# Vulnerability Assessment for Web Applications

Susana Paola Lainez Garcia
susana.lainezgarcia@utdallas.edu

Amy S. Abraham
amy.abraham@utdallas.edu

Kristina Kepic
Kristina.kepic@utdallas.edu

Ebru Celikel Cankaya
ebru.cankaya@utdallas.edu

Department of Computer Science
University of Texas at Dallas
Richardson 75080, TX, USA

## Abstract

Considering the everlasting need for security in network platforms, this study investigates various penetration testing tools in the abundance of options when it comes to network security. We present the experimental run results of select penetration testing tools on deliberately vulnerable network traffic, as well as the comparison of those tools. We test three vulnerability assessment tools: ZAP, Vega and Arachni as part of this research in the hope to provide current and practical data for the research community in the field. The selection criteria we adopted were being current, ease of use, reliability (stability), and speed performance. Our results demonstrate that each vulnerability assessment tool depicts its own advantages and disadvantages by being better at one or more criteria than the others, but not prevailing in all.

**Keywords:** Vulnerability Assessment, Penetration Testing, Web Applications, SQL Injection, Cross-Site Scripting (XSS).

### 1. INTRODUCTION

In today's society, many companies are faced with security threats, particularly through the use of third-party web-applications. As a result, it is imperative for them to find the security vulnerabilities in their systems that can be exploited by black-hat hackers in order to determine whether such applications can be used by their employees. We propose the use of security penetration tools, specifically open-source security testing tools for web applications. There is a wide array of tools available on the market. Ultimately, we selected tools that were ubiquitous, free, open-source, and easy to use as this can be particularly beneficial for smaller companies, which are quite abundant. So, we ultimately decided to analyze and compare the tools ZAP (Owasp Zap”), Vega (Vega Vulnerability Scanner), and Arachni (Web application security scanner framework).

Our goal is to contribute to the field of data security and privacy with an in-depth analysis and comparison of free, open-source security testing

tools that can aid a company in making the decision for the tool that would work best for their specific cybersecurity needs. With ever-increasing interconnectivity through third-party web applications, this research provides the means to fill the gap of a need for companies to instill tougher security guidelines since those web applications can so easily be exploited by attackers.

## 2. BACKGROUND AND RELATED WORK

### Background

Penetration testing simulates an attack by an ethical hacker and is used for evaluating the security of a network or computer system. The ultimate goal of penetration testing is to increase the data security for a given party, whether that be an individual or a company. Penetration tests are typically done with a license and requires a signed contract with a company. The output of the penetration testing is provided as a report to disclose the weakness found in the system. It is very important for companies to have high data security since their data is one of their most valuable assets. In addition, with the rise of internet usage in various fields like medicine, finance, and the military, web security has become an increasing concern. As a result, penetration tools for web applications are especially critical when it comes to maintaining data security (Mirjalili, Nowroozi, & Alidoosti, 2013).

### Penetration Tools

This study explores three free, open-source penetration tools named Zap, Vega, and Arachni for vulnerability assessment purposes. Zap is a free, open-source, GUI-based application that can be used on Windows, Linux, and Mac ("Owasp zap"). It is officially called the Owasp Zap. Zap can be used to test the security of web applications with penetration testing. It can also be used to find security vulnerabilities such as Cross-Site Scripting, SQL injections, and information leaks. Zap has an automated scanner and monitors the responses to requests it sends to the web application to find vulnerabilities. The application was programmed using a combination of Java, JavaScript, HTML, Python, and PHP. We ran Zap on a Windows OS computer and supplied it with the Spotify Web Player and a deliberately vulnerable target website with the URL http://testphp.vulnweb.com.

Vega is also a free, open-source, GUI-based web security scanner and testing platform that tests web application security ("Vega Vulnerability Scanner"). Vega can find vulnerabilities such as Cross-Site Scripting (XSS), SQL injection, and inadvertently disclosed information. Vega has an automated scanner for fast tests and an intercepting proxy to observe the interaction between clients and servers. The operating system that we ran Vega on is Windows, but it can be run on Linux and MAC OS as well.

