USE OF EXPERT SYSTEMS TO PREDICT ATTACKS ON WEB-BASED SERVERS

M. Zainal Ariffin, Hildhan Fauzul Hakim

Department of Electrical Engineering, Universitas Negeri Malang, Malang 65145, Indonesia
*Corresponding author, email: hildhan.fauzul.2005356@students.um.ac.id

doi: 10.17977/um068.v4.i2.2024.3

Keywords
Cyber Attacks
System Security
Expert System
Attack Prediction
Penetration Testing
OWASP Zap

Abstract
Cyber-attacks are on the rise, and various types of threats can compromise data confidentiality, integrity, and availability. Reports from the National Cyber and Crypto Agency (BSSN) and research by Check Point indicate a significant increase in cyber-attacks. These attacks often occur due to a lack of understanding and security testing of systems. In this context, the fundamental rules of the CIA (Confidentiality, Integrity, and Availability) become a crucial foundation for system security. Self-testing through penetration testing methods emerges as a solution to identify security vulnerabilities. Therefore, this research aims to develop an expert system using the OWASP Zap penetration testing tool to predict attacks on web-based servers. Utilizing a rule-based algorithm, the output of this expert system will provide results containing the type of attack, CIA classification, score, solutions, and more. In this study, testing and evaluation of the expert system are conducted on domains within the State University of Malang as the target. The test results indicate a satisfactory expert system performance with an accuracy rate of 91.62 percent. This evaluation is expected to provide a comprehensive insight into the expert system's performance in securing the system, enabling developers or campus administrators to address any issues promptly.

1. Introduction

Cyber Attacks are attempts to steal, alter, or destroy data with unauthorized access to computer systems. These attacks are often orchestrated by individuals or specific groups for political, criminal, or personal purposes to either disrupt or gain unauthorized access to confidential information [1].

According to the annual report by the *National Cyber and Crypto Agency* (BSSN), in 2022, Indonesia recorded 976,429,996 cases of cyber attacks. Additionally, BSSN reported a total of 236 cyber complaints in 2022, showing a significant increase compared to the previous year, which had only 79 complaints. Based on the 2022 complaint data, the types of cyber attacks conducted were misconfigurations (37%), cybercrimes or fraud (21%), ransomware (15%), and others (27%) [2].

Furthermore, research conducted by Check Point, an Israel-based cybersecurity company, revealed a 38% increase in cyber attacks in 2022 compared to the previous year. This upward trend is expected to continue annually, especially with the emergence of AI technologies such as ChatGPT, which can diversify and enhance cyber attacks [3]. This aligns with the findings of Gupta et al. [4], where ChatGPT can assist hackers in performing malicious actions such as phishing, payload generation, ransomware creation, and more.

The various mentioned attacks underscore the negative impact of human negligence due to a lack of understanding of system security. Faced with target and time constraints, developers often prioritize the functionality of a system over testing its security [5] [6].

System security is closely related to the three fundamental CIA rules: 1. Confidentiality, 2. Integrity, and 3. Availability. These three basic principles serve as a reference for identifying security vulnerabilities in a system [7]. If these three fundamental factors cannot be met, the system can be considered insecure or susceptible to unauthorized access [8].

To address this, a self-test security solution is proposed. Self-testing involves evaluating the security of a system to identify vulnerabilities and promptly address them [9]. Penetration testing is a self-testing method, akin to hacking activities but conducted legally [10].

Therefore, this research aims to develop an expert system capable of predicting web-based server attacks using penetration testing methods. The chosen tool for attack detection in this expert system is OWASP Zap, an open-source penetration testing tool specifically designed to assess website security vulnerabilities. After testing, the expert system will generate a report containing the attack type and classification based on CIA rules, scores, solutions, and more.

The choice of an expert system is motivated by its knowledge base, allowing non-experts to use it effectively [11]. In a prior study by Taufiq et al., an expert system for predicting diabetes proved beneficial in helping users improve their lifestyles by providing insights into the risk of diabetes. In the context of system security, an expert system can mitigate the risk of cyber-attacks through its knowledge.

This research focuses on testing and evaluating the expert system using the State University of Malang as the testing target. As an educational institution, the State University of Malang is a relevant choice to assess the reliability of the expert system in detecting attacks. The expected outcome of this evaluation is to provide a comprehensive understanding of the expert system's performance, enabling developers or system administrators in the campus environment to identify and address security vulnerabilities more effectively.

2. Method

In this section, the research methodology and the algorithm used in the expert system (rule-based) will be explained. The research method consists of the following stages.

