that often arises and occurs is Cross-Site Scripting (XSS). XSS is one of the Top 10 Open Web Application Security Projects (OWASP) list. Several alternatives can be used to prevent cyber-attack like OWASP Security Shepherd. The OWASP Security Shepherd project allows users to learn or develop their manual penetration testing skills. In this research, there are several case examples of challenges that we can use as a simulation of the role of OWASP Security Shepherd to detect this XSS. The purpose of this paper is to conduct a brief and clear review

of technology on OWASP Security Shepherd. This technology was chosen as an appropriate and inexpensive alternative for users to ward XSS attacks.

**Keywords:** *Cross Site Scripting (XSS), web application security, OWASP Security Shepherd*

## 1. Introduction

In the second quarter of 2020 in April, an institution, focusing on research related to digital data, technology, and the internet, released that 4.54 billion people in the world used the internet with 59% penetration [1]. According to Wearesocial Data (Digital Around the world, April 2020), the Internet has made rapid progress in 2020 and people in the world have a high dependency on Internet technology with increasing Web application services. People become very dependent on the internet and technology. This condition is also triggered by the Covid-19 pandemic or also known as Corona Virus.

This makes the technology that emerges and is created in network applications becomes sensitive and vulnerable to cyber-attacks that appear on the internet. In 2017 Open Web Application Security Project (OWASP) published the Top 10 list of vulnerabilities that often appear on the internet [2][3][5]. Cross Site Scripting, also known as XSS, is the 7th most frequent and most critical web application security risk, which means detection of attacks of XSS must still be considered and is important in terms of cybersecurity.

XSS attacks insert injection scripts that are dangerous and are most encountered in this type of attack, which occurs when an attacker in cyberspace injects destroyers of similar content such as JavaScript into a webpage to be executed by browser users [3] [5].

This type of script is quite dangerous and unwittingly the victim will execute and attempt a type of piracy of user sessions, the appearance of misinformation, and damage to web pages. Other cases are phishing attacks, accessing the user's business data, and controlling the user's browser [4] [5]. What is even scarier is that XSS is combined with other types of attacks or other vulnerabilities such as Remote Code Execution (RCE), and Cross Site Request Forgery (CSRF) which threatens internet users [1-5].

Several alternatives can be done to prevent cyber-attack. In this paper, the author chooses OWASP Security Shepherd. OWASP Security Shepherd can be used to prevent the effects of XSS attacks. The OWASP Security Shepherd project allows users to learn or develop their manual penetration testing skills. In this research, there are several case examples of challenges that can be used as a simulation of the role of OWASP Security Shepherd to detect XSS. The purpose of this detection is to find out the type of XSS that can be learned as a step to create a secure Website. The purpose of this paper is to conduct a brief and clear review of technology on OWASP Security Shepherd. Hopefully, this tool can be useful and help overcome the existence of XSS.

## 2. Literature Review

### 2.1. OWASP

The Open Web Application Security Project (OWASP) is a non-profit organization that helps organizations create, buy, and manage trusted software applications. OWASP offers convenience and ease of access to free and open source for [2][8][10]:

- Application of security software and standards
- Full books on security testing, security-code creation, and security code analysis
- The reports and the videos
- Optional security tests and repositories
- Local branches all over the world
- Cutting the bottom of science
- Extensive conferences around the world, etc

OWASP aims to inform developers, designers, architects, and business owners about the risks associated with the most growing security vulnerabilities in web applications [2]. OWASP has published a famous Top Ten List that describes the most dangerous security vulnerabilities in web applications and makes suggestions about how to deal with those flaws. Table 1 shows OWASP Top 10 – 2017 (Ten Most Important Security Threats for Web Application).

**Table 1.** Types of Security Threats

No.	Types of Security Threats
1	Injection
2	Broken Authentication
3	Sensitive Data Exposure
4	XML External Entities (XXE)
5	Broken Access Control
6	Security Misconfiguration
7	Cross-Site Scripting (XSS)
8	Insecure Deserialization
9	Using Component with Known Vulnerabilities
10	Insufficient Logging & Monitoring

### 2.2. OWASP Security Shepherd

The OWASP Security Shepherd project is a website and mobile security training program for applications. Security Shepherd has been developed to cultivate and improve security knowledge across a wide variety of demographic skills. The goal of this project is to sharpen the penetration testing skills of AppSec novices or seasoned engineers to the level of security experts [2]. The reasons for using Shepherd Security is that this tool provides Broad Topic Scope, Gentle Learning Curve, Real-World Examples, Scalability, Highly Scalable, User Management, and Robust Service.

