International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Assessing Vulnerability Detection Tools for Authentication

Saru Chandrakar[1], Siddharth[2], Dr. Ani Thomas[3]

Submitted: 29/12/2023 **Revised**: 05/02/2024 **Accepted**: 13/02/2024

Abstract: In this research paper, we perform a comprehensive analysis of the current state of vulnerability detection tools for authentication. The increasing number of data breaches and cyber-attacks has made it essential for organizations to regularly assess the security of their authentication systems. The purpose of this research is to evaluate the effectiveness and efficiency of several commonly used vulnera- bility assessment tools for authentication/authorization and related areas. The study includes a comparison of the features, capabilities, and scope of the selected tools. The results of the analysis provide valuable insights into the strengths and limitations of the different tools and can help bring light to some flaws. The paper concludes by providing recommendations for future research in this field.

Keywords: *Vulnerability assessment, Cyber attacks, Computer security, VAPT.*

1. Introduction

Organizations are under increasing pressure to protect the security of their authentication systems due to the rise in cyberthreats and data breaches. Authentication is a crucial part of cybersecurity and is in charge of confirming users' identities and giving access to resources that are secured. Thus, it is crucial to routinely evaluate the authentication systems' weaknesses and take steps to reduce any dangers. Tools for vulnerability detection are created to assist organisations in locating and resolving potential issues with their authentication systems. These tools employ a number of methodologies, including network scans, penetration testing, and code analysis, to find possible vulnerabilities and offer remediation recommendations that may be put into prac tise. Yet, it might be difficult to determine the scope of detection given the growing number of tools available. Many of the tools mentioned in this paper have their own unique features, strengths and weaknesses and the choice of tool should be based on an organization's specific needs. There are wide range of tools, open-source tools for example Nessus or OpenVAS, commercial tools for example Qualys or Acunetix. Some may prefer tools that are cloud- based, while others may prefer tools that can be installed on- premises. This research paper aims to provide a comprehensive anal ysis of the current state of vulnerability detection tools for authentication *(Also some view of the authentication techniques that need focus)*. The study evaluates the features, capabilities, and usability of several commonly used tools. We will look at the tools on the basis of their authentication vulnerability detection capabilities. The paper provides valu- able insights into the strengths and limitations of the different tools and helps organizations choose the right tool for their needs. The findings of this research will contribute to the state of the field of cybersecurity and provide a foundation for motivation of future research in this area.

2. Authentication Methods and Related Vulnerabilities

There are many different authentication methods in use today, but we will be looking at a few on the basis of security vulnerability caused during development and which one of those a tool can have testing methods on.

Note : There are many types of attacks possible in these authentication methods, we have only recog- nised those which are not caused by user error, phishing or spoofing attacks but rather result of au thentication.

These are the most used auth methods:

A. Passwords:

This is perhaps the most common authenti- cation method, where a user must enter a specific word or phrase in order to access a system. However, pass- words can be easily compromised if they are weak or if they are not properly secured. Some of those attack methods which can be detected in production are

- **Brute force attacks**: In this type of attack, attack- ers try every possible combination of characters until the correct password is found. This type of attack can be successful if the password is weak orshort and some systems are not designed for large scale brute force attack.

[1] *Bhilai Institute of Technology, Durg, Chhattisgarh, India saru.chandrakar@gmail.com*
[2] *Bhilai Institute of Technology, Durg, Chhattisgarh, India sid291210@gmail.com*
[3] *Bhilai Institute of Technology, Durg, Chhattisgarh, India ani.thomas@bitdurg.ac.in*

- **HTTP monitoring attacks**: Passwords are most susceptible to HTTP monitoring attacks because there is no default encryption on the content of the request. Usually username and passwords are sent over plain text in the request which would enable the attacks to steal the credentials.

B. Two-factor authentication:

This method adds an addi- tional layer of security by requiring a user to provide two forms of identification, such as a password and a security token. This makes it more difficult for attack- ers to gain access, but two-factor authentication is not foolproof and can still be vulnerable to certain types of attacks, the subtype of 2FA are.

- SMS-based 2FA: In this method, the user is sent a one-time passcode (OTP) via text message to their registered phone number. The user then enters the OTP to complete the login process.
- App-based 2FA: In this method, the user installs an authentication app on their mobile device, which generates OTPs. The user enters the OTP to complete the login process.
- Hardware token-based 2FA: In this method, the user carries a small hardware device (such as a key fob) that generates OTPs. The user enters the OTP to complete the login process.

