Design Challenges and Assessment of Modern Web Applications Intrusion Detection and Prevention Systems (IDPS)

Yassine Sadqi(✉) and Manal Mekkaoui

Laboratory LIMATI, FPBM, USMS University, Beni Mellal, Morocco
yassine.sadqi@gmail.com, manalmekkaoui1@gmail.com

Abstract. Intrusion detection and prevention systems (IDPS) are primarily designed to observe, detect and prevent malicious activity on the network. However, the characteristics of traditional network attacks are very different from those of web attacks. The first targets the network layer while the second focuses on the weaknesses of the application layer of the TCP/IP stack. The aim of this paper is to present the essential information on IDPS exclusively proposed for web applications in order to contribute to the design of secure and efficient IDPS. To do this, first, we present a comprehensive study of intrusion detection and prevention systems. Second, we identify several specific challenges that make it difficult for an IDPS to monitor and detect web attacks. Finally, we evaluate four of the most deployed open-source IDPS, namely AppSensor, ModSecurity, Shadow Daemon, and AQTRONIX WebKnight. The assessment is based on security features that a web IDPS must incorporate in order to surpass the identified IDPS challenges. The results show that none of the evaluated IDPS integrates all the required security features.

Keywords: Intrusion detection and prevention · IDPS · Web applications security · Web attacks · IDS · IPS

1 Introduction

The use of web-based applications has experienced phenomenal growth over the last two decades [1–3]. Applications such as online banking, e-commerce, online blogs and social networking sites have become a common platform for the transmission of information and the provision of services online. Since these applications deal with sensitive data and operations, they are an easy, lucrative and potential target for attackers. For instance, acquisition of confidential users' data, financial gain, and the performance of several illegal activities. According to Symantec Internet Security Threat Report (ISTR) [4], over 76% of scanned web applications were rated vulnerable, and there was a 62% increase in malicious botnets targeting web applications.

In this context, the worldwide revenue for Corporate Web Security solutions is expected to grow from nearly $3.7 billion in 2019, to an estimated $6.1 billion by 2023 [5]. In fact, the long-term security goal of web applications is to maintain the trust

M. Ben Ahmed et al. (Eds.): SCA 2020, LNNS 183, pp. 1087–1104, 2021.
https://doi.org/10.1007/978-3-030-66840-2_83

of users. Thus, the security policy for web applications must therefore be defined to guarantee security objectives such as [6]:

- Confidentiality: which ensures that only authorized users have access to information and exchanged resources intended for them.
- Integrity: which determines the absence of inappropriate alterations of information to ensure the accuracy of information, not modified by unauthorized third parties.

In order to meet these security objectives, several solutions have been proposed [1, 6, 7] to ensure adequate security for web applications.

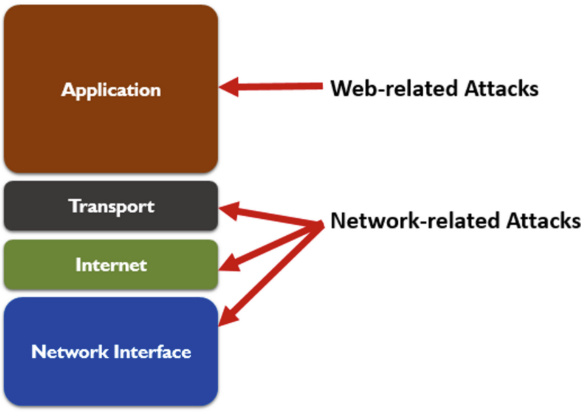

Fig. 1. The logical mapping between TCP/IP stack and cyber-security attacks. Web-related attacks target the application layer and network-related attacks focus on the other layers.

Among recent protection solutions, the use of intrusion detection systems (IDS) and intrusion prevention systems (IPS) for modern web applications [1, 8]. An IDS (Intrusion Detection System) is a defense tool used to detect and report intrusions to the administrator. An IPS (Intrusion Prevention System) is just an extension of an IDS capable of responding to attacks [9]. IDS and IPS are primarily designed to observe, detect, and prevent malicious activity on the network. However, the characteristics of traditional network attacks are very different from those of web attacks [1, 2, 10]. The first targets the network layer while the second focuses on the weaknesses of the application layer (Fig. 1).

In addition, modern web applications architecture and its running environments are complex (Fig. 2), managed by cross-platform databases, and typically created by developers with limited computer security skills [6, 10, 11]. Therefore, designing an IDPS to recognize and prevent suspicious activity on a web application requires a very different approach to that of an IDPS designed to monitor network traffic.

To help the cybersecurity community in building secure and efficient IDPS for moderns web applications, this paper aims to:

1. overview the core concepts of intrusion detection and prevention;

2. present the design challenges of an IDPS specifically proposed for the web;
3. evaluate certain open-source IDPS based on criteria related to the current context of modern web applications.

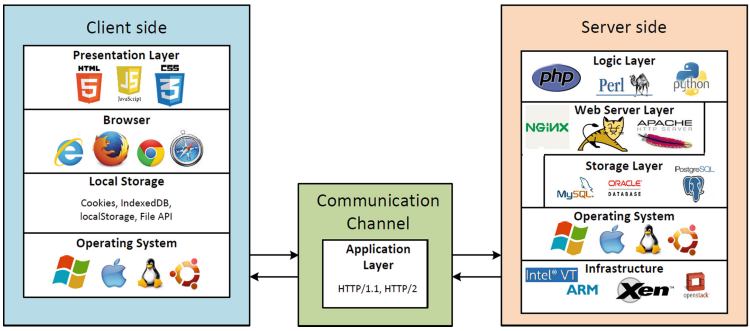

Fig. 2. An Abstract model of modern Web application architecture and its running Environments [12]

The rest of the paper is structured as follows. Section 2 overviews the core concepts of intrusion detection and prevention research area. In Sect. 3, we identify and discuss several specific challenges that make it difficult for an IDPS to monitor, detect, and prevent web attacks. Section 4 evaluates four of the most deployed open-source IDPS, namely AppSensor, ModSecurity, Shadow Daemon, and AQTRONIX WebKnight. The assessment is based on security features that a web IDPS must incorporate in order to surpass the identified IDPS challenges. Finally, Sect. 5 concludes the paper.