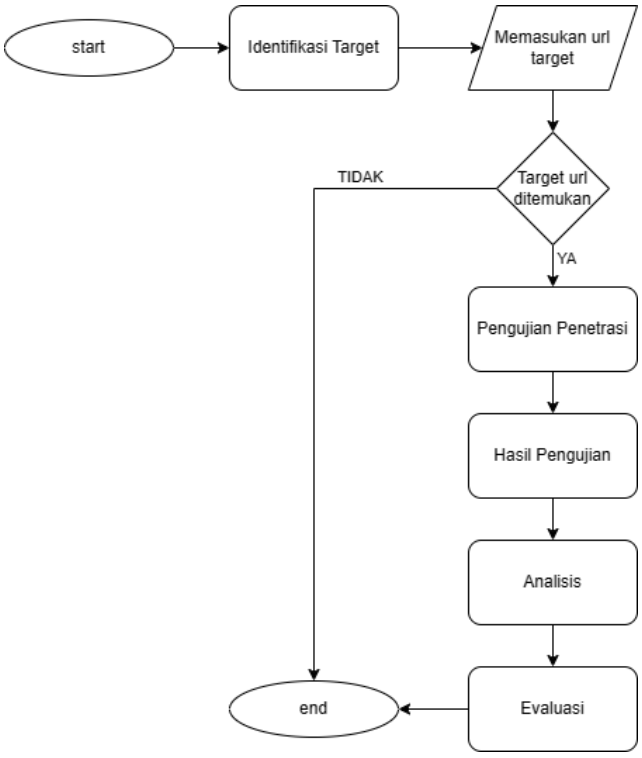


Figure 1. Flowchart research method

2.1. Target Identification (Foot printing)

The first step in this research is to identify the target for prediction, such as finding a website's IP address or URL. In this study, the researcher identified websites within the um.ac.id domain (State University of Malang) by selecting 8 websites within that domain. The selected domains include journal2.um.ac.id, konseling.um.ac.id, mulok.library.um.ac.id, pakar.um.ac.id, repository.um.ac.id,

tutorial.um.ac.id, sipejar.um.ac.id, and elektro.um.ac.id. The decision to choose eight domains was based on time constraints and to avoid a heavy testing burden, considering each domain requires more effort and time when tested manually. Additionally, testing with eight domains is an improvement from the previous research conducted by Hidayatullah & Saptadiaji [12], which focused on five target domains at Universitas ARS.

2.2. Automated Testing

Automated testing is the penetration testing stage, which is conducted automatically using the expert system. For testing, the expert system uses Python version 3.10.11 and the library zapv2.4 , a penetration testing tool from OWASP Zap. Overall, the flow of automated testing by the expert system can be seen in the flowchart in Figure 2. The first step is to enter the URL or IP of the website into the provided input form. The expert system then performs scanning and checks whether the user input is correct. After finding the target URL, the expert system performs penetration testing and generates a dashboard report. This dashboard will contain reports on attacks or vulnerabilities found, CIA classification, scores, solutions, and more.

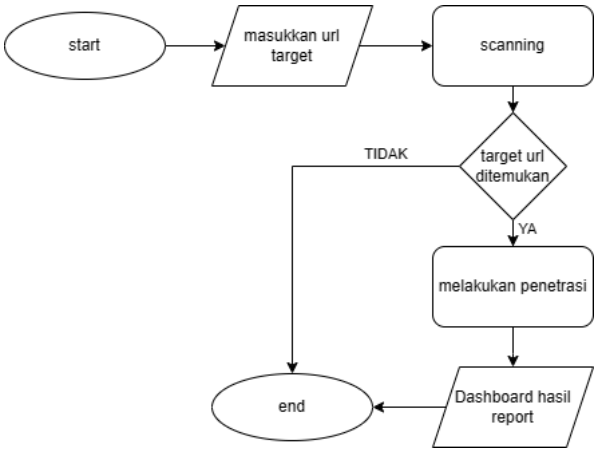


Figure 2. Flowchart expert system

There is a background on why the researcher uses OWASP Zap as a penetration testing tool. Based on research conducted by Albahar et al. [13], there are six pentest tools: OWASP Zap, Burp Suite Professional, Qualys WAS, Arachni, Wapiti3, and Fortify WebInspect. Albahar et al. divided these six tools into two usage cases: commercial and non-commercial. The testing found that OWASP Zap (non-commercial) has a score almost equivalent to that of commercial tools.

