2017 IEEE 3rd International Conference on Big Data Security on Cloud

Web Application Security Tools Analysis

Abdulrahman Alzahrani, Ali Alqazzaz, Huirong Fu, Nabil Almashfi
Department of Computer Science and Engineering
Oakland University
Rochester, MI, USA
{aalzahrani, aalqazzaz, fu, nalmashfi}@oakland.edu

Ye Zhu
Electrical and Computer Engineering Department
Cleveland State University
Cleveland, OH, USA
y.zhu61@csuohio.edu

***Abstract*—Strong security in web applications is critical to the success of your online presence. Security importance has grown massively, especially among web applications. Dealing with web application or website security issues requires deep insight and planning, not only because of the many tools that are available but also because of the industry immaturity. Thus, finding the proper tools requires deep understanding and several steps, including analyzing the development environment, business needs, and the web applications' complexity. In this paper, we demonstrate the architecture of web applications then list and evaluate the widespread security vulnerabilities. Those vulnerabilities are: Insufficient Transport Layer Protection, Information Leakage, Cross-Site Scripting, and SQL Injection. In addition, this paper analyzes the tools that are used to scan for these widespread vulnerabilities in web applications. Finally, it evaluates tools due to security vulnerabilities and gives recommendations to the web applications' users and administrators aiming to educate them.**

***Index Terms*—Web application, web application security, web application vulnerabilities**

I. Introduction

Most businesses depend on the power of websites to interact with their customers and sell products. Some technologies are often developed to take care of the different tasks of a website. Thus web applications have been used increasingly to provide critical security services [1]. Most of the web applications interact with back-end databases so those valuable services are targeted by attacks. As a result, those threats may compromise web applications' security by breaching an enormous amount of information, which could lead to severe economic losses or cause damages.

Real-world websites are complex systems that exchange and integrate data with other systems and store and process data in many different places. In other words, they consist of different numbers of components and technologies, including web browser and client-side tools (such as JavaScript and Flash) and web server and server-side application development tools [2].

Web application security deals with different software bugs in the attempt to get the application to do something bad. In most of the cases, software is developed with the focus on functionality and future needs. Businesses are able to maintain a competitive advantage by providing easier usage. Testing plans are based on making sure the program does what it is supposed to when given good information [3]. The bad news is that there is a number of people out there that are testing your web application as well, but with a different attempt. They check your web server to see if it is vulnerable to unpatched flaws.

This paper studies the underline of web application to understand web application vulnerabilities. A very high percentage of web applications deployed on the Internet are exposed to security problems. According to the Web Application Security Consortium [4], about 49% of the web applications that have been reviewed contain vulnerabilities of high-risk level. In addition, this paper analyzes the tools that have being used to scan for the most widespread vulnerabilities in web applications, which are: Insufficient Transport Layer Protection, Information Leakage, Cross-Site Scripting, and SQL Injection.

The rest of this paper is organized as follows: Section II briefly explains the architecture of the web application and how it works. The most widespread vulnerabilities and the security tools used for each type of vulnerability are discussed in Section III. This is followed by a comparison between all the security tools in Section IV. We also include some policy and recommendations for web developers and administrators in Section V. Finally, the paper concludes in Section VI.

II. Understand How Web Applications Work

A web app is any program that runs in a web browser and can be accessed over the web. This covers lots of territory and includes a multitude of examples and apps. They provide functionality to the user without having to download and install software. They can also be updated, and the updates do not have to be sent to individual computers. Basically, they consist of three layers. First layer is the user-side and consists of a basic browser that displays the content of the web pages. Second layer is the server-side, which generates the dynamic content pages. It contains variety of generation tools, such as Java, active server pages, and PHP. Third layer is the back-end databases, where the data is stored. Figure 1 illustrates the architecture of the web application.

