

Web Application Security Tools Analysis

Abdulrahman Alzahrani¹, Ali Alqazzaz^{1,2}, Nabil Almashfi¹, Huirong Fu¹, Ye Zhu²

¹Department of Computer Science and Engineering, Oakland University, Rochester, MI, United States

²Electrical and Computer Engineering Department, Cleveland State University, Cleveland, OH, United States

Correspondence: Abdulrahman Alzahrani, Department of Computer Science and Engineering, Oakland University, Rochester, MI, United States.

Received: September 21, 2017

Accepted: October 23, 2017

Online Published: November 8, 2017

doi:10.11114/smc.v5i2.2663

URL: <https://doi.org/10.11114/smc.v5i2.2663>

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: Fingerprinting, Insufficient Transport Layer Protection, Information Leakage, Cross-Site Scripting, SQL Injection, and HTTP Splitting. 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.

Keywords: web application, web application security, web application vulnerabilities

1. 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 (Livshits & Lam). 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 (Curphey & Arawo, 2005).

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. 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 (Dhanjani & Clarke, 2006).

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 (Ristic, 2014), 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: Fingerprinting, Insufficient Transport Layer Protection, Information Leakage, Cross-Site Scripting, SQL Injection, and HTTP Splitting.

This paper is organized as follows: Section II briefly explains the architecture of the web application and how it works. Next, 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. Aiming to educate, this paper

includes some policies and recommendations for web developers and administrators in Section V. Finally, the paper concludes in Section VI.

2. Understand How Web Applications Work

A web application is any program that runs in a web browser. In other words, web applications are programs that can be accessed over the web. This covers lots of territory and includes a multitude of examples and applications. Web applications 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. Furthermore, web applications require only Internet access and a web browser. Web applications consist of three layers. The first layer is called the user side and consists of a basic browser that displays the content of the web pages. The second layer is called the server side and generates the dynamic content pages. It contains variety of generation tools such as Java, active server pages, and PHP. The third layer is called back-end databases, where the data is stored. Figure 1 shows each of these levels.

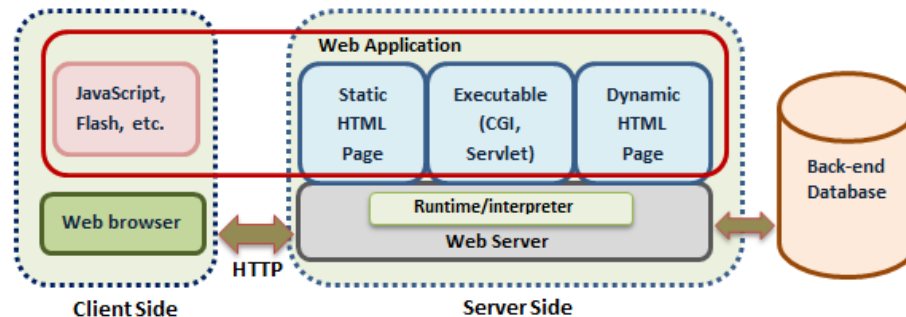


Figure 1. Web application architecture

3. Common Web Application Vulnerabilities and Security Tools

Web application insecurity is caused by several factors as following. First, the web has evolved into complex application platforms on top of which more and more sophisticated applications have been developed. Moreover, web specifications have grown massively to meet rising demands, and browsers and web-development languages fought a “feature war” to win market proportion. Unfortunately, security issues have been grown beside that and left as an afterthought. Second, hackers increasingly target web applications since software vendors become smarter at writing secure code and developing and distributing patches to counter traditional forms of attack such as buffer overflows. Finally, current security technologies such as network firewalls and anti-virus software provide comparatively secure protection at the host and network levels, but not at the application level (Curphey et al., 2002). Generally, the public interfaces to web applications become the target of attacks when network and host-level entry points become relatively secure (Curphey et al., 2002; Meier et al., 2003).

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 corporate Management Information Systems engineers (MIS), most of whom have less training and experience in secure software development compared to engineers at large software firms, such as IBM, Sun, Microsoft, and Facebook. According to The web Application Security Consortium (Huang & Lee, 2005), the most widespread vulnerabilities are Cross-Site Scripting, Information Leakage, SQL Injection, Insufficient Transport Layer Protection, Fingerprinting, and HTTP Response Splitting. Figure 2 shows the percentage of the most widespread vulnerabilities.

As a rule, the causes of Cross-Site Scripting, SQL Injection, and HTTP Response Splitting vulnerabilities are design errors, while Information Leakage, Insufficient Transport Layer Protection, and Fingerprinting are often caused by insufficient administration. This section briefly describes each type of these vulnerabilities. Then, it studies some security tools for each type.