We should note that in the literature the terms "ID/IP systems" [13], "IDP" [14, 15], and "IDPS" [9, 16–19] can be used interchangeably to refer to intrusion detection and prevention systems. In addition, many vendors and web security experts consider Web Application Firewalls (WAFs) a special case of IDPS that work at the TCP/IP application layer [20, 21]. In this paper, we use IDPS to designate intrusion detection and prevention systems in the context of modern web applications.

2 An Overview of Intrusion Detection and Prevention

2.1 Intrusion Detection and Prevention

Intrusion detection was introduced in 1980 by J.P. Anderson [22] with the aim of identifying the use of a computer system for unauthorized purposes and detecting possible breaches of a system's security policy. Anderson defines an intrusion as a violation of the system's security policy, that is, a violation of one of the confidentiality, integrity or availability properties of the system. The purpose of intrusion detection is to report intrusions to a computer system's security administrator, so that they can take appropriate action. IDPS are required to follow two criteria [9]:

1. The reliability of the IDPS: any intrusion must electively give rise to an alert. An unreported intrusion constitutes a failure of the IDS (false negative).
2. The relevance of alerts: We can consider four alerts types listed in Table 1. Any alert must correspond to an effective intrusion. Any false alarm (false positive) decreases the relevance of the IDS. A good IDPS should have as low a number of false positives as possible.

Table 1. Types of IDPS alerts

	No alert	Alert
Normal activity	True Negative (TN)	False Positive (FP)
Attack	False Negative (FN)	True Positive (TP)

Intrusion detection research also includes the notion of automatic response to intrusions; in addition to warning the security administrator, the intrusion detection system takes measures to block the intrusion using an intrusion prevention system (IPS) [9]. An IPS is an extension of an IDS with all of the functionality of the latter, but it is also capable of responding automatically to attacks. For this, IPSs use several response techniques, which can be divided into the following groups [18]:

- The IPS stops automatically the attack without the network admin intervention. For example, the IDPS terminates the network connection or the user session used for the attack.
- The IPS changes the security environment: IDPS could change configuration of other security controls to disrupt an attack. Some IDPS can even cause a host to be patched if it detects that the host has vulnerabilities.
- The IPS modifies the attack content: Some IDPS technologies can remove or replace malicious parts of an attack to make it harmless

2.2 Detection Methods in IDPS

In general, there are three main approaches to intrusion detection [7, 9, 23, 24]: signature-based (known also as misuse detection), anomaly-based detection, and hybrid detection. Each detection approach operates on a specific set of principles. Table 2 gives a summary about the advantages and disadvantages of each detection methodology.

1. *Signature-based detection:* Signature-based intrusion method base their detection on the recognition, in the flow of events generated by one or more probes, of attack signatures that are contained in a signature database. Thus, it uses a set of signatures representing patterns of attacks already known to filter malicious activity [23]. A signature-based intrusion detection system consists of [18]: (1) one or more probes,

generating a flow of events, which can be network or host type, (2) a signature database, and (3) a pattern recognition system in the flow of events.
Anomaly-based detection: In the literature, they are so many definition of anomaly based on its problem domain [25, 26] (e.g., network intrusion detection, fraud detection, Natural Language Processing, Image Processing, etc.). One of the most acceptable definition is "An anomaly is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism [27]". Anomaly detection was first presented in 1987 by Denning in her seminal work on the host-based IDES system [28]. The problem of anomaly detection can be

Table 2. Comparison between IDPS detections methods

Detection method	Advantages	Disadvantages
Signature-based	– Easy to install and deploy – Associated with specific attacks – Describe attacks that may or may not succeed – Very effective in detecting attacks without producing a large number of false alarms (false positive) – In case of alerts, it is easy to identify exactly the attack	– Detect only known attacks – Generalization of signatures can lead to a decrease in the relevance of the system – The exploitation of new vulnerabilities between the time of their discovery and their actual description as signatures – Cannot keep up with the daily disclosure of web-related vulnerabilities
Anomaly-based	– Detection of unknown intrusion is possible – Do not seek to characterize the intrusions but the expected behavior – The system can gradually learn intrusive behaviors introduced by an attacker – The ability to detect symptoms of known and unknown attacks without specific knowledge of the details	– Notoriously prone to both false positives and false negatives – The learning dataset must be free of intrusions. Otherwise, the system would risk learning intrusive behaviors – A malicious user can slowly change their behavior in order to get used to the system – Lack of information on detected attacks
Hybrid-based	– Better detection accuracy – Offers better performance using the strength of multiple approaches	– Hybrid IDSs can have a layered or parallel architecture, but opting for one is a prerequisite – In a layered architecture, deciding the correct sequence of multiple components for processing events is a big challenge – Classification conflicts, since one method may classify an event as a normal activity and the other may declare the same as an intrusion

modeled as a classification problem [29]. Anomaly-based IDSs refers to systems that are able to model the network and/or system normal behavior (legitimate actions), and after that identify outlier or deviation of a normal behavior to detect attacks (malicious actions). In particular, an anomaly-based approach is used to recognize unknown attacks (called also zero-day attacks). Due the complexity of today technologies, it is very hard to define precisely the normal behavior, which lead to high false-positive rates [30, 31]. In the literature, they are very rare purely anomaly-based IDSs, because most systems rely on hybrid-based approaches [30].

