



# USE OF “OWASP TOP 10” IN WEB APPLICATION SECURITY

Nikola Nedeljković<sup>1</sup>   
Natalija Vugdelija<sup>2</sup>   
Nenad Kojić<sup>3</sup> 

DOI: <https://doi.org/10.31410/ITEMA.2020.25>

---

**Abstract:** *Web application security vulnerabilities can lead to various attacks on users, some of which can have major consequences. It is important to point out the weaknesses that allow abuse, because often increased risk awareness is the first step in protecting web applications. Some of the most critical security risks that organizations face today have been analyzed and uncovered using OWASP Top 10. This paper presents concrete examples of attacks and abuse of web applications. Through the implementation and analysis of attacks on web applications, weaknesses that need to be eliminated in order to protect against potential new attacks are identified. Especially, suggestions to help protect web applications from each type of attack listed and described are provided.*

**Keywords:** *Web security, Web attack, Weaknesses of the web application.*

---

## INTRODUCTION

Nowadays, with many jobs being done online to prevent the spread of Covid-19 virus infection, the importance of web application security has increased even further. Normally, a successful business requires some kind of interaction – with other users or with back-end databases, which could be susceptible to attacks. As stated by Alzahrani, Alqazzaz, Zhu, Fu and Almashfi (2017) “threats may compromise web applications’ security by breaching an enormous amount of information, which could lead to severe economic losses or cause damages” (p. 237). In addition to reliable transmission systems and secure communication, another matter of great importance is application-level web security, which, according to Scott, Sharp (2002) “refers to vulnerabilities inherent in the code of a web-application itself (irrespective of the technologies in which it is implemented or the security of the web-server / back-end database on which it is built)” (p. 396). The communication chain that connects end users is as secure as its weakest link. Therefore, it is important to identify any vulnerabilities that may lead to abuse during a multimedia session and to provide secure and reliable communication with appropriate security services and mechanisms. Connecting to a global, public network requires a global approach to using web applications safely, as is noted by Andrian, Fauzi (2019) “application security must be applied to all infrastructure that supports web applications, including the web application itself.” (p. 68). The greater the number of significant transactions performed over the Internet, the more motivated hackers are to carry out an attack. According to Rafique, Humayun, Gul, Abbas, Javed (2015) “hackers in recent years are increasingly targeting web applications, since most networks are closely

---

<sup>1</sup> Enreach Labs, Omladinskih brigada 90 V, Belgrade, Serbia

<sup>2</sup> Academy of Technical and Art Applied Studies Belgrade (ATUSS) – Department ICT College for vocational studies, Zdravka Čelara 16, Belgrade, Serbia

<sup>3</sup> Academy of Technical and Art Applied Studies Belgrade (ATUSS) – Department ICT College for vocational studies, Zdravka Čelara 16, Belgrade, Serbia

monitored through Intrusion Detection Systems (IDS) and firewalls” (p. 29). Web application developers have a very difficult task to create a quality web application that will be protected from malicious attacks, especially if we keep in mind that an army of hackers is working to detect the weaknesses of web applications and that they are very motivated because it often allows them significant financial gain. This is also indicated by Shahriar (2018) “Despite the awareness of web application developers about safe programming practices, there are still many aspects in web applications that can be exploited by an attacker” (p. 1). Additional problem is that there are many inexperienced developers who are not aware of the potential consequences of security vulnerabilities in a web application.

## USING OWASP TOP 10

The Open Web Application Security Project (OWASP) is a nonprofit community of software developers, engineers, and freelancers that provides resources and tools for web application security. Every few years, OWASP publishes a report of the top 10 security risks to web applications. The top 10 risks were first published in 2003 and since then the report has been constantly updated and published every 3-4 years. Current version is from 2017. and can be seen on the OWASP site. Many standards, books and tools list OWASP Top 10 as one of the best resources in the field of web application security, which also notify Rafique, Humayun, Gul, Abbas, Javed (2015) “OWASP Top 10 is aimed at analyzing the security of a system by identifying the vulnerabilities of web applications.” (p. 37).

Information technologies are evolving very fast and we are all witnessing constant changes. The development of web applications is also very intensive and one might think that the list of the biggest risks is changing in the same way, but that is not the case. Most of the problems in the “OWASP Top Ten 2017” are the same, or very similar, as in the first list. The web has advanced significantly, but the security of web applications has lagged far behind that progress. As security risks recur, in order to better understand the security of web applications, certain attack methods are explained in detail.

### Listed are some of the OWASP Top Ten (2017) web application security risks

1. **A1-Injection:** Most web applications use basic systems or functions such as database or e-mail functionality. An application often uses user input to execute a command. If the application does not calculate the user input correctly, malicious code can be inserted, which would lead to the execution of unauthorized commands.
2. **A2-Broken Authentication:** In many cases, web applications contain functionality for user authentication and session management. These functions could be incorrectly implemented which allows attackers to access confidential information. In web applications, ‘logged in’ users are often identified using a session. The application sees if the user is logged in using that session, but it also sees malicious code.
3. **A3-Sensitive Data Exposure:** Leaks of confidential information can have major consequences. An example when not enough security measures are taken is sending a login form containing the username and password to the server via http. Some of the confidential information, in addition to the codes, are credit card details, email addresses and private data.
4. **A8-Insecure Deserialization:** The vulnerability applies to applications that use user-defined input as a serialized object without input validation. It can lead to remote code execution or to perform attack, including replay attacks, injection attacks, and privilege escalation attacks.