Table 1. Comparison of scores for six pentest tools

Use Case	Tools	Score
Commercial	Burp Suite Professional	38
	Qualys WAS	36
	Fortify WebInspect	32
Non-commercial	OWASP Zap	32
	Wapiti3	15
	Arachni	20

2.3. Manual Testing

Manual testing is the stage of manually penetrating the target. This testing can be done by injecting SQL or XSS payloads into vulnerable areas such as login forms, search forms, and URLs. In addition to payloads, manual testing can be performed by inspecting elements or using the "curl" command to gather header information. Manual testing results will be used to evaluate automated testing for detected attacks.

2.4. Testing Results

The testing results include the dashboard report generated by the expert system. The results will include at least the following points:

1. Risk Category: Vulnerability category based on low, medium, and high.
2. Attack Type: Type of attack or vulnerability that occurred.
3. CIA Classification: Classification of attacks with CIA basic rules.
4. Score: Score for each attack and its average.
5. Detail: Details of each attack found, such as description, evidence, confidence, solution, and so on.

2.5. Analysis

The analysis phase involves a detailed examination of the testing results, focusing on Risk Category, Attack Type, CIA Classification, and Score.

2.6. Evaluation

The evaluation stage validates the automated testing conducted by the expert system or OWASP Zap against manual testing. The evaluation is structured with TRUE and FALSE values. To calculate accuracy for each domain, the following formula is used:

$$\text{Accuracy}=(\text{Number of TRUE value})/(\text{Total Attacks}) \times 100\%$$

The total accuracy is then calculated as the average accuracy across all domains:

$$\text{Final Accuracy}= (\text{Accuracy for each domain})/8 \times 100\%$$

The number 8 is derived from the total number of domains tested.

2.7. Rule-Based Algorithm

In this research, the researcher employs a rule-based method or algorithm to classify CIA rules (Confidentiality, Integrity, and Availability) and assign scores. The rule-based algorithm operates by applying predefined rules and then classifying based on these rules.

The rule-based method is an inference technique used to make decisions based on predefined rules or knowledge bases in the expert system. The knowledge base includes a table of attack types and CIA rules. The correlation between the table of attack types and CIA rules is presented below:

Table 2. Correlation between attack types and CIA rules

Jenis Serangan	Aturan CIA
S1- Directory Browsing	*Confidentiality* (C)
S2- Private IP Disclosure	*Confidentiality* (C)
S3 - Session ID in URL Rewrite	*Integrity* (I)
S4 - Referer Exposes Session ID	*Integrity* (I)
S5 - Path Traversal	*Integrity* (I)
S6 - Remote File Inclusion	*Integrity* (I)
S7 - Source Code Disclosure – Git	*Confidentiality* (C)
S8 - Source Code Disclosure – SVN	*Confidentiality* (C)
S9 - Source Code Disclosure - File Inclusion	*Confidentiality* (C)
S10 - Vulnerable JS Library	*Integrity* (I)
S11 - In Page Banner Information Leak	*Confidentiality* (C)
S12 - Cookie No HttpOnly Flag	*Confidentiality* (C)
S13 - Cookie Without Secure Flag	*Confidentiality* (C)
S14 - Re-examine Cache-control Directives	*Confidentiality* (C)
S15 - Web Browser XSS Protection Not Enabled	*Integrity* (I)
S16 - Cross-Domain JavaScript Source File Inclusion	*Integrity* (I)
S17 - Content-Type Header Missing	*Availability* (A)
...	...

Source: https://www.zaproxy.org/docs/alerts/ (2023).

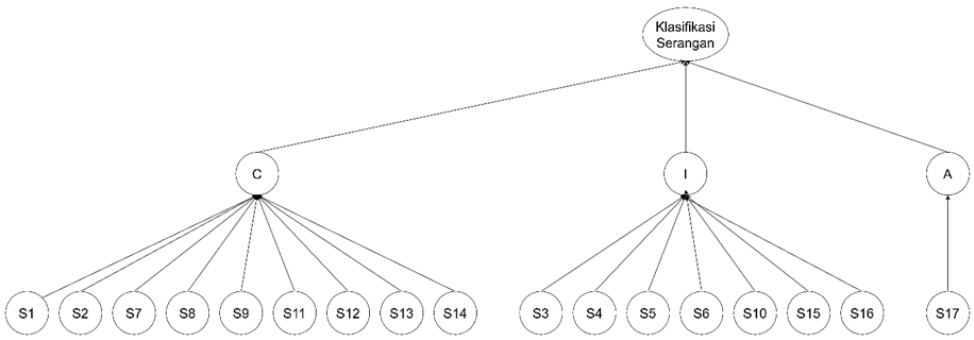


Figure 3. Inference flow model

To view the complete correlation between attack types and CIA rules, please refer to Appendix 1.