III. Common Web Application Vulnerabilities and Security Tools

Web app insecurity is caused by several factors. First, the web has evolved into complex application platforms on top of which more sophisticated applications have been developed. Unfortunately, security issues have been increased beside that and left as an afterthought. Second, hackers increasingly

978-1-5090-6296-6/17 $31.00 © 2017 IEEE
DOI 10.1109/BigDataSecurity.2017.47

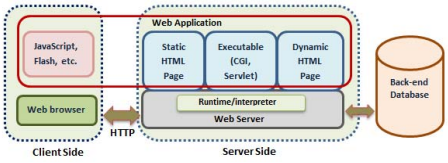

Fig. 1: Web application architecture

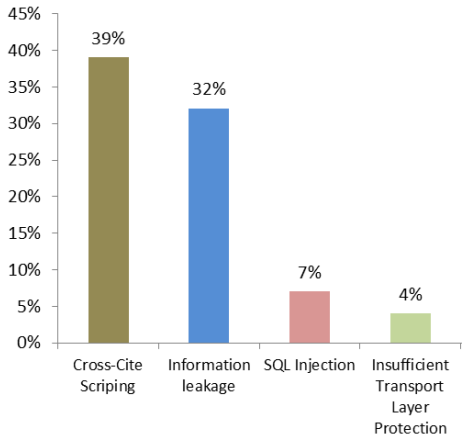

Fig. 2: The percentage of the most widespread vulnerabilities

target web applications since software vendors have become smarter at writing secure code and developing and distributing patches to counter traditional forms of attack, such as buffer overflows. Third, current security technologies including network firewalls and anti-virus software provide comparatively secure protection at the host and network levels, but not at the application level. Generally, the public interfaces to web applications become the target of attacks when network and host-level entry points become relatively secure [5], [6].

Web application vulnerabilities are hard to eliminate because of two reasons. First, most web applications go through rapid development phases with extremely short turnaround time. Second, they are developed in-house by some Management Information Systems engineers (MIS), most of whom have less training and experience in secure software development if compared to engineers at large software firms, such as IBM, Sun, Microsoft, and Facebook. According to The Web Application Security Consortium [7], the most widespread vulnerabilities are Cross-Site Scripting, Information Leakage, SQL Injection, Insufficient Transport Layer Protection, and Fingerprinting. Figure 2 shows the percentage of these top widespread vulnerabilities.

In this section, we will briefly describe each type of vulnerability. Then, we will study some security tools for each type.

A. *Insufficient Transport Layer Protection Vulnerability*

It is a security weakness that is caused when the network traffic is not protected [8]. It aims to compromise a web application and/or steal sensitive data. Typically, websites use

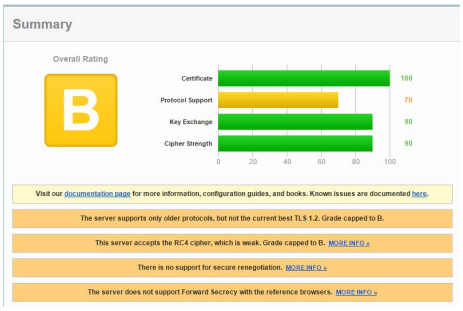

Fig. 3: SSLyze Run

Secure Sockets Layer/Transport Layer Security (SSL/TLS) to provide encryption at the transport layer. However, if the website does not support the SSL/TLS and administrated properly, the website may be vulnerable to traffic interception and modification [9].

When the transport layer is not encrypted, all communications between the website and clients are sent in clear-text, which leaves it open to interception, injection, and redirection (also known as a man-in-the-middle/MITM attack) [4]. Attackers may passively intercept the communication, giving them access to any sensitive data that is being transmitted, such as usernames and passwords. They also actively can inject/remove content from the communication, allowing them to forge and omit information, inject malicious scripts, or cause the client to access untrusted content [4], [10]. However, there are many security tools used for network scanning and vulnerability management, such as SSLyze, Qualys SSL Labs, and OpenSSL.

a) SSLyze: is a stand-alone Python tool that can analyze the SSL configuration of a server by connecting to it. It is designed to be fast and comprehensive, and should help organizations and testers to identify misconfiguration that affect their SSL servers. It provides the advanced users with the opportunity to customize the app via a simple plug-in interface [11]. The following are some features of SSLyze, whereas Figure 3 shows the typical output of a SSLyze run:

- Insecure renegotiation testing.
- Scan for weak ciphers.
- Check for SSLv2, SSLv3 and TLSv1 versions.
- Server certificate information dump and basic validation.
- Support for client certificate authentication.

b) Qualys SSL Labs: is a well-known cloud security provider for network security scanning and vulnerability management. It checks details of HTTPS secured websites. Therefore, users have to keep in mind that they're allowing another company to scan their SSL settings. In other words, SSL Labs is Qualys's research effort to understand SSL/TLS and PKI, as well as to provide tools and documentation to assist with assessment and configuration. Hundreds of thousands of assessments have been performed using the free online assessment tool since 2009. Additionally, there are other projects run by SSL Labs include periodic Internet-wide surveys of

Fig. 4: Qualys SSL Labs result

SSL configuration and a monthly scan of about 170,000 most popular SSL-enabled websites in the world [11], [12]. Qualys' approach consists of four steps:

1) Verify the certificate to insure that it is valid and trusted.
2) Inspect server configuration in three categories: Protocol support, Key exchange support, and Cipher support.
3) Combine the category scores into an overall score (expressed as a number between 0 to 100). A zero in any category will push the overall score to zero. Then, a letter grade is calculated.
4) Apply a series of rules to assign a letter grade (A+, A- B, C, D, E, or F) to some aspects of server configuration that cannot be evaluated via numerical scoring. Most rules will reduce the grade if they encounter an unwanted feature, whereas some rules will increase the grade (up to A+), to reward exceptional configurations [11].

However, some changes were announced in 2014. Here is a list of the most important changes:

- Support for TLS 1.2 is now required to get the A grade. Without it, the grade is capped at B.
- Keys below 2048 bits are considered weak, and the grade capped at B.
- Keys under 1024 bits are considered insecure and the configuration will receive an F.
- This version introduces warnings as part of its rating criteria. In most cases, warnings are about to be issues that do not affect the grade yet, but likely will in the future. Thus, server administrators are advised to correct the issues as soon as possible.
- A new grade A- is introduced for servers with generally good configuration that have one or more warnings.
- A new grade A+ is introduced for servers with exceptional configurations. At the moment, this grade is awarded to servers with good configuration, no warnings, and HTTP Strict Transport Security support with a max-age of at least 6 months.
- MD5 certificate signatures are now considered insecure and the configuration will receive an F.

Figure 4 shows a program-run example of Qualys SSL Labs.

TABLE I: A list of the basic commands in OpenSSL

Command	Function
ca	To create certificate authorities
dgst	To compute hash functions
enc	To encrypt/decrypt using secret key algorithms
genrsa	To generate a pair of public/private key for the RSA algorithm
password	Generation of "hashed passwords"
rand	Generation of pseudo-random bit strings
rsa	RSA data management
rsautl	To encrypt/decrypt or sign/verify signature with RSA
verify	Checking for X509

c) OpenSSL: is an open-source implementation of the SSL and TLS protocols. The core library implements basic cryptographic functions and provides various utility functions. Wrappers allowing the use of the OpenSSL library in a variety of computer languages are available [13]. As of this writing, it's estimated that 66% of all Web servers use OpenSSL. The OpenSSL toolkit is licensed under an Apache-style license [11]. Instead of transmitting the packets in plain text, it allows the information to be encrypted using a strong algorithm. Both the source and destination perform a so-called "handshake" where they agree on a long and complicated key to decrypt the data when it safely arrives at the destination. As a result, even if a malicious hacker gained access to a connecting device and read the packets traveling through it, they would simply look like gibberish. While it is possible to decrypt them without the key, such an attempt can take days, if not weeks, of continuous efforts. Needless to say, it is a strong deterrent [14]. OpenSSL has got many useful commands to use the various cryptography functions of OpenSSL's crypto library from the shell. Table I contains a list of basic commands used in OpenSSL.