3.1 Fingerprinting Vulnerability

Web server/application fingerprinting is the passive collection of configuration attributes from a remote device during the application layer. So, we can think of it as the TCP/IP fingerprinting except that it is focused on the Application Layer of the OSI model instead of on the Transport Layer. In other words, the goal of web application fingerprinting is to create an accurate profile about the target’s configurations, software, and network topology/architecture by analyzing the HTML headers, cookies, and directory structures. Consequently, the attacker may use this information to develop an attack, which will effectively exploit vulnerability in the software type/version being utilized by the target host. Thus, the web administrators should try to hide the version of software being displayed by the server response header (Kals, Kirda, Kruegel, & Jovanovic, 2006; Stuttard & Pinto, 2011).

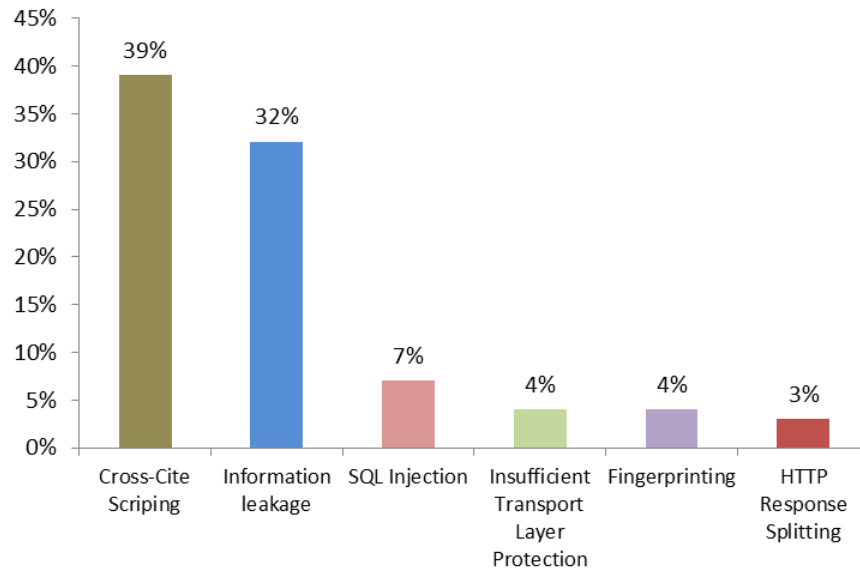


Figure 2. The percentage of the most widespread vulnerabilities

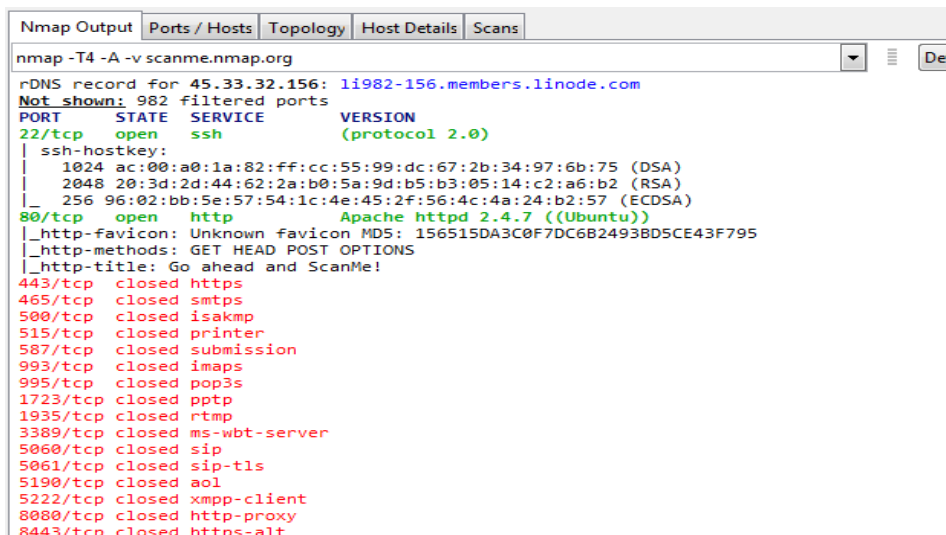
However, there are a number of tools that are being used to accurately fingerprint a web server such as Httprint, Nmap, Netcraft, SHODAN, WhatWeb, BlindElephant, and Wappalyzer. We will take a look at each tool and its features.

3.1.1 Nmap

Nmap (Network Mapper) is a security scanner used to scan a computer network and discover its hosts and services. Thus, it creates a 'map' of the network. After it finds the host, Nmap sends specially crafted packets to the target host then analyzes the responses to accomplish its goal.

