Borneo Akademika

e-ISSN: 2735-2250

Available online at
https://borneoakademika.uitm.edu.my

Volume 8(2) December 2024, 176-187

Borneo
Akademika

Exploring Vega: A Tool for Scanning Vulnerabilities in Penetration Testing within Web Applications

Sulastri Putit[1*], Lenny Yusrina Bujang Khedif[1]

[1]*College of Computing, Informatics and Mathematics, Universiti Teknologi MARA Sarawak Branch*

ARTICLE INFO

Article history:
Received 1 October 2024
Accepted 15 November 2024
Online first
Published 20 December 2024

Keywords:
Vega
Penetration testing
Scanning vulnerabilities
Web applications
cybersecurity

ABSTRACT

In the realm of cybersecurity, penetration testing is essential for identifying and mitigating vulnerabilities before they can be exploited by attackers, particularly within web applications. Vega, an open-source web security scanner, stands out due to its comprehensive scanning capabilities and user-friendly interface, making it a valuable tool for vulnerability detection. This paper explores Vega's core features, including automated scanning, manual testing, and customisable scanning profiles. It examines Vega's effectiveness in detecting common web vulnerabilities like SQL injection and Cross-Site Scripting (XSS) and assesses its role in enhancing the penetration testing process. Real-world case studies are discussed, demonstrating Vega's capabilities and limitations within practical testing environments. While Vega offers strong scanning capabilities, this paper posits that its effectiveness is significantly enhanced when combined with other security tools, highlighting Vega's potential in strengthening security postures and suggesting avenues for future development, including enhanced machine learning integration to improve detection accuracy.

INTRODUCTION

In the rapidly evolving landscape of cybersecurity, penetration testing has emerged as a critical practice for identifying and mitigating vulnerabilities within systems. Penetration testing, commonly known as ethical hacking, is a proactive approach to identifying security vulnerabilities in web applications. The increasing complexity of cyber threats necessitates robust security measures, with penetration testing being a proactive approach to identifying system vulnerabilities. Penetration testing is utilised to mitigate risks and potential information loss by identifying the existing gaps and providing a comprehensive mitigation plan. It also evaluates the effectiveness of implemented security measures (Altulaihan et.al, 2023). The primary objective of

[1*] Corresponding author. *E-mail address*: sulastri@uitm.edu.my

pen testing is to proactively discover and identify security vulnerabilities before attackers can exploit them (Albahar et al., 2022). In a pen test, an unauthorised attack is simulated on a target system using either manually operated tools, automated tools, or a combination of both to prevent potential web security breaches. The results are then applied to address security issues in compliance with accepted security standards. While manual vulnerability assessment in a web application can be effective, it is also highly time-consuming, error-prone, and costly, thereby limiting its viability and scalability (Stefinko et al., 2016). As cyber threats evolve, the need for efficient and reliable tools to perform penetration testing has become critical.

Vega is a tool that offers both automated scanning and manual testing capabilities, making it a valuable asset for cybersecurity professionals. Vega is an open-source web vulnerability scanner that provides a comprehensive suite of features that is designed to assist security professionals in identifying potential security weaknesses in web applications. Vega, an open-source web vulnerability scanner, provides a platform for security professionals to conduct thorough assessments of web applications. Its features facilitate the identification of security flaws, thereby enhancing the overall security posture of organisations (Phong & Yan, 2014; Denis et al., 2016). The significance of this exploration lies in the increasing complexity of cyber threats and the necessity for robust tools that can adapt to these challenges. This paper will explore the functionalities of Vega, focusing on its role in the reconnaissance phase of penetration testing and its integration with other tools and methodologies.

LITERATURE REVIEW

Overview of Penetration Testing

Penetration testing is a method used by organisations to evaluate their security posture by identifying vulnerabilities within systems, networks, and applications. This method involves simulating cyber-attacks on a system. This process is essential for organisations to understand their vulnerabilities and the potential impact of a successful attack. The methodology typically involves several steps, including planning, reconnaissance, scanning, exploitation, and reporting (Zennaro & Erdődi, 2020).

Penetration testing simulates real-world attacks to uncover vulnerabilities within systems, applications, and networks. The primary goal is to identify weaknesses that could be exploited by attackers, thereby allowing organisations to remediate these issues before they can be exploited (Phong & Yan, 2014; Denis et al., 2016). Various methodologies exist for conducting penetration tests, including black-box, white-box, and grey-box testing, and each offers unique advantages, depending on the testing objectives and the environment (Bertoglio & Zorzo, 2017; Tetskyi et al., 2021). The effectiveness of penetration testing is heavily reliant on the tools employed, which can significantly influence the outcomes of the assessments. Vega fits into this landscape as a tool that enhances the reconnaissance and scanning phases of penetration testing.

The Role of Vulnerability Scanners

Vulnerability scanners play a pivotal role in the penetration testing process by automating the identification of security flaws. These tools can scan networks, systems, and applications to detect known vulnerabilities, misconfigurations, and compliance issues. The integration of vulnerability scanners into penetration testing workflows enhances efficiency and accuracy, allowing security professionals to focus on more complex tasks such as exploitation and remediation (Amankwah et al., 2020). Vega, as a web vulnerability scanner, offers unique features that cater specifically to web applications, making it a valuable asset for organisations seeking to bolster web security.

Comparative Analysis with Other Tools

Research has highlighted and analysed the tools utilised in penetration testing, including Vega. To understand the effectiveness of Vega, it is essential to compare it with other vulnerability scanning tools. Research indicates that while commercial scanners often provide extensive support and features, open-source tools like Vega can offer comparable capabilities at a lower cost (Amankwah et al., 2020). A study comparing commercial and open-source web vulnerability scanners found that both categories have their strengths and weaknesses, with the open-source tools being particularly advantageous for organisations with limited budgets (Amankwah et al., 2020).

Moreover, the adaptability of Vega in various environments is noteworthy, as its open-source nature allows for community contributions and continuous updates, ensuring that it remains relevant in the face of emerging threats. This aspect is particularly critical given the dynamic nature of cybersecurity, where new vulnerabilities are constantly being discovered (Modesti, 2024).