B. Information Leakage Vulnerability

It is a weakness in web applications where system data or debugging information is revealed. This information might be exploited and helps an adversary to attack the system. Designers of secure systems need to pay close attention to information leakage as it may lead to damaging attacks [15], [16]. All tools in terms of XSS and SQL-injection vulnerabilities can be used to detect any information disclosure. A notable tool here is Netcraft.

a) Netcraft: is a very popular tool sued by security professionals to gather information related to a target domain. It can give some good foot-printing reconnaissance information. As a security professional, this tool can be used to determine what information is leaking from your organization as failing to properly secure data collected from your organization's site may leave you open to possible attacks. Netcraft runs your own domain and see what you need to fix and what you need to defend against. Netcraft provides web server and web hosting market-share analysis, including web server and operating system detection. It can also tell how long the servers have been up, what their uptime is, the last time they were

reboot and so forth. Netcraft is very easy to use. Users can visit netcraft.com and put the desired domain information to get all details of their targets [17], [18].

C. Cross-Site Scripting Vulnerability

(XSS) vulnerabilities are a type of injection issue, which means that malicious scripts are injected by hackers into trusted web pages. These vulnerabilities are widespread and occur when attackers use web applications to send malicious code. Mostly, they inject browsers into a different end user. Flaws in web application development allow these attacks to be executed especially when there is an exchange of data between users and servers [19], [20], [21]. XSS also exploits vulnerabilities in dynamically generated web pages. This form can be defined into two sub-types: internal and external:

Internal XSS: is done in the same website where the code would be injected. In other words, hackers will post the malicious code as a comment in a website that has been shown to them, like Facebook or Twitter. In this type, the code will be run and executed in the computer of anyone who has opened that injected page or comment.

External XSS: injection can be done from outside using the browser. In this type, the malicious code is not saved in the website, which means that the hacker is the only one who can see the malicious code.

Cybercriminals use XSS to send their malicious code to the targeted users. The malicious script seems trustful because users' browsers can't determine whether the requested page has been injected and untrusted or not. As a result, it will be executed and run the script code. Malicious code can obtain cookies or any other important information that user may send to the other end [22]. These scripts can change the contents of the page by rewriting the HTML. Steps that hackers follow in this attack are:

1) Find vulnerable websites and issue the needed cookies.
2) Build their malicious code and verify that it will be executed as they expect.
3) Build URLs. They also could embed the code in web pages and emails.
4) Try to trick users to execute that malicious code, which will end up with hijacking the account or gathering sensitive data.

Threats in XSS could be Client Side Code Injection, Cookie Stealing, XSS Tunneling, DoS, Malware Spreading, etc. To prevent CSS, we can implement more validations that will take care of those web pages' inputs. Also, we can patch those vulnerable programs. There are many tools used to test dynamic web pages, such as XSS Server, XSSer, and OWASP Xenotix. However, these tools could be used by attackers to find vulnerable web pages as well [21], [23].

a) XSS Server: This is server-side loot, which is used to exploit XSS vulnerabilities. The purpose of this tool is to gather sensitive data from users when they execute an XSS code or when they access an injected web page that has embedded XSS code. That victim's sensitive data could be cookies, victim's IP address, web page contents, username and passwords, etc.

b) Cross Site Scripter "XSSer": is an automatic framework that is used to exploit and detect CSS vulnerabilities in web applications. This tool is an open source penetration testing tool that contains many techniques that help to bypass certain filters. It also has some various options of code injection. XSSer requires Python and runs on many platforms [24].

c) OWASP Xenotix XSS: is an advanced framework that is used for XSS vulnerability detection and exploitation. "This tool supports both manual mode and automated time sharing based test modes. It includes a XSS encoder, a victim side keystroke logger, and an Executable Drive-by downloader". Here are some of this tool's Framework features: Xenotix API, Python Scripting Engine with Triple, Payloads, Zero False Positive, XSS Key logger, Browser Engine Rendering and XSS, XSS Executable Drive-by downloader, and Automatic XSS Testing and Encoder [25]. Unlike other tools, this tool is described as one of the most powerful tools in case of XSS detection and exploitation and can be used to detect most of XSS threats by performing penetration tests on the web pages against XSS vulnerabilities [26]. The following are some of its features:

- Payloads: including HTML5 compactable XSS injection payloads, this tool has more than 380 payloads.
- XSS Key logger is one way of grabbing users' information that is typed on web pages. The action of recording the keys struck on a keyboard is called Keylogging, which can be used to spy on someone and get gain access.
- XSS Encoder: This feature allows encoding in different forms, such as URL Encoding, HTML, Base64 and HEX Encoding to bypass web application firewalls and other filters. Moreover, there are many available websites allow users to generate an XSS code to check their input validation filters against XSS like XSS String Encoder.
- XSS Testing: Xenotix has an automatic testing mode, which tests every payload, based on a period of time that users have to specify according to the needed time to load a web page, which depends on their bandwidth.