The last tool we examined was Arachni ("Web application security scanner framework"). In regard to the Operating System, we present results for the Arachni tool running on the Windows platform, but it can be run on Linux and Mac OS X as well. Arachni is an open-source, modular web application security scanner framework. It focuses on identifying, classifying, and logging vulnerabilities in web applications. It is licensed under the Arachni Public Source License v1.0. It is important to note that for commercialization a non-free license is required. It also requires 2GB of memory and 10 GB of available disk space. Furthermore, the distributed architecture of Arachni allows remote control of the scan. This is done by deploying agents on remote servers.

### Related Work

In Mirjalili et al. (2014), authors introduce various types of penetration tools based on the parameters as whether the tool is manual or automatic, and whether it offers black-box, white-box, or grey-box testing. Furthermore, this paper explains different web vulnerabilities like injection, broken authentication, session management, and cross-site scripting (XSS). The authors look specifically at black-box web vulnerability scanners, both open-source and commercial. Some of such tools are Websecurity, Wapiti, ZAP, and Acunetix. The paper compares whether those tools are GUI based or not, and rate how good their configuration, usability, stability, and performance are. Similar to this paper, we too compare different penetration tools for web applications. However, we focus more on comparing the results of three different tools: Vega, Arachni, and ZAP. Additionally, we analyze their vulnerability results and how they compare to one another.

In Fonseca, Vieira, & Madeira (2007), authors delineate how different penetration tools produce different results, which is also verified by our study. In addition, they also explain how several vulnerabilities were missed by some of the penetration tools they tested, and oftentimes they tended to have a significant rate of false positives.

Khera, Kumar, Sujay, & Garg (2019) discusses

and analyzes the VAPT (vulnerability assessment and penetration tools) and life cycle. Some of the assessment tools that this paper explores for network security include Wire shark, Nmap, and Metasploit. A case study is shown where Nmap is used to target the Metasploitable virtual machine, which is a vulnerable version of Linux Ubuntu designed for testing purposes. In addition, Khera et al. discuss both the advantages and disadvantages of VAPT as a cyber defense technology. In our study, we also discuss different vulnerability assessment tools, but focus on those targeting web applications, as opposed to network security like Khera et al. In addition, we too have a case study where we use a vulnerable web application, but instead of testing it using a single tool such as Nmap, we test three tools Vega, ZAP, and Arachni and compare the results.

Zakaria, Phin, Mohmad, Ismail, Kama, & Yusop (2019) explains how there is no standardized format of penetration testing reports. As a result, they analyze eight different penetration testing reports found online in order to compare their similarities and patterns to aid them in creating a standardized format of the report. This format focuses on catering to both security personnel and upper management of the organization. This lack of standardization is important to note since in our case study we witness three very different reports from three different tools for a single web application.

In Abu-Dabaseh & Alshammari (2018), the authors discuss the standards for penetration testing tools. A detailed comparison of automated versus manual penetration testing techniques is provided in this paper. The paper compares the current methodologies used to build an automated penetration testing system. These methodologies target HTTP/TCP/IP and SIP attacks, all protocols and services, and databases. The different tools, phases, methods of implementation, and aim of the methodologies are considered. Tools such as ZAP and Metasploit are referenced in this paper as well. Lastly, the importance of automating the process of penetration testing is discussed. Those presented in Abu-Dabaseh & Alshammari (2018) all contribute as a basis framework for our paper.

Nagpure & Kurkure (2017) discusses the different vulnerabilities of web applications. The attacks discussed in this paper include SQL injection, Session Hijacking, Cross-Site Request Forgery, Security Misconfigurations, Buffer over Flows, Privilege Escalation, Cross-Site Scripting (and the different types), etc. The paper reviews and compares two testing methods: automated vs. manual testing. Lastly, a comparison of three penetration testing tools of web applications is presented. The paper compares the features of ZAP, Acunetix, and Burpsuit.

Kang, Lee, Kim, & Kim (2016) discusses how penetration testing can be implemented into businesses within the financial sector to make them more secure. The model put forth in this article gives practical steps companies can take to implement penetration testing tools into their security testing. It also gives the definitions of SQL Injections and Cross-Site Scripting. These are both vulnerabilities we focus on in our paper.

