An OWASP Top Ten Driven Survey on Web Application Protection Methods

Ouissem Ben Fredj[1](✉), Omar Cheikhrouhou[2,3], Moez Krichen[4,5], Habib Hamam[6], and Abdelouahid Derhab[7]

[1] University of Sousse, Sousse, Tunisia
ouissem.benfredj@gmail.com
[2] Taif University, Taif, Saudi Arabia
o.cheikhrouhou@tu.edu.sa
[3] University of Monastir, Monastir, Tunisia
[4] Al-Baha University, Al Bahah, Saudi Arabia
[5] University of Sfax, Sfax, Tunisia
[6] University of Moncton, Moncton, Canada
[7] King Saud University, Riyadh, Saudi Arabia

Abstract. Web applications (WAs) are constantly evolving and deployed at broad scale. However, they are exposed to a variety of attacks. The biggest challenge facing organizations is how to develop a WA that fulfills their requirements with respect to sensitive data exchange, E-commerce, and secure workflows. This paper identifies the most critical web vulnerabilities according to OWASP Top Ten, their corresponding attacks, and their countermeasures. The application of these countermeasures will guarantee the protection of the WAs against the most severe attacks and prevent several unknown exploits.

Keywords: Survey · Security · Web · Attacks · OWASP top ten · Countermeasures

1 Introduction

During the latest period, the organizations have been using the web not only as a tool to advertise their images, product, and services, but also to perform their daily tasks, including sensitive data and complex workflows. Moreover, due to the popularity and the spread of sophisticated hand-held devices, several applications are moving from the regular desktop-based versions to the web-based ones to target more devices with low cost of portability [32]. On the other hand, the number attackers is continuously growing, and their attack techniques are becoming increasingly sophisticated and dangerous, which impose real security challenges on the organizations to secure their web applications (WAs). Hence, the security of WAs has become an important research area, and several solutions have been proposed to protect the WA.

From another point-of-view, the security administrators usually deploy WAFs (WA Firewalls) to protect the WAs. However, as will be shown later in this paper, the WAFs often use trivial protection methods instead of the advanced techniques suggested by the researchers (see Sect. 4). There is a large gap between

© Springer Nature Switzerland AG 2021
J. Garcia-Alfaro et al. (Eds.): CRiSIS 2020, LNCS 12528, pp. 235–252, 2021.
https://doi.org/10.1007/978-3-030-68887-5_14

the state of the art web protection methods and those employed by the existing WAFs. This paper tries to narrow down this gap by identifying the most severe web attacks as well as the appropriate countermeasures against each attack. The critical attacks are determined based on the most known web vulnerabilities, which were released by the OWASP project [2]. We have reviewed the security countermeasures to provide the readers with the smallest set of protection methods that prevent the broadest range of critical attacks.

The rest of the paper is organized as follows: In Sect. 2, the most critical vulnerabilities, as released by the OWASP project, are presented. For each vulnerability, the corresponding attacks are identified. Section 3 is tailored to the analysis of the most recent progress in the security countermeasures for each attack. A focus will be made on the runtime and server-side web protection methods. Section 4 deals with the use of Firewalls for the WA protection. Section 5 is dedicated for the adoption of formal methods for this same purpose. Section 6 concludes the paper and give future directions.

2 Security Attacks Against Web Applications

In this section, we describe the possible attacks that could target a Web Application (OWASP [2]).

2.1 Injection

The injection attacks consist in injecting (sending) untrusted information for an interpreter. This injection is a part of an instruction: command/query. By providing malicious information, the attacker can mislead the interpreter and cause unintended commands. The most critical injection attacks are the following:

– **SQL Injection:** It consists in injecting (inserting) SQL commands into input forms or queries to get access to a database (DB) or manipulate its data, for example: modification or deletion of database content.
– **Code Injection:** This attack consists in injecting code that the application interprets and runs, which exploits poor processing of untrusted data.
– **XPATH Injection:** This attack takes place when a WA uses user-input information for building an XPath query corresponding to XML data.

For more information about SQL injection attacks, the reader can refer to [32].

2.2 Broken Authentication and Session Management

In case of broken authentication and session management attack, the intruder tries to exploit the vulnerabilities of the authentication procedure in order to access the WA or to use the credentials of other authorized users. This attack is classified into the following categories:

– **Brute Force Attack:** It consists in trying a combination of characters to guess the password of a given user.

- **Dictionary Attack:** If the attacker has some knowledge on the victim, he can prepare dictionary (set of valid words). Then, he combines these words to guess the victim password.
- **Credential Enumeration Attack:** Under this kind of attack, the intruder attempts to harvest valid usernames for a password-guessing campaign, by using verbose error of message telling whether the login is a valid username or not.
- **Session Fixation Attack:** In this attack, the hacker fixes the session ID, which will be used by user before the user logins into the server.
- **Cookie Poisoning Attack**: It consists in modifying a cookie by an intruder to obtain unauthorized information about the user for the purpose to perform for example identity theft.