While theoretically two factor authentication is amazing when implemented properly, but in reality that has many problems [1] [2] in these studies various challenges with multi factor factor and **OAuth** as well as its vari- ous other forms have been described as per implementation basis.

C. Single sign-on (SSO):

SSO allows a user to use a sin- gle set of login credentials to access multiple systems or applications. While SSO can be convenient, it can also be a vulnerability if the login credentials are com- promised, as an attacker can potentially gain access to multiple systems. This is usually done using OAuth in most scenarios, this suffers from the same problems [3] as described earlier, theoretically its very secure and it doesn't cause problems most of the times but VAPT tools need to analyse the code for various implemen- tation bugs.

D. Bio-metric authentication:

This method uses physical or behavioral characteristics, such as fingerprints, facial recognition, or voice recognition, to verify a user's iden- tity. While biometric authentication can be very secure, it is not always reliable and can be easily defeated if an attacker has access to a copy of the biometric data.

3. Related Works and Case Study

a. Related Studies

Several studies have been conducted to evaluate the effective- ness of vulnerability detection tools. For example, Al-Aziz and Kim [4] conducted a comparative study of several vul- nerability detection tools and found that different tools have varying levels of effectiveness in detecting vulnerabilities. They identified that tools such as Nessus and OpenVAS were more effective in detecting vulnerabilities compared to oth- ers. Similarly, Zhang et al. [5] evaluated the effectiveness of several vulnerability detection tools and found that while these tools are useful in detecting known vulnerabilities, they may not be effective in detecting unknown vulnerabilities.

In addition there have been several other studies evaluating the effectiveness of vulnerability detection tools. For exam- ple, Goyal and Singh [6] evaluated the effectiveness of sev- eral commercial and open-source vulnerability scanners and found that open-source scanners were more effective in de- tecting vulnerabilities. They also found that no single scan- ner was able to detect all vulnerabilities and recommended the use of multiple scanners to increase the chances of de- tecting vulnerabilities.

Similarly, Vaarandi and Vilo [7] conducted a study to evaluate the effectiveness of vulnerability scanners in de- tecting web application vulnerabilities. They found that the most effective scanners were those that used a combination of signature-based and behavior-based detection techniques. They also recommended the use of manual testing in addition to automated scanning to ensure comprehensive vulnerability detection.

Another study by Hars et al. [8] evaluated the effective- ness of vulnerability scanners in detecting vulnerabilities in industrial control systems. They found that existing vulnera- bility scanners were not effective in detecting vulnerabilities specific to industrial control systems and recommended the development of specialized scanners for these systems.

Overall, although all these studies were not focused on au- thentication as this paper is, these studies highlight the im- portance of evaluating the effectiveness of vulnerability de- tection tools and the need for a combination of automated scanning and manual testing to ensure comprehensive vul- nerability detection. They also emphasize the need for spe- cialized scanners for specific types of systems and applica- tions, such as industrial control systems.

b. Case Study

There have been numerous instances where vulnerability as- sessment and penetration testing (VAPT) tools have been instrumental in identifying and mitigating

vulnerabilities in software systems. In this section, we will discuss some ex- amples of incidents where VAPT tools helped bring out bugs. In 2020, a leading e-commerce company in India hired a VAPT service provider to conduct a comprehensive security audit of its web application and mobile app. The VAPT team discovered several critical vulnerabilities, such as SQL in- jection, cross-site scripting, insecure file upload, and broken authentication. The team also performed a simulated attack on the application and successfully compromised sensitive data, such as customer details, payment information, and or- der history. The VAPT report provided detailed recommen- dations on how to fix the vulnerabilities and prevent future breaches. [9]

In 2019, a multinational bank in Europe engaged a VAPT service provider to assess the security of its online banking system. The VAPT team performed a thorough vulnerabil- ity assessment of the system's network infrastructure, web servers, databases, APIs, and web interfaces. The team also conducted a penetration test to exploit the identified vulner- abilities and gain access to privileged accounts, confiden- tial documents, and financial transactions. The VAPT report highlighted the security gaps and suggested best practices on how to enhance the system's security controls. [10]

In 2018, a government agency in Australia contracted a VAPT service provider to evaluate the security of its cloud- based applications. The VAPT team performed a compre- hensive analysis of the applications' architecture, configura- tion, code quality, encryption mechanisms, and access poli- cies. The team also executed a penetration test to bypass the applications' security defenses and access sensitive data stored in cloud storage services. The VAPT report revealed several high-risk vulnerabilities that could have resulted in data leakage or unauthorized modification of data.