The OWASP Security Shepherd project allows users to learn or develop their manual penetration testing skills. This is done by introducing safety risk principles to users in lessons followed by challenges [2] [10]. A lesson from OWASP provides a layman with the help of a user on a particular security problem and allows them to manipulate the textbook version of the issue [2]. The challenges include weak security protection for vulnerabilities that have left scope for users to exploit [2] [8].

By using the OWASP Top Ten as a challenging testbed, the security vulnerabilities can be explored and their effect on the system can be understood [2]. The by-product of this competitive game is the learned ability to harden the player's world from OWASP's top ten security threats. The modules have been developed to not only challenge security novices, but also security professionals [2][6].

The safety risks of Shepherd are borne by hardened actual vulnerabilities that cannot be exploited to compromise the application or its environment. Shepherd does not model security threats in such a way that all and all attack vectors can work to ensure real-world response [2].

### 2.3. Cross Site Scripting (XSS)

Cross Site Scripting (XSS) attacks [3-6] are the most common vulnerabilities found in web applications. The injection occurs at the client-side by embedding a file. XSS attacks can cause sensitive data to be tampered with and exposed. The most popular method is accessing sessions or stealing cookies to collect confidential information. We can see XSS impact and weakness from Figure 1 and Figure 2 [2][3][5].

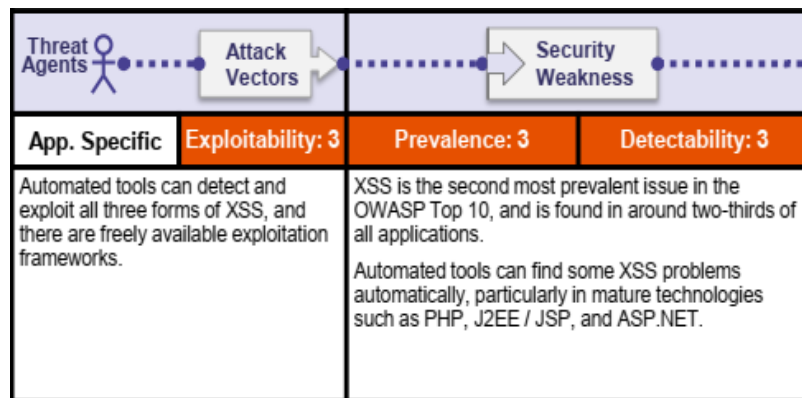


Figure 1. XSS Attack Vector to Security Weakness

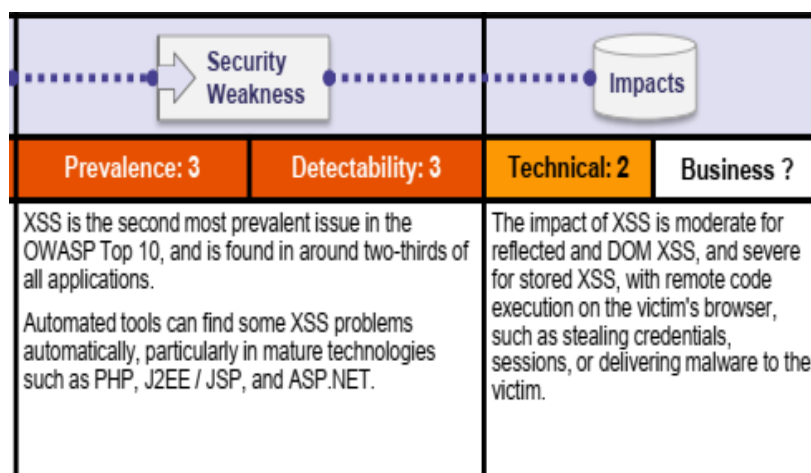


Figure 2. XSS Attack Vector to Impact on XSS

Cross-Site Scripting (XSS) is a known web-based attack. It happens when malicious web code is sent or executed, usually in script form, from a victim's computer browser using their web applications. With this execution, personal information can be filtered, or the cookies can be stolen from the user [4] to hijack the identity in a fraudulent session. This also gives attackers the possibility of stealing confidential data or even taking control of other computers. XSS poses 40% attack attempts, SQL injection (SQLi) 24%, and attack called cross-section 7%, the inclusion of local files (LFI) 4%, and in the last place, the DDoS (Denial of Distributed Services). According to Imperva [19-21] results, XSS attacks reflect the highest number of web application vulnerabilities in 2017. Their number has doubled in comparison to 2016. Also, according to Imperva's projections, the most frequent offensives would occur in 2018.

The 25 most dangerous software errors [13] are classified into three groups according to the Specific Weakness Enumeration (CWE / SANS):

1. Unsafe interaction of components (6 errors)
2. Risky resource management (8 bugs)
3. Porous protections (11 bugs).

XSS also happens when [14]:

1. Untrusted data enters a web application.
2. The same web application continuously produces untrusted data.
3. A victim visits the website that has been created by a web browser and has been infected with a malicious XSS script containing untrusted data.