This tool provides many features for probing computer networks, including discovering a host and the type of services that it provides. It also has an ability to detect the operating system. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features (Lyon, 2009) such as:

- Host discovery: Identifying hosts on a network.
- Port scanning: Enumerating the open ports of target hosts.
- Version detection: Interrogating network services on remote devices to determine application name and version number.
- OS detection: Determining the operating system and hardware characteristics of network devices.
- Scriptable interaction with the target.
- Auditing the security of a device or firewall by identifying the network connections, which can be made to or through it.
- Network maintenance and asset management.
- Generating traffic to hosts on a network.
- Finding and exploiting vulnerabilities in a network.



```

nmap -T4 -A -v scanme.nmap.org
rDNS record for 45.33.32.156: li982-156.members.linode.com
Not shown: 982 filtered ports
PORT      STATE SERVICE          VERSION
22/tcp    open  ssh              (protocol 2.0)
| ssh-hostkey:
|   1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75 (DSA)
|   2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2 (RSA)
|_  256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57 (ECDSA)
80/tcp    open  http             Apache httpd 2.4.7 ((Ubuntu))
|_ http-favicon: Unknown favicon MD5: 156515DA3C0F7DC6B24938D5CE43F795
|_ http-methods: GET HEAD POST OPTIONS
|_ http-title: Go ahead and ScanMe!
443/tcp   closed https
465/tcp   closed smtps
500/tcp   closed isakmp
515/tcp   closed printer
587/tcp   closed submission
993/tcp   closed imaps
995/tcp   closed pop3s
1723/tcp  closed pptp
1935/tcp  closed rtmp
3389/tcp  closed ms-wbt-server
5060/tcp  closed sip
5061/tcp  closed sip-tls
5190/tcp  closed aol
5222/tcp  closed xmpp-client
8080/tcp  closed http-proxy
8443/tcp  closed https-alt

```

Figure 3. Shows the scan results with different tags. They are “Nmap Output,” “Ports / Hosts,” “Topology,” “Host Details,” and “Scans” (Lyon, 2009).

3.1.2 SHODAN

This is a search engine based tool that allows users to find different types of computers (routers, servers, etc.) connected to the internet using different types of filters. Some have also described it as a search engine of service banners, which are meta-data the server sends back to the client. This can be information about the server software, what options the service supports, a welcome message, or anything else that the client can find out before interacting with the server.

SHODAN currently collects data mostly on web servers (HTTP port 80), but there is also some data from FTP (21), SSH (22) Telnet (23), SNMP (161), and SIP (5060) services. SHODAN is highly useful tool to people interested in security including researchers, developers, and anyone who has a need for the information it provides (Curphey et al., 2002). To simply search for information that would be found in a standard banner (such as the software version in use), one can input the query into the search box, and if needed, select the country on the drop-down map to limit results by country. An example of this can be seen in Figure 4.

In this example, the information for all devices in the US that have “default password” in their banner will be returned to the user. A standard result can contain anything from the IP address, HTTP headers, and targets’ locations.

One can see that this information discloses that this device is a wireless access point running a Boa webserver that is accessible to all Internet addresses and that the default password for this device is ‘connect’. If the administrator of this device has not changed the default password, anyone with this information would be able to log in and remotely manage the device (Stallings, 2007).

Despite the fact that a basic search will be sufficient for most users, the real power of SHODAN lies in the filters available to help refine your search to a specific, targeted subset of the results. There are many filters available, some of which include:

- City: Can be used to refine a search to devices around a certain city. Example: default password city: Rochester Hills.
- Country: Same as city, but to narrow results down by country. Countries are specified via their two letter country codes and can be used to further specify where the city searched for is located (in case there are two cities with the same name).
- Hostname: Searches for computers that include the specified value in their hostname. Example: hostname: twitter.com.
- OS: Specifies which operating system the machine should have. Example: apache os:linux.
- Port: Restricts search to only certain ports. Possible values are 21,22,23 and 80. Example: openssh port: 21.

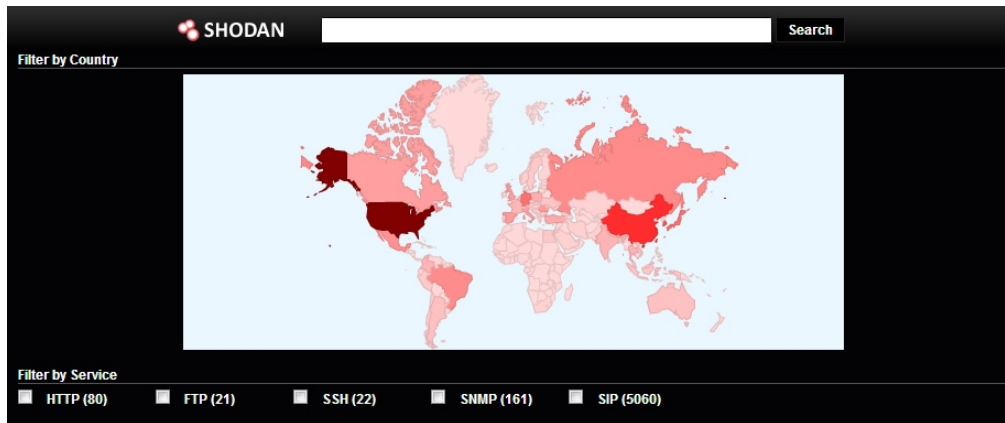


Figure 4. SHODAN interface

3.1.3 BlindElephant

BlindElephant is a python application that fingerprints web applications such as wordpress, joomla, mediawiki, drupal, phpbb, and many others to determine the version of the web application. It is an open source tool and can be freely downloaded. It works using a new technique of fetching static elements of the web app, such as .js, .css, and other core files, then running a checksum to compare sizes to those files from released versions (Thomas, 2010).