2.3 Cross-site Scripting (XSS)

It consists in injecting malicious code/scripts into web responses, which are returned back by the trusted WA, to be executed by the web browser. The following three main kinds of XSS exist according to the way the malicious code is injected:

- **Stored XSS Attack:** It takes place when the user input (such as message forum, database data, comment field, visitor log, etc.) is stored on the WA server. Then, a victim may get back the stored data from the WA without making it safe.
- **Reflected XSS Attack:** It occurs when a client receives data in an HTTP request and uses the data in an unsafe manner within the immediate response.
- **DOM Based XSS Attack:** In this attack, the whole malicious data flow from source to sink occurs within the browser. It means that the data source is in the Document Object Model (DOM), the sink is in DOM as well, and the data flow does not leave the browser.

A recent survey about the XSS attacks can be found in [23].

2.4 Insecure Direct Object References

A Direct Object Reference takes place whenever a programmer presents references to internal implementation objects. It may be a database key, directory, or file. When there is no access control or other security measures, intruders may exploit such references to reach unauthorized data. This vulnerability may lead to the following several attacks:

- **Path Traversal Attack:** It is a kind of attack, in which insecure direct object reference to directories and files which are placed outside the web root folder or in hidden places including system and configuration files.
- **Direct Request Attack:** (also called forced browsing) It consists in using brute force procedures to access unlinked contents in the main directory. The attacker may use google crawler to list hidden pages and files.

- **Authorization Bypass Through User-Controlled SQL Primary Key Attack:** It occurs when the attacker manipulates a DB table primary key, which is used in an SQL statement, in order to reach inaccessible records.

2.5 Security Misconfiguration

Security misconfiguration problem occurs when one or more of the components of the system such as the applications, the frameworks, the application server, the web server, the DB server, the network router, and the platform are not well configured. Secure settings have to be defined, implemented, and maintained. Default settings are very often the cause of such a risk [50]. The attacker could exploit this flaw to perform several attacks. The severity of the attack depends on the misconfiguration level and place.

2.6 Sensitive Data Exposure

IT systems always store in a DB users personal data like passwords, home addresses, phone numbers, credit card details, etc. Once the systems are not properly secured from forbidden access, there is a strong likelihood of an attacker exploiting that vulnerability and stealing the information. There are three attacks, which are related to the sensitive data exposure:

- **Information Leakage Attack:** it occurs when a WA reveals sensitive data, such as error messages or developer comments. These sensitive data, which give an attacker useful guidance, can be exploited to attack the system [4–6,58].
- **Transmission Attack:** When the communication is not encrypted, all data exchanged between the client and the web server is sent in clear-text which leaves it exposed to interception, injection and redirection.
- **Database Theft:** when the sensitive data in the DB is not protected using strong encryption or access policies, attacker could steal this data. Three database attacks are possible: Brute-force attack; SQL injection and Privilege escalation.

2.7 Missing Function Level Access Control

Some WAs check access rights to function level before making the feature available to the user. Nevertheless, once each feature is accessed, applications must achieve the same access control check for the server. Whenever requests are not checked, attackers can access the features without proper permission. Examples of attacks that may exploit this vulnerability are the following:

- **Local File Inclusion Attack:** The attacker tries to find a page that receives as input a path to a file to be included in the calling page.
- **Remote File Inclusion Attack:** it is the same as the Local File Inclusion Attack but instead of including files located in the same server, the attacker manipulates the user input to include remote files.

- **Command Injection Attack:** it is another attack that accesses the OS functions with unauthorized manner. The attacker tries to find a piece of code in the WA that accepts untrusted input to build OS commands without proper sanitization.

2.8 Cross-site Request Forgery (CSRF)

According to [7], a WA is vulnerable to CSRF attacks (sometimes referred to as XSRF or Session Riding) when it does not verify that any request done by a trusted user has actually been intentionally done by that user only. There is a big difference between CSRF vulnerabilities and XSS vulnerabilities. The CSRF attack exploits an authenticated user to make a request on their behalf. Thus, a web site that uses cookies for authentication may be vulnerable, as well as those web application that use Basic or Digest authentications, because the browser automatically sends the cookies and the server will rely on that browser.

2.9 Using Components with Known Vulnerabilities

Software Components, like frameworks, libraries, and other kinds of modules, often execute with maximum privilege. Whenever a weak component is attacked, it may lead to serious threat. Depending on the vulnerabilities of the components, any kind of attack is eventually possible. For example, if a website is using a library vulnerable to SQL injection, the whole website will be vulnerable to such an attack. The open source libraries, framework, and content management systems (CMSs) are the source of many attacks.

2.10 Unvalidated Redirects and Forwards

WAs usually forward and redirect users to other websites and pages, and exploit input data to identify new potential destinations. Without proper checking and authentication of the input data, users can be redirected to malware or phishing. Attackers may also exploit forwards to reach unauthorized zones. For instance, http parameter can include, or part of, a URL value, which could be exploited by the WA to redirect the request to the considered URL. An attacker can execute a phishing scam and capture user information by changing the URL address to a hostile site. Since the server in the updated connection has the same name as the original (trusted) site attempts at phishing look more trustworthy.

3 Countermeasures Against Attacks

In this section, we present the main proposed solutions to mitigate web attacks described in the previous section.