3. *Hybrid-based detection*: A hybrid system is the fusion of different intrusion detection approaches into a single integrated detection system. Hybrid systems provide better performance by using the strength of several approaches to overcome the limitations of individual techniques. Hybrid-based detection was first explored in [32]. This method tries to increase detection accuracy and in the same time make detection more efficient.

2.3 Basic Architecture of an IDPS

As illustrated in Fig. 3, a typical IDPS is composed of four components [33]:

1. *Event generators (E-Box):* This type of block is made up of sensor elements that monitor the target system, thus capturing events and information to be analyzed by other blocks. For example, the decoding process would be included in this phase.

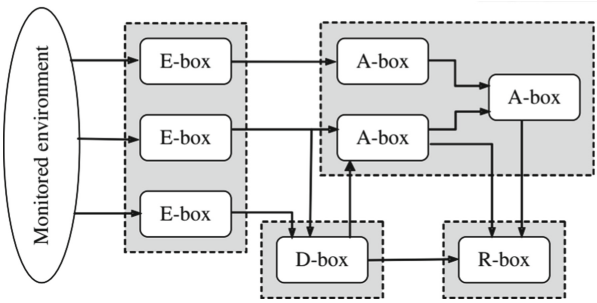

Fig. 3. Typical architecture for IDPS systems.

2. *Event analyzers (A-Box)*: Processing modules for event analysis and detection of potentially hostile behavior, so that a certain type of alarm is generated if necessary. In anomaly-based IDPS, this component should include two sub-components:

   a. *Preprocessor* (preprocessing phase): performs all actions before the classification of the incoming requests, such as dataset creation, data cleaning, normalization,

parsing the requests, feature extraction and feature selection. This sub-phase is very critical in anomaly-based IDPS [34]

b. *detection engine* (detection phase): An engine that analyzes the requests searching for intrusions

3. *Event databases (D-Box)*: These are elements intended to store information from blocks E for further processing by boxes A and R.
4. *Response units (R-Box)*: The main function of this type of block is the execution, if any intrusion occurs, of a response to thwart the detected threat. The action depends on the type of system. IDS raise an alarm when intrusions are found, while IPSs block the requests to avoid them reaching the target server.

3 Web IDPS Design Challenges

Intrusion detection and prevention is still immature in the field of web application security [1, 8]. Intrusion detection and prevention systems are primarily used as a network security appliance. However, the design of Web IDPS requires a different approach than the traditional network IDPS to manage the complexities associated with modern Web applications [1]. In our study, we identified several specific issues that make it difficult for an IDPS to monitor and detect web-related security issues. In this section, we present what we truly believe the main design challenges of IDPS in the context of modern web apps.

3.1 Web-Related Security Issues

In general, web specific security problems are very different from traditional network-related attacks. In fact, a web security threat or issue is defined as a potential malicious activity that exclusively targets one or multiple components of web application's architecture such as the user's browser or the web app hosting server [2]. Li and Xue [6] classified web security issues into three core web-specific vulnerabilities:

1. **Input validation or injection vulnerabilities:** Validating inputs data (for example, data entered by a user into an authentication form) is an important part of a security policy for a web application. For this reason, web developers use validation and sanitization routines to identify, clean up unreliable user input, and let pass only secure data by filtering or avoiding suspicious characters. Incorrect or insufficient input validation could cause various injection attacks, such as SQL injection, XSS attacks, code injection attacks or memory buffer overflow attacks. This makes it possible to alter program executions, make unexpected commands or obtain unauthorized access to resources and sensitive data of users of the application. For example, authentication interfaces where the user can enter malicious code and access the customer area, without a reliable verification of their authentication data.
2. **Business logic vulnerabilities:** Business logic designates the set of rules and algorithms implemented in an application to manage the flow exchange between the web application web browser and the back-end servers (e.g. database server). Business

Logic or logic vulnerabilities allow the legitimate processing flow of an application to be used in a way that has negative consequences for the application, which lead to various attacks such as access control bypass and application flow bypass attacks.

3. **Session management vulnerabilities:** Modern web applications rely mainly on the HTTP protocol to send requests and receive resources from the web application's servers [35]. However, HTTP is a stateless protocol; it treats each request as independent of all the others, this design is not suitable for modern web applications (for example, e-commerce applications and online banking), which require a mechanism above the HTTP protocol to manage user sessions. The exchange of information between the user and the webserver is done by the creation of a web session by the server; it is a sequence of the HTTP request and response transactions associated with the same user. Designing and implementing a secure and efficient web session management is a complex task, which usually leads to several security attacks (e.g. session hijacking attacks, session fixation, and CSRF) that compromise web applications security [36].