However, it's important to note that VAPT tools are not a substitute for a comprehensive security program. While these tools can be effective in detecting certain types of vul- nerabilities, they may not detect all vulnerabilities, partic- ularly those related to social engineering and human error. Therefore, it's crucial for organizations to employ a multi- layered security approach that includes both technical and non-technical measures.

4. Comparison Between VAPT

There are a lot of tools developed for detecting bugs and vul- nerabilities overtime because of its significance in a software based product. These are some of the popular tools currently being used for vulnerability detection in context of authenti- cation.

- **Nessus**: Nessus is a comprehensive vulnerability scan- ning tool that can detect a wide range of security is- sues, including authentication vulnerabilities. It can scan for weak passwords, unsecured authentication pro- tocols, and other vulnerabilities that could compromise authentication.
- OpenVAS: OpenVAS is a free and open source vulner- ability scanner that can detect security issues in authen- tication systems, as well as other areas of an IT infras- tructure. It can be configured to scan for specific types of authentication vulnerabilities, such as password reuse or weak authentication protocols.
- Burp Suite: Burp Suite is a web application security testing tool that includes a suite of tools for testing au- thentication and authorization systems. It can be used to test for common authentication vulnerabilities, such as session hijacking and brute force attacks.
- Metasploit: Metasploit is a popular penetration testing tool that includes a wide range of modules for testing the security of various systems, including authentication systems. It can be used to test for vulnerabilities such as weak passwords, unsecured protocols, and other issues that could compromise authentication.
- OWASP ZAP: OWASP ZAP is a free and open source web application security scanner that can detect vulnerabilities in authentication systems, as well as other areas of web applications. It can scan for common authentication vulnerabilities, such as session fixation and cookie manipulation.
- QualysGuard: QualysGuard is a cloud-based vulnerability management tool that can be used to scan for authentication vulnerabilities, as well as other types of security issues. It can be used to scan networks, servers, and web applications for vulnerabilities.
- Acunetix: Acunetix is a web application security scanner that can be used to test for authentication vulnerabil ities, such as weak passwords and session management issues. It can also scan for other types of web applica- tion vulnerabilities, such as SQL injection and cross-site scripting.
- Nexpose: Nexpose is a vulnerability management tool that can be used to scan for authentication vulnerabilities, as well as other types of security issues. It can scan networks, servers, and web applications for vulnerabili ties.
- Nikto: Nikto is a web server scanner that can be used to test for authentication vulnerabilities, as well as other types of web server vulnerabilities. It can scan

for is- sues such as weak authentication protocols and directory traversal vulnerabilities.

a. Platform support

Platform support is a critical factor to consider when selecting a vulnerability assessment tool. It is essential to choose a tool that supports the platforms used in the development environment. Table 1 provides a comparison of the overall features of vulnerability assessment tools in general. These tools are evaluated based on their support for web, mobile, API, cloud, network, database, and reporting. Burp Suite, Nessus, OpenVAS, Metasploit, Nexpose, Qualys, ZAP, App- Scan, Acunetix, and WebInspect are some of the most widely used vulnerability assessment tools. Burp Suite, Nessus, and OpenVAS provide support for web and network scanning, while Metasploit and Nexpose provide support for network and database scanning. Qualys provides support for web, API, cloud, network, and database scanning. Acunetix provides support for web, mobile, API, and cloud scanning. ZAP and AppScan provide support for web and API scanning. WebInspect provides support for web and API scanning. This table shows the best most popular tools avaible and their support range.