3.1 Countermeasures Against Injection Attacks

Many solutions have been adopted to address the SQL Injection, as it presents the most widely spread attack [32]. The authors of [60] proposed a framework based on information theory for detecting SQLI attacks. The proposed framework statically estimates query's entropy based on the distribution of token probability of a query. First, the system computes the entropy of every query included in the program source code before the deployment of the application. Then, during the execution of the application, the system computes again the entropy of each invoked SQL query to detect if there is any change in the measured entropy. In [51], the authors proposed a WAF based on Artificial Neural Network (ANN) to avoid SQLIAs. The system consists of a pair of steps: Training step and Working step. During the training step, a collection of normal and malicious data is fed to the system to train the ANN. During the working step, the obtained ANN is integrated into the WA firewall to detect the WA attacks. The authors of [48] proposed a semantic comparison based scheme. The semantic comparison is made between the two syntax trees of a query during training and run-time. If the two trees are similar, then the query is evaluated as benign query, else it is evaluated as malicious one. Authors in [30] have proposed WASP, a tool for avoiding SQLIAs using the notion of positive tainting and on syntax-aware evaluation. The idea of positive tainting is to identify and track trusted data, instead of tainting untrusted data in traditional (negative) tainting approach. The advantages of the positive tainting over the negative one is that it generates false positives instead of false negatives, in case of incompleteness.

3.2 Countermeasures Against Broken Authentication and Session Management

For session hijacking, the traditional countermeasure technique consists in binding the client IP address. More precisely, in this technique, the web server binds a user's session to a fixed IP address, and then discard any request coming from a distinct IP address. This technique requires that each client possess a different and unchanging public IP address. However, a network generally uses NAT protocol to share the same IP address to multiple clients and, therefore, make this technique ineffective [25]. Another technique to mitigate session hijacking is based on tracking user browser fingerprint. A browser fingerprint consists of numerous characteristics of the user browser. Any modification of the user browser fingerprint might represent an attacker stealing a session [52]. SessionLock [9] adds an integrity checks to every client request based on a secret shared with the server. If a session identifier is stolen, a valid request cannot be computed since the value of the secret is unknown. One limit of SessionLock is its vulnerability to script-based attacks. To mitigate session hijacking attacks and inspired by the concept of Kerberos service tickets, the authors in [24] proposed to replace the static session identifier with disposable tokens per request. Macaroons [33] targets cloud services and restricts access to cooki.e. Macaroons uses chains of nested Hash-based Message Authentication Codes (HMACs), constructed from a shared secret and a chain of messages.

3.3 Countermeasures Against XSS Attacks

A first defense line against XSS, at the server-side, is to adopt a user-input validation to enforce the security. Validation can use either blacklisting or whitelisting techniques. Moreover, once user-input is found to be malicious, it can either be sanitised or rejected [3]. However, the secure input handling method cannot achieve full protection, especially for complex website.

A second defense line, which is becoming more and more implemented in web-servers, is based on Content Security Policy, which generally defines trusted origins that the browser is allowed to download resources (can be a script, a style-sheet, an image, etc.) from them. Therefore, although an intruder is able to inject vulnerable content into the website, the CSP method may block its execution. Authors in [63] proposed a secure WA proxy for detecting and blocking Cross Site Scripting (XSS) attacks. The proposed framework contains a reverse proxy intercepting the returned HTML messages first, then using an altered web browser to locate vulnerable scripts.

The authors in [59] proposed to use Kullback-Leibler Divergence (KLD) measure to provide a proxy-level detection methodology for the XSS attacks. The idea is based on the intuition that legitimate WAs JavaScript code should remain comparable or very similar to a rendered web page's JavaScript code. For this purpose, the authors proceed to the tokenization of the considered script code into unique elements and calculate the probabilities of their occurrences in order to construct two sets P (legitimate JS code available in the application page) and Q (observed JS code available in the response page). Then, KLD computes the distance separating these two proposed probability distributions. An XSS attack is detected in case of a significant divergence between the two sets.

3.4 Countermeasures Against Insecure Direct Object References and Missing Function Level Access Control

To secure the access to the resources and the utilization of internal functions of a WA, most of security systems have used access control mechanisms. For instance in Role-Based Access Control (RBAC) [27], programmers control objects by permissions, assign permissions to roles and assign roles to users. Permission authorizes a user for a role in a given session. The Separation of Duty Constraints prevent a user from acquiring two or more conflicting roles. For example, Cisco ACE WA Firewall uses RBAC to define the administration roles of the WAF itself. In [53], the authors describe an implementation of RBAC with role hierarchies on the Web by secure cookies. The user's role information is injected in a set of secure cookies and transmitted to the corresponding Web servers. In order to verify the cookies, they use PGP (Pretty Good Privacy) to define cookie-verification procedures.

In [12], the authors proposed an access control method for open web service applications. Their work is based on the eXtensible Access Control Markup Language (XACML) which belongs to the class of access control languages.

3.5 Countermeasures Against Sensitive Data Exposure

As presented in Sect. 1, the following three categories of sensitive data Exposure flaw exist:

– **Information Leakage:** As for this flaw, only the developer can improve security by paying attention to what he leaves in the code and to handle in a secure way the errors that can occur.
– **Transmission Attacks:** this kind of attacks is mainly avoided by a strong encryption mechanism and we do not know a well known approaches used in WAFs.
– **Database Thefts:** to deal with this attack, cryptography is a key solution together with a good security policy to access database. In [26], the authors proposed a dynamic database security policies as a solution for this kind of attack.