Key Features of Vega

Vega is an open-source web security scanner and testing platform, designed to find vulnerabilities in web applications through automated scans and manual testing tools. Itis a free and open-source web security scanner and web security testing platform aims to test the security of web applications. Vega can help to find and validate SQL Injection, Cross-Site Scripting (XSS), inadvertently disclosed sensitive information, and other vulnerabilities. It is written in Java, GUI based, and runs on Linux, OS X, and Windows. It can help you find vulnerabilities such as: reflected cross-site scripting, stored cross-site scripting, blind SQL injection, remote file includes, shell injection, and others while it also probes for TLS / SSL security settings and identifies opportunities for improving the security of your TLS servers. Vega includes an automated scanner for quick tests and an intercepting proxy for tactical inspection. The Vega scanner finds XSS (cross-site scripting), SQL injection, and other vulnerabilities, in which can be extended using a powerful API in the language of the web: Javascript. (Subgraph, n.d.) Vega's user-friendly interface, combined with its powerful scanning capabilities, makes it a popular choice among cybersecurity professionals.

Vega offers several key features making it a valuable tool for penetration testers. Its user-friendly interface allows for easy navigation, making it accessible even for those with limited experience in cybersecurity (Barik et al., 2021; Aar & Sharma, 2017). The tool supports automated scanning, which significantly reduces the time required to identify vulnerabilities. Additionally, Vega provides detailed reports that outline the vulnerabilities discovered, their severity, and recommendations for remediation (Sarker et al., 2023; Kamarudin et al., 2019). This reporting capability is crucial for organisations to understand their security posture and prioritize remediation efforts effectively.

Vega is distinguished by several key features that enhance its utility in scanning vulnerabilities. Firstly, it provides a user-friendly interface that simplifies the scanning process, making it accessible to both novice and experienced users. The tool supports automated scanning, which allows for the rapid identification of vulnerabilities without extensive manual intervention (Aisyah, 2024). Furthermore, Vega provides customisable scanning profiles, allowing users to tailor scans to meet specific requirements or compliance standards.

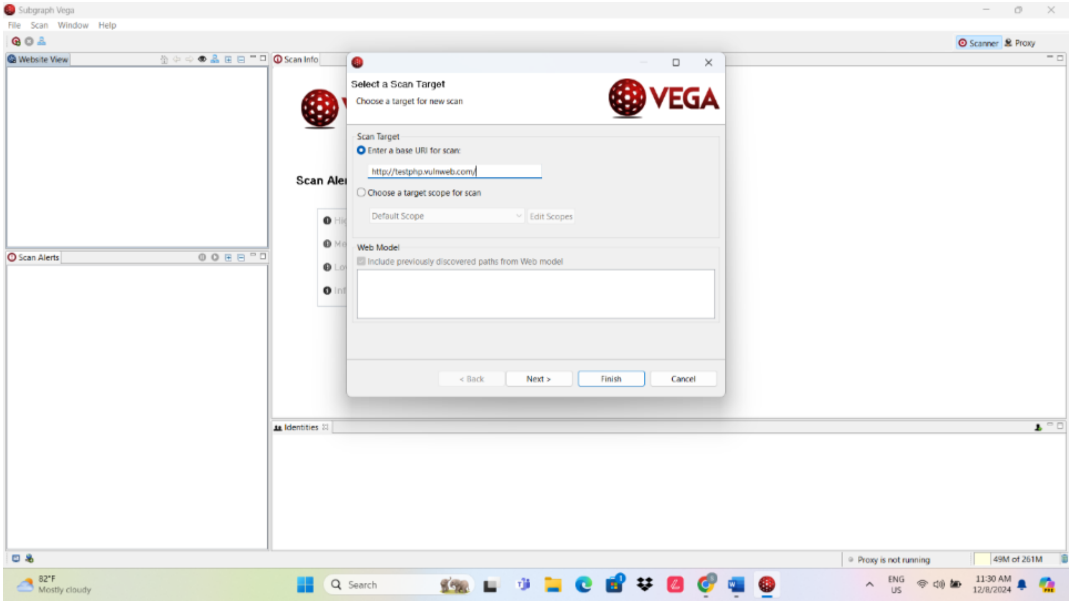
Another notable feature of Vega is its ability to perform both active and passive scanning. Active scanning involves sending requests to the target application to identify vulnerabilities, while passive scanning analyses traffic to detect potential issues without direct interaction (Priambodo et al., 2023). This dual approach enhances the comprehensiveness of the assessments conducted using Vega.

Furthermore, Vega includes a built-in reporting feature that generates detailed reports on identified vulnerabilities, including their severity levels and recommended remediation steps. This functionality is crucial for organisations to prioritise their response efforts and allocate resources effectively (Modesti, 2024). Integrating such reporting capabilities aligns with the best practices in vulnerability management, facilitating a structured approach to addressing security weaknesses.

CASE STUDIES AND PRACTICAL APPLICATIONS

Numerous case studies illustrate the effectiveness of Vega in real-world penetration testing scenarios. For example, organisations have successfully utilised Vega to assess the security of their web applications, leading to the identification of critical vulnerabilities that were subsequently remediated (Sarker et al., 2023; Kamarudin et al., 2019). These practical applications underscore the importance of incorporating tools like Vega into the penetration testing process, as they provide tangible benefits in enhancing security measures.

The paper included practical case studies demonstrating Vega's effectiveness and capabilities in real-world penetration testing scenarios. These case studies showcase how Vega can be used to uncover vulnerabilities in various web applications, highlighting both its strengths and limitations. A case study using the web application http://testphp.vulnweb.com demonstrated Vega’s capability to identify various critical vulnerabilities, such as SQL injection and XSS. This is an example PHP application, which is intentionally vulnerable to web attacks.