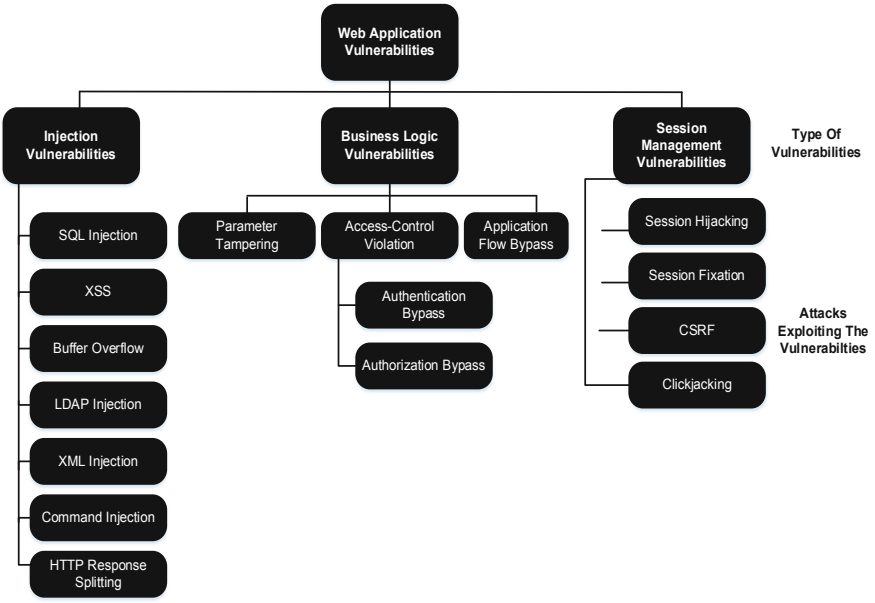

Fig. 4. Deepa and Thilagam [2] web apps security issues classification.

Deepa and Thilagam [2] extended Li and Xue [6] taxonomy by introducing attacks that exploit each vulnerability category. Figure 4 illustrates the scholars' classification.

3.2 Placement of the IDPS

According to our study, an IDPS for web apps has four possible deployment approaches.

1. *Database-side*: The database is a sensitive component in the architecture of web applications (Fig. 2). Multiple attacks are usually made possible by the fact that a single back-end database is used to store all of a web application's persistent information. Therefore, if malicious queries are allowed to access data stored in the main database or by exploiting a code vulnerability intended to gain access to a limited portion of the database content, it is possible to extend the access to the database and retrieve sensitive information. In [10], authors have proposed anomaly-based IDS in the database-side. This system is able to detect malicious SQL query and withstand SQL injection attacks. An overview of this the database-side anomaly detector is illustrated in Fig. 5.

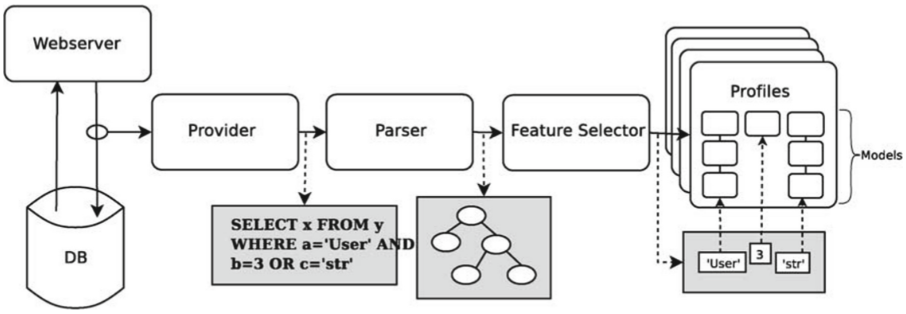

Fig. 5. Architecture of the database anomaly detector proposed by Vigna et al. [10].

2. *Web server-side*: To mitigate the security risks associated with web servers, IDPS are deployed to analyze and filter incoming requests. The goal is to quickly detect malicious activity and potentially prevent more serious damage. G.Vigna et al. presented in [37] WebSTAT, an intrusion detection system, which analyzes web requests for evidence of malicious behavior. This system is based on signature-based detection methodology. It is capable of detecting and describing attack scenarios, as well as allowing the detection of variants of attacks similar to the malicious behavior already specified. Another signature-based IDS presented by M.Almgren et al. in [38]. The proposed system is able to analyze log entries to recognize malicious activity on the web server and includes mechanisms to reduce the number of false alarms.

3. IDS/IPS as a proxy: It is also possible to deploy an IDS or and IPS as a proxy that intercepts incoming requests and outgoing responses from client and web servers, respectively. This placement option is also known as standalone appliance. In this placement, all traffic will pass from this point and be analyzed. The authors in [39] have presented an intrusion detection system that relies on anomaly-based detection to identify attacks against web applications. The system analyzes client requests referencing programs at the server level and creates models for a wide range of functionalities related to those requests. Another proxy IDPS proposed by N. Agarwal

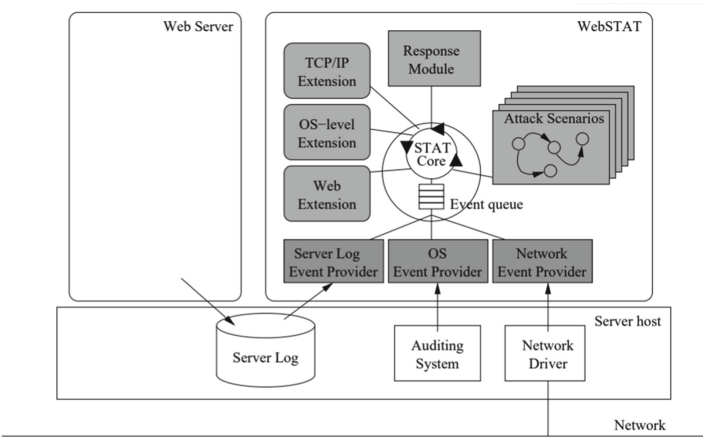

Fig. 6. WebSTAT architecture [37]

and S.Z. Hussain in [8]. It is a web intrusion detection system with a prevention mechanism that acts as a reverse proxy that intercepts incoming requests and outgoing responses from client and server.