3. Result and Discussion

3.1. Implementation of the Expert System

The scanning process is the first step in the expert system after the user enters the target URL or IP. In this test, the researcher used eight targets within the domain of um.ac.id or Universitas Negeri Malang. The tested targets include: journal2.um.ac.id, konseling.um.ac.id, mulok.library.um.ac.id, pakar.um.ac.id, repository.um.ac.id, tutorial.um.ac.id, sipejar.um.ac.id, and elektro.um.ac.id. After entering the target URL, the Expert System will scan the target using the OWASP Zap API to determine if the target URL is found. Once the URL is found, penetration testing will be performed on the target.

After the penetration testing process is complete, a list of possible attacks on the target website will be displayed. The output of the penetration testing can be seen in the following figure:


```
{
    "sourceid": "3",
    "other": "",
    "method": "GET",
    "evidence": "<script type=\"text/javascript\" src=\"//www.google.com/jsapi\"></script
    "pluginId": "10017",
    "cweid": "829",
    "confidence": "Medium",
    "wascid": "15",
    "description": "The page includes one or more script files from a third-party domain
    "messageId": "1048",
    "inputVector": "",
    "url": "https://journal2.um.ac.id/?locale=en-US",
    "tags": {
        "OWASP_2021_A08": "https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_
    },
    "reference": "",
    "solution": "Ensure JavaScript source files are loaded from only trusted sources, an
    "alert": "Cross-Domain JavaScript Source File Inclusion",
```


Figure 4. Penetration Testing Output

The above image shows an example of the total attacks found on the target. The attack output will be in JSON format and will be stored in the database. The output results from the penetration testing stored in the database will be classified into three categories based on the CIA rules (Confidentiality, Integrity, and Availability).

The score index is the final step in the expert system. The score index for each attack is determined using the parameters of confidence and risk. The final program of the expert system will be shown in the following figure:

Overall Score

No	Alert	Confidence	Risk	Score
1	Content Security Policy (CSP) Header Not Set	High	Medium	7
2	Strict-Transport-Security Header Not Set	High	Low	4
3	Cross-Domain JavaScript Source File Inclusion	Medium	Low	3
4	Cookie No HttpOnly Flag	Medium	Low	3
5	Cookie without SameSite Attribute	Medium	Low	3
6	Cookie Without Secure Flag	Medium	Low	3
7	HTTPS to HTTP Insecure Transition in Form Post	Medium	Medium	6
8	Secure Pages Include Mixed Content (Including Scripts)	Medium	Medium	6
9	Vulnerable JS Library	Medium	Medium	6

Overall Score : 4.7

Figure 5. Final Score Index Result

The final score index results indicate how vulnerable the website is to attacks. It is emphasized here that a high score comes from high confidence and vulnerability. Therefore, the higher the obtained score, the higher the potential for dangerous attacks. The determination of the score index itself is based on the Common Vulnerability Scoring System (CVSS), an open industry standard for assessing attacks on systems based on research by the National Infrastructure Advisory Council (NIAC) [14].

3.2. Risk Category

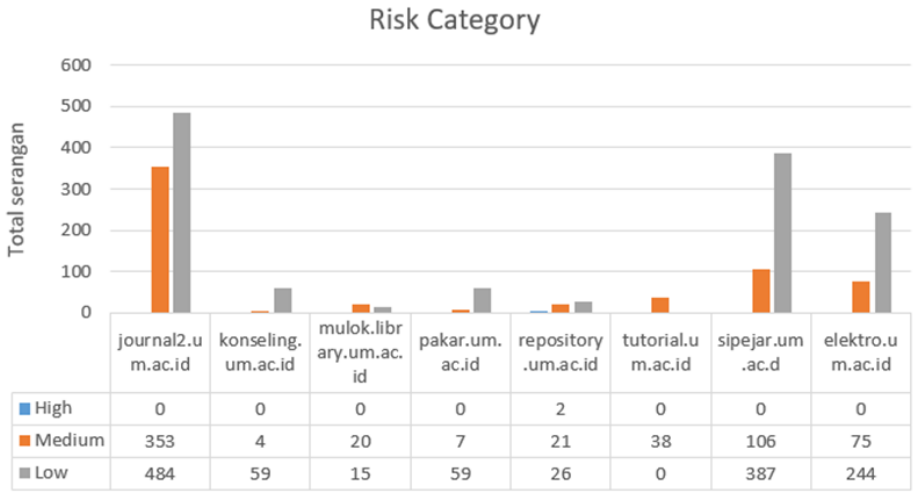


Figure 6. Bar Chart of Risk Category in the um.ac.id domain

The above figure shows a bar chart diagram of security vulnerabilities found by the expert system. Vulnerability categories are divided into three categories: High, Medium, and Low, where each category represents the effects of the security vulnerabilities found. Looking at the data above, the low category dominates most domains, with a total of 1274 low-level attacks. Except for the mulok.library.um.ac.id and tutorial.um.ac.id domains, which have fewer low categories compared to other categories, even zero. Next, in the medium category, a total of 624 medium-level attacks were found. All domains have at least one medium attack found. Then there is interesting data on the repository.um.ac.id domain, where 2 high-level attacks were found, making it the only high-level attack found.