Select a Scan Target

Choose a target for new scan

Scan Target

● Enter a base URI for scan: http://testphp.vulnweb.com/

○ Choose a target scope for scan

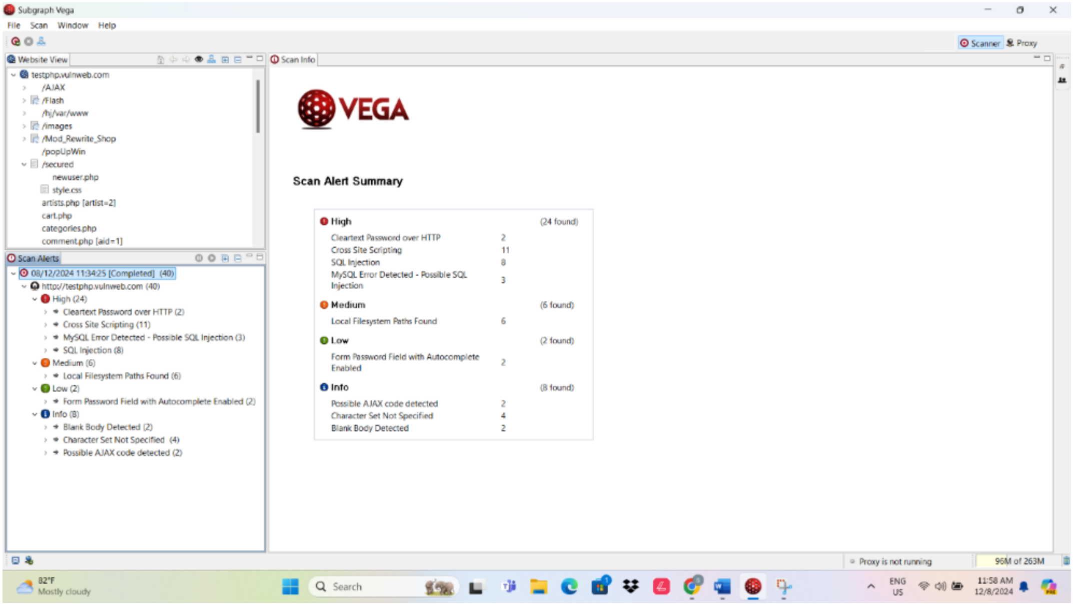
Default Scope Edit Scopes

Web Model

□ Include previously discovered paths from Web model

Figure 1. User friendly interface in Vega

Figure 1 shows the user-friendly interface in Vega which makes it accessible to both novice and experienced penetration testers.



```
testphp.vulnweb.com
    /AJAX
    /Flash
    /hj/var/www
    /images
    /Mod_Rewrite_Shop
    /popUpWin
    /secured
        newuser.php
      style.css
    artists.php [artist=2]
    cart.php
    categories.php
    comment.php [aid=1]
```

```
08/12/2024 11:34:25 [Completed]  (40)
  http://testphp.vulnweb.com (40)
    High (24)
      Cleartext Password over HTTP (2)
      Cross Site Scripting (11)
      MySQL Error Detected - Possible SQL Injection (3)
      SQL Injection (8)
    Medium (6)
      Local Filesystem Paths Found (6)
    Low (2)
      Form Password Field with Autocomplete Enabled (2)
    Info (8)
      Blank Body Detected (2)
      Character Set Not Specified  (4)
      Possible AJAX code detected (2)
```

Scan Alert Summary

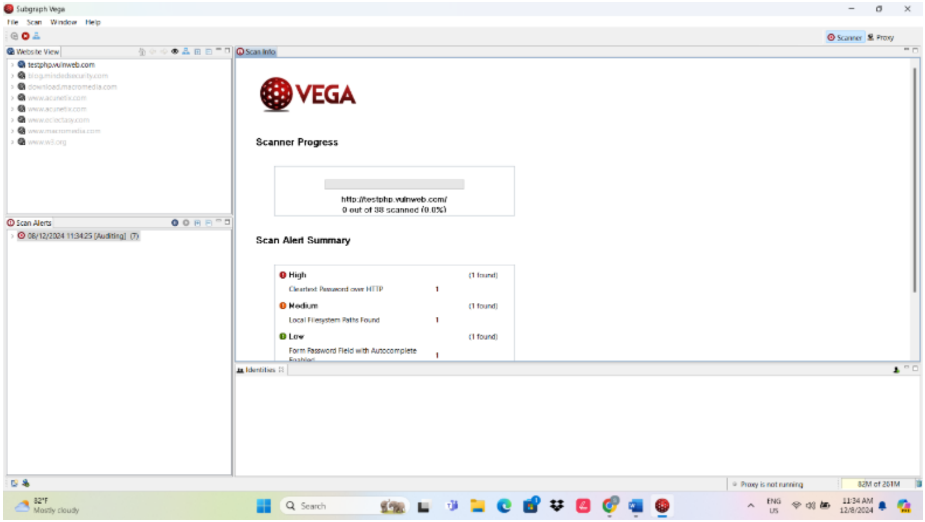
High		(24 found)
Cleartext Password over HTTP	2	
Cross Site Scripting	11	
SQL Injection	8	
MySQL Error Detected - Possible SQL Injection	3	
Medium		(6 found)
Local Filesystem Paths Found	6	
Low		(2 found)
Form Password Field with Autocomplete Enabled	2	
Info		(8 found)
Possible AJAX code detected	2	
Character Set Not Specified	4	
Blank Body Detected	2	

Figure 2. Summary of the Vulnerability Scanning

Vega has comprehensive scanning, as shown in Figure 2 Vega's automated scanning capabilities cover a broad spectrum of vulnerabilities, making it a versatile tool for web application security. Once the scan is complete, a detailed report of all vulnerabilities discovered in the target web application is displayed. The vulnerability scan results are summarised and organised by severity (High, Medium, and Low). The report includes details about each type of vulnerability, affected URLs, and technical support with request and response snippets. In addition, it offers recommendations for corrective action, references to outside sources, metadata from the scan (such as the parameters and duration), and compliance data about security standards. Understanding security risks, prioritising fixes, assuring compliance, and directing ongoing security improvements are all made possible by this report, which can be exported in a variety of formats.