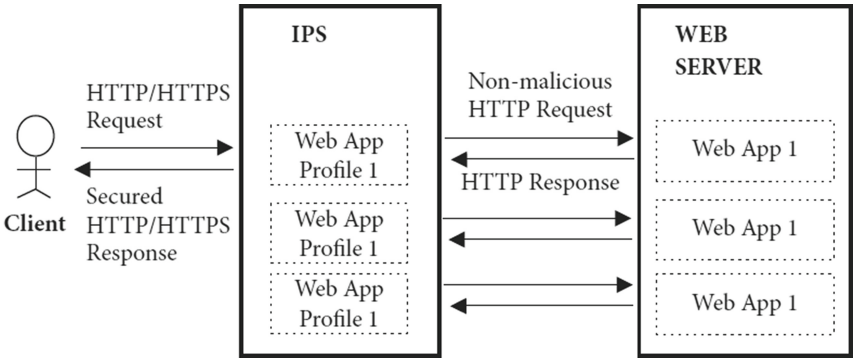

Fig. 7. Block diagram of the IDPS proposed by N. Agarwal and S.Z. Hussain [8].

4. *Browser-side*: A browser-side approach is considered to be one of the latest approaches or trends in IDPS deployment [40]. The idea here is to extend browsers with intrusion detection and prevention functionalities. This has an advantage for web applications, which will be automatically protected against a large number of browser-side bugs and vulnerabilities. In [40], the authors have introduced WPSE (Web Protocol Security Enforcer), a browser-side attack detection and prevention system that addresses the unique challenges of web protocols such as OAuth2.0 and

SAML. The current prototype of this solution is implemented as an extension for Google Chrome.

Table 3 gives an overview of some IDS/IPS offered in the literature and their placement. After all, the question then arises as to whether there is an ideal approach that should be privileged when deploying an IDPS for web apps. Unfortunately, each one of the above deployment strategies is vulnerable to specific attacks [3]. Moreover, it is very difficult to build an IDPS that will withstand all types of web attacks, without affecting the system efficiency. Therefore, we believe that the ideal solution would be to design an IDPS based on a distributed approach deployed across all the four basic placements.

Table 3. Overview of some IDS/IPS designed for web applications

Paper	Authors	Contribution	IDS/IPS Placement
[38]	Almgren et al.	A real-time intrusion-detection system to withstand web server attacks	Web server-side
[10]	Vigna et al.	An anomaly-based intrusion detection system for identifying SQL injection attacks	Database-side
[37]	Vigna et al.	A stateful signature-based IDS based on the STAT framework	Web server-side
[39]	Kruegel et al.	An anomaly-based IDS, which analyzes web client queries that reference server-side programs and creates models for a wide-range of different features of these queries	Proxy
[8]	Agarwal and Hussain	A conceptual framework for an ideal IDPS for web applications	Proxy
[40]	Calzavara et al.	A signature-based IDPS that monitor the browser-side functionalities to detect web protocols attacks	Browser-side

3.3 Communication Protocol (HTTP/HTTPS)

Attackers exclusively use HTTP / HTTPS protocols to exploit vulnerabilities in web applications. Hypertext Transfer Protocol (HTTP) is a request-response protocol designed to facilitate communication between client and server, and HTTPS provides secure and encrypted connection. When HTTPS traffic is observed from an IDPS perspective, the packet data exists in the encrypted form that the system fails to inspect. However, these systems can verify HTTPS traffic if they have access to the private key of the SSL certificate. However, due to security considerations, it is so inadvisable to share the private key of the web server.

Additionally, HTTP/HTTPS requests and responses carry a variety of values, and choosing the appropriate validation approach (whitelist or blacklist) depends largely on the type of value defined. The positive validation approach (whitelist) defines the data expected by the application. It includes several types of validation checks, such as data type (string, integer), minimum and maximum length, and specific patterns. In contrast, the negative (blacklist) approach involves filtering out the values containing the attack patterns. Signature-based systems include both positive (whitelist) and negative (blacklist) validation, while anomaly-based systems only deal with positive validations.

3.4 Users and Session Management

Web applications are generally accessible to several users with different privilege levels [36]. These privileges are controlled by the authorization process, which ensures that the user performs only authorized operations. Applications use a session management mechanism [35] to track individual client-server interaction and map the request to a particular user in order to decide whether the request should be processed or denied. Allowing different users with a different set of privileges places several demands on the IDPS:

- the functionality to track user sessions to link user's requests to the specified session;
- the ability to monitor the integrity of a session to ensure that the session is being used by the same person who logged in to the application;
- the ability to perform continuous monitoring of resource usage and user activity during a session.

3.5 Continuous Changes and Performance

Todays' web apps change pretty often. This rapid and continuous change over time are a major challenge for IDPS. IDPS should also be tuned and maintained over time to accommodate changes to an application. In the context of anomaly detection: frequent modifications handicap the system that is to say that the current model ignores the modified version of the application [41]. Apparently, this would incorrectly classify the new legitimate behavior as intrusive. These detectors need to be reconverted to accommodate the changes. In the signature-based IDPS: the blacklist is not affected too much, because the attack patterns remain the same, but the whitelist should be updated according to the change in application behavior [8].

In addition, the great complexity and interactivity of modern web apps profoundly affect the performance of intrusion detection systems [1]. It makes IDPS job more difficult, and the more interactivity, the harder it is for IDS to detect intruders.

3.6 Bots Requests

Web traffic can be generated by bots rather than humans. Bots are automated scripts designed to perform a specific set of activities on web applications. Automated attacks are comparatively cheaper than manual attacks because they allow adversaries to target

large numbers of web applications with less time and effort. For example, Angler [4] a sophisticated exploitation kit which was the source of many advanced attack techniques. Angler was able to download and run malware from memory, without having to write files to disk, to avoid detection by traditional security technology. Signature-based systems are not strong enough to recognize requests from scripts designed to mimic user activities, because malicious requests differ from legitimate requests by intent, but not by content, and the signature-based IDPS uses malicious content rather than intentional.