4. An XSS-type script sent by a server is executed on a web page, i.e. in the same sense as the domain of a web server. This attack does not exploit the vulnerabilities of a single browser, it affects web servers where web applications are hosted.

According to Elhakeem and Barry [15], the impact of cross-site scripts can be described as follows:

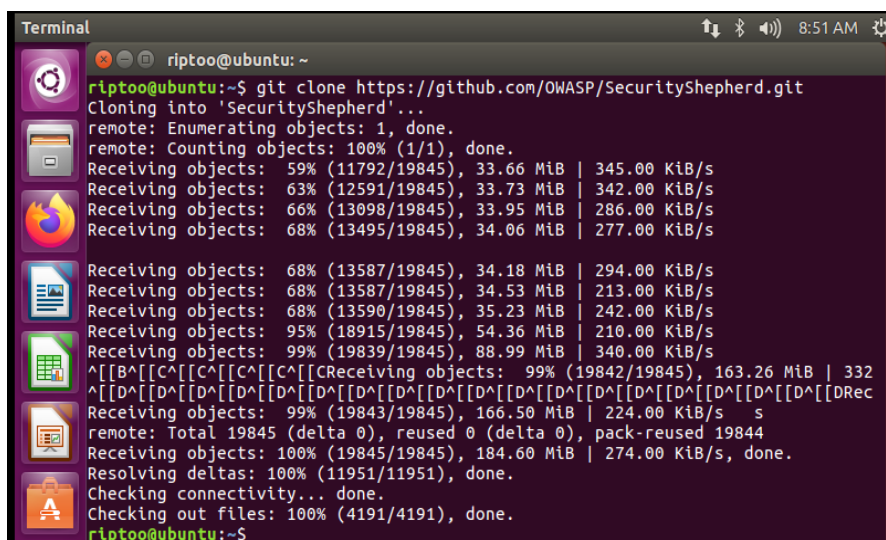
1. The content of a web application may be changed by inserting scripts that show false advertising, affect the credibility of commercial websites or deceive the consumer
2. The theft of session cookies could be carried out in open sessions to collect information while such sessions remain online
3. Possibility of hijacking the identity of legitimate users, stealing sensitive personal information.

### 3. Experimental Tools

OWASP Security Shepherd as we know is an application security training platform. This platform is designed to train and increase the security awareness of web technology users with various levels of knowledge. We use this platform because the author feels it can be used to sharpen penetration capabilities. We ran this experiment on the Ubuntu operating system. We do this because we believe OWASP Security Shepherd is a license-free application and runs well on Ubuntu Linux. We use Cross-Site Scripting (XSS) because it is known as one of the web-based attacks. XSS attacks reflect the highest number of web application vulnerabilities in 2017 [21].

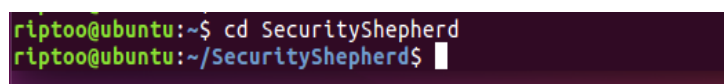
### 4. Simulation with OWASP Security Shepherd

OWASP where we can do a simulation directly on OWAS Security Shepherd. The first step to do the simulation is to install Shepherd on your computer. Shepherd can be downloaded at Clone the Github repository (<https://github.com/OWASP/SecurityShepherd.git>). A successful installation can be seen in Figure 3 and Figure 4.



```
Terminal
riptoo@ubuntu: ~
riptoo@ubuntu:~$ git clone https://github.com/OWASP/SecurityShepherd.git
Cloning into 'SecurityShepherd'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
Receiving objects: 59% (11792/19845), 33.66 MiB | 345.00 KiB/s
Receiving objects: 63% (12591/19845), 33.73 MiB | 342.00 KiB/s
Receiving objects: 66% (13098/19845), 33.95 MiB | 286.00 KiB/s
Receiving objects: 68% (13495/19845), 34.06 MiB | 277.00 KiB/s
Receiving objects: 68% (13587/19845), 34.18 MiB | 294.00 KiB/s
Receiving objects: 68% (13587/19845), 34.53 MiB | 213.00 KiB/s
Receiving objects: 68% (13590/19845), 35.23 MiB | 242.00 KiB/s
Receiving objects: 95% (18915/19845), 54.36 MiB | 210.00 KiB/s
Receiving objects: 99% (19839/19845), 88.99 MiB | 340.00 KiB/s
Receiving objects: 99% (19842/19845), 163.26 MiB | 332
Receiving objects: 99% (19843/19845), 166.50 MiB | 224.00 KiB/s
remote: Total 19845 (delta 0), reused 0 (delta 0), pack-reused 19844
Receiving objects: 100% (19845/19845), 184.60 MiB | 274.00 KiB/s, done.
Resolving deltas: 100% (11951/11951), done.
Checking connectivity... done.
Checking out files: 100% (4191/4191), done.
riptoo@ubuntu:~$
```