Table 1: Comparison of Overall Features of Vulnerability Assessment Tools in General

Tool	Web	Mobile	API	Cloud	Network	Database	Reporting
Burp Suite	✓	✓	✓				✓
Nessus	✓			✓	✓	✓	✓
OpenVAS	✓			✓	✓	✓	✓
Metasploit	✓	✓	✓		✓	✓	✓
Nexpose	✓			✓	✓	✓	✓
Qualys	✓		✓	✓	✓	✓	✓
ZAP	✓		✓				✓
AppScan	✓	✓	✓				✓
Acunetix	✓	✓	✓	✓			✓
WebInspect	✓		✓				✓
Nikto	✓		✓				
Nmap					✓		
Acunetix	✓	✓	✓	✓			✓

b. Features and Pricing

The table 2 is a comparison of different vulnerability assess- ment tools for their authentication features. It includes infor- mation such as the vendor, license type, supported protocols, and authentication methods for each tool. The supported pro- tocols include HTTP, HTTPS, SSH, FTP, and Telnet, while the authentication methods include Password, SSH Key, Ker- beros, Basic Auth, Digest Auth, and Form-Based. Nessus, OpenVAS, Nikto, Nmap, Burp Suite, Acunetix, Qualys, and AppSpider are included in this table. Nessus, OpenVAS, Nikto, Nmap, and Qualys support a variety of authentica- tion methods, including password-based, SSH key-based, and Kerberos. Burp Suite and Nikto, on the other hand, sup- port only basic and digest authentication. Acunetix and App- Spider support form-based, basic, and digest authentication.

Table 2: Comparison of Features of Vulnerability Assessment Tools for Authentication

Tool	Vendor	License	Supported Protocols	Authentication Methods
Nessus	Tenable	Commercial	HTTP, HTTPS, SSH, FTP, Telnet	Password, SSH Key, Kerberos
OpenVAS	Greenbone	Open-Source	HTTP, HTTPS, SSH, FTP, Telnet	Password, SSH Key, Kerberos
Nikto	CIRT	Open-Source	HTTP, HTTPS	Basic Auth, Digest Auth
Nmap	Nmap Project	Open-Source	HTTP, HTTPS, SSH, FTP, Telnet	Password, SSH Key, Kerberos
Burp Suite	PortSwigger	Commercial	HTTP, HTTPS	Basic Auth, Digest Auth
Acunetix	Acunetix Ltd.	Commercial	HTTP, HTTPS, FTP	Form-Based*, Basic Auth*, Digest Auth*

Qualys	Qualys, Inc.	Commercial	HTTP, HTTPS, SSH	Password, SSH Key, Kerberos
AppSpider	Rapid7	Commercial	HTTP, HTTPS, FTP	Form-Based*, Basic Auth*, Digest Auth*

The comparison table 3 provides a comprehensive overview of the pricing and release dates of some of the most widely used vulnerability assessment tools for authentica- tion. These tools play a crucial role in detecting bugs and vulnerabilities in software products, which is why they have become a significant part of the development process. Nes- sus, OpenVAS, Nikto, Nmap, Burp Suite, Acunetix, Qualys, and AppSpider are some of the most popular tools used for vulnerability detection, and they offer different features and pricing plans to cater to the varying needs of organizations. While some tools like Nessus and Burp Suite come with a commercial license and a starting price, others like Open- VAS and Nikto are open-source and free to use. Qualys and AppSpider are examples of tools that require users to contact the vendor for pricing information. Understanding the pric- ing in conjunction with 2 and 1 is important before selecting a tool.

Table 3: Comparison of Pricing, and Release Dates of Vulnerability Assessment Tools for Authentication

Tool	Vendor	License	Pricing (USD)	Release Dates
Nessus	Tenable	Commercial	Starts at 2, 190*/year*	1998
OpenVAS	Greenbone	Open-Source	Free	2005
Nikto	CIRT	Open-Source	Free	2001
Nmap	Nmap Project	Open-Source	Free	1997
Burp Suite	PortSwigger	Commercial	Starts at 399*/year*	2006
Acunetix	Acunetix Ltd.	Commercial	Starts at 4, 990*/year*	2004
Qualys	Qualys, Inc.	Commercial	Contact vendor for pricing	1999
AppSpider	Rapid7	Commercial	Contact vendor for pricing	1997

c. Problems with current tool-chain

As mobile and native applications become increasingly pop ular, there is a growing need for vulnerability assessment tools to support these platforms. However, most of the currently available tools are designed primarily for web applications and have limited support for mobile and native appli cations. Tools like Nessus, OpenVAS, Nikto, and Nmap are primarily designed for network and web application scanning and have limited support for mobile and native applications. Acunetix and Burp Suite do provide some support for mobile applications, but it is limited and not as robust as their web application support. This gap in tool support for mobile and native applications presents a significant challenge for organizations seeking to ensure the security of their software products across all platforms.