As conclusion, there are no known approaches that can be used by WAFs to overcome sensitive data exposure flaw.

3.6 Countermeasures Against CSRF

The are some countermeasures at the server-side to mitigate CSRF attacks [11, 26, 36]. OWASP developed a project called CSRFGuard [1]. It is a library, which implements a variant of the Synchronizer Token Pattern to minimize the risk of CSRF attacks. The authors of [34] defined a server-side proxy named NoForge, which could be plugged into the considered system to discover and avoid CSRF attacks and it is transparent to users and applications. This proxy primarily detects and protects PHP applications against CSRF attacks. Zeller et al. in [64] enumerated the characteristics of server-side precautions to protect users. They also developed a plug-in at the server side for preventing users from the attacks.

3.7 Countermeasures Against Unvalidated Redirects and Forwards

The authors in [57] categorized the phishing countermeasures into four categories: blacklist-based, heuristic-based, visual similarity-based, and machine learning based. The blacklist-based techniques build a repository of discovered phishing URLs, which should be updated regularly. The most representative works under this category are the Google Safe Browsing API [8], PhishNet [54], which predicts the phishing URLs based on the known phishing URLs, and Automated Individual White-List (AIWL) [20] that keeps a list of trusted Login User Interfaces (LUI). However, this list suffers from the problem of untrusted LUI prediction. Generally, the blacklists offer good True-Positive (TP) rates but suffer from False-Positive (FP) rates. SPHERES [28] is a WAF implemented in the WA server based on behaviour, and prevents the phishing attack by defining a profile for each parameter provided by the web client.

Table 1. Attacks classification and their countermeasures

Attacks	Sources	Sinks	Countermeasure techniques
SQL injection	User input, cookies, server variables	Database	Information theory based; compare the query entropy before deployment and during execution
			Artificial neural network
			Semantic comparison
			Positive tainting and on syntax-aware evaluation
			Syntactic structures comparison of the programmer-intended query and the actual query
			Software-testing techniques
			The model is expressed as a grammar that only accepts legal queries
			Taint based approach
Code injection		System Web site	Technique based on multitier compilation
			Constructs a control flow graph for each function
Brute force attack, Dictionary attack, Credential enumeration	User input	Session	Picture-based
Session hijacking, Session fixation, Cookie poisoning	Cookies	Website, URL, Session	Time signature based
			Shared secret
			Token per request
			Chains of nested HMAC
Stored XSS, reflected XSS, DOM XSS	User input	Data base Website	Per-page security policies
			Probability distributions of tokens extracted from the script code
			Creation of shadow pages that reflects the set of scripts that a web application intends to create
			XSD schema file
			Reverse proxy
			Boundary injection and policy generation
CSRF	User input	Database	Server side changes and captcha
Privilege escalation	Use input	Database	Automatically instrument application source code program analysis to check for authorization state consistency in a web application
Transmission attacks	User input	ALL	Cryptography
Directory traversal attacks, Path Traversal attack, The direct request attack	User input	System, website	Access control using RBAC
			Simple filtering rule
			RBAC for cookies
			Security Policy Description Language
			Access control using RBAC for WS-BPEL processes
Authorization bypass through user-controlled key	User input	DB	Access control using RBAC
Local file inclusion, Command injection, Remote file inclusion	User input	System, DB, NET, website	Access control using RBAC
Phishing attack	User input	DB (user credentials)	Recognize fake URLs
			Recognize whitelist URLs

3.8 Discussion

The most severe, critical and widespread flaw as classified by the OWASP top ten is the injection flaw [2]. The main attack under this flaw is SQL injection. Several solutions were proposed to mitigate this attack and they can be classified mainly as grammar-based, entropy-based, machine learning-based and tainting-based. Grammar-based methods are efficient but require to write a grammar model for each possible query, which is error prone. Moreover, these methods can not discover stored procedure type attacks and database management systems (DBMS) specific subqueries. In addition, the time complexity of these methods is high, and hence it is impossible to discover the attack in real-time [47].

The entropy methods are based on probabilistic models and so far are unstable. Taint-based approaches are time consuming as they need to monitor every variable in the web site. Machine learning techniques are not well adapted to this context as they need a long training phase and the results can include several false negatives and positives [47].

The second severe flaw is related to authentication and session management. As for authentication, the value of the authenticated cookie must be updated each time the level of authorization of the user takes a new value to combat potential session vulnerabilities [16]. The web developer should enhance the authentication method using picture-based or time-signature-based authentication scheme. The common protection of session attacks prevents JavaScript access to session cookies. Another promising idea is based on defining a collection of security policies.

For the XSS attack, many defense solutions are adopted, and existing industrial approaches mostly rely on user input sanitizing [55]. Some approaches use probability distribution of tokens in a web page [59]. Other approaches are based on page code modification either by creation of shadow page [14], or by inserting a script ID [63], or using boundaries injection, [29].

The fourth and the seventh flaws lead to similar attacks. The fourth category encompasses attacks that lead to a misuse of the objects that exist in the web structure, and the seventh category encompasses the attacks that misuse the functions provided by the web application. Both categories could be secured by controlling and managing the roles, the objects and the permissions to handle both of them.

Regarding the fifth flaw, the web administrator should fine-tune the configuration entries of the web application during the deployment and use of the application. Thus, the default values usually known by the attacker will be minimized and the security of the component will be maximized. A static scan of the server configuration could help in this stage.