Automated Scanning Capabilities

One of the standout features of Vega is its automated scanning capabilities (Figure 3). By utilising a combination of predefined scanning profiles and customisable settings, the users can tailor the scanning process to meet specific organisational needs (Bu, 2024; Akhilesh et al., 2022). This flexibility allows for comprehensive assessments of web applications, identifying a wide range of vulnerabilities, including SQL injection, cross-site scripting (XSS), and insecure server configurations (Zhang, 2023; Kumar et al., 2024). The automation feature not only improves efficiency but also reduces the potential for human error during the testing process.



testphp.vulnweb.com
blog.mindedsecurity.com
download.macromedia.com
www.acunetix.com
www.acunetix.com
www.eclectasy.com
www.macromedia.com
www.w3.org

08/12/2024 11:34:25 [Auditing] (7)

Scanner Progress

http://testphp.vulnweb.com/
0 out of 38 scanned (0.0%)

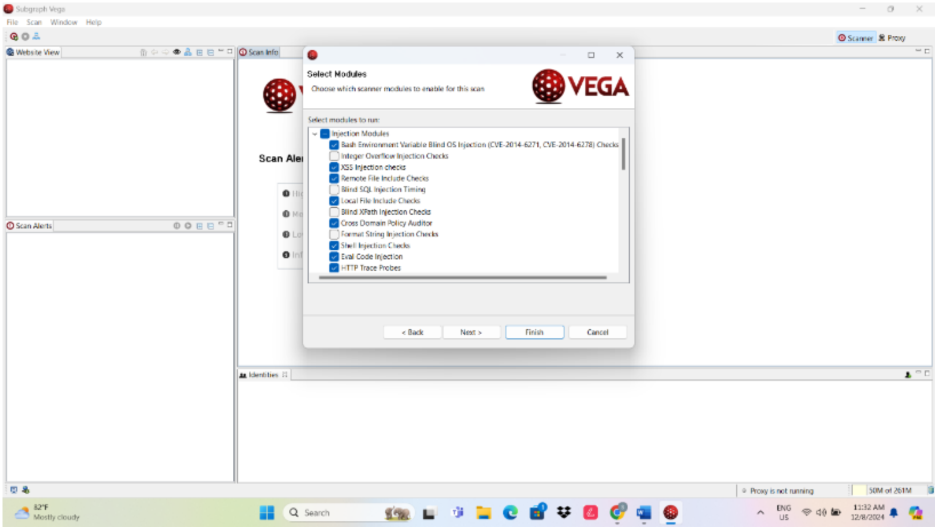
Scan Alert Summary

High		(1 found)
Cleartext Password over HTTP	1	
Medium		(1 found)
Local Filesystem Paths Found	1	
Low		(1 found)
Form Password Field with Autocomplete Enabled	1	

Proxy is not running
32M of 201M

Figure 3. Automated Scanning using Vega

Vega also provides a Scanner Module, which includes various customised scanner modules based on the specific needs of the penetration tester (Figure 4). Vega provides Customizable Modules, and these allow testers to tailor Vega to specific testing scenarios.



Select Modules
Choose which scanner modules to enable for this scan

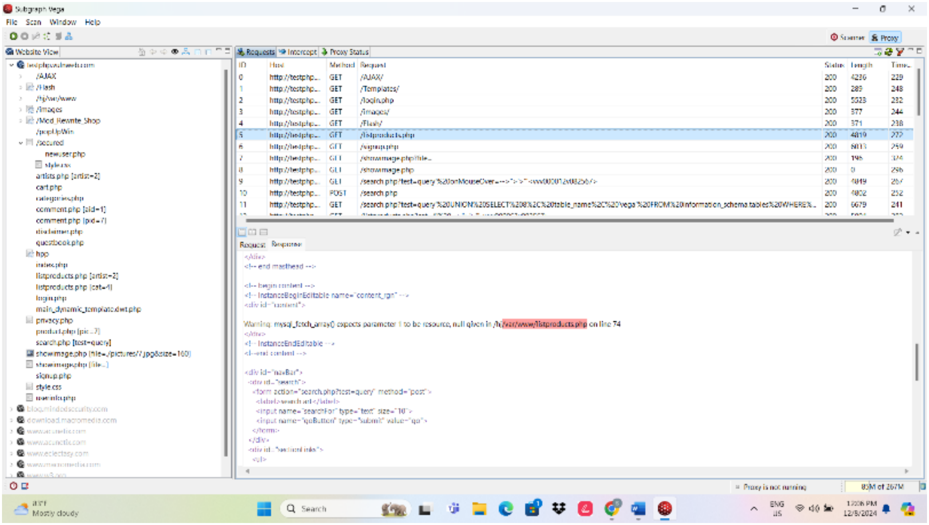
Select modules to run:

- Injection Modules
  - ☑ Bash Environment Variable Blind OS Injection (CVE-2014-6271, CVE-2014-6278) Checks
  - □ Integer Overflow Injection Checks
  - ☑ XSS Injection checks
  - ☑ Remote File Include Checks
  - □ Blind SQL Injection Timing
  - ☑ Local File Include Checks
  - □ Blind XPath Injection Checks
  - ☑ Cross Domain Policy Auditor
  - □ Format String Injection Checks
  - ☑ Shell Injection Checks
  - ☑ Eval Code Injection
  - ☑ HTTP Trace Probes

Proxy is not running
30M of 201M

Figure 4. Scanner Module in Vega

Vega also supports manual testing (Figure 5), allowing the testers to perform in-depth analysis and validation of vulnerabilities identified during automated scans.