Figure 3. Shepherd installation from Github



```
riptoo@ubuntu:~$ cd SecurityShepherd
riptoo@ubuntu:~/.SecurityShepherd$
```

Figure 4. Create a directory and call Shepherd

In Figure 18 it can be seen the function 'cd' on the Security Shepherd that creates a new directory and calls Security Shepherd. Whereas in Figure 19 Docker is formed and processed. If the process is successful there will be information regarding Pull Complete. If the process is successful, there are steps to determine the package that is patched and the process of viewing the database. The next step is to open the package and set up the patch.

```
mukhammad@mukhammad-Lenovo-ideapad-330S-14IKB:~/SecurityShepherd$ sudo docker-compose up
Creating network "securityshepherd_default" with the default driver
Building db
Step 1/18 : ARG MYSQL_VERSION
Step 2/18 : FROM mysql:${MYSQL_VERSION}
5.7.26: Pulling from library/mysql
0a4690c5d889: Pulling fs layer
98aa2fc6cbeb: Pulling fs layer
0777e6eb0e6f: Pulling fs layer
2464189c041c: Waiting
b45df9dc827d: Waiting
b42b0086160: Waiting
bb93567627c7: Waiting
48acc32b4863: Waiting
6257d2da4815: Waiting
1cd5ed3b2653: Waiting
f4ba7ff24ae9: Waiting
[+]
91d1e510d72b: Pull complete
98ce65c663bc: Pull complete
27d4ac9d012a: Pull complete
Digest: sha256:2c90303e910d7d5323935b6dc4f8ba59cc1ec99cf1b71fd6ca5158835cfff9c
Status: Downloaded newer image for tomcat:8.5.51-jdk8-openjdk
--> 4e7840b49fad
Step 3/34 : ENV RUN_USER tomcat
--> Running in f85abb30c1f6
Removing intermediate container f85abb30c1f6
--> 65ab0f2a77ac
Step 4/34 : RUN apt-get -qq update && apt-get install -y patch
--> Running in ef9bd9794e53
Reading package lists...
Building dependency tree...
Reading state information...
Suggested packages:
  ed diffutils-doc
The following NEW packages will be installed:
  patch
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 126 kB of archives.
After this operation, 249 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main amd64 patch amd64 2.7.6-3+deb10u1 [126 kB]
^[[2-debconf: delaying package configuration, since apt-utils is not installed
Fetched 126 kB in 0s (272 kB/s)
Selecting previously unselected package patch.
(Reading database ... 12552 files and directories currently installed.)
Preparing to unpack ../patch_2.7.6-3+deb10u1_amd64.deb ...
Unpacking patch (2.7.6-3+deb10u1) ...
Setting up patch (2.7.6-3+deb10u1) ...
```

Figure 5. Docker Compose to call Shepherd

After the steps are completed, Shepherd can be called on the localhost computer and do a simulation. The results can be seen in Figure 6 - 8. In the first process, if the previous step was successful, a login area will appear on localhost and users can start creating a new account by selecting Register. After the registration process is completed, the next step is to return to the login menu to enter the username and password (shown in Figure 6).

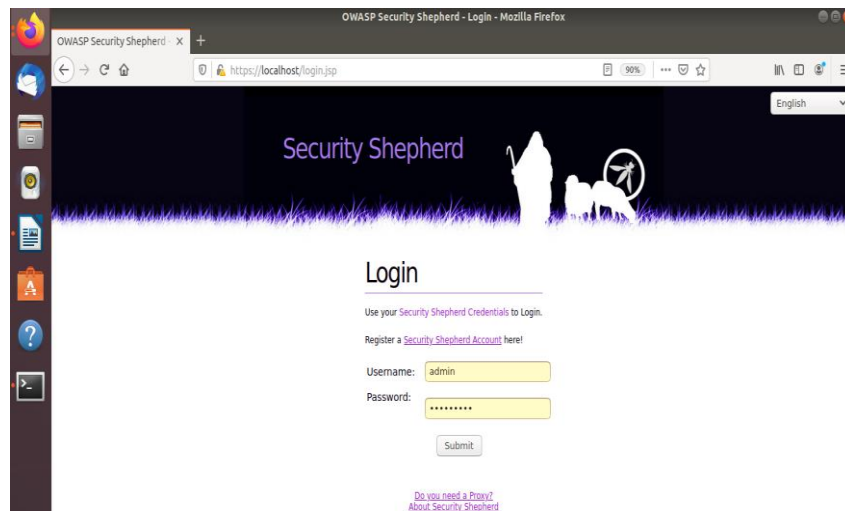


Figure 6. Web Login Shepherd

If the login is successful then we can access the main menu with the main panel there are “ADMIN”, “Scoreboard” and “Get The Next Challenge”. Next step, we have to open the whole Module first on Shepherd. This step is shown in Figure 7.