Regarding the sixth flaw, the web server must use secured connection when sensitive data are exchanged with the client (Emails, banking transaction, etc.). The system administrator must choose the right database access policies and a strong cryptography of sensitive data stored in database. The web developer must pay attention to what he leaves in the code source and must handle the system errors perfectly.

A good and very well known way to overcome the CSRF attacks, i.e., the eight flaw, is using captcha. We need to apply strong models to avoid bypassing it. Many others works are proposed to handle this attack. They add some code at the web server [64] and enhance also the client side by some routines.

Regarding the ninth flaw, the web developer should handle with care the external components used in the website especially the open source libraries and frameworks. The developer can minimize the risks produced by these components by rewriting their interface for example. However, it will be difficult to use a subsequential version of the component.

The tenth category in the top ten may lead to the phishing attach, which could be mitigated by blocking fake URLs using existing black lists [8,54] or white lists [20]. A summary of the studied approaches is given in Table 1.

4 Protection Methods for WA Firewalls

WA firewalls (WAFs) are the primary front-end protection mechanism for web-based applications which are continuously under attack. We can find two categories of WAF: open source and commercial.

4.1 Open Source WAF

Examples of open source solutions that can be used to deploy a firewall to protect web applications are the following:

- **AQTRONIX WebKnight**: It is an open source WA Firewall (WAF) for Internet Information Services (IIS). AQTRONIX WebKnight is an ISAPI filter that tries to secure the target web server by blocking certain requests. To do so, a scanning and processing of all requests is performed according to filter rules, which do not come from a dataset of attack signatures requiring regular updates.
- **ModSecurity:** It is a toolkit for real-time WA access control, logging and monitoring. This toolkit supports a pair of deployment options: reverse and embedded proxy deployment. This method enables protecting the WA against a wide range of attacks. It also offers the monitoring of HTTP traffic, its logging as well as the real-time analysis of it.

4.2 Commercial WAF

Examples of commercial solutions that can be used to deploy a firewall to protect web applications are the following:

- **dotDefender:** it is a WA Firewall installed on Apache or Microsoft IIS Server. This WAF claims preventing the following attacks: XSS, SQLIAs, Credit Card Disclosure, DoS, etc.

- **Imperva SecureSphere:** [31] it may be used as a reverse proxy or as a transparent bridge, and when deployed out-of-band, it operates passively as a sniffer, detection and alteration without protection against attacks.
- **Barracuda:** [13] it is designed to protect WA and Web sites from application vulnerabilities to instigate data theft, application-layer DoS attacks, or defacement of the Web site of an organization. Th WAF offers protection against attacks like XSS, Brute Force and SQL Injection.

5 Model-Based Testing and Formal Methods for Web Service Security

In this section, we give an overview on WA verification, i.e., model-based testing and formal methods, and present their applications for web applications. WA verification can be classified under the following two methods:

- **Model Based Testing (MBT):** It is a methodology [37–39] where the behavior of the System Under Test (SUT) is encoded by means of an abstract model. This methodology permits to automatically produce abstract test scenarios from the considered model. Regarding testing security aspects, the authors of [41] proposed an MBT methodology for validating security aspects of IoTs in Smart Cities. The proposed methodology takes advantage of the adoption of the standard testing language TTCN- 3 [44] and a cloud-oriented architecture [45]. Similarly, the authors of [42,43] proposed an MBT methodology in order to validate security properties of IoTs. In [40], a set of optimization techniques was adopted in order to diminish the complexity of MBT procedures.
- **Formal Methods (FM):** When establishing computer systems (CS), the complete detection and correction of design errors remain remarkably hard in the context of manual simple verification techniques and functional testing activities. Consequently, in the early 1980s, scientists [18,21,56] started to make CS verification methodologies more rigorous, specially by making them more automatic. In fact, with the emergence of new mathematical languages for the specification and description of dynamic systems, the first formal verification methodologies have appeared.

Model-based testing methods for WA security can be classified as follows:

- **Modelling HTTP Requests:** In [19], an approach named Chained Attack is proposed. The proposed approach considers HTTP requests as a starting point, produces models, and extracts scenarios of attacks from these models using model-checking procedures.
- **Formalizing Vulnerabilities into Test Purposes:** The authors of al [46] proposed an MBT security validation methodology, which allows to formalize vulnerability test patterns in form of test purposes. The authors defined the behavior of the considered web applications and purposes of tests, and adopted model-checking procedures in order to produce testing scenarios.

- **Consideration of an Attacker Model:** The authors of [15] adopted an MBT methodology, in which the formal models of attacker are considered to validate web applications and functionalities.
- **Technique Inspired by Mutation Testing:** In [17], the authors presented an MBT approach for validating security properties of WA. The proposed approach is closely inspired by mutation testing techniques.
- **Use of UMLsec Tool:** In [35], the authors proposed an MBT testing approach for the automatic production of security test-scenarios. The approach takes advantage of the UMLsec tool. It aims at testing the security properties of the Common Electronic Purse Specifications.
- **Use of Alloy Analyser:** In [10], a methodology using the Alloy Analyzer for inspecting several web applications and mechanisms was proposed. The authors adopted threat models such as an intruder taking control over a website or a whole part of the network.
- **Mobster Tool:** In [49], the authors introduced the MobSTer tool, which is a Model-based Security Testing Framework that may help security analysts in testing security aspects of WA. This framework combines model-checking procedures with the knowledge obtained from penetration testing guidelines and checklists.