ID	Host	Method	Request	Status	Length	Time...
0	http://testphp...	GET	/AJAX/	200	4236	229
1	http://testphp...	GET	/Templates/	200	289	248
2	http://testphp...	GET	/login.php	200	5523	232
3	http://testphp...	GET	/images/	200	377	244
4	http://testphp...	GET	/Flash/	200	371	238
5	http://testphp...	GET	/listproducts.php	200	4819	272
6	http://testphp...	GET	/signup.php	200	6013	259
7	http://testphp...	GET	/showimage.php?file...	200	196	374
8	http://testphp...	GET	/showimage.php	200	0	296
9	http://testphp...	GET	/search.php?test=query'%20onMouseOver=-->">'>'"<vwv000012v00825b/>	200	4849	257
10	http://testphp...	POST	/search.php	200	4802	252
11	http://testphp...	GET	/search.php?test=query'%20UNION%20SELECT%208%2C%20table_name%2C%20'vega'%20FROM%20information_schema.tables%20WHERE%...	200	6679	241

```
</div>
<!-- end masthead -->

<!-- begin content -->
<!-- InstanceBeginEditable name="content_rgn" -->
<div id="content">

Warning: mysql_fetch_array() expects parameter 1 to be resource, null given in /hj/var/www/listproducts.php on line 74
</div>
<!-- InstanceEndEditable -->
<!--end content -->

<div id="navBar">
  <div id="search">
    <form action="search.php?test=query" method="post">
      <label>search art</label>
      <input name="searchFor" type="text" size="10">
      <input name="goButton" type="submit" value="go">
    </form>
  </div>
  <div id="sectionLinks">
    <ul>
```

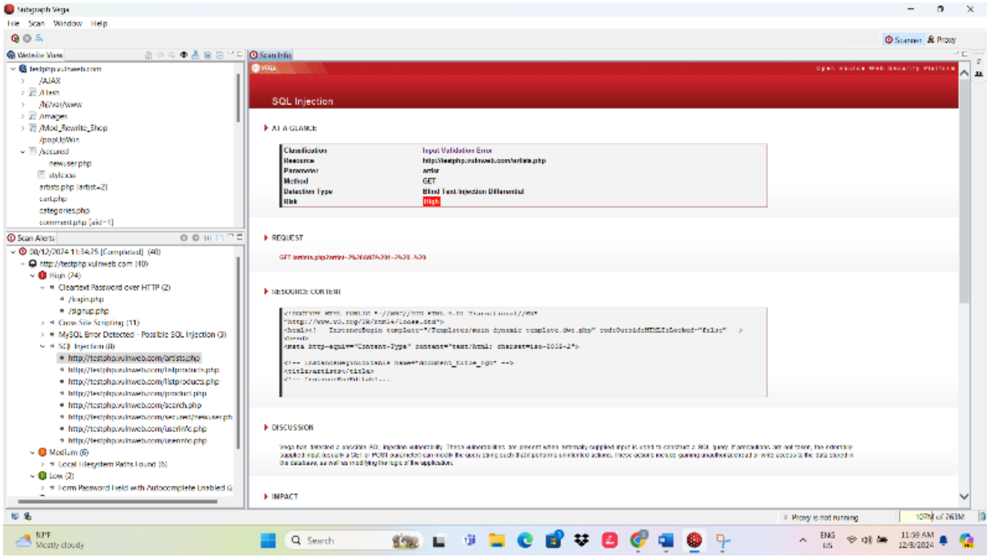
Proxy is not running 85M of 267M

Figure 5. Manual Testing in Vega

Vulnerability Detection Mechanisms

Vega's vulnerability detection mechanisms are based on signature and heuristic analysis. In this case study of the web application http://testphp.vulnweb.com/, the results revealed several vulnerabilities. The Vega tool employs various techniques to identify common web vulnerabilities, such as:

SQL Injection: In Figure 6, Vega can detect SQL injection vulnerabilities by analysing the input fields and URL parameters for potential injection points.



Scan Alerts
- 08/12/2024 11:34:25 [Completed] (40)
  - http://testphp.vulnweb.com (40)
    - High (24)
      - Cleartext Password over HTTP (2)
        - /login.php
        - /signup.php
      - Cross Site Scripting (11)
      - MySQL Error Detected - Possible SQL Injection (3)
      - SQL Injection (8)
        - http://testphp.vulnweb.com/artists.php
        - http://testphp.vulnweb.com/listproducts.php
        - http://testphp.vulnweb.com/listproducts.php
        - http://testphp.vulnweb.com/product.php
        - http://testphp.vulnweb.com/search.php
        - http://testphp.vulnweb.com/secured/newuser.ph
        - http://testphp.vulnweb.com/userinfo.php
        - http://testphp.vulnweb.com/userinfo.php
    - Medium (6)
      - Local Filesystem Paths Found (6)
    - Low (2)
      - Form Password Field with Autocomplete Enabled (2

SQL Injection

AT A GLANCE

Classification	Input Validation Error
Resource	http://testphp.vulnweb.com/artists.php
Parameter	artist
Method	GET
Detection Type	Blind Text Injection Differential
Risk	High

REQUEST

GET /artists.php?artist=2%20AND%201=2%20..%20

RESOURCE CONTENT

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><!-- InstanceBegin template="/Templates/main_dynamic_template.dwt.php" codeOutsideHTMLIsLocked="false" -->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