5. **A9-Using Components with Known Vulnerabilities:** Using CMS such as WordPress, Joomla! and Drupal often contain vulnerabilities that are known to the general public. Today, there are automatic scanners that use databases to scan a special platform.
6. **A10-Insufficient Logging & Monitoring:** Detecting an attack on a website is only possible if there is enough logging and monitoring of the system. Recording should be by levels of communication with the network, operating system and application.

## EXAMPLES OF POSSIBLE ATTACKS

According to the law, one cannot attack without the permission of the owner, and certain sanctions follow for such an act. Unfortunately, the legal system and supporting institutions are often unable to monitor the development of cybercrime successfully enough, so the perpetrators in many cases remain undetected or unpunished. This makes space for attacks by people who do not plan to commit a crime for personal gain but for various motives, who sometimes try to find and attack vulnerable systems and applications just for fun and cause great damage. Attackers intensively explore areas of vulnerability and attacks are increasingly targeted at web applications. Ethical hackers use their knowledge and skills to analyze and increase security. In order to better protect the system from future attacks, ethical hackers must think like real attackers and act like malicious users. Although ethical hackers may abuse their knowledge at some point, it is easy to distinguish ethical from malicious hackers; according to Engebretson (2010) „differences can be boiled down to three key points: authorization, motivation, and intent” (p. 3).

The following are examples of possible attacks that could seriously compromise the confidentiality, integrity, and availability of data and devices:

1. Example of an attack using social engineering and Cross-site scripting:
  - Step 1:** The attacker sends a special URL to his victim, the text of the message sent by the attacker can be for example: "Can you tell me exactly what THIS means", where THIS is a link to the attacker's website.
  - Step 2:** The victim clicks to open the given URL.
  - Step 3:** JavaScript sends a cookie to the attacker's web server.
  - Step 4:** The attacker saved the cookie to a text file.
  - Step 5:** The victim returns to the original site unaware that she has just given a session cookie to the attacker.
  - Step 6:** The attacker can log in to the site via a session cookie
2. Example of an attack using SQL injection: An attacker searches an application to find fields for user input. He checks whether this input is used for an SQL query, trying to provoke an error message by, for example, adding single quotes, which corrupts the SQL query. If an error occurs, the attacker knows it is vulnerable to SQL injection. Some of the attempts to obtain useful information are:
  - a. `1' union select user(), database() - a`
  - b. `1' union select table_schema, table_name from information_schema.tables where table_schema = '*****' -- a`
3. Example of an attack bypassing login and security barriers:
  - a. Sometimes it is enough for the attacker to, browsing a web application and navigating through the pages, discover access to one of the pages on which users normally need to be logged in to abuse it.
  - b. Also, if by browsing through the cookies the attacker finds a cookie of type `login = false`, which in many cases is a way for the application to differentiate

between logged in and unregistered users, it is enough for the attacker to change the cookie value to *login = true*, or in some cases *login = (username )* to access the application as a logged in user.

4. Example of an attack using insecurities in a CMS: A large number of CMS-s being used have a similar appearance and are very recognizable. Also, the name of the CMS used is written somewhere in the page. By viewing the application code, it is possible to determine exactly which one it is, if it is not directly indicated. An attacker can use some of the well-known CMS scanners – wpscan (Wordpress) and joomscan (Joomla) to find vulnerabilities in the version used by the user. This vulnerability can later be exploited to attack the system.

## METHODS OF DEFENSE

It is important to point out the weaknesses that allow abuse, because often increased risk awareness is the first step in safeguarding web applications. According to Parimi and Babu (2020) “Companies will be responsible for the personal data in future” (p. 924).

There are published lists of activities, like the one on the OWASP site, that can prevent attacks or at least reduce the possibility of abuse. According to Rafique, Humayun, Gul, Abbas, Javed (2015) “OWASP is major source to construct and validate web security processes and standards” (p. 28). The following section lists some of the ways to defend against malicious attacks that are not too demanding for users and that have proven to be very useful in securing web applications.