Formal methods for WA security can be classified as follows:

- **Security by Construction:** Some works in the literature [61] aim at defining new formal languages and abstraction techniques in order to make the Web safer. For this purpose, these works attempt to identify the main limitations in the current conception techniques of the Web and suggest a paradigm evolution to ameliorate it. This set of suggestions is adequate for dealing with the principle source of security problems. However, they mostly need a deep change to current WA and technologies.
- **Modelling, Verification and Enforcement:** Some other research works adopt an other strategy which consists in considering appropriate algorithms and models for dealing with the FV of the security properties of modern Web technologies. These research works attempt to exploit available standards and frameworks as best as they can. This approach may be in many cases sub-optimal and not very effective. However, the main advantage is that this procedure does not impact a lot the existing Web technologies. For example with respect to scripting languages, different solutions [62] based on the adoption of rigorous semantics for the considered language were adopted.
- **An industrial Application:** In [22], the authors applied FV techniques for the security of Amazon Web Services (AWS). Two main goals were considered with this respect. The first one consists in raising the level of security of the provided products and the second one helping customers securing themselves against possible attacks.

6 Conclusions and Future Work

This paper presents an up to date survey about web applications vulnerabilities, attacks and server-side countermeasures. The main vulnerabilities and attacks, which targets current WAs according to the OWASP top 10 classification, have been described. After that we surveyed the countermeasures solutions proposed in the last decades to protect WAs against these attacks. The literature includes hundreds of works about server-side web protection methods, and many of them propose enhanced protection models. The existing WAFs include only simple protection rules, which does not take into account the advances in the field. There is a big gap between the research products and the WAF methods. The developers of WAFs try to propose a global protection tools that mitigate a wide range of attacks using simple methods that fail to deal with the complexity of new attacks. As a future work, we plan to design and develop a WA firewall that is lightweight and adaptable to current WAs needs.

References

1. Category: OWASP CSRFGuard project - OWASP. https://www.owasp.org/index.php/Category:OWASP_CSRFGuard_Project. Accessed 30 July 2020
2. Category: OWASP top ten project - OWASP. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project. Accessed 230 July 2020
3. Excess XSS: A comprehensive tutorial on cross-site scripting. http://excess-xss.com/. Accessed 30 July 2020
4. Information leakage - OWASP. https://www.owasp.org/index.php/Information_Leakage. Accessed 30 July 2020
5. InfoSecPro.com - computer, network, application and physical security consultants. http://www.infosecpro.com/applicationsecurity/a52.htm. Accessed 30 July 2020
6. The web application security consortium/information leakage. http://projects.webappsec.org/w/page/13246936/Information%20Leakage. Accessed 30 July 2020
7. Website. https://lthieu.wordpress.com/2012/11/22/cross-site-request-forgery-a-small-demo. Accessed 30 July 2020
8. Website. https://developers.google.com/safe-browsing/. Accessed 30 July 2020
9. Adida, B.: Sessionlock: securing web sessions against eavesdropping. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, New York, NY, USA, pp. 517–524. ACM (2008)
10. Akhawe, D., Barth, A., Lam, P.E., Mitchell, J., Song, D.: Towards a formal foundation of web security. In: 2010 23rd IEEE Computer Security Foundations Symposium, pp. 290–304, July 2010. https://doi.org/10.1109/CSF.2010.27
11. Anwar, D., Anwar, R.: Transparent data encryption-solution for security of database contents. Int. J. Adv. Comput. Sci. Appl. **2**(3) (2011)
12. Ardagna, C.A., di Vimercati, S.D.C., Paraboschi, S., Pedrini, E., Samarati, P., Verdicchio, M.: Expressive and deployable access control in open web service applications. IEEE Trans. Serv. Comput. **4**(2), 96–109 (2011)
13. Barracuda: Barracuda WAF. White paper (2019)
14. Bisht, P., Venkatakrishnan, V.N.: XSS-GUARD: precise dynamic prevention of cross-site scripting attacks. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 23–43. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70542-0_2
15. Blome, A., Ochoa, M., Li, K., Peroli, M., Dashti, M.T.: Vera: a flexible model-based vulnerability testing tool. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, pp. 471–478, March 2013. https://doi.org/10.1109/ICST.2013.65