<!-- InstanceBeginEditable name="document_title_rgn" -->
<title>artists</title>
<!-- InstanceEndEditabl...
```

DISCUSSION

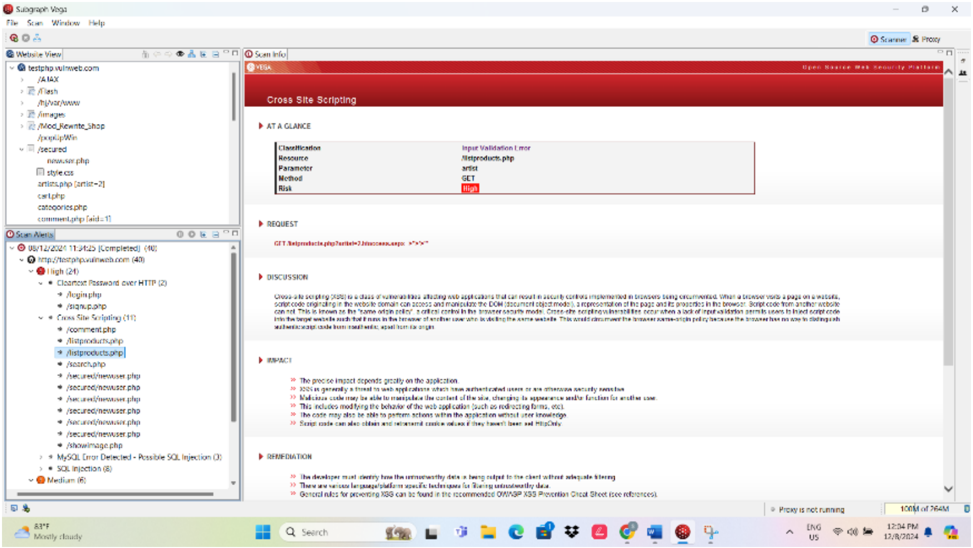
Vega has detected a possible SQL injection vulnerability. These vulnerabilities are present when externally-supplied input is used to construct a SQL query. If precautions are not taken, the externally supplied input (usually a GET or POST parameter) can modify the query string such that it performs unintended actions. These actions include gaining unauthorized read or write access to the data stored in the database, as well as modifying the logic of the application.

IMPACT

Proxy is not running 107M of 263M

Figure 6. SQL Injection Vulnerabilities

Cross-Site Scripting (XSS): Vega identifies XSS vulnerabilities by simulating malicious scripts that can be injected into web applications as shown in Figure 7.



Cross Site Scripting

AT A GLANCE

Classification	Input Validation Error
Resource	/listproducts.php
Parameter	artist
Method	GET
Risk	High

REQUEST

GET /listproducts.php?artist=2.htaccess.aspx >"'>'"

DISCUSSION

Cross-site scripting (XSS) is a class of vulnerabilities affecting web applications that can result in security controls implemented in browsers being circumvented. When a browser visits a page on a website, script code originating in the website domain can access and manipulate the DOM (document object model), a representation of the page and its properties in the browser. Script code from another website can not. This is known as the "same origin policy", a critical control in the browser security model. Cross-site scripting vulnerabilities occur when a lack of input validation permits users to inject script code into the target website such that it runs in the browser of another user who is visiting the same website. This would circumvent the browser same-origin policy because the browser has no way to distinguish authentic script code from inauthentic, apart from its origin.

IMPACT

- The precise impact depends greatly on the application.
- XSS is generally a threat to web applications which have authenticated users or are otherwise security sensitive.
- Malicious code may be able to manipulate the content of the site, changing its appearance and/or function for another user.
- This includes modifying the behavior of the web application (such as redirecting forms, etc).
- The code may also be able to perform actions within the application without user knowledge.
- Script code can also obtain and retransmit cookie values if they haven't been set HttpOnly.

REMEDIATION

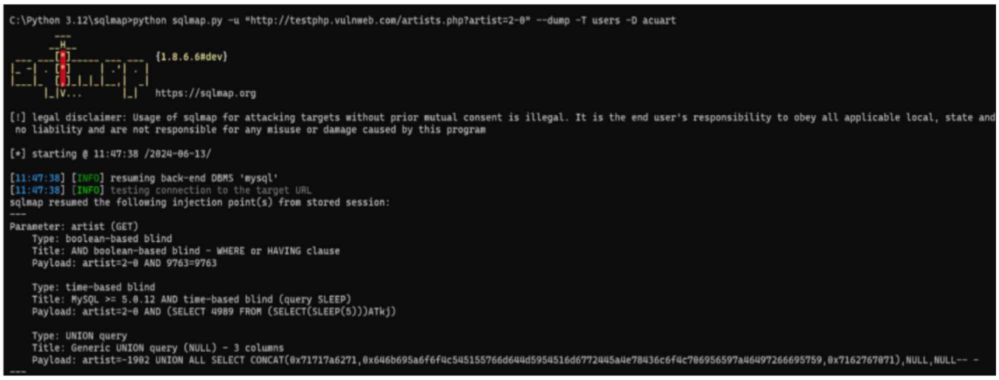
- The developer must identify how the untrustworthy data is being output to the client without adequate filtering.
- There are various language/platform specific techniques for filtering untrustworthy data.
- General rules for preventing XSS can be found in the recommended OWASP XSS Prevention Cheat Sheet (see references).

Figure 7. Cross-Site Scripting Vulnerabilities

Integration Capabilities

Vega can be effectively integrated with other penetration testing tools and frameworks, enhancing its functionality and expanding its capabilities. For instance, it can work alongside tools like rp Suite and OWASP ZAP to provide a more comprehensive testing environment (Rushing et al., 2015; Abu-Dabaseh & Alshammari, 2018). This integration allows penetration testers to leverage the strengths of multiple tools, facilitating a more thorough examination of web application security. Furthermore, the ability to export results to various formats enables seamless collaboration among team members and stakeholders (Kollepalli, 2024; Zheng et al., 2020).

Vega can also be integrated with other security tools to enhance its utility in comprehensive penetration testing strategies. In this case study, several tools can be integrated with Vega, particularly for SQL vulnerabilities. In this phase, Vega passes the process to SQLMap (Figure 8) for the next step, which is exploitation.

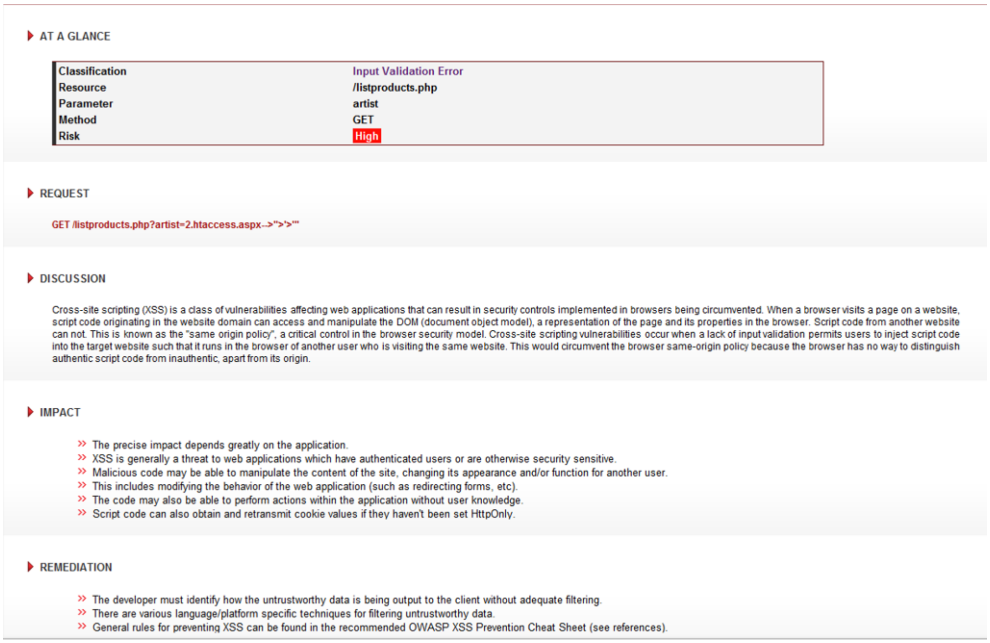