It is important to understand that BlindElephant was not designed to check for vulnerabilities to a specific exploit, but shows what version of applications are running on a particular web site. It can help to detect outdated versions, which in many cases, tend to be less secure. On the technical side, for each application that BlindElephant supports, it consumes a number of version directories. All directories and files are processed, and a hash is computed for each file. Then, this hash is stored in a temporary table as well as the path and version of the application it came from (Thomas, 2010). In addition, BlindElephant is capable of guessing the CMS type (Drupal, Joomla, Moodle, etc.) of the application. Also, it has the ability to determine the version of it.

3.1.4 Httpprint

Httpprint is a web application fingerprinting tool. It is used to identify the web server type/application on which the site is hosted, despite the fact that it may have been obfuscated by changing the server banner strings. Httpprint suggests the type of the web server and gives confidence ratings, a percentage of how much sure it is about the type of the web server. In addition to that, it has the ability to detect web enabled devices which do not have a server banner string, such as wireless access points, routers, switches, cable modems, etc.

Httpprint is a complete re-write, a multithreaded scanner to process multiple servers in parallel. This is a new feature that has been added to the tool. Another feature is the ability to gather SSL certificate information, which helps you identify expired SSL certificates, ciphers used, certificate issuer, and other such SSL related details. It also has the Ability to import web servers from Nmap network scans. Httpprint is easy to use as there is no key/registration needed. However, a target website/IP must be provided. It is not an open source but it is available for free use and it is available on Linux, Mac OS X, FreeBSD (command line only) and Win32 (command line and GUI) (Shah, 2004).

3.1.5 WhatWeb

WhatWeb is a tool for fingerprinting. As the designers of the tool say that the goal of this tool is to answer the question 'what is that website?' This tool is used to fingerprint a web application, web server and other technologies of a web page including content management systems (CMS), blogging platforms, statistic/analytics packages, JavaScript libraries, web servers, and embedded devices. It examines the web server, HTTP Headers and the HTML source of a web page to determine technologies in use. Sometimes a single webpage visit contains enough information to identify a website but when it does not, WhatWeb can interrogate the website further.

WhatWeb has over 1500 plug-ins where each plug-in is used to recognize something different by running a few tests. For example, most WordPress websites can be identified by the meta HTML tag, e.g. < meta name="generator" content="WordPress 2.6.5" > but a few of them remove this identifying tag but WhatWeb still runs other tests to find out. WhatWeb is available for free use and it is available on Linux, Mac OS X, FreeBSD and Win32 (Shrivastava, 2011).

3.2 Insufficient Transport Layer Protection Vulnerability

Insufficient transport layer protection is a security weakness caused when the network traffic is not protected (Stallings,

2007). It aims to compromise a web application and/or steal sensitive information. In general, websites typically use 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 (Ristic, 2014).

When the transport layer is not encrypted, all communications between the website and clients are sent in cleartext, which leaves it open to interception, injection, and redirection (also known as a man-in-the-middle/MITM attack). Attackers may passively intercept the communication, giving them access to any sensitive data that is being transmitted such as, usernames and passwords. Attackers may also actively inject/remove content from the communication, allowing the attackers to forge and omit information, inject malicious scripting, or cause the client to access remote untrusted content. Moreover, they may redirect the communication in such a way that the website and client are no longer communicating with each other, but instead are unknowingly communicating with the attacker in the context of the other trusted party (Ristic, 2014; McClure, Scambray, Kurtz, & Kurtz 2009).

However, there are different tools used for network security scans and vulnerability management. This paper focuses on three tools. These tools are:

3.2.1 SSLyze

This 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 identify misconfiguration affecting their SSL servers. So, it provides the advanced users with the opportunity to customize the application via a simple plug-in interface (Ristic, 2009).

SSLyze Features:

- Insecure renegotiation testing.
- Scan for weak strength ciphers.
- Check for SSLv2, SSLv3 and TLSv1 versions.
- Server certificate information dump and basic validation.
- Session resumption capabilities and actual resumption rate measurement.
- Support for client certificate authentication.
- Simultaneous scanning of multiple servers, versions and Ciphers.

Figure 5 shows the typical output of a SSLyze run.

3.2.2 Qualys SSL Labs

Qualys is a known cloud security provider for network security scans and vulnerability management. It is a free tool to check details of HTTPS secured websites. Therefore, users have to keep in mind that you're allowing another company to scan your 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. According to them, hundreds of thousands of assessments have been performed using the free online assessment tool since 2009. In addition, there are other projects run by SSL Labs include periodic Internet-wide surveys of SSL configuration and a monthly scan of about 170,000 most popular SSL-enabled websites in the world (Ristic, 2009; Qualys, 2014).