As mentioned, The current vulnerability assessment tools have limited support for mobile and native applications. However, with the increasing number of desktop applications for Windows, Mac, and Linux, it is important to note that these tools also have limited support for desktop applications. In addition, static analysis tools can help with the situation, but they are not ideal for VAPT purposes since they are de- signed to analyze the source code for potential vulnerabilities rather than testing the application in a live environment.

Desktop applications are a crucial component of modern software development, and the number of desktop applications has increased rapidly in recent years. However, traditional vulnerability assessment tools are mostly focused on web applications and lack the necessary features to thoroughly assess desktop applications. This lack of support for desktop applications creates a significant gap in the security of software products.

One solution to this problem is the use of static analysis tools. These tools analyze the source code of an application to identify potential vulnerabilities. However, static analysis tools have limitations as they are not designed to test applications in a live environment, and they may not be able to detect certain types of vulnerabilities, such as those that occur at runtime.

Moreover, the use of static analysis tools is only effective in detecting vulnerabilities during the development phase of an application. Once an application is deployed, dynamic analysis tools become more effective in identifying vulnerabilities. As a result, traditional vulnerability assessment tools need to expand their capabilities to cover not only web applications but also desktop applications and mobile applica- tions.

Another challenge with current VAPT tools is that many of them are geared towards specific types of vulnerabilities, which means that organizations may need to use multiple tools in order to identify all potential vulnerabilities. For

ex- ample, some tools may be designed to test for network vulnerabilities, while others may be better suited for testing web applications. This can lead to a lot of extra work for security teams, who may need to run multiple tools in order to get a complete picture of their organization's security posture.

Finally, another challenge with current VAPT tools is that many of them rely on static analysis to identify potential vul- nerabilities. While static analysis tools can be helpful, they are not ideal for VAPT testing, as they are not able to accurately simulate the behavior of real-world attackers. This means that security teams may be missing out on potential vulnerabilities that could be exploited by attackers, simply because their static analysis tools didn't identify them.

d. Static analysis tools

In addition to general VAPT tools staitc analysis tools are also important for finding bugs during the development cycle. Static analysis tools are a valuable resource for identify- ing implementation bugs and other vulnerabilities in soft- ware code, including issues related to encryption keys. These tools analyze the source code of an application or system and identify potential vulnerabilities before the code is executed. One common application of static analysis tools is in identifying implementation bugs related to authentication mech- anisms. For instance, a static analysis tool can detect vul nerabilities in an application's password storage mechanism, such as plaintext password storage or weak password hashing algorithms. By identifying these vulnerabilities early on in the development process, developers can take steps to rec- tify them before they are exploited by malicious actors. Various static analysis tools can effectively detect authentication

vulnerabilities and encryption key issues. These include:

- Fortify: Fortify is a static code analysis tool that is de- signed to identify security vulnerabilities in software code. It can be used to analyze a wide range of pro- gramming languages, including Java, .NET, C++, and Python. Fortify provides a comprehensive set of vulner- ability detection rules that can identify issues such as buffer overflows, SQL injection, and cross-site script- ing (XSS).
- Coverity: Coverity is a code analysis tool that is focused on identifying critical software defects. It can be used to analyze C, C++, Java, and C# code, and provides a set of advanced analysis techniques such as control flow analysis and data flow analysis. Coverity is known for its ability to identify complex defects such as memory leaks and concurrency issues.
- SonarQube: SonarQube is an open source tool that pro- vides continuous code inspection for identifying code smells, bugs, and security vulnerabilities. It supports a wide range of programming languages, including Java, C#, and Python, and provides a set of quality metrics and visualizations for tracking code quality over time.
- PVS-Studio: PVS-Studio is a static analysis tool that is designed to detect errors and potential vulnerabilities in C and C++ code. It provides a set of advanced analysis techniques such as data flow analysis and symbolic exe- cution, and can detect issues such as null pointer deref- erences, division by zero, and buffer overflows.
- Klocwork: Klocwork is a code analysis tool that is fo- cused on detecting software defects such as race con- ditions, memory leaks, and buffer overflows. It sup- ports C, C++, Java, and C# code, and provides a set of advanced analysis techniques such as interprocedural analysis and path-sensitive analysis.
- Checkmarx: Checkmarx is a static analysis tool that is designed to identify security vulnerabilities in software code. It supports a wide range of programming lan- guages, including Java, .NET, and Python, and provides a comprehensive set of vulnerability detection rules for identifying issues such as SQL injection, XSS, and au- thentication bypass.
- ESLint: ESLint is a static analysis tool for JavaScript development that is designed to identify code quality is- sues and enforce coding standards. It can detect poten- tial security vulnerabilities such as XSS and injection attacks, and provides a set of quality metrics and visu- alizations for tracking code quality over time.