D. SQL Injection Vulnerability

Mostly, databases contain extremely significant information, such as users' credit cards, passwords, etc. Not only database is paramount, but also Database Management System (DBMS) implementation and file system can be affected on the database centralization and distribution. The most common threat in this section is SQL injection which has an ability to shut down the whole system if underlying file system is modified [2]. Therefore, SQL injection vulnerabilities are widely common, which consist of insert or "inject" SQL query via input data "valid statement" from the user to the application. In other words, SQL Injection involves bypassing malicious SQL queries and statements directly to a database through the input fields in the web application in order to access, manipulate, or delete contents [27]. If an injected query successfully done, its

exploitation can read sensitive data, modify it, or even execute administration operations on the database. Moreover, DBMS implementation can be affected on the database centralization and distribution. Thus, SQL injection may result in some operating system's error [28].

SQL injection allows attackers to spoof IDs and modify current data. As a result, it will cause repudiated issues like changing and rejecting transactions. It can also exposure the data on the system or destroy it [29]. Attackers in these types of threats can play as administrators' roles in the database server. In general, the threats of SQL Injection attacks highly depend on the hackers' skills and imagination of the database design and rules configuration [30], [31]. There are many tools used to detect SQL injection, such as SQL Inject-ME, SQLninja, SQLMap, and Havij. However, some of these tools can be used for penetration purposes.

a) SQL Inject-Me: This is an Exploit-Me tool based on Firefox that is used to discover SQL Injection vulnerabilities. It is a suite of tools and applications that is designed to help with application security testing. Users in this tool submit the HTML form and it substitutes the form values with strings that look representative of the SQL Injection vulnerabilities. In fact, it sends database escape strings via those fields in the form then it looks for the error messages that occur as an output. Sql Inject-Me does not make any threats to the system. It works to find any possible way for such as attack against the system, so there is no password hacking or packet sniffing has been done by this tool. It also does not provide any port scanning or firewall attacks [32].

b) SQLninja: This is a tool that is designed to detect SQL Injection vulnerabilities on web applications that use Microsoft SQL Server as its back-end. The main goal of this tool is to provide a remote access on a vulnerable DB server. It is very powerful tool, so it can work even in a very hostile environment. When SQL injection vulnerability occurs, this tool should be used to help the tester to take over the DB Server manage the process [33]. Here are some features of this tool:

- Provide a fingerprint of the remote SQL Server, such as its version, user performing the queries, and database authentication mode.
- In case of removing the original custom xp_cmdshell is destroyed, it is able to create a new one.
- When there is no TCP/UDP port available, it can do DNS-tunneled pseudo-shell and ICMP-tunneled shell.
- Bruteforce of 'sa' password.
- Scan for TCP/UDP ports on the target SQL Server to the attacking machine, to see if there is a port allowed by the target's firewall and use it.

c) Havij: This is an automated tool that helps penetration testers to detect sql Injection vulnerabilities on web applications. This tool can provide a fingerprint of the back-end database. It is one of the most powerful tools based on SQL Injection vulnerabilities due to its unique methods. It has been rated above 95% based on its success of attack vulnerable targets [34], [35]. Here are some of its abilities:

TABLE II: Tools comparison

Tool	Free Source	Feature
OpenSSL	yes	powerful SSL cryptographic library, File encryption and hashing
Qualys SSL Labs	yes	Free, cloud security provider, online
SSLyze	yes	Fast and full-featured SSL scanner, Support StartTLS handshakes
Netcraft	yes	Free, server-side impact, could be used for other types of vulnerability
XSS Server	no	You can generate an XSS script by using this tool
XSSer	yes	Requires Python
Xenotix XSS	yes	2nd largest XSS Payloads
SQL Inject-Me	yes	No security threats related, can't be installed on Windows 8 and Firefox 19.0.2
SQLninja	no	Successfully tested on Linux, FreeBSD, Mac OS X, iOS
Havij	yes	Friendly GUI, New bypass method for MySQL, automated configuration and heuristic detections

- Obtain data from the DB, dump tables and columns and execute sql statements against the server.
- Ability to recover DBMS usernames and password hashes.