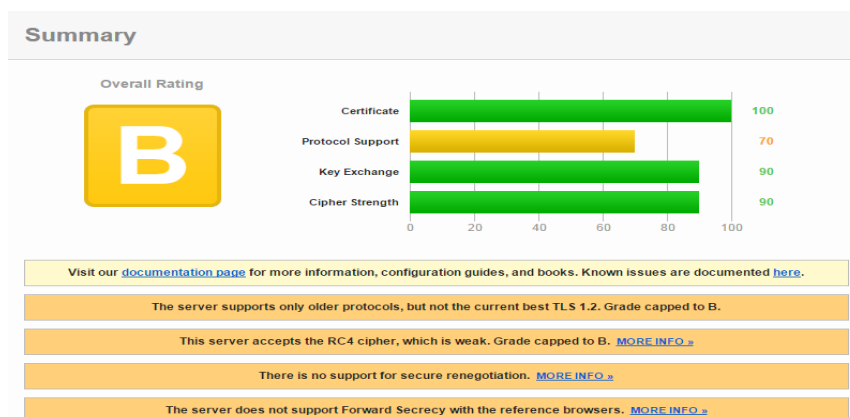


Figure 5. SSLyze run

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.

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 (e.g., 1024) are now considered weak, and the grade capped at B.
- Keys under 1024 bits are now 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 issues that do not yet affect the grade but likely will in the future. 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 maxage of at least 6 months.
- MD5 certificate signatures are now considered insecure and the configuration will receive an F. Figure 6 shows a program-run example of Qualys SSL Labs.

```

$ ./sslyze.py --regular example.org:443
SCAN RESULTS FOR EXAMPLE.ORG:443
-----
* Compression :
  DEFLATE Compression:           Disabled

* Session Renegotiation :
  Client-initiated Renegotiations: Rejected
  Secure Renegotiation:          Supported

* TLSV1_2 Cipher Suites :
  Rejected Cipher Suite(s): Hidden

  Preferred Cipher Suite:
  DHE-RSA-AES256-GCM-SHA384      256 bits      HTTP 200 OK

  Accepted Cipher Suite(s):
  DHE-RSA-CAMELLIA256-SHA        256 bits      HTTP 200 OK
  DHE-RSA-AES256-SHA256         256 bits      HTTP 200 OK
  DHE-RSA-AES256-SHA            256 bits      HTTP 200 OK
  DHE-RSA-AES256-GCM-SHA384     256 bits      HTTP 200 OK
  CAMELLIA256-SHA               256 bits      HTTP 200 OK
  AES256-SHA                    256 bits      HTTP 200 OK
  DHE-RSA-CAMELLIA128-SHA       128 bits      HTTP 200 OK
  DHE-RSA-AES128-SHA256         128 bits      HTTP 200 OK
  DHE-RSA-AES128-SHA           128 bits      HTTP 200 OK
  DHE-RSA-AES128-GCM-SHA256     128 bits      HTTP 200 OK
  CAMELLIA128-SHA              128 bits      HTTP 200 OK
  AES128-SHA                   128 bits      HTTP 200 OK

[...]
```

Figure 6. Qualys SSL Labs result

3.2.3 OpenSSL

OpenSSL is an open-source implementation of the SSL and TLS protocols. The core library, written in the C programming language, implements basic cryptographic functions and provides various utility functions. According to the OpenSSL Project (1990), wrappers allowing the use of the OpenSSL library in a variety of computer languages are available. 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 (Ristic, 2009). Instead of transmitting the packets in plain, human readable text, it allows the information to be encrypted using a strong algorithm. Both the user computer and the destination server 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 was to gain 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 (Ristic, 2009). In addition, OpenSSL has got many useful commands to use the various cryptography functions of OpenSSL's crypto library from the shell. Table 1 shows a brief list of the basic commands and what they are used for.

Table 1. 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

3.3 Information Leakage Vulnerability

Information Leakage 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 (Ristenpart, Tromer, Shacham, & Savage, 2009; Alvim et al, 2016). All tools in terms of cross site scripting, SQL-injection, and fingerprinting vulnerabilities may be also used to detect any information disclosure. A notable tool here is Netcraft.

3.3.1 Netcraft

It is a very popular tool used 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 (Netcraft, 2009; Bruns, 2016).

3.4 HTTP Splitting Vulnerability

HTTP response splitting is web application vulnerability that is widespread through web applications and it may also lead to different kinds of attacks. HTTP response splitting attack takes place when an attacker sends a single HTTP request with malicious data included in the request header. The server receives the HTTP request and produces a response that is interpreted as two responses by the target. The malicious data usually contains CR (carriage return) and LF (line feed) which must be allowed by the application for this attack to succeed. HTTP response splitting attack may be extended and make it possible to various kinds to be instantiated. Examples of these attacks are Cross-site scripting, Web-cache poisoning, Cross-user attacks and Browser-cache poisoning (SecureTeam, 2005; Auger, 2010).