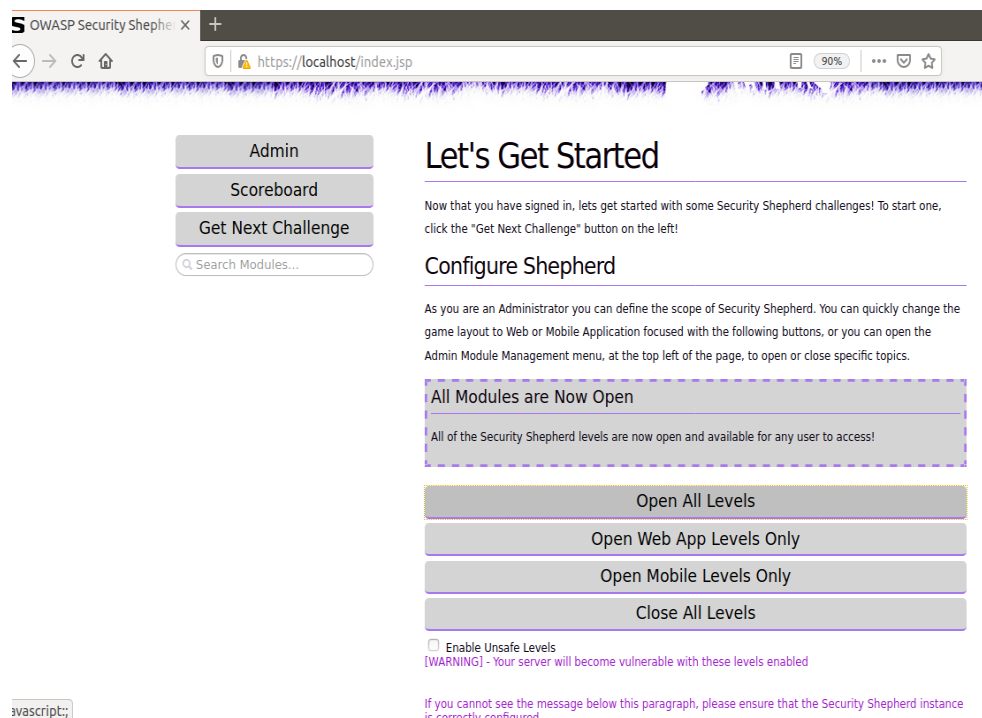


Figure 7. Configuration of Shepherd

If all Modules have been opened, there will be a Challenges option that we can choose to determine web security. This can be seen on the right side of the screen, there are options from Insecure Cryptographic Storage, Insecure Data Storage, Insecure Direct Object References, XSS, Poor Authentication, Poor Data Validation, Reverse Engineering, SQL Injection, and several more options. Then XSS or Cross Site Scripting (Figure 8 and Figure 9) was chosen.