To better understand what attacks or vulnerabilities were found, the following is a table of attacks or vulnerabilities found in the um.ac.id domain:

Table 3. Attacks Found in Each Domain

Detected Attacks	Risk	Domain							
		J	K	Mu	P	R	T	S	E
Big Redirect Detected (Potential Sensitive Information Leak)	L							√	
Content Security Policy (CSP) Header Not Set	M	√	√	√	√	√	√	√	√
Cookie No HttpOnly Flag	L	√	√		√			√	
Cookie without SameSite Attribute	L	√	√	√				√	
Cookie Without Secure Flag	L		√		√			√	
Cross-Domain JavaScript Source File Inclusion	L	√	√			√			√
Cross Site Scripting (Reflected)	H					√			
Format String Error	M					√			
Hidden File Found	M			√		√		√	√
Missing Anti-clickjacking Header	M	√							
Server Leaks Information via ""X-Powered-By"" HTTP Response Header Field(s)	L			√					
Strict-Transport-Security Header Not Set	L		√		√			√	√
Vulnerable JS Library	M		√	√	√		√	√	
Weak Authentication Method	M					√			
X-Content-Type-Options Header Missing	L	√						√	

*L = Low, M = Medium, H = High, J = journal2.um.ac.id, K = konseling.um.ac.id, Mu = mulok.library.um.ac.id, P = pakar.um.ac.id, R = repository.um.ac.id, T = tutorial.um.ac.id, S = sipejar.um.ac.id, E = elektro.um.ac.id

Each domain has two or more vulnerabilities found, as seen in the table above. Each identified vulnerability has its own number of instances, such as the tutorial.um.ac.id domain, which has two vulnerabilities found: Content Security Policy (CSP) Header Not Set with a total of 37 attacks and Vulnerable JS Library with a total of 1 attack, resulting in a total of 38 medium attacks found (See Figure 6).

The table above shows that the Content Security Policy (CSP) Header Not Set vulnerability is found in every domain. Content Security Policy (CSP) is an additional security layer that helps detect and mitigate specific types of attacks, including Cross-Site Scripting (XSS) and data injection attacks. In other words, CSP helps control which resources can load on that web.

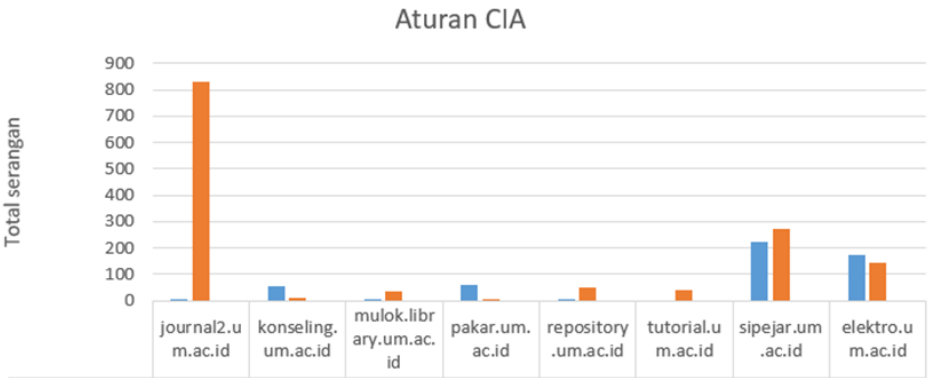
In addition to CSP, three other vulnerabilities often appear: Cookie No HttpOnly Flag, Cross-Domain JavaScript Source File Inclusion, and Vulnerable JS Library. What is interesting here is the Vulnerable JS Library vulnerability, where it turns out that many domains still use outdated Javascript files. However, the use of outdated versions can be exposed to high-risk potential attacks such as XSS.

If we look at the table again, there is the Cross-Site Scripting (Reflected) or XSS attack on the repository.um.ac.id domain. This attack is very dangerous and falls into the high-risk category. However, after the researcher tested it manually, it seemed that the attack was not real. In other words, it is safe from high-risk XSS attacks.