Additionally, cross user defacement enables the attacker to steal information from the user such as login information by making a fake login screen for the website, thus allowing account compromise. In the browser cache poisoning attack, the attacker forces the browser to cache the web page so forming a long lasting defacement till the browser's cache has been cleaned. In the web cache poisoning, the cache is poisoned which is used by multiple users and therefore making them think the site has been defaced, or that the site they are seeing is the genuine site when it is not. Consequently, it is very critical that developers of any application fully validate user input and ensure that no malicious data is injected. However, there are many tools that can be used to detect if a system is prone to HTTP response splitting attack. One of those tools and the most powerful one is AppScan (SecureTeam, 2005; Auger, 2010).

3.4.1 AppScan

This is an automated dynamic security testing tool. It enhances web application security and improves application security program management throughout the application development lifecycle. It helps to identify security

vulnerabilities early in the development phase when it is least expensive to fix such problems. AppScan runs various tests that probe for known vulnerabilities and produces the results into reports so that it is easy to understand the threats to the application. These reports also include information that can help you to implement the essential protections. AppScan scans for many common vulnerabilities, such as cross site scripting, HTTP response splitting, parameter tampering, hidden field manipulation, backdoors/debug options, buffer overflows and more (Nassar, Hoyos, & Anderson, 2012; AppScan, 2009).

3.5 Cross-Site Scripting Vulnerability

Cross-Site Scripting (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 (Nava & Lindsay, 2009; Shar, Tan, & Briand, 2013; S. Gupta & B. Gupta, 2017). XSS also exploits vulnerabilities in dynamically generated web pages. XSS or CSS can be defined into two sub-types: internal and external:

- Internal XSS. It 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. The injection can be done from outside by 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 cannot 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 (Catteddu, 2010). 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 XSS, 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, which are used to test dynamic web pages, such as XSS Server, XSSer, XSS-Me, OWASP Xenotix and Exploit-Me. However, these tools could be used by attackers to find vulnerable web pages as well (Catteddu, 2010; Snehi & Dhir, 2013). The following section is to introduce these tools.

3.5.1 XSS Server

This is server-side tool, which is used to exploit XSS vulnerabilities. The purpose of this tool is to gather sensitive data from victim 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.

3.5.2 Cross Site Scripter "XSSer"

XSSer is an automatic framework that is used to exploit and detect XSS 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 (Punter, 2014).

3.5.3 OWASP Xenotix XSS

This 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 (Open Web Application Security Project [OWASP], 2017).

Unlike other tools, this tool is easy to use and described as one of the most powerful tools in case of XSS detection and can be used to detect most of XSS threats by performing penetration tests on the web pages against XSS vulnerabilities (Bau, Bursztein, Gupta, & Mitchell, 2010). The following are some of its features:

- **Payloads:** including HTML5 compactable XSS injection payloads, this tool has more than 380 payloads. The advantage of using Xenotix XSS can be seen if compared to other XSS filters, which are mostly implemented using html entities filter and html special characters filter String Replace filter. Because of the weakly designed of those filters, hackers can bypass them by specific XSS payloads present in the inbuilt payload list.
- **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. Attackers are mostly interested in uses session cookies, which help them in such web pages that ask for passwords frequently to confirm some actions.
- **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. 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 (Bau et al, 2010).

In short, XSS is the most popular website security threat. Xenotix XSS can be used to detect most of those threats by performing penetration tests on the web pages against XSS vulnerabilities. There are many programs for bug bounty have been offered by companies such as Google Vulnerability Reward Program and Facebook Bounty to discover and solve any vulnerability that may cause such security threats against them.

3.6 SQL Injection Vulnerability

Databases mostly 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 (Curphey & Arawo, 2006).

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 (Kiani, Clark, & Mohay, 2008). If an injected query successfully done, its exploitation can read sensitive data, modify it, or even execute administration operations on the database. Moreover, database management system (DBMS) implementation can be affected on the database centralization and distribution. Thus, SQL injection may result in some operating system's error (Subashini & Kavitha, 2011).

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 (Shahriar, North, & Chen, 2013). 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 (Shulman & Co-founder, 2006; Shegokar & Manjaramkar, 2014). 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.

3.6.1 SQL Inject-Me

This is an Exploit-Me tool based on Firefox used to discover SQL Injection vulnerabilities. Exploit-Me 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.

3.6.2 SQLninja

SQLninja 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. Here are some features of this tool:

- It can 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.

3.6.3 Havij

It is an automated tool that helps penetration testers to detect sql Injection vulnerabilities on web applications (Pundlik, Kumar, Gaikwad, Aadhale, & Waghmare, 2013). 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 (Pundlik et al, 2013; Nagpal, Singh, Chauhan, & Panesar, 2015). Here are some of its abilities:

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

4. Comparison and Evaluation of Security Tools

This section compares the tools that are covered in each type of these vulnerabilities. Table 2 shows a comparison between the security tools for the fingerprinting and Information Leakage vulnerabilities. Table 3 shows a comparison between the security tools for the Insufficient Transport Layer Protection vulnerability (ITLP). Table 4 shows a comparison between the security tools for the Cross-Site Scripting vulnerabilities. Table 5 shows a comparison between the security tools for the SQL-Injection vulnerabilities.