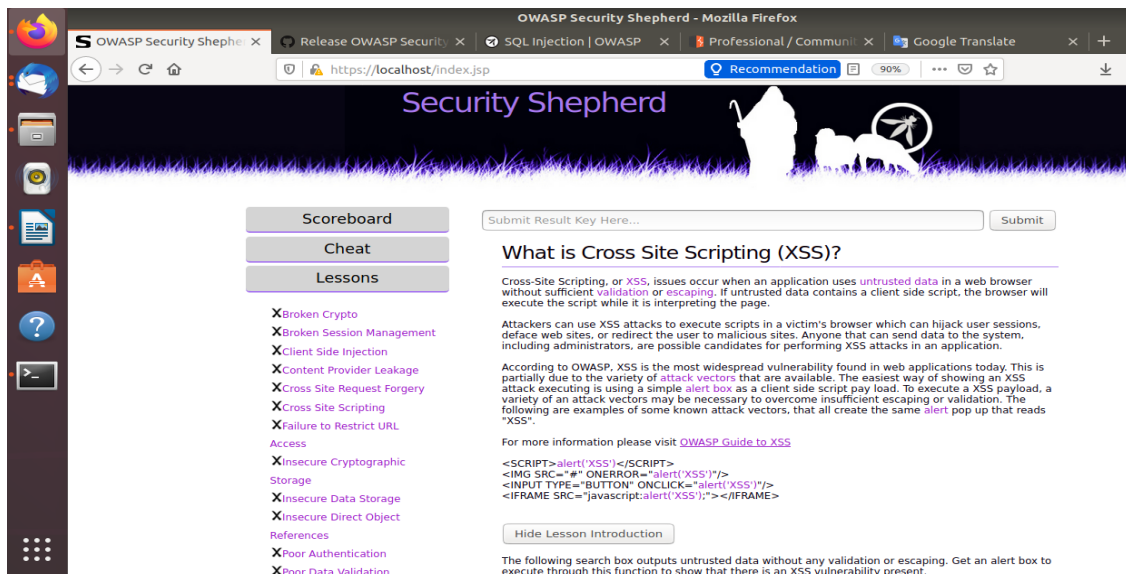


Figure 8. Choose XSS

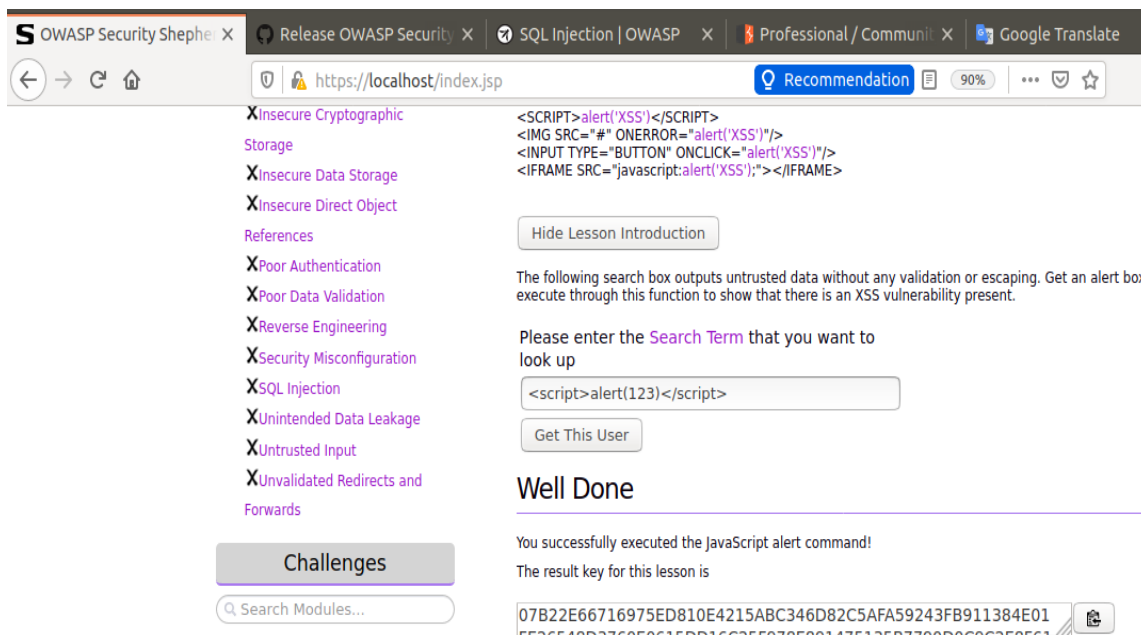


Figure 9. Simulation of Shepherd

Then, users can write this script in search term: `<script> alert (123) </script>`. The results or the numbers dialog box will appear if successful. Then, the Result Key will appear. The users should copy the result key on Submission Area. The results will appear if the process is successful (Figure 10). The example of the Result Key is:  
07B22E66716975ED810E4215ABC346D82C5AFA59243FB911384E01FE26548D3760E0615DD16C25F978E891475135B7790D0C9C2E8F6147437EE78494E51EAB49.