16. Braun, B., Pauli, K., Posegga, J., Johns, M.: LogSec: adaptive protection for the wild wild web. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 2149–2156. ACM (2015)
17. Büchler, M.: Semi-automatic security testing of web applications with fault models and properties. Ph.D. thesis, Technical University Munich (2015). http://nbn-resolving.de/urn:nbn:de:bvb:91-diss-20151218-1273062-1-3
18. Bugliesi, M., Calzavara, S., Focardi, R.: Formal methods for websecurity. J. Log. Algebr. Methods Program. **87**, 110–126 (2017). https://doi.org/10.1016/j.jlamp.2016.08.006. http://www.sciencedirect.com/science/article/pii/S2352220816301055
19. Calvi, A., Viganò, L.: An automated approach for testing the security of web applications against chained attacks. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, SAC 2016, New York, NY, USA, pp. 2095–2102. ACM (2016). https://doi.org/10.1145/2851613.2851803. http://doi.acm.org/10.1145/2851613.2851803
20. Cao, Y., Ye, C., Weili, H., Yueran, L.: Anti-phishing based on automated individual white-list. In: Proceedings of the 4th ACM Workshop on Digital Identity Management - DIM 2008 (2008)
21. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982). https://doi.org/10.1007/BFb0025774. http://dl.acm.org/citation.cfm?id=648063.747438
22. Cook, B.: Formal reasoning about the security of amazon web services. In: Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, 14–17 July 2018, Proceedings, Part I, pp. 38–47 (2018). https://doi.org/10.1007/978-3-319-96145-3_3
23. Cui, Y., Cui, J., Hu, J.: A survey on XSS attack detection and prevention in web applications. In: Proceedings of the 2020 12th International Conference on Machine Learning and Computing, pp. 443–449 (2020)
24. Dacosta, I., Chakradeo, S., Ahamad, M., Traynor, P.: One-time cookies: preventing session hijacking attacks with stateless authentication tokens. ACM Trans. Internet Technol. **12**(1), 1:1–1:24 (2012)
25. De Ryck, P., Desmet, L., Piessens, F., Johns, M.: Primer on client-side web security. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12226-7
26. Doshi, J., Trivedi, B.: Sensitive data exposure prevention using dynamic database security policy. Int. J. Comput. Appl. Technol. **106**(15), 18600–9869 (2014)
27. Ferraiolo, D., Cugini, J., Kuhn, D.R.: Role-based access control (RBAC): features and motivations. In: Proceedings of 11th Annual (1995)
28. Fredj, O.B.: Spheres: an efficient server-side web application protection system. Int. J. Inf. Comput. Secur. **11**(1), 33–60 (2019)
29. Gupta, S., Gupta, B.B.: XSS-SAFE: a server-side approach to detect and mitigate cross-site scripting (XSS) attacks in javascript code. Arab. J. Sci. Eng. **41**, 897–927 (2015). https://doi.org/10.1007/s13369-015-1891-7
30. Halfond, W., Orso, A., Manolios, P.: WASP: protecting web applications using positive tainting and syntax-aware evaluation. IEEE Trans. Software Eng. **34**(1), 65–81 (2008)
31. Imperva: WAF gateway. White paper pp. 1–2 (2019)
32. Jemal, I., Cheikhrouhou, O., Hamam, H., Mahfoudhi, A.: SQL injection attack detection and prevention techniques using machine learning. Int. J. Appl. Eng. Res. **15**(6), 569–580 (2020)

33. Johns, M., Martin, J., Bastian, B., Michael, S., Joachim, P.: Reliable protection against session fixation attacks. In: Proceedings of the 2011 ACM Symposium on Applied Computing - SAC 2011 (2011)
34. Jovanovic, N., Kirda, E., Kruegel, C.: Preventing cross site request forgery attacks. In: Securecomm and Workshops, 2006. pp. 1–10. ieeexplore.ieee.org, August 2006
35. Jürjens, J.: Model-based security testing using UMLsec. Electron. Notes Theor. Comput. Sci. **220**(1), 93–104 (2008).https://doi.org/10.1016/j.entcs.2008.11.008
36. Kiernan, J., Jerry, K., Rakesh, A., Haas, P.J.: Watermarking relational data: framework, algorithms and analysis. VLDB J. Int. J. Very Large Data Bases **12**(2), 157–169 (2003)
37. Krichen, M.: Model-based testing for real-time systems. Ph.D. thesis, PhD thesis, PhD thesis, Universit Joseph Fourier, December 2007
38. Krichen, M.: A formal framework for conformance testing of distributed real-time systems. In: Lu, C., Masuzawa, T., Mosbah, M. (eds.) OPODIS 2010. LNCS, vol. 6490, pp. 139–142. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17653-1_12
39. Krichen, M.: Contributions to model-based testing of dynamic and distributed real-time systems. Ph.D. thesis, École Nationale d'Ingénieurs de Sfax (Tunisie) (2018)
40. Krichen, M.: Improving formal verification and testing techniques for internet of things and smart cities. Mobile Netw. Appl. 1–12 (2019)
41. Krichen, M., Alroobaea, R.: A new model-based framework for testing security of IoT systems in smart cities using attack trees and price timed automata. In: 14th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2019 (2019)
42. Krichen, M., Cheikhrouhou, O., Lahami, M., Alroobaea, R., Jmal Maâlej, A.: Towards a model-based testing framework for the security of internet of things for smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) SCITA 2017. LNICST, vol. 224, pp. 360–365. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94180-6_34
43. Krichen, M., Lahami, M., Cheikhrouhou, O., Alroobaea, R., Maâlej, A.J.: Security testing of internet of things for smart city applications: a formal approach. In: Mehmood, R., See, S., Katib, I., Chlamtac, I. (eds.) Smart Infrastructure and Applications. EICC, pp. 629–653. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-13705-2_26
44. Lahami, M., Fakhfakh, F., Krichen, M., Jmaiel, M.: Towards a TTCN-3 test system for runtime testing of adaptable and distributed systems. In: Nielsen, B., Weise, C. (eds.) ICTSS 2012. LNCS, vol. 7641, pp. 71–86. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34691-0_7
45. Lahami, M., Krichen, M., Alroobaea, R.: TEPaaS: test execution platform as-a-service applied in the context of e-health. Int. J. Auton. Adapt. Commun. Syst. **12**(3), 264–283 (2019)
46. Lebeau, F., Legeard, B., Peureux, F., Vernotte, A.: Model-based vulnerability testing for web applications. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops. pp. 445–452, March 2013. https://doi.org/10.1109/ICSTW.2013.58
47. Lee, I., Jeong, S., Yeo, S., Moon, J.: A novel method for SQL injection attack detection based on removing SQL query attribute values. Math. Comput. Modell. **55**(1–2), 58–68 (2012). https://doi.org/10.1016/j.mcm.2011.01.050. http://www.sciencedirect.com/science/article/pii/S0895717711000689. Advanced Theory and Practice for Cryptography and Future Security