Table 2. Fingerprinting and Information Leakage vulnerabilities' tools

Tool	Free Source	Features
Nmap	yes	Popular, well documented, easy-to-use, Portable, free, Powerful, flexible
SHODAN	yes	Free, easy-to-use, online search engine
BlindElephant	yes	Fast, support for 15 commonly deployed web apps, support for web app plugins
Httprint	no	Free, server-side impact
Netcraft (information leakage vulnerabilities)	yes	Free, server-side impact, could be used for other types of vulnerability
WhatWeb	yes	Free, server-side impact

Table 3. ITLP vulnerability tools

Tool	Free Source	Features
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

Table 4. Cross-Site scripting vulnerability tools

Tool	Free Source	Features
XSS Server	no	Allows users to generate an XSS script by using this tool
XSSer	yes	Requires Python
Xenotix XSS	yes	2nd largest XSS Payloads

Table 5. SQL-Injection vulnerability tools

Tool	Free Source	Features
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

5. Policy and Recommendations

Vulnerabilities such as Cross-Site Scripting, SQL Injection, and HTTP Response Splitting are occurred by design errors. On the other hand, Information Leakage, Insufficient Transport Layer Protection, and Fingerprinting are often caused by insufficient administration. Thereby, this section elucidates the usage of the tools that are 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.

5.1 Cross-Site Scripting

Tools that are listed in Cross-Site Scripting 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.

5.2 SQL Injection Vulnerability

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

5.3 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.

5.4 Fingerprinting

Nmap, as we mentioned previously, can be installed in both sides, however, in our case we indicated Nmap as a server-side tool. SHODAN is the tool that can be used by the user before establishing a connection with a host server to find out more information about it, so it is a client-side tool. BlindElephant is also client-side tool that is used to show the versions of the running applications on a particular web site.

5.5 Information Leakage

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

5.6 HTTP Splitting

It is recommended that the web developers pay more attention to the user inputs and validate them to ensure that no malicious code has been injected. Tools in this type such as AppScan, which is the most common tool, are server-side tools. They help administrators to run different tests throughout the application development lifecycle.

6. Conclusions

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.

Acknowledgements

This research work is partially supported by the National Science Foundation under Grants CNS-1460897 and 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

- Alvim, M. S., Chatzikokolakis, K., McIver, A., Morgan, C., Palamidessi, C., & Smith, G. (2016, June). Axioms for information leakage. In *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th* (pp. 77-92). IEEE.
- AppScan, I. R. (2009). Security AppScan: Identify and fix vulnerabilities in web and mobile applications prior to deployment [WWW page]. URL <https://www.ibm.com/us-en/marketplace/ibm-appscan-source>
- Auger, R. (2010). HTTP Response Splitting [WWW page]. URL <http://projects.webappsec.org/w/page/13246931/HTTP%20Response%20Splitting>
- Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2010, May). State of the art: Automated black-box web application vulnerability testing. In *Security and Privacy (SP), 2010 IEEE Symposium on* (pp. 332-345). IEEE.
- Bruns, A. (2016). Prosumption, Produsage. *The International Encyclopedia of Communication Theory and Philosophy*. <https://doi.org/10.1002/9781118766804.wbiect086>
- Catteddu, D. (2010). Cloud Computing: benefits, risks and recommendations for information security. In *Web application security* (pp. 17-17). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16120-9_9
- Curphey, M., & Arawo, R. (2006). Web application security assessment tools. *IEEE Security & Privacy*, 4(4), 32-41. <https://doi.org/10.1109/MSP.2006.108>
- Curphey, M., Endler, D., Hau, W., Taylor, S., Smith, T., Russell, A., & Klien, A. (2002). A guide to building secure web applications. *The Open Web Application Security Project*, 1(1).
- Dhanjani, N., & Clarke, J. (2005). *Network Security Tools: Writing, Hacking, and Modifying Security Tools*. "O'Reilly Media, Inc."
- Gupta, S., & Gupta, B. B. (2017). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, 8(1), 512-530. <https://doi.org/10.1007/s13198-015-0376-0>
- Huang, Y. W., & Lee, D. T. (2005). Web Application Security—Past, Present, and Future. In *Computer Security in the 21st Century* (pp. 183-227). Springer US. https://doi.org/10.1007/0-387-24006-3_12
- Kals, S., Kirda, E., Kruegel, C., & Jovanovic, N. (2006, May). Secubat: a web vulnerability scanner. In *Proceedings of the 15th international conference on World Wide Web* (pp. 247-246). ACM. <https://doi.org/10.1145/1135777.1135817>
- Kiani, M., Clark, A., & Mohay, G. (2008, March). 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* (pp. 47-55). IEEE. <https://doi.org/10.1109/ARES.2008.123>
- Livshits, V. B., & Lam, M. S. (2005, August). Finding Security Vulnerabilities in Java Applications with Static Analysis. In *USENIX Security Symposium* (Vol. 14, pp. 18-18).
- Lyon, G. (2009). Nmap [WWW page]. URL <http://nmap.org/>
- McClure, S., Scambray, J., Kurtz, G., & Kurtz. (2009). Hacking exposed: network security secrets and solutions.
- Meier, J. D., Mackman, A., Dunner, M., Vasireddy, S., Escamilla, R., & Murukan, A. (2003). Improving web application security: threats and countermeasures. *Microsoft Corporation*, 3.
- Nagpal, B., Singh, N., Chauhan, N., & Panesar, A. (2015, May). Tool based implementation of SQL injection for penetration testing. In *Computing, Communication & Automation (ICCCA), 2015 International Conference on* (pp. 746-749). IEEE.
- Nassar, N., Hoyos C., & Anderson, D. (2012). Introduction to AppScan policies [WWW page]. URL <https://www.ibm.com/developerworks/library/se-appscan/index.html>
- Nava, E. V., & Lindsay, D. (2009). Our favorite XSS filters/IDS and how to attack them. *Black Hat USA*.
- Netcraft, L. T. D. (2009). Netcraft Extension [WWW page]. URL <http://toolbar.netcraft.com/>
- Bruns, A. (2016). OWASP (2017). About the open web application security project [WWW page]. URL https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project
- Pundlik, S., Kumar, R., Gaikwad, B., Aadhale, A., & Waghmare, V. (2013). SQLIJS: SQL Injection Attack Handling System. *International Journal of Engineering Research & Technology (IJERT) ISSN*.
- Punter, M. (2014). Detecting and exploiting XSS injections using XSSer tool [WWW page]. URL