In Muñoz, Armas Vega, & Villalba (2016), the efficiency and false positive rates of multiple penetration testing tools are examined. This article examined both OWASP ZAP and Arachni, but it did not examine Vega. OWASP ZAP was found to be more time efficient than Arachni. However, Arachni generates more requests to the server. In this test OWASP ZAP had one vulnerability that was found by the application's requests but was not reported to the user. Arachni did not have any false positive reports during this experiment. We use those results for comparing with our results.

## 3. METHODOLOGY

**ZAP**

Upon opening the application, the Automated Scan button is pressed to start the scan. The URL to test must be given. Once the web application URL has been entered the attack button is pressed. The scan tests the website and any associated URL to find vulnerabilities. Once the scan is completed, the alerts section will be populated with any vulnerabilities found. By going to the alerts section, details about each vulnerability can be analyzed.

**Vega**

The first step for running a Vega Scan is clicking the "Scan" tab and creating a new scan. Then we select which modules to enable for this scan (we leave the default selections). We then press the "Finish" button so that the scan can start. Furthermore, each vulnerability of the web application can be clicked on for a further explanation of the vulnerability.

**Arachni**

In order to run Arachni, the folder \arachni-1.6.1-0.6.1-windows-x86_64\bin must be opened. Inside the folder, arachni_web.bat file must be run by double-clicking the file. The file

will open the command line that displays the address it's listening on. The address can then be copied and pasted onto a web browser which will prompt the log-in screen. The default parameters can be found in the Arachni documentation. This then opens the Arachni web interface. A new scan can be started by clicking the Scans tab and then selecting New Scan. The URL for the web application can then be inserted. Clicking the Go button will then begin the scan. The vulnerabilities are then shown, where each one can be selected for further inspection.

## 4. EXPERIMENTAL WORK

In our study, we tested three vulnerability assessment tools: ZAP, Vega and Arachni. Specifically, each tool ran a vulnerability assessment of the same malicious test website called Acuart ("Acunetix Web Vulnerability Scanner - test websites") and we furthered compared the results of one another. It is important to note that Acuart is one of several vulnerable test websites provided by Acunetix. The results of this test are shown below.

### ZAP Tool Results

As seen in Figure 1, Zap found three types of high-risk vulnerabilities. Two of these are different types of Cross-Site Scripting, DOM Based and Reflected. The other high-risk vulnerability found were SQL Injections. Further information can be found on each case by clicking on the requests in the drop-down menu for each category. There were four types of medium-risk vulnerabilities found. There were seven instances of .htaccess Information Leaks, 47 cases of CSP Headers not being set, 40 cases of an Absence of Anti-CSRF Tokens, and 44 cases of Missing Anti-Clickjacking Headers. In total 138 medium-risk vulnerabilities were found. Two different types of low-risk vulnerabilities were found by Zap. There were 62 cases of Server Leaks found and 67 X-Content-Type-Options. In total 129 different low-risk vulnerabilities were found. For each vulnerability found, information on the exact call that exposed the vulnerability can be found under the vulnerability for each one. The right side of the screen gives additional information on vulnerabilities.

### Vega Tool Results

The results shown in Figure 2 for the test website display a total of 36 vulnerabilities. These vulnerabilities are categorized into high, medium, low, and info alerts. There was a total of 21 high alerts, 6 medium alerts, 2 low alerts, and 7 info alerts.