4 Assessment of Open-Source Web IDPS

The evaluation of intrusion detection and prevention systems is a very important concept in improving the security of computer systems [9, 24]. It helps provide essential data and conclusions to help developers improve their IDPS and let users know the capabilities and limitations of the IDPS system in use. In this section, we will evaluate some intrusion detection and prevention tools designed for the web. The assessment is based on various security features that must be incorporated in an IDPS. In particular, to overcome the IDPS design challenges discussed in the previous section, we argue that an IDPS for web applications is required to integrate the following functionalities:

- *Correct input validation*: The main problem with insecure web applications is that users have complete control over what data is submitted to the server. They can provide any arbitrary entry in the settings including header fields, or even change the values stored by the application on the client side in the form of hidden fields, cookies and URL parameters [42]. Incorrect validation or no input validation on the server side is the root cause of most vulnerabilities in web applications, including XSS, SQL Injection, and uncommitted redirects [2, 43]. The validation process checks whether the input meets a predefined set of rules to prevent insecure data from entering the application. It includes checking all input parameters, including URLs, form data, cookies, and query strings.
- *Output sanitization*: In general, output validation or sanitization protects a web application against unintentional disclosure of sensitive information. For example, the web application can also expose internal details if it does not handle error messages. Specifically, this attack technique takes advantage of overly descriptive error messages returned by the database when a query is rejected [1].
- *Session verification*: A large part of web attacks involve session hijacking (session hijacking), session fixation (session fixation) and session replay (session replay) [11, 35, 36]. When an attacker steals or overwrites a user's session ID to impersonate the valid user and perform operations on their behalf, the session verification performed by an IDPS checks if the session is used by the same user as the one who logged in to the application. It may also include detecting any attempt by an adversary to illegitimately obtain the session ID.
- *Access control*: Access control defines the policies to regulate the privileges granted to the user of the application. Some detection systems also provide the functionality of monitoring users' activities to prevent unauthorized access to information or services offered by the application [23]. Systems can also detect requests that attempt to

gain unauthorized access to objects (such as files and directories) that are mistakenly exposed via URL or a form.
- *Web bots detection*: In order to resist automated attacks, an IDPS must also be able to differentiate the requests of normal users from the requests generated by certain malicious bots [44]. More specifically, an IDPS should have the potential to distinguish between good and bad bots. Non-malicious bots are automated programs that are beneficial to the service provider, such as search engine bots that help improve the ranking of a web application.
- *Response-time*: The required time of an IDPS to deal with a potential threat is a very critical issue. For instance, it was demonstrated that less than fifteen minutes some sophisticated attacks were able to stop a large area of the Internet from normal functioning [45]. Thus, online detection also known as real-time detection is considered to be more efficient and secure than off-line detection [9, 46, 47].

Using these above critical functionalities, we assess some of the most used open-source IDPS exclusively designed for securing web applications. The evaluated IDPS are: OWASP AppSensor [48], ModSecurity [49], Shadow Daemon [50], and AQTRONIX WebKnight [51]. OWASP AppSensor [48] is a conceptual framework that offers prescriptive guidance to implement application intrusion detection and automated response in real-time. OWASP offers a reference implementation of this framework in a Java [19].

Since modern web application firewalls (WAFs) also offer security services similar to IDPS designed for web applications [1], we will also evaluate three commonly used open source WAF (Web Application Firewalls), namely ModSecurity a signature detection based web application firewall, Shadow Daemon [43] free software that intercepts requests and filters malicious settings and AQTRONIX WebKnight [44].

The results of the assessment are shown in Table 4. Three of the four evaluated IDPS rely on signature-based detection methodology. This demonstrate that like in the network security context, signature-based approach is widely deployed in web applications [52]. As discussed in Sect. 2, this detection methodology has several advantages such as easy installation and deployment. Real-time response is integrated in all the evaluated IDPS. For the session verification criterion only ModSecurity which contains a function to fix insecure session cookies. AQTRONIX WebKnight is a good example of an IDPS integrating access control and bot detection, as it can monitor access to certain important files or limit the number of requests from a single IP address. In particular, AQTRONIX is the only IDPS that include web bots detection using four possible ways:

1. A large database to block known bad bots or any additional bots the administrator specifies.
2. Bad bot trap mechanism that enable blocking bots that are note included in (1).
3. Aggressive Bot Trap filter, which prevent bots that are requesting too many web pages in a short period of time,
4. WebKnight 2.5 and later supports URL rewriting to prevent certain robots or hackers from seeing the true contents of your robots.txt file.

Table 4. Assessment of some IDPS exclusively designed for web applications

Functionality	AppSensor	ModSecurity	Shadow Daemon	AQTRONIX
Detection Method	Hybrid-based	Signature-based	Signature-based	Signature-based
Input validation	Yes	Yes	Yes	Yes
Output sanitization	No	No	No	No
Session verification	No	Yes	No	No
Access control	Yes	No	No	Yes
Bots detection	No	No	No	Yes
Response-time	Real-time	Real-time	Real-time	Real-time
Placement	Server-side	Reverse-proxy, Server-side	Server-side	Server-side

5 Conclusion

The rapid evolution and advance in web technologies have made the structure and interaction between the web client-side and server-side components of modern web applications more and more complexes, which have led to several security issues. In this context, intrusion detection and prevention systems (IDPS) are among the recent security solutions to protect web applications. In fact, IDPS are primarily designed to observe, detect, and prevent malicious activity on the network. However, the characteristics of traditional network attacks are very different from those of web-related attacks. The first targets the TCP/IP network layer while the second focuses on the weaknesses of the application layer.

In this paper, we overviewed the core concepts in the area of intrusion detection and prevention. Next, we discussed IDPS main design challenges for moderns' web applications, which make it difficult for an IDPS to monitor, detect, and prevent web-related security attacks. Finally, we evaluated four of the open-source and most deployed IDPS exclusively proposed for web apps security.