Then there is another interesting finding, namely Hidden File Found. Hidden information obtained from this attack includes server status and composer. lock, and info.php. These files should not be publicly visible.

Regarding confidentiality, the sipejar.um.ac.id domain has the highest value of 222, followed by the elektro.um.ac.id domain with a value of 173. Both domains may not have adequately secured confidentiality by hiding sensitive data, given their relatively high values compared to other domains. On the other hand, the tutorial.um.ac.id domain has a value of zero in terms of confidentiality. This indicates that the tutorial.um.ac.id domain has implemented confidentiality by hiding sensitive data.

3.3. CIA Classification



	journal2.um.ac.id	konseling.um.ac.id	mulok.library.um.ac.id	pakar.um.ac.id	repository.um.ac.id	tutorial.um.ac.id	sipejar.um.ac.id	elektro.um.ac.id
Confidentiality	6	54	1	59	1	0	222	173
Integrity	831	9	34	7	48	38	271	146
Availability	0	0	0	0	0	0	0	0

Figure 7. Bar Chart of CIA Classification in the um.ac.id Domain

Moving on to another aspect regarding integrity, the journal2.um.ac.id domain has the highest value, 831. This value is quite significant compared to other domains' integrity values. This is because, besides the high number of attacks in this domain, the attacks also fall within the integrity aspect. If we observe, attacks with the integrity aspect undoubtedly occur in all domains. This suggests that domains within Universitas Negeri Malang may still not demonstrate their security in terms of integrity. The konseling.um.ac.id and pakar.um.ac.id domains can be considered relatively good in integrity security as they show low values of 9 and 7, respectively.

Next, regarding availability, all domains have the same value: zero. This is interesting because the expert system does not detect attacks that threaten availability. In other words, domains within UM have implemented security regarding availability, ensuring that resources remain available when needed. From the above data, it can also be concluded that the likelihood of finding availability aspects is very small, even zero.

3.4. Score

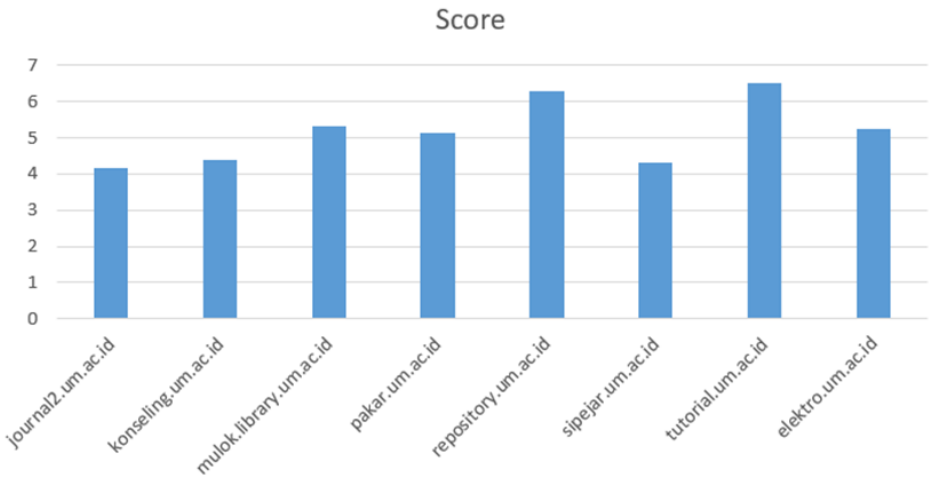


Figure 8. Bar Chart of Scores in the um.ac.id Domain

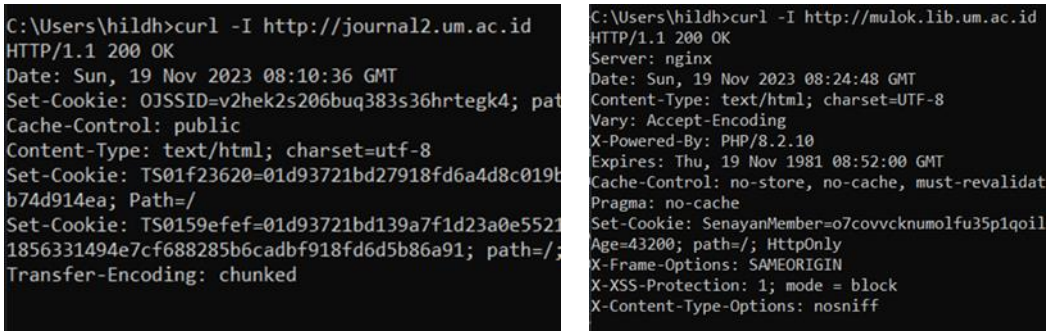
Entering the final discussion, which is the score. The scores in the above diagram represent the average score for each domain. It can be seen that the repository.um.ac.id and tutorial.um.ac.id domains have relatively high scores of 6.3 and 6.5, respectively. Both domains have high scores because attacks with the High category have been found, as in the case of the repository.um.ac.id

domain. The tutorial.um.ac.id domain, although it does not have High-category attacks, still has a high score because the attacks found on this domain have only one category: Medium.