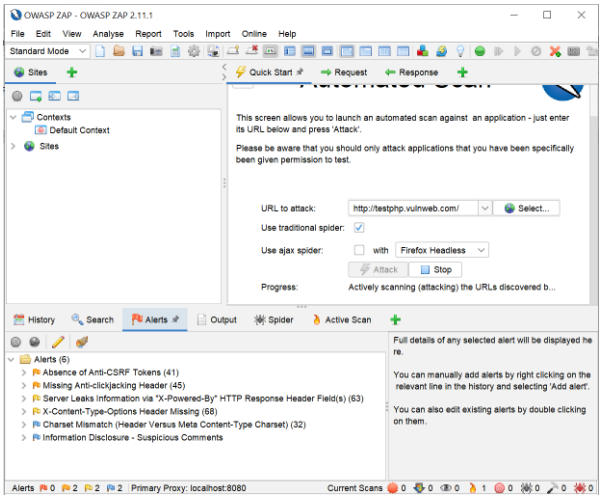


**Figure 1: ZAP Interface with the Vulnerability Alerts for the Test Website Shown**

For each alert, an explanation of what the vulnerability is, the request, resource content, a discussion, impact, and resources are available. The high-level alerts include 2 cleartext password over HTTP alerts, 10 Cross-Site Scripting alerts, 3 MySQL Error Detect- Possible SQL Injection, and 6 SQL Injection alerts. The medium-level alerts include 6 Local Filesystem Paths Found alerts. The low-level alerts include 2 Form Password Fields with Autocomplete Enabled alerts. The info-level alerts include 2 Possible AIAX Code detected, 4 Character Set Not Specified, and 1 Blank Body detected.

### Arachni Tool Results

In the results shown in Figure 3, we see that it found a total of 69 alerts where 37 were of high risk, 7 medium risk, 10 low risk, and 24 that are informational. In regard to the high-risk alert, the largest number of alerts fall under the category of Cross-Site Scripting (XSS). XSS can make the system vulnerable where clients can inject scripts into a request and the server then returns in the response the script to the client. Another high-risk vulnerability found in this malicious web application by Arachni is SQL injection. Web applications use SQL queries to retrieve data from databases. SQL injections occur when a value from the client's request is used in the SQL query without sanitization. This makes it vulnerable to attackers executing arbitrary SQL code to steal data or even take control of more server components by exploiting the additional functionalities of the database server. It is important to note that this is one of the most common web application vulnerabilities. This specific vulnerability was detected by Arachni by causing the server to respond to a request with a

database-related error. Some of the medium risk alerts include common directory and unencrypted password form, while the low-level risks include common sensitive file and password fields with auto-complete.

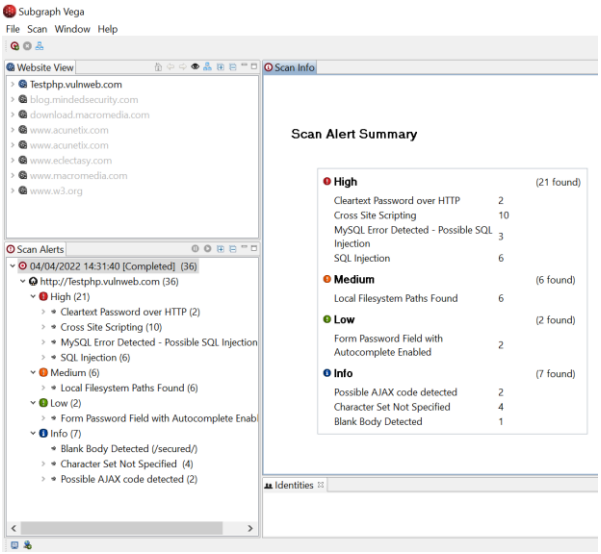


**Figure 2: Vega Interface with the Vulnerability Alerts for the Test Website http://testphp.vulnweb.com**

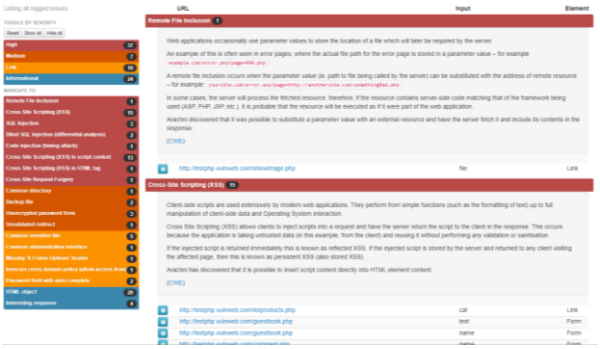

**Figure 3: Arachni Interface with the Vulnerability Alerts for the Test Website http://testphp.vulnweb.com**

**Tool Comparison**

Below we have constructed a table that compares the vulnerabilities of ZAP, Vega, and Arachni:

| | ZAP | Vega | Arachni |
|---|---|---|---|
| **Total Vulnerabilities** | 341 | 36 | 69 |
| **High Alerts** | 41 | 21 | 37 |
| **Medium Alerts** | 138 | 6 | 7 |
| **Low Alerts** | 129 | 2 | 10 |