References

1. 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
2. Deepa, G., Thilagam, P.S.: Securing web applications from injection and logic vulnerabilities: approaches and challenges. IST. **74**, 160–180 (2016). https://doi.org/10.1016/j.infsof.2016.02.005
3. Rodríguez, G.E., Torres, J.G., Flores, P., Benavides, D.E.: Cross-site scripting (XSS) attacks and mitigation: a survey. Comput. Netw. **166**, 106960 (2020). https://doi.org/10.1016/j.comnet.2019.106960
4. Symantec: 2019 Internet Security Threat Report (2019)

5. RADICATI GROUP: Corporate Web Security Market, 2019–2023. https://www.radicati.com/wp/wp-content/uploads/2019/01/Corporate_Web_Security_Market_2019-2023_Executive_Summary.pdf
6. Li, X., Xue, Y.: A survey on server-side approaches to securing web applications. ACM Comput. Surv. **46**, 54:1–54:29 (2014). https://doi.org/10.1145/2541315
7. Iqbal, S., Kiah, M.L.M., Dhaghighi, B., Hussain, M., Khan, S., Khan, M.K., Choo, K.-K.R.: On cloud security attacks: a taxonomy and intrusion detection and prevention as a service. J. Netw. Comput. Appl. **74**, 98–120 (2016). https://doi.org/10.1016/j.jnca.2016.08.016
8. Agarwal, N., Hussain, S.Z.: A closer look at intrusion detection system for web applications. Secur. Commun. Netw. **2018**, 1–27 (2018). https://doi.org/10.1155/2018/9601357
9. Guezzaz, A., Asimi, A., Asimi, Y., Tbatous, Z., Sadqi, Y.: A global intrusion detection system using PcapSockS sniffer and multilayer perceptron classifier. **21**, 438–450 (2019). https://doi.org/10.6633/IJNS.20190521(3).10
10. Vigna, G., Valeur, F., Balzarotti, D., Robertson, W., Kruegel, C., Kirda, E.: Reducing errors in the anomaly-based detection of web-based attacks through the combined analysis of web requests and SQL queries. J. Comput. Secur. **17**, 305–329 (2009). https://doi.org/10.3233/JCS-2009-0321
11. Calzavara, S., Focardi, R., Squarcina, M., Tempesta, M.: Surviving the web: a journey into web session security. ACM Comput. Surv. **50**, 13:1–13:34 (2017). https://doi.org/10.1145/3038923
12. Taguinod, M., Doupé, A., Zhao, Z., Ahn, G.-J.: Toward a moving target defense for web applications. In: 2015 IEEE International Conference on Information Reuse and Integration, pp. 510–517 (2015). https://doi.org/10.1109/IRI.2015.84
13. Chan, G.-Y., Lee, C.-S., Heng, S.-H.: Discovering fuzzy association rule patterns and increasing sensitivity analysis of XML-related attacks. J. Netw. Comput. Appl. **36**, 829–842 (2013). https://doi.org/10.1016/j.jnca.2012.11.006
14. Chan, G.-Y., Chua, F.-F., Lee, C.-S.: Intrusion detection and prevention of web service attacks for software as a service: fuzzy association rules vs fuzzy associative patterns. J. Intell. Fuzzy Syst. **31**, 749–764 (2016). https://doi.org/10.3233/JIFS-169007
15. Chan, G.-Y., Chuaa, F.-F., Leeb, C.-S.: Fuzzy association rules vs fuzzy associative patterns in defending against web service attacks. In: 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 524–529. IEEE, Zhangjiajie (2015). https://doi.org/10.1109/FSKD.2015.7381997
16. Guezzaz, A., Asimi, A., Sadqi, Y., Asimi, Y., Tbatou, Z.: A new hybrid network sniffer model based on Pcap language and sockets (Pcapsocks). ijacsa **7** (2016). https://doi.org/10.14569/IJACSA.2016.070228
17. Guezzaz, A., Asimi, A., Asimi, Y., Tbatou, Z., Sadqi, Y.: A lightweight neural classifier for intrusion detection. GLM 2 (2017). https://doi.org/10.31559/GLM2016.2.2.4
18. Scarfone, K., Mell, P.: Guide to intrusion detection and prevention systems (IDPS). Natl. Inst. Standards Technol. (2007). https://doi.org/10.6028/NIST.SP.800-94
19. Watson, C., Groves, D., Melton, J.: Application-Specific Real Time Attack Detection & Response Version 2.0. OWASP (2015)
20. Desmet, L., Piessens, F., Joosen, W., Verbaeten, P.: Bridging the gap between web application firewalls and web applications. In: Proceedings of the Fourth ACM Workshop on Formal Methods in Security, pp. 67–77. Association for Computing Machinery, Alexandria, Virginia, USA (2006). https://doi.org/10.1145/1180337.1180344
21. Garcia-Teodoro, P., Diaz-Verdejo, J.E., Tapiador, J.E., Salazar-Hernandez, R.: Automatic generation of HTTP intrusion signatures by selective identification of anomalies. Comput. Secur. **55**, 159–174 (2015). https://doi.org/10.1016/j.cose.2015.09.007
22. Anderson, J.P.: Computer security threat monitoring and surveillance. Technical Report, James P. Anderson Company (1980)