Then, if we look at the average score across all domains at Universitas Negeri Malang, the average score is 5.16. This score can be classified as Medium, meaning the security score in UM domains can be considered in the middle. It is neither too small, indicating safety, nor too large, indicating danger.

3.5. Evaluation

The implementation of the evaluation involves manual testing, which includes inspecting elements or using the "curl" command. For example, in the journal2.um.ac.id domain, a vulnerability was found in the Cookie No HttpOnly Flag. By manually checking using the "curl" command in the cmd, the following results were obtained:



(a) (b)

Figure 9. Cookie No HttpOnly Flag: (a) TRUE; (b) FALSE

The "curl -I" command displays headers on the domain. The image above shows an example result of the curl command that shows the Cookie. If HttpOnly is not found in the cookie, then the Cookie No HttpOnly Flag attack is considered TRUE, and vice versa.

After obtaining TRUE and FALSE results in manual testing against automatic testing, accuracy values can be calculated for each domain and their averages, as presented in the table below:

Table 4. Evaluation and Accuracy

No	Domain	Detected Attacks	Validation	Accuracy
1	http://journal2.um.ac.id	*Content Security Policy (CSP) Header Not Set*	*TRUE*	100%
		Missing Anti-clickjacking Header	*TRUE*	
		X-Content-Type-Options Header Missing	*TRUE*	
		Cookie No HttpOnly Flag	*TRUE*	
		Cookie without SameSite Attribute	*TRUE*	
		Cross-Domain JavaScript Source File Inclusion	*TRUE*	
2	https://konseling.um.ac.id	*Content Security Policy (CSP) Header Not Set*	*TRUE*	85,71%
		Strict-Transport-Security Header Not Set	*FALSE*	
		Cookie No HttpOnly Flag	*TRUE*	
		Cookie without SameSite Attribute	*TRUE*	
		Cookie Without Secure Flag	*TRUE*	
		Cross-Domain JavaScript Source File Inclusion	*TRUE*	
		Vulnerable JS Library	*TRUE*	
3.	http://mulok.lib.um.ac.id	*Content Security Policy (CSP) Header Not Set*	*TRUE*	100%
		Cookie without SameSite Attribute	*TRUE*	
		"Server Leaks Information via ""X-Powered-By"" HTTP Response Header Field(s)"	*TRUE*	
		Vulnerable JS Library	*TRUE*	
		Hidden File Found	*TRUE*	
4.	https://pakar.um.ac.id	*Content Security Policy (CSP) Header Not Set*	*TRUE*	80%
		Strict-Transport-Security Header Not Set	*FALSE*	
		Cookie No HttpOnly Flag	*TRUE*	
		Cookie Without Secure Flag	*TRUE*	
		Vulnerable JS Library	*TRUE*	
5.	http://repository.um.ac.id	*Content Security Policy (CSP) Header Not Set*	*TRUE*	83,33%
		Cross-Domain JavaScript Source File Inclusion	*TRUE*	
		Weak Authentication Method	*TRUE*	

No	Domain	Detected Attacks	Validation	Accuracy
		Cross Site Scripting (Reflected)	*FALSE*	
		Format String Error	*TRUE*	
		Hidden File Found	*TRUE*	
6.	http://tutorial.um.ac.id	*Content Security Policy (CSP) Header Not Set*	*TRUE*	100%
		Vulnerable JS Library	*TRUE*	
7.	https://sipejar.um.ac.id	*Content Security Policy (CSP) Header Not Set*	*TRUE*	88,88%
		Cookie No HttpOnly Flag	*TRUE*	
		Cookie without SameSite Attribute	*TRUE*	
		Cookie Without Secure Flag	*TRUE*	
		Strict-Transport-Security Header Not Set	*FALSE*	
		Big Redirect Detected (Potential Sensitive Information Leak)	*TRUE*	
		X-Content-Type-Options Header Missing	*TRUE*	
		Vulnerable JS Library	*TRUE*	
		Hidden File Found	*TRUE*	
8.	https://elektro.um.ac.id	*Content Security Policy (CSP) Header Not Set*	*TRUE*	75%
		Cross-Domain JavaScript Source File Inclusion	*TRUE*	
		Hidden File Found	*TRUE*	
		Strict-Transport-Security Header Not Set	*FALSE*	
		Rata-Rata Akurasi		91.62%

The overall evaluation results on the expert system show excellent performance in detecting security vulnerabilities. By averaging the accuracy of all domains, the expert system achieves an accuracy value of 91.62%. The general evaluation results can reflect the reliability and effectiveness of the expert system in detecting attacks.