```
C:\Python 3.12\sqlmap>python sqlmap.py -u "http://testphp.vulnweb.com/artists.php?artist=2-0" --dump -T users -D acuart
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.8.6.6#dev}
|_ -| . [']     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and
 no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 11:47:38 /2024-06-13/

[11:47:38] [INFO] resuming back-end DBMS 'mysql'
[11:47:38] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=2-0 AND 9763=9763

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=2-0 AND (SELECT 4989 FROM (SELECT(SLEEP(5)))ATkj)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-1902 UNION ALL SELECT CONCAT(0x71717a6271,0x646b695a6f6f4c545155766d644d5954516d6772445a4e78436c6f4c706956597a46497266695759,0x7162767071),NULL,NULL-- -
---
```

Figure 8. SQL Map Tool use in Exploitation Phase

Reporting and Analysis

The reporting features of Vega are designed to provide actionable insights into the identified vulnerabilities during scanning (Figure 9). Reports generated by Vega include detailed descriptions of vulnerabilities, their potential impact, and suggested remediation steps (Ariyadi, 2023; Kamarudin et al., 2019). This level of detail is essential for organisations to prioritise their response efforts effectively. Moreover, the ability to generate standardised reports aligns with the industry's best practices, ensuring that findings are communicated clearly to both technical and non-technical stakeholders (Barik et al., 2021; Bertoglio & Zorzo, 2017).



AT A GLANCE

Classification	Input Validation Error
Resource	/listproducts.php
Parameter	artist
Method	GET
Risk	High

REQUEST