**Table 1: Comparison of the Total Vulnerabilities and Types of Vulnerabilities for the Test Website in Each of the Tools**

From Table 1 above, we can see that ZAP has a total of 341 vulnerabilities that it detects. This is significantly higher than both Vega and Arachni, which have 36 and 69 vulnerabilities, respectively. Additionally, ZAP has a larger number of high, medium, low, and information alerts in comparison to the other tools. When comparing Vega and Arachni, Arachni detected more vulnerabilities in total and in each of the types of vulnerabilities.

| | ZAP | Vega | Arachni |
|---|---|---|---|
| **SQL Injection** | 7 | 6 | 5 |
| **XSS** | 34 | 10 | 29 |

**Table 2: Comparison of the Two Different Vulnerabilities Present in the Test Website for Each of the Tools**

Table 2 shows the comparison of the two different vulnerabilities present in each of the tools: SQL injection and Cross-Site Scripting (XSS). ZAP found the most SQL injections and Cross-Site Scripting vulnerabilities. ZAP found 7 SQL injections and 34 Cross-Site Scripting vulnerabilities, 18 of which were DOM-based and 16 that were reflections. In comparison, Arachni found 29 Cross-Site Scripting vulnerabilities and only 5 SQL injections, 2 of which were Blind SQL injections. Lastly, Vega found 10 Cross-Site Scripting vulnerabilities, which is considerably lower than its counterparts, and 6 SQL injections and 3 possible SQL injections. Altogether, ZAP showed the most proficiency in finding SQL injections and Cross-Site Scripting vulnerabilities.

| | ZAP | Vega | Arachni |
|---|---|---|---|
| **Interface Platform** | Yes | Yes | For the most part |
| **Ease of Use** | Simple | Very Simple | Complex |
| **Performance** | Med (4-5 min) | High (2-3 min) | Low (8-9 min) |
| **Stability** | High | Low | Very Low |

**Table 3: Comparison of Penetration Tool Characteristics**

Table 3 shows a comparison of the tool characteristics such as the interface, ease of use, performance, and stability. ZAP and Vega had great interfaces, and Arachni has a good interface but involves additional work from the command line to start the application.

Vega was the easiest tool to use because it only involved downloading the software and inputting the URL of the application we needed to scan. ZAP was also simple to use as it also only involved

downloading the software and inputting the URL of the application we needed to scan. However, ZAP was a little more difficult to use versus Vega in that assessing the results wasn't as simple. Arachni was the hardest to use because we needed to download the software, run a .bat file, which opened the command line, and had to copy a URL from the command line that is used in order to access Arachni's GUI. After the GUI was opened, a password and email were needed from Arachni's documentation that would allow us to use the tool. Then we could input the URL of the application we needed to scan and assess the vulnerabilities.

Furthermore, Vega's performance was the highest with the scan running in 2-3 minutes. ZAP had the next best performance, taking 3-4 minutes. Arachni had the lowest performance and took about 8-9 minutes to run. We also assess the tool's stability. ZAP was a very stable tool as it did not give us any problems when running. Vega, on the other hand, would not work on a public Wifi and affected our computer's internet access as well. Arachni had the worst stability as it also did not work on the university Wifi and crashed multiple times.

As a result, from this analysis, ZAP is the best penetration testing tool for web applications when compared to Vega and Arachni. This is the tool that we expected to be the best of the three tools based on our initial research. Other previous works compare ZAP to other tools, and these other tools may have a better implementation than ZAP and have the potential to find more vulnerabilities. However, ZAP does a decent job for a free, open-source tool available to everyone.