IV. Comparison and Evaluation of Security Tools

In this section, we compare the tools in each type of vulnerabilities. Also, we have given a big picture of the security tools we have described earlier. Table II shows a comparison between the security tools covered in this paper in terms of availability and features

V. Policy and Recommendations

Some vulnerabilities, such as Cross-Site Scripting, and SQL Injection are occurred by design errors, while Information Leakage, Insufficient Transport Layer Protection, and Fingerprinting are often caused by insufficient administration. So, in this section we elucidate the usage of the tools used to prevent these vulnerabilities in terms of installation and practice. Generally, the administrator should be aware of some testing skills that the corresponding system may need during its lifecycle.

Based on the web application environment, tools are installed either in the browser side "user" or server side. However, some tools could be used in both sides for various purposes, such as Netcraft. This tool for instance, can be used in client-side by the attacker to find some weak spots and gather information about the target host. On the other hand, it can be used by the server administrator to check for information leakage in the host system.

A. Cross-Site Scripting

Tools that we listed in XSS are server-side tools. Because attackers from client-side inject malicious codes in this type of vulnerabilities, tools are developed to prevent attacks in the sever-side.

B. SQL Injection

Tools that are used to prevent SQL injection vulnerabilities are server-side based since it is related to the database server.

C. Insufficient Transport Layer Protection

Insufficient Transport Layer Protection tools, such as SSLyze, are used to test the SSL server by the organizations. Qualys SSL Labs is another server-side tool that is used by the administrator to test the server. OpenSSL is a server-side tool. About 66% of all web servers use OpenSSL.

D. Information Leakage

In this type, many tools that we mentioned earlier can be used to prevent information leakage. However, we listed Netcraft here, which is multi-purpose tool as well, due to its usage popularity in preventing vulnerabilities in this type. Additionally, Netcraft is an open-source tool that can be used in the server-side.

VI. Conclusions and Future Work

Security testing a web application or website requires careful thought and planning due to both tool and industry immaturity. As a result, finding the right tool is not an easy task. In order to choose the appropriate tool, it helps to understand a typical website and its security vulnerabilities. This paper discusses, studies, and compares the security tools that are used for the most common web application vulnerabilities. In addition, it shows the advantages and disadvantages of each tool so that the users can determine and chose the most appropriate tool based on their needs.

Acknowledgment

This research work is partially supported by the National Science Foundation under Grants CNS-1338105, CNS-1343141, CNS-1460897, DGE-1623713. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] V. B. Livshits and M. S. Lam, "Finding security vulnerabilities in java applications with static analysis." in *Usenix Security*, vol. 2013, 2005.

[2] M. Curphey and R. Arawo, "Web application security assessment tools," *IEEE Security & Privacy*, vol. 4, no. 4, pp. 32–41, 2006.

[3] N. Dhanjani and J. Clarke, *Network Security Tools: Writing, Hacking, and Modifying Security Tools*. " O'Reilly Media, Inc.", 2005.

[4] T. W. A. S. Consortium. (2014) Insufficient transport layer protection. [Online]. Available: http://projects.webappsec.org/w/page/13246927/FrontPage

[5] M. Curphey, D. Endler, W. Hau, S. Taylor, T. Smith, A. Russell, G. McKenna, R. Parke, K. McLaughlin, N. Tranter *et al.*, "A guide to building secure web applications," *The Open Web Application Security Project*, vol. 1, p. 1, 2002.

[6] J. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla, and A. Murukan, *Improving web application security: threats and countermeasures*. Microsoft Redmond, WA, 2003.