1. A1-Injection
  - a. Use verified server validations of user entries.
  - b. Use output characters for all dynamic arrays.
  - c. Use LIMIT and other SQL control commands to reduce the number of results obtained, thus making the outflow of information as small as possible in case of a security breach.
2. A2-Broken Authentication
  - a. Implement multifactor authentication.
  - b. Do not place an application with basic credentials.
  - c. Make sure that weak passwords are not being used.
  - d. Align passwords with standards for creating passwords.
  - e. Use the same error messages for all errors.
  - f. Limit the number of incorrect password attempts.
  - a. Use the server to generate a session ID
3. A3-Sensitive Data Exposure
  - a. Classify data and arrange them by sensitivity, and store and dispose of those with the highest sensitivity carefully.
  - b. Do not store sensitive data locally with the user.
  - c. Encrypt all sensitive data.
  - d. Use the latest strong encryption algorithms.
  - e. Encrypt all data in transit, not only sensitive data.
  - f. Disable caching for response that contain sensitive data.
  - g. Verify independently the effectiveness of the server and system configuration.
4. A8-Insecure Deserialization
  - a. Implement integrity checks such as digital signature.
  - b. Enforcing strict type constraints during deserialization before object creation.

- c. Isolate code that deserializes.
- d. Log deserialization exceptions and failures.
- 5. A9-Using Components with Known Vulnerabilities
  - a. Remove unused library and file.
  - b. Continuously inventory the versions of both client-side and server-side components and their dependencies using tools.
  - c. Only obtain components from official sources over secure links.
  - d. Monitor for libraries and components that are unmaintained or do not create security patches for older versions.
- 6. A10-Insufficient Logging & Monitoring
  - a. Ensure all login, access control failures, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts.
  - b. Ensure that logs are generated in a format that can be easily consumed by a centralized log management solution.
  - c. Ensure transactions have an audit trail with integrity controls.
  - d. Establish effective monitoring and alerting such that suspicious activities are detected.
  - e. Establish or adopt an incident response and recovery plan.

## **FUTURE RESEARCH DIRECTIONS**

Web applications are being more and more used, so there is a growing awareness of the importance of their security. More users will begin to see the need for a good security policy. A comprehensive analysis of potential risks is important for a good security mechanism. It would be useful to analyze other different types of attacks, such as brute force attacks or attacks using an insecure file upload. In future research, the impact of attacks on the rest of the risks from the OWASP Top Ten list should be paid attention to, as well as which of them are easier and which are more difficult to abuse.

## **CONCLUSION**

Through carrying out and analyzing attacks on web applications, it can be noticed that some weaknesses are easier to abuse than others, as well as that attacks can be more or less harmful to the application and its users, which is why it is necessary to develop an appropriate security policy. Using Components with Known Vulnerabilities is a flaw that can be exploited relatively easily, and depending on the type of weakness detected, the damage can be either minor or major. Some of the vulnerabilities that an attacker can most easily take advantage of are injection and corrupt authentication. By finding an opening for injection, an attacker can easily add some of the well-known codes, after which he would easily attack the application server itself. In contrast, the most difficult flaw for an attacker to abuse is insecure deserialization. It is often difficult to detect and exploit such a flaw, but it is the most harmful to the application and can have major consequences. Also, injection, corrupt authentication and detection of sensitive data are security vulnerabilities that lead to the greatest damage to a web application. These can lead to system crashes, the attacker getting admin rights, sensitive data, such as credit card data and personal data, being released to the public, and the like. The flaw from the list which causes the least damage per attack is insufficient monitoring, but with this flaw the future defense of the system is much more difficult, as is the elimination of the current attack.

## REFERENCES

- Alzahrani, A., Alqazzaz, A., Zhu, Y., Fu, H. and Almashfi, N. (2017). Web Application Security Tools Analysis, *IEEE 3rd international conference on big data security on cloud (bigdatasecurity)*, *IEEE international conference on high performance and smart computing (hpsc)*, and *IEEE international conference on intelligent data and security (ids)*, Beijing, 2017, (pp. 237-242), doi: 10.1109/BigDataSecurity.2017.47.
- Andrian, R., Fauzi, A. (2019). Security Scanner for Web Applications Case Study: Learning Management System *JOIN (Jurnal Online Informatika)* Volume 4 No. 2 | December 2019., 63-68
- Engelbreton, P. (2010) *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy*, Elsevier, USA: Syngress
- OWASP. (n.d.). Top 10-2017 Top 10. Available at: OWASP: [https://www.owasp.org/index.php/Top\\_10-2017\\_Top\\_10](https://www.owasp.org/index.php/Top_10-2017_Top_10)
- Parimi, M., R., Babu, S. (2020) Critical Analysis of Software Vulnerabilities through Data Analytics, *Proceedings of the International Conference on Industrial Engineering and Operations Management* Dubai, UAE, March 10-12, 2020, (pp. 923-934)
- Rafique, S., Humayun, M., Gul, Z., Abbas, A. and Javed, H. (2015) Systematic Review of Web Application Security Vulnerabilities Detection Methods. *Journal of Computer and Communications*, 3, 28-40. doi: 10.4236/jcc.2015.39004.
- Scott, D., Sharp, R. (2002) Abstracting application-level web security *WWW '02: Proceedings of the 11th international conference on World Wide Web* May 2002 (pp. 396–407) <https://doi.org/10.1145/511446.511498>
- Shahriar, H. (2018) Web Security Vulnerabilities: Challenges and Solutions *A Tutorial Proposal for ACM SAC 2018*,” (pp. 1–5),