23. Peng, J., Choo, K.-K.R., Ashman, H.: User profiling in intrusion detection: a review. J. Netw. Comput. Appl. **72**, 14–27 (2016). https://doi.org/10.1016/j.jnca.2016.06.012
24. Zhang, J., Zulkernine, M., Haque, A.: Random-forests-based network intrusion detection systems. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **38**, 649–659 (2008). https://doi.org/10.1109/TSMCC.2008.923876
25. Ahmed, M., Naser Mahmood, A., Hu, J.: A survey of network anomaly detection techniques. J. Netw. Comput. Appl. **60**, 19–31 (2016). https://doi.org/10.1016/j.jnca.2015.11.016
26. Resende, P.A.A., Drummond, A.C.: A survey of random forest based methods for intrusion detection systems. ACM Comput. Surv. **51**, 48:1–48:36 (2018). https://doi.org/10.1145/3178582
27. Hawkins, D.M.: Identification of Outliers. Chapman and Hall (1980)
28. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **SE-13**, 222–232 (1987). https://doi.org/10.1109/TSE.1987.232894
29. Kakavand, M., Mustapha, A., Tan, Z., Yazdani, S.F., Arulsamy, L.: O-ADPI: online adaptive deep-packet inspector using Mahalanobis distance map for web service attacks classification. IEEE Access **7**, 167141–167156 (2019). https://doi.org/10.1109/ACCESS.2019.2953791
30. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutorials **18**, 1153–1176 (2016). https://doi.org/10.1109/COMST.2015.2494502
31. Sommer, R., Paxson, V.: Outside the closed world: on using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy, pp. 305–316 (2010). https://doi.org/10.1109/SP.2010.25
32. Depren, O., Topallar, M., Anarim, E., Ciliz, M.K.: An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert Syst. Appl. **29**, 713–722 (2005). https://doi.org/10.1016/j.eswa.2005.05.002
33. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**, 18–28 (2009). https://doi.org/10.1016/j.cose.2008.08.003
34. Davis, J.J., Clark, A.J.: Data preprocessing for anomaly based network intrusion detection: a review. Comput. Secur. **30**, 353–375 (2011). https://doi.org/10.1016/j.cose.2011.05.008
35. Sadqi, Y., Asimi, A., Asimi, Y.: Short: a lightweight and secure session management protocol. In: International Conference on Networked Systems, pp. 319–323. Springer, Heidelberg (2014)
36. Sadqi, Y., Asimi, A., Asimi, Y.: A secure and efficient user authentication scheme for the web. Int. J. Internet Technol. Secured Trans. **6**, 43–63 (2015)
37. Vigna, G., Robertson, W., Vishal Kher, Kemmerer, R.A.: A stateful intrusion detection system for World-Wide Web servers. In: 19th Annual Computer Security Applications Conference Proceedings, pp. 34–43 (2003). https://doi.org/10.1109/CSAC.2003.1254308
38. Almgren, M., Debar, H., Dacier, M.: A lightweight tool for detecting web server attacks. In: NDSS Symposium 2000. Internet Society, San Diego, California (2000)
39. Kruegel, C., Vigna, G., Robertson, W.: A multi-model approach to the detection of web-based attacks. Comput. Netw. **48**, 717–738 (2005). https://doi.org/10.1016/j.comnet.2005.01.009
40. Calzavara, S., Focardi, R., Maffei, M., Schneidewind, C., Squarcina, M., Tempesta, M.: WPSE: fortifying web protocols via browser-side security monitoring. In: 27th USENIX Security Symposium USENIX Security 18, pp. 1493–1510 (2018)
41. Maggi, F., Robertson, W., Kruegel, C., Vigna, G.: Protecting a moving target: addressing web application concept drift. In: Kirda, E., Jha, S., Balzarotti, D. (eds.) Recent Advances in Intrusion Detection, pp. 21–40. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04342-0_2

42. Scholte, T., Robertson, W., Balzarotti, D., Kirda, E.: Preventing input validation vulnerabilities in web applications through automated type analysis. In: 2012 IEEE 36th Annual Computer Software and Applications Conference, pp. 233–243. IEEE (2012)
43. Scholte, T., Balzarotti, D., Kirda, E.: Have things changed now? An empirical study on input validation vulnerabilities in web applications. Comput. Secur. **31**, 344–356 (2012). https://doi.org/10.1016/j.cose.2011.12.013
44. Jonker, H., Krumnow, B., Vlot, G.: Fingerprint surface-based detection of web bot detectors. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) Computer Security – ESORICS 2019, pp. 586–605. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29962-0_28
45. Herrero, Á., Navarro, M., Corchado, E., Julián, V.: RT-MOVICAB-IDS: addressing real-time intrusion detection. Future Gen. Comput. Syst. **29**, 250–261 (2013). https://doi.org/10.1016/j.future.2010.12.017
46. Rathore, M.M., Ahmad, A., Paul, A.: Real time intrusion detection system for ultra-high-speed big data environments. J. Supercomput. **72**, 3489–3510 (2016). https://doi.org/10.1007/s11227-015-1615-5
47. Carrascosa, C., Bajo, J., Julian, V., Corchado, J.M., Botti, V.: Hybrid multi-agent architecture as a real-time problem-solving model. Expert Syst. Appl. **34**, 2–17 (2008). https://doi.org/10.1016/j.eswa.2006.08.031
48. OWASP: AppSensor - Application Intrusion Detection and Response. https://www.appsensor.org/. Accessed 20 July 2020
49. Apache: ModSecurity: Open Source Web Application Firewall. https://modsecurity.org/. Accessed 20 July 2020
50. Zecure: Shadow Daemon Open-Source Web Application Firewall. https://shadowd.zecure.org/overview/introduction/. Accessed 20 July 2020
51. AQTRONiX: AQTRONiX WebKnigh. https://www.aqtronix.com/?PageID=99. Accessed 20 June 2020
52. Schmitt, I., Schinzel, S.: WAFFle: Fingerprinting filter rules of web application firewalls. In: 6th USENIX Workshop on Offensive Technologies, pp. 34–40 (2012)