- <http://securityxploded.com/detecting-exploiting-xss-using-xsser-tool.php>
- Qualys, S. S. L. (2014). labs. SSL server test.
- Ristenpart, T., Tromer, E., Shacham, H., & Savage, S. (2009, November). 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* (pp. 199-212). ACM.
- Ristic, I. (2009). How well do you know SSL: If you want to learn more about the technology that protects the Internet, you've come to the right place. [WWW page]. URL <https://www.ssllabs.com/index.html>
- Ristic, I. (2014). Insufficient Transport Layer Protection [WWW page]. URL <http://projects.webappsec.org/w/page/13246945/Insufficient%20Transport%20Layer%20Protection>
- SecureTeam (2005). Introduction to HTTP Response Splitting [WWW page]. URL <http://www.secureteam.com/securityreviews/5WP0E2KFGK.html>
- Shah, S. (2004). An introduction to HTTP fingerprinting [WWW page]. URL http://www.net-square.com/httpprint_paper.html. Available E-mail: saumil@net-square.com
- Shahriar, H., North, S., & Chen, W. C. (2013, June). Client-side detection of sql injection attack. In *International Conference on Advanced Information Systems Engineering* (pp. 512-517). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38490-5_46
- Shar, L. K., Tan, H. B. K., & Briand, L. C. (2013, May). Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis. In *Proceedings of the 2013 International Conference on Software Engineering* (pp. 642-651). IEEE Press. <https://doi.org/10.1109/ICSE.2013.6606610>
- Shegokar, A. M., & Manjaramkar, A. K. (2014). A survey on SQL injection attack, detection and prevention techniques. *Int. J. Comput. Sci. Inf. Technol*, 5(2), 2553-2555.
- Shrivastava, A. (2011). Web application finger printing [WWW page]. URL https://anantshri.info/articles/web_app_finger_printing.html
- Shulman, A., & Co-founder, C. T. O. (2006). Top ten database security threats. *How to Mitigate the Most Significant Database Vulnerabilities*.
- Snehi, J., & Dhir, R. (2013). Web Client and Web Server approaches to Prevent XSS Attacks. *International Journal of Computers & Technology*, 4(2b1), 345-352.
- Stallings, W. (2007). *Network security essentials: applications and standards*. Pearson Education India.
- Stuttard, D., & Pinto, M. (2011). *The web application hacker's handbook: finding and exploiting security flaws*. John Wiley & Sons.
- Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1), 1-11. <https://doi.org/10.1016/j.jnca.2010.07.006>
- The OpenSSL Project (1990). OpenSSL cryptography and SSL/TLS Toolkit [WWW page]. URL <https://www.openssl.org/>
- Thomas, P. (2010). BlindElephant web application fingerprinter [WWW page]. URL <http://blindelephant.sourceforge.net/>

Copyrights

Copyright for this article is retained by the author(s), with first publication rights granted to the journal.

This is an open-access article distributed under the terms and conditions of the [Creative Commons Attribution license](https://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.