JSHint: JSHint is a static analysis tool for JavaScript de- velopment that is focused on identifying potential errors and code quality issues. It provides a set of customiz- able rules for enforcing coding standards and detecting potential security vulnerabilities.

- Flow: Flow is a static type checker for JavaScript de- velopment that is designed to detect potential type- related errors in code. It can detect issues such as null pointer dereferences and type mismatches, and provides a set of annotations for enforcing typing constraints in JavaScript code.

Table 4: Comparison of Features for Popular Static Analysis Tools.

Tool	Programming Languages	Open Source	Integration with IDE	GUI	Reporting	Community Support
Fortify	C/C++, Java, .NET			✓	✓	✓
Coverity	C/C++, Java, Python, Ruby, C#		✓	✓	✓	✓
SonarQube	27 languages, including C/C++, Java, Python, JavaScript	✓	✓	✓	✓	✓
PVS-Studio	C/C++, C#		✓	✓	✓	✓
Klocwork	C/C++, Java, C#, JavaScript, Python		✓	✓	✓	✓
Checkmarx	C/C++, Java, .NET, JavaScript		✓	✓	✓	✓
ESLint	JavaScript	✓	✓	✓	✓	✓
JSHint	JavaScript	✓		✓	✓	✓
Flow	JavaScript	✓	✓	✓	✓	✓

Based on the comparison table in Table 4, we can see that there are several popular static analysis tools available for detecting implementation bugs and other vulnerabilities in software code, including Fortify, Coverity, SonarQube, PVS- Studio, Klocwork, Checkmarx, ESLint, JSHint, and Flow. Each tool has its own set of strengths and weaknesses, with some tools offering better support for certain programming languages and others offering more comprehensive vulnera- bility detection capabilities. For example, Fortify and Cover- ity are known for their advanced vulnerability detection ca- pabilities and support for a wide range of programming lan- guages, while ESLint, JSHint, and Flow are specialized tools for JavaScript development. When selecting a static analysis tool for VAPT, it's important to consider the specific needs and requirements of your organization, as well as the pro- gramming languages and technologies used in your software development projects.

Static analysis tools are not foolproof, however. These tools are best used in conjunction with other vulnerability detection tools and techniques to provide a comprehensive approach to vulnerability detection and mitigation. Addition- ally, static analysis tools may generate false positives or miss certain vulnerabilities altogether, making it essential to use them with other methods of vulnerability detection.

e. Zero day bugs : Drawbacks of static analysis tools

Zero-day vulnerabilities refer to previously unknown secu- rity vulnerabilities that attackers can exploit. They are called "zero-day" because there is zero-day between the discovery of the vulnerability and the first attack exploiting it. There- fore, zero-day vulnerabilities pose a significant threat to sys- tems, and detecting them before attackers is critical. The de- tection of zero-day vulnerabilities typically follows a work- flow that involves identifying and analyzing vulnerabilities, reproducing the attack, and verifying the fix.

Static analysis tools can help identify code-level vulnera- bilities during the development phase, reducing the chances of zero-day vulnerabilities. However, since zero- day vulner- abilities are unknown, static analysis tools might not be effective in detecting them. Therefore, VAPT teams typically use a combination of tools, techniques, and knowledge to identify zero-day vulnerabilities.

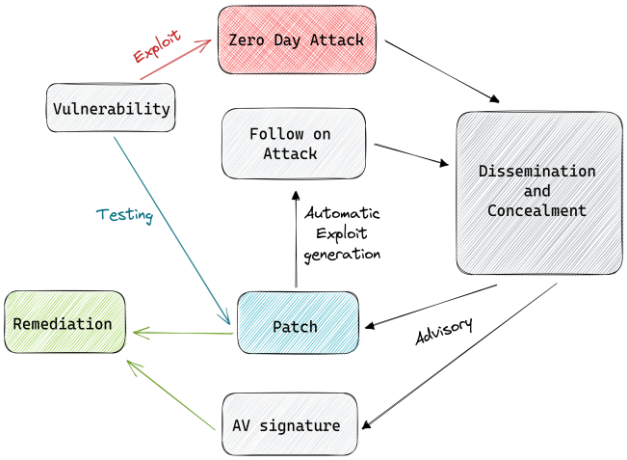

Fig. 1: Workflow diagram of a vulnerability detection and remedy.