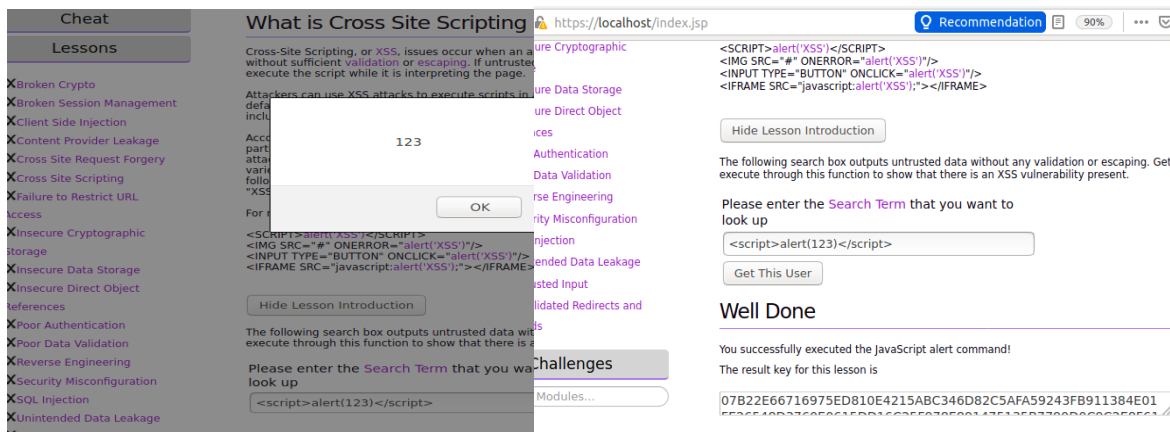


Figure 10. Write a script on the submission area

After completing the process, a message will appear and show Congratulation which means Shepherd has been successful. Testing other challenges of XSS can be done by selecting the options (Figure 11).

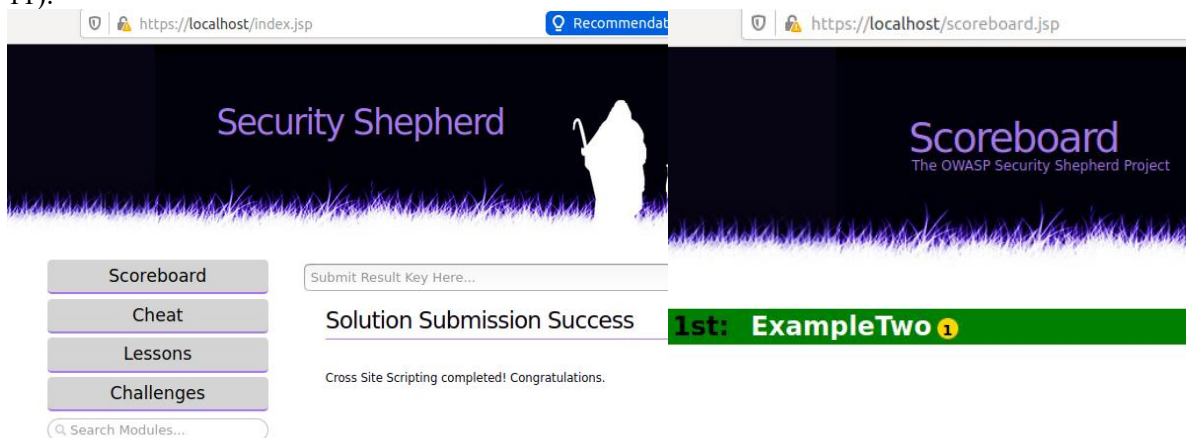


Figure 11. Solution Submission Success

## 5. Conclusion

For Preventing XSS Attack, and some of the method: the input data of the XSS Attacks filter. Encode the data when printing to display. To ensure that you do not run HTML code, always use text-type headers to ensure that your browsers interpret the answers in the way you want.

The research that has the most impact and uses advanced technology and is needed is OWASP so we use this research [6] [8] [19] [27]. Shepherd offers technology and support that is not inferior. If the technology is used, it can help prevent vulnerabilities on the Web in the XSS case. Henceforth we can combine Shepherd or with other technologies such as ZAP to measure other web security. We can also use ISO or other security methods that can check the evaluation process precisely and accurately [28]. Further research is evaluating Shepherd, testing accuracy and comparison, and combining to optimal preventive measures for cyber-attacks.