We show that ZAP is the best penetration testing tool for web applications when compared to Vega and Arachni. This can be seen in our results as ZAP found the most vulnerabilities, and specifically, more SQL injection and Cross-Site Scripting vulnerabilities when compared to its counterparts. ZAP was also simple to use, had a great interface, had a performance time of 3-4 minutes, and was a very stable tool as it did not give us any problems when running.

## 5. CONCLUSION AND FUTURE WORK

Considering the everlasting request for reliable, current, and fast network penetration testing tools, in the hope to provide current and practical reference for researchers and practitioners, this work presents the results and comparison of test running three vulnerability assessment tools: ZAP, Vega and Arachni. While selecting those tools the criteria we considered were being current, ease of use, reliability (stability), and performance (speed). The comprehensive test run of each tool yields that each tool possesses its own advantages and deficiencies, with each being better at one or more criteria than the others, but not prevailing in all.

The span of network vulnerability tools is large and is expanding even more, thanks to constant developments in network and PC technologies. Hence, to be more inclusive, we plan to explore more penetration tools for testing purposes and provide a comparative study on a wider span of such tools. In particular, we intend to include Vulcan (Vulcan), Invicti (Invicti), Intruder (Intruder), and BeyondTrust (BeyondTrust) as our next step of expanding this research for a more comprehensive analysis of network vulnerability assessment tools.

## 6. REFERENCES

"What I Should Do If No Author Listed." (2022). New England Online Education 7(12). Retrieved March 19, 2022 from http://giveaddress.com/xyz

Vega Vulnerability Scanner. (n.d.). Retrieved August 8, 2022, from https://subgraph.com/vega/

Web application security scanner framework. Arachni. (n.d.). Retrieved August 8, 2022, from https://www.arachni-scanner.com/

Mirjalili, M., Nowroozi, A., & Alidoosti, M. (2013). A survey on web penetration test. ACSIJ Advances in Computer Science: an International Journal, 3(6).

Fonseca, J., Vieira, M., & Madeira, H. (2007). Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks. 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007). https://doi.org/10.1109/prdc.2007.55

Khera, Y., Kumar, D., Sujay, & Garg, N. (2019). Analysis and impact of Vulnerability Assessment and penetration testing. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). https://doi.org/10.1109/comitcon.2019.8862224

Zakaria, M. N., Phin, P. A., Mohmad, N., Ismail, S. A., Kama, M. N., & Yusop, O. (2019). A review of standardization for penetration

testing reports and documents. 2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS). https://doi.org/10.1109/icriis48246.2019.9073393

Abu-Dabaseh, F., & Alshammari, E. (2018). Automated penetration testing : An overview. Computer Science & Information Technology. https://doi.org/10.5121/csit.2018.80610

Nagpure, S., & Kurkure, S. (2017). Vulnerability assessment and penetration testing of web application. 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA). https://doi.org/10.1109/iccubea.2017.8463920

Kang, W., Lee, G., Kim, S., & Kim, J.-B. (2016). Diagnostic Model of Vulnerability based on Penetration Testing. International Information Institute (Tokyo). Information, 19(6), 2257–2262.

Muñoz, F. R., Armas Vega, E. A., & Villalba, L. J. (2016). Analyzing the traffic of penetration testing tools with an IDS. The Journal of Supercomputing, 74(12), 6454–6469. https://doi.org/10.1007/s11227-016-1920-7

Acunetix Web Vulnerability Scanner - test websites. Acunetix Web Vulnerability Scanner - Test websites. (n.d.). Retrieved May 25, 2022, from http://www.vulnweb.com/

Vulcan Network Vulnerability Assesment Tool, Retrieved September 21, 2022, from https://vulcan.io/

Invicti Network Vulnerability Assesment Tool, Retrieved September 21, 2022, from https://www.invicti.com/

Intruder Network Vulnerability Assesment Tool, Retrieved September 21, 2022, from https://www.intruder.io/

BeyondTrust Network Vulnerability Assesment Tool, Retrieved September 21, 2022, from https://www.beyondtrust.com/