In the first step of the zero-day vulnerability detection workflow, the VAPT team scans the system for any potential vulnerabilities. This is where vulnerability assessment tools such as Nessus, OpenVAS, and Qualys come in. They help to identify known vulnerabilities in the system that could be exploited by attackers.

In the second step, the VAPT team analyzes the vulnera- bilities that have been identified. This step involves attempt- ing to exploit the vulnerabilities to determine their impact on the system. This is where penetration testing

tools such as Metasploit and Burp Suite are used. They enable the VAPT team to simulate attacks and determine how vulnerabilities can be exploited.

In the third step, the VAPT team verifies the fix to en- sure that the vulnerability has been resolved. This step involves retesting the system to ensure that the vulnerability is no longer present. Once the fix has been verified, the VAPT team provides recommendations to improve the security pos- ture of the system.

In conclusion, detecting zero-day vulnerabilities is a challenging task that requires a combination of tools and expertise. While static analysis tools can help to identify code-level vulnerabilities, VAPT teams typically use a combination of vulnerability assessment and penetration testing tools to identify and analyze zero-day vulnerabilities. The key to detecting zero-day vulnerabilities is to stay up-to-date with the latest security trends, techniques, and tools.

5. Future Works and Conclusions

In conclusion, vulnerability assessment tools in conjunction with static analysis tools are essential for identifying and ad- dressing security vulnerabilities in authentication systems. Through our evaluation of several popular tools, we have found that the current generation of tools provides reliable and effective detection of vulnerabilities in web-based authentication systems. However, it is important to note that most of these tools have limited support for native desk- top applications and mobile devices, which are becoming increasingly prevalent in modern computing environments. Thus, while current tools are highly useful, there remains a need for continued research and development to improve their capabilities, particularly in the area of native applica- tion authentication assessment. Nevertheless, the tools we evaluated in this research paper are highly recommended for organizations seeking to improve the security of their web- based authentication systems. Perhaps future works can be based on these findings.

References

[1] M. Tolbert, “Vulnerabilities of multi-factor authentication in modern computer networks,” Ph.D. dissertation, Worcester Polytechnic Insti- tute, 2021.

[2] Y. He, W. Wang, Y. Teng, Q. Wang, M. Wang, and J. Lin, “2022 ieee wireless communications and networking conference (wcnc),” 2022, pp. 992–997.

[3] PortSwigger. (2022) Oauth 2.0 authentication vulnerabilities. portswigger.net https : / / portswigger . net / web-security / oauth (10/07/2022).

[4] Al-Aziz and H. Kim, “Comparative study of vulnerability detec- tion tools,” International Journal of Advanced Computer Science and Applications, vol. 6, no. 7, pp. 214–220, 2015.

[5] Y. Zhang, X. Chen, and D. Chen, “Evaluating vulnerability detection tools,” Journal of Cyber Security Technology, vol. 1, no. 3, pp. 164– 178, 2017.

[6] Goyal and Y. Singh, “A comparative study of commercial and open source vulnerability scanners,” International Journal of Computer Sci- ence and Mobile Computing, vol. 8, no. 10, pp. 78–85, 2019.

[7] R. Vaarandi and M. Vilo, “Evaluating web application vulnerability scanners,” Journal of Information Security and Applications, vol. 27, pp. 63–73, 2016.

[8] Hars, Y. Liu, and S. Jajodia, “Evaluating the effectiveness of vulner- ability scanners for industrial control systems,” Computers & Security, vol. 53, pp. 73–93, 2015.

[9] S. T. P. Ltd. How we helped india’s leading e-commerce company secure its web application and mobile app with vapt. [Online]. Available: https://securelayer7.net/blog/how-we-helped-indias-leading-e-commerce-company-secure-its-web-application-and-mobile-app-with-vapt/

[10] C. C. Ltd. How we performed vapt for a multinational bank in europe and improved its online banking security. [Online]. Available: https://cybersecconsulting.com/case-studies/vapt-for-bank.html