```
GET /listproducts.php?artist=2.htaccess.aspx-->">'>'"
```

DISCUSSION

Cross-site scripting (XSS) is a class of vulnerabilities affecting web applications that can result in security controls implemented in browsers being circumvented. When a browser visits a page on a website, script code originating in the website domain can access and manipulate the DOM (document object model), a representation of the page and its properties in the browser. Script code from another website can not. This is known as the "same origin policy", a critical control in the browser security model. Cross-site scripting vulnerabilities occur when a lack of input validation permits users to inject script code into the target website such that it runs in the browser of another user who is visiting the same website. This would circumvent the browser same-origin policy because the browser has no way to distinguish authentic script code from inauthentic, apart from its origin.

IMPACT

» The precise impact depends greatly on the application.
» XSS is generally a threat to web applications which have authenticated users or are otherwise security sensitive.
» Malicious code may be able to manipulate the content of the site, changing its appearance and/or function for another user.
» This includes modifying the behavior of the web application (such as redirecting forms, etc).
» The code may also be able to perform actions within the application without user knowledge.
» Script code can also obtain and retransmit cookie values if they haven't been set HttpOnly.

REMEDIATION

» The developer must identify how the untrustworthy data is being output to the client without adequate filtering.
» There are various language/platform specific techniques for filtering untrustworthy data.
» General rules for preventing XSS can be found in the recommended OWASP XSS Prevention Cheat Sheet (see references).

Figure 9. Reporting and Analysis for each Vulnerability found

CHALLENGES AND LIMITATIONS

Challenges in Vulnerabilities Scanning

Despite its advantages, the use of Vega and similar tools is not without challenges. One significant issue is the potential for false positives, where the scanner identifies vulnerabilities that do not exist. This can lead to wasted resources and diminished trust in the scanning process (Amankwah et al., 2020). The tool may produce false positives, which can lead to unnecessary remediation efforts and resource allocation ("A Process of Penetration Testing Using Various Tools", 2023; Roshanaei, 2024). Additionally, while Vega is effective for web application testing, it may not cover all aspects of a comprehensive penetration test, necessitating the use of complementary tools to achieve a holistic assessment (Denis et al., 2016; Aar & Sharma, 2017). Another challenge lies in the interpretation of scan results. Security professionals must possess the expertise to analyse the findings and determine the appropriate remediation actions. This requirement underscores the importance of training and continuous education in the field of cybersecurity (Zennaro & Erdődi, 2020).

Limitations of Vega

Vega's effectiveness depends on maintaining an up-to-date vulnerability database, which is critical for accurate and reliable scanning. Regular updates are necessary to keep pace with the latest security threats. Furthermore, interpreting the scan results requires expertise, as security professionals must analyse findings and determine appropriate remediation actions. This requirement underscores the importance of training and continuous education in cybersecurity

(Amankwah et al., 2020; Zennaro & Erdődi, 2020). Understanding these limitations is crucial for penetration testers to effectively leverage Vega in their assessments.

FUTURE DIRECTIONS FOR VEGA

As the cybersecurity landscape continues to evolve, there are several avenues for improving Vega's capabilities used to combat threats. Future developments for Vega could include enhanced machine learning capabilities to improve vulnerability detection accuracy and reduce false positives. The integration of artificial intelligence could enable the tool to learn from previous scans and adapt its scanning techniques accordingly (Pozdniakov et al., 2020), and the integration of threat intelligence feeds to provide context around identified vulnerabilities (Bu, 2024; Akhilesh et al., 2022). Additionally, expanding the tool's capabilities to include more comprehensive reporting features, such as integration with ticketing systems for remediation tracking, could further enhance its usefulness for organisations (Modesti, 2024). These advancements would further solidify Vega's position as a leading tool in the penetration testing domain.

CONCLUSION

In conclusion, Vega represents a significant advancement in vulnerability scanning for web applications, offering penetration testers a powerful tool to enhance their scanning capabilities. Its user-friendly interface, customisable scanning options, automated features, and robust reporting capabilities make it an invaluable tool for penetration testing. These make Vega an invaluable asset in the cybersecurity toolkit. Yet, the challenges, such as false positives and the need for expert interpretation of results, must be addressed to maximise its effectiveness. As cybersecurity threats continue to evolve, ongoing development and adaptation of tools like Vega will be essential in maintaining organisations' robust security postures. By integrating Vega into their penetration testing processes, organisations can proactively identify and remediate vulnerabilities, ultimately strengthening their security posture against evolving cyber threats. Continued research and development in this area will ensure that tools like Vega remain relevant and effective in the face of emerging challenges.

ACKNOWLEDGEMENTS

The authors would like to extend their gratitude to the lecturers of Universiti Teknologi MARA (UiTM), Sarawak Branch, Samarahan Campus, for their valuable input and contribution to this paper.

CONFLICT OF INTEREST

The authors agree that this paper was conducted in the absence of any self-benefits, commercial or financial conflicts and declare the absence of conflicting interests with the funders.

AUTHORS' CONTRIBUTIONS

Sulastri Putit developed the concept of the article, wrote the initial draft, and made revisions. She also led the review and modifications and approved the final submission. While Lenny Yusrina Bujang Khedif contributed to the article's conception and supervised its progress.

REFERENCES

Aar, P. and Sharma, A. (2017). Analysis of penetration testing tools. International Journal of Advanced Research in Computer Science and Software Engineering, 7(9), 36. https://doi.org/10.23956/ijarcsse.v7i9.408

Aisyah, N. S., Puspitasari, F. Z., Angga, K. O., & Shandi, B. R. (2024). Identify vulnerabilities on the Ministry of Health's Ayo Sehat website through penetration testing. *Engineering and Technology Journal*, *09*(07). https://doi.org/10.47191/etj/v9i07.11

Akhilesh, R., Bills, O., Chilamkurti, N., & Chowdhury, M. J. M. (2022). Automated penetration testing framework for smart-home-based iot devices. Future Internet, 14(10), 276. https://doi.org/10.3390/fi14100276Altulaihan, E. A., Alismail, A., & Frikha, M. (2023). A survey on web application penetration testing. Electronic, 12(5), 1229.

Albahar, M., Alansari, D., & Jurcut, A. (2022). An empirical comparison of pen-testing tools for detecting web app vulnerabilities. Electronics, 11(19), 2991.

Amankwah, R., Chen, J., Kudjo, P. K., & Towey, D. (2020). An empirical comparison of commercial and open-source web vulnerability scanners. *Software: Practice and Experience*, *50*(9), 1842–1857. https://doi.org/10.1002/spe.2870

Ariyadi, T. and Pohan, M. R. (2023). Implementation of penetration testing tools to test wi-fi security levels at the directorate of innovation and business incubators. Jurnal Penelitian Pendidikan IPA, 9(12), 10768-10775. https://doi.org/10.29303/jppipa.v9i12.5551

Dalalana Bertoglio, D., & Zorzo, A. F. (2017). Overview and open issues on penetration test. *Journal of the Brazilian Computer Society*, *23*(1). https://doi.org/10.1186/s13173-017-0051-1

Denis, M., Zena, C., & Hayajneh, T. (2016). Penetration testing: Concepts, attack methods, and defense strategies. *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. https://doi.org/10.1109/lisat.2016.7494156

Home of Acunetix Art. (n.d.). Testphp.vulnweb.com. http://testphp.vulnweb.com/

Kousik Barik, A Abirami, Das, S., Konar, K., & Banerjee, A. (2021). Penetration Testing Analysis with Standardized Report Generation. *Atlantis Highlights in Computer Sciences*. https://doi.org/10.2991/ahis.k.210913.045

Kumar, P., Srinivasa, J., Bellam Lakshman Sai, Natarajan, A., Senthilkumar Mathi, & Ramalingam, V. (2024). An Experimental Study on Detecting and Mitigating Vulnerabilities in Web Applications. *International Journal of Safety and Security Engineering*, *14*(2), 523–532. https://doi.org/10.18280/ijsse.140219

Modesti P, Golightly L, Holmes L, Opara C, Moscini M. (2024). Bridging the Gap: A Survey and Classification of Research-Informed Ethical Hacking Tools. *Journal of Cybersecurity and Privacy*. 4(3):410-448. https://doi.org/10.3390/jcp4030021

Phong, C. T. and Yan, W. (2014). An overview of penetration testing. International Journal of Digital Crime and Forensics, 6(4), 50-74. https://doi.org/10.4018/ijdcf.2014100104

Pozdniakov, K., Alonso, E., Stankovic, V., Tam, K., & Jones, K. (2020). Smart Security Audit: Reinforcement Learning with a Deep Neural Network Approximator. *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*.

https://doi.org/10.1109/cybersa49311.2020.9139683

Priambodo, D. F., Rifansyah, A. D., & Hasbi, M. (2023). Penetration Testing Web XYZ Berdasarkan OWASP Risk Rating. *Teknika*, *12*(1), 33–46. https://doi.org/10.34148/teknika.v12i1.571

Roshanaei, M. (2024). Enhancing Mobile Security through Comprehensive Penetration Testing. *Journal of Information Security*, *15*(2), 63–86. https://doi.org/10.4236/jis.2024.152006

Stefinko, Y., Piskozub, A., & Banakh, R. (2016). Manual and automated penetration testing: Benefits and drawbacks. Modern tendency. In 2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET) (pp. 488-491). IEEE. https://doi.org/10.1109/TCSET.2016.7451845

Tetskyi, A., Kharchenko, V., Uzun, D., & Nechausov, A. (2021). Architecture and model of neural network based service for choice of the penetration testing tools. International Journal of Computing, 513-518. https://doi.org/10.47839/ijc.20.4.2438

Vega Vulnerability Scanner. (n.d.). Subgraph.com. https://subgraph.com/vega/

Zennaro, Fabio Massimo, & Erdodi, L. (2020). Modeling Penetration Testing with Reinforcement Learning Using Capture-the-Flag Challenges: Trade-offs between Model-free Learning and A Priori Knowledge. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2005.12632