## References

- [1] Digital Around the World from We Are Social. Available at <https://wearesocial.com/blog/2020/04/digital-around-the-world-in-april-2020> .Accessed Januari 21th, 2020.
- [2] OWASP, OWASP Top 10 – 2017. Available at [https://owasp.org/www-pdf-archive/OWASP\\_Top\\_10-2017\\_%28en%29.pdf](https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf) . Accessed May 1, 2020.
- [3] Gupta S, Gupta B B. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art[J]. International Journal of System Assurance Engineering and Management, 2017, 8(1): 512-530.
- [4] Hadpawat T, Vaya D. Analysis of Prevention of XSS Attacks at Client Side[J]. Analysis, 2017, 173(10).
- [5] Gupta M K, Govil M C, Singh G. Text-mining based predictive model to detect XSS vulnerable files in Web applications[C]//India Conferen (INDICON), 2015 Annual IEEE. IEEE, 2015: 1-6.
- [6] Mohammadi M, Chu B, Lipford H R. Detecting Cross-Site Scripting Vulnerabilities through Automated Unit Testing [C]//Software Quality, Reliability and Security (QRS), 2017 IEEE International Conference on. IEEE, 2017: 364- 373.
- [7] Nithya V, Pandian S L, Malarvizhi C. A survey on detection and prevention of cross-site scripting attack[J]. International Journal of Security and Its Applications, 2015, 9(3): 139-152.
- [8] The OWASP Foundation, “ZAP Proxy.”
- [9] R. Mardisalu, “14 Most Alarming Cyber Security Statistics in 2019,” 2019. [Online]. Available: <https://thebestvpn.com/cyber-securitystatistics2019/>.
- [10] I. Riadi, R. Umar, and W. Sukarno, “Vulnerability of Injection Attacks Against The Application Security of Framework Based Websites Open Web Access Security Project (OWASP),” J. Inform., vol. 12, no. 2, pp. 53–57, 2018
- [11] The OWASP Foundation, “OWASP Risk Rating Methodology,” 2019. [Online]. Available: [https://www.owasp.org/index.php/Threat\\_Risk\\_Modeling](https://www.owasp.org/index.php/Threat_Risk_Modeling).
- [12] D. Saputra and I. Riadi, “Network Forensics Analysis of Man in the Middle Attack Using Live Forensics Network Forensics Analysis of Man in the Middle Attack Using Live Forensics Method,” Int. J. Cyber Security Digit. Forensics, vol. 8, no. 1, pp. 66–73, 2019.
- [13] K. Pandey, “A Bug Tracking Tool for Efficient Penetration Testing,” Int. J. Educ. Manag. Eng., vol. 8, no. 3, pp. 14–20, 2018.
- [14] IT-Digital, El 100% de las aplicaciones web contienen vulnerabilidades, 2018, <http://discoverthenew.ituser.es/security-and-risk-management/2018/04/el-100-de-lasaplicaciones-web-contienen-vulnerabilidades>.
- [15] J. Shanmugam, M. Ponnaivaikko, XSS application worms: New internet infestation and optimized protective measures, in: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007), vol. 3, 2007, pp. 1164–1169.
- [16] J. Bozic, F. Wotawa, Purity: a planning-based security testing tool, in: 2015 IEEE International Conference on Software Quality, Reliability and Security-Companion, 2015, pp. 46–55.
- [17] H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, H.Y. Youm, Preventing abuse of cookies stolen by XSS, in: 2013 Eighth Asia Joint Conference on Information Security, 2013, pp. 85–89.
- [18] L.K. Shar, H.B.K. Tan, Auditing the XSS defence features implemented in web application programs, IET Softw. 6 (4) (2012) 377–390.
- [19] M.K. Gupta, M.C. Govil, G. Singh, Predicting cross-site scripting (XSS) security vulnerabilities in web applications, in: 2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2015, pp. 162–167.

- [20] Verizon, 2017 data breach investigations report, 2018, <http://www.ictsecuritymagazine.com/wp-content/uploads/2017-Data-BreachInvestigations-Report.pdf>.
- [21] Imperva, The state of web application vulnerabilities in 2017, 2018, <https://www.imperva.com/blog/2017/12/the-state-of-web-application-vulnerabilities-in-2017/>.
- [22] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey," *Comput. Networks*, vol. 166, 2020, doi: 10.1016/j.comnet.2019.106960.
- [23] I. Altaf, F. Ul Rashid, J. A. Dar, and M. Rafiq, "Vulnerability assessment and patching management," *Int. Conf. Soft Comput. Tech. Implementations, ICSCIT 2015*, pp. 16–21, 2016, doi: 10.1109/ICSCIT.2015.7489631.
- [24] Chunlei, Wang, Liu Li, and Liu Qiang. "Automatic fuzz testing of web service vulnerability." *Information and Communications Technologies (ICT 2014)*, 2014 International Conference on. IET, 2014.
- [25] P. S. Shinde and S. B. Ardhapurkar, "Cyber security analysis using vulnerability assessment and penetration testing," *IEEE WCTFTR 2016 - Proc. 2016 World Conf. Futur. Trends Res. Innov. Soc. Welf.*, pp. 1–5, 2016, doi: 10.1109/STARTUP.2016.7583912.
- [26] Sunardi, I. Riadi, and P. A. Raharja, "Vulnerability analysis of E-voting application using open web application security project (OWASP) framework," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 11, pp. 135–143, 2019, doi: 10.14569/IJACSA.2019.0101118.
- [27] T. Vieira and C. Serrao, "Web security in the finance sector," *2016 11th Int. Conf. Internet Technol. Secur. Trans. ICITST 2016*, pp. 255–259, 2017, doi: 10.1109/ICITST.2016.7856707.
- [28] R. M. Wibowo, P. A. Erna, and I. Hidayah, "Heuristic evaluation and user testing with ISO 9126 in evaluating of decision support system for recommendation of outstanding marketing officer," *Proc. - 2017 Int. Conf. Sustain. Inf. Eng. Technol. SIET 2017*, vol. 2018-January, no. April 2019, pp. 454–458, 2018, doi: 10.1109/SIET.2017.8304181.