[7] Y.-W. Huang and D. Lee, "Web application securitypast, present, and future," in *Computer Security in the 21st Century*. Springer, 2005, pp. 183–227.

[8] W. Stallings, *Network security essentials: applications and standards*. Pearson Education India, 2007.

[9] T. Dierks, "The transport layer security (tls) protocol version 1.2," 2008.

[10] S. McClure, J. Scambray, G. Kurtz, and Kurtz, *Hacking exposed: network security secrets and solutions*. McGraw-Hill/Osborne New York, 2005.

[11] Q. Inc. (2015) Ssl server rating guide. [Online]. Available: https://www.ssllabs.com/

[12] S. Qualys, "labs. ssl server test," 2014.

[13] O. S. Foundation. (2015) Openssl news. [Online]. Available: https://www.openssl.org/news/

[14] J. Viega, M. Messier, and P. Chandra, *Network Security with OpenSSL: Cryptography for Secure Communications*. " O'Reilly Media, Inc.", 2002.

[15] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 199–212.

[16] M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith, "Axioms for information leakage," in *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*. IEEE, 2016, pp. 77–92.

[17] N. Ltd. (2015) Netcraft tool. [Online]. Available: http://toolbar.netcraft.com/

[18] A. Bruns, "Prosumption, produsage," *The International Encyclopedia of Communication Theory and Philosophy*, 2016.

[19] E. V. Nava and D. Lindsay, "Our favorite xss filters/ids and how to attack them," *Black Hat USA*, 2009.

[20] L. K. Shar, H. B. K. Tan, and L. C. Briand, "Mining sql injection and cross site scripting vulnerabilities using hybrid program analysis," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 642–651.

[21] S. Gupta and B. Gupta, "Cross-site scripting (xss) attacks and defense mechanisms: classification and state-of-the-art," *International Journal of System Assurance Engineering and Management*, pp. 1–19, 2015.

[22] D. Catteddu, "Cloud computing: benefits, risks and recommendations for information security," in *Web Application Security*. Springer, 2010, pp. 17–17.

[23] J. Snehi and D. R. Dhir, "Web client and web server approaches to prevent xss attacks," *International Journal of Computers & Technology*, vol. 4, no. 2, 2013.

[24] XSSer. (2016) Cross site scripter. [Online]. Available: http://xsser.03c8.net/

[25] OWASP. (2015) Fingerprint web application framework. [Online]. Available: https://www.owasp.org

[26] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the art: Automated black-box web application vulnerability testing," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 332–345.

[27] M. Kiani, A. Clark, and G. Mohay, "Evaluation of anomaly based character distribution models in the detection of sql injection attacks," in *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*. IEEE, 2008, pp. 47–55.

[28] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of network and computer applications*, vol. 34, no. 1, pp. 1–11, 2011.

[29] H. Shahriar, S. North, and W.-C. Chen, "Client-side detection of sql injection attack," in *Advanced Information Systems Engineering Workshops*. Springer, 2013, pp. 512–517.

[30] A. Shulman and C. Co-founder, "Top ten database security threats," *How to Mitigate the Most Significant Database Vulnerabilities*, 2006.

[31] P. Kumar and R. Pateriya, "A survey on sql injection attacks, detection and prevention techniques," in *Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on*. IEEE, 2012, pp. 1–5.

[32] Imbalife. (2014) How to use sql inject me. [Online]. Available: https://www.imbalife.com/how-to-use-sql-inject-me

[33] A. Singh, *Instant Kali Linux*. Packt Publishing Ltd, 2013.

[34] S. Pundlik, R. Kumar, B. Gaikwad, A. Aadhale, and V. Waghmare, "Sqlijhs: Sql injection attack handling system," in *International Journal of Engineering Research and Technology*, vol. 2, no. 6 (June-2013). ESRSA Publications, 2013.

[35] B. Nagpal, N. Singh, N. Chauhan, and A. Panesar, "Tool based implementation of sql injection for penetration testing," in *Computing, Communication & Automation (ICCCA), 2015 International Conference on*. IEEE, 2015, pp. 746–749.