48. Mamadhan, S., Manesh, T., Paul, V.: SQLStor: blockage of stored procedure SQL injection attack using dynamic query structure validation. In: 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 240–245 (2012)
49. Meo, F.D., Viganò, L.: A formal approach to exploiting multi-stage attacks based on file-system vulnerabilities of web applications. In: Engineering Secure Software and Systems - 9th International Symposium, ESSoS 2017, Bonn, Germany, July 3–5, 2017, Proceedings, pp. 196–212 (2017). https://doi.org/10.1007/978-3-319-62105-0_13
50. Mnif, A., Cheikhrouhou, O., Jemaa, M.B.: An ID-based user authentication scheme for wireless sensor networks using ECC. In: ICM 2011 Proceeding, pp. 1–9. IEEE (2011)
51. Moosa, A.: Artificial neural network based web application firewall for SQL injection. Proc. World Acad. Sci. Eng. Technol. **64**, 12–21 (2010)
52. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., Vigna, G.: Cookieless monster: exploring the ecosystem of web-based device fingerprinting. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 541–555. ieeexplore.ieee.org, May 2013
53. Park, J.S., Sandhu, R., Ghanta, S.L.: RBAC on the web by secure cookies. In: Atluri, V., Hale, J. (eds.) Research Advances in Database and Information Systems Security. ITIFIP, vol. 43, pp. 49–62. Springer, Boston, MA (2000). https://doi.org/10.1007/978-0-387-35508-5_4
54. Prakash, P., Kumar, M., Kompella, R.R., Gupta, M.: PhishNet: predictive blacklisting to detect phishing attacks. In: 2010 Proceedings IEEE INFOCOM, pp. 1–5. ieeexplore.ieee.org, March 2010
55. Prokhorenko, V., Choo, K.K.R., Ashman, H.: Web application protection techniques: a taxonomy. J. Netw. Comput. Appl. **60**, 95 – 112 (2016).https://doi.org/10.1016/j.jnca.2015.11.017. http://www.sciencedirect.com/science/article/pii/S1084804515002908
56. Queille, J.P., Sifakis, J.: Specification and verification of concurrent systems in CESAR. In: Dezani-Ciancaglini, M., Montanari, U. (eds.) Programming 1982. LNCS, vol. 137, pp. 337–351. Springer, Heidelberg (1982). https://doi.org/10.1007/3-540-11494-7_22. http://dl.acm.org/citation.cfm?id=647325.721668
57. Scott, D., Sharp, R.: Specifying and enforcing application-level web security policies. IEEE Trans. Knowl. Data Eng. **15**(4), 771–783 (2003)
58. Shabtai, A., Elovici, Y., Rokach, L.: A Survey of Data Leakage Detection and Prevention Solutions. Springer, Boston (2012). https://doi.org/10.1007/978-1-4614-2053-8
59. Shahriar, H., Hossain, S., Sarah, N., Wei-Chuen, C., Edward, M.: Design and development of Anti-XSS proxy. In: 8th International Conference for Internet Technology and Secured Transactions (ICITST 2013) (2013)
60. Shahriar, H., Zulkernine, M.: Information-theoretic detection of SQL injection attacks. In: 2012 IEEE 14th International Symposium on High-Assurance Systems Engineering (HASE), pp. 40–47 (2012)
61. Swamy, N., et al.: Gradual typing embedded securely in javascript. In: Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2014, New York, NY, USA, pp. 425–437. ACM (2014). https://doi.org/10.1145/2535838.2535889. http://doi.acm.org/10.1145/2535838.2535889
62. Taly, A., Erlingsson, U., Mitchell, J.C., Miller, M.S., Nagra, J.: Automated analysis of security-critical javascript APIs. In: 2011 IEEE Symposium on Security and Privacy, pp. 363–378, May 2011. https://doi.org/10.1109/SP.2011.39

63. Wurzinger, P., Platzer, C., Ludl, C., Kirda, E., Kruegel, C.: SWAP: Mitigating XSS attacks using a reverse proxy. In: 2009 ICSE Workshop on Software Engineering for Secure Systems, SESS 2009, pp. 33–39. IEEE (2009)
64. Zeller, W., Felten, E.W.: Cross-site request forgeries: Exploitation and prevention. NY Times, pp. 1–13 (2008)