4. Conclusion

This study implemented and evaluated a rule-based expert system for penetration testing based on OWASP Zap at Universitas Negeri Malang. The evaluation results showed an accuracy level of 91.62%, while the score analysis on university domains described a medium-risk level. Suggestions for future research include using other penetration testing tools, in-depth crawling, and weighted assessments. Developers should periodically update and hide sensitive files and implement a Content Security Policy (CSP) in headers. Overall, this research provides insights and practical recommendations to enhance system security in an academic environment, creating a foundation for developing more effective security solutions in the future.

5. Reference

[1] Microsoft, "What is Cyber Attack?," 2023. [Online]. Available: https://www.microsoft.com/id-id/security/business/security-101/what-is-a-cyberattack.

[2] BSSN, "Lanskap 2022," 20 February 2023. [Online]. Available: https://www.bssn.go.id/lanskap2022/.

[3] C. P. Research, "Check Point 2022 Cyber Security Report," 2023. [Online]. Available: https://resources.checkpoint.com/cyber-security-resources/2022-cyber-security-report.

[4] M. Gupta, C. Akiri, K. Aryal, E. Parker and L. Praharaj, "From chatgpt to threatgpt: Impact of Generative AI in Cybersecurity and Privacy," IEEE Access, vol. 11, pp. 80218-80245, p. Https://doi.org/10.1109/ACCESS.2023.3300381, 2023.

[5] A. M. Dwika, STUDI KEAMANAN SISTEM INFORMASI BERBASIS WORDPRESS TERHADAP SERANGAN SQL INJECTION DI SITUS CAHUNNES.COM, Semarang: UNIVERSITAS NEGERI SEMARANG, 2017.

[6] K. Hughes-Lartey, M. Li, F. E. Botchey and Z. Qin, "Human factor, a critical weak point in the information security of an organization's Internet of things," Heliyon, 7(3), 2021.

[7] G. Guntoro, L. Costaner and M. Musfawati, "ANALISIS KEAMANAN WEB SERVER OPEN JOURNAL SYSTEM (OJS) MENGGUNAKAN METODE ISSAF DAN OWASP (STUDI KASUS OJS UNIVERSITAS LANCANG KUNING)," JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika) 5(1): 45-55, p. Https://doi.org/10.29100/jipi.v5i1.1565, 2020.

[8] A. P. Dewanto, Penetration Testing Pada Domain UUI.AC.ID menggunakan OWASP 10, Yogyakarta: Universitas Islam Indonesia, 2018.

[9] G. Kusuma, "IMPLEMENTASI OWASP ZAP UNTUK PENGUJIAN KEAMANAN SISTEM INFORMASI AKADEMIK," Jurnal Teknologi Informasi: Jurnal Keilmuan dan Aplikasi Bidang Teknik Informatika 16.2 (2022): 178-186., pp. Https://e-journal.upr.ac.id/index.php/JTI/article/download/3995/3679, 2022.

[10] A. W. Kuncoro, PENGUJIAN AUTENTIKASI DAN OTORISASI WEB MI-GATEWAY UII BERDASARKAN DOKUMEN owaspwstgv4.2, Yogyakarta: Univeristas Islam Indonesia, 2022.

[11] Taufiq, E. Hasmin, C. Susanto and K. Aryasa, "Expert System for Predicting Diabetes Using the Android-Based K-nnmethod," cogito Smart Journal, 8(2), 359–370, pp. Https://doi.org/10.31154/cogito.v8i2.406.359-370, 2022.

[12] Hidayatulloh and Saptadiaji, "Penetration Testing pada Website Universitas ARS Menggunakan Open Web Application Security Project (OWASP)," Jurnal Algoritma 18.1 (2021): 77-86, pp. Https://doi.org/10.33364/algoritma/v.18-1.827, 2021.

[13] M. Albahar, D. Alansari and A. Jurcut, "An empirical comparison of pen-testing tools for detecting web app vulnerabilities," Electronics 11.19 (2022): 2991., p. Https://doi.org/10.3390/electronics11192991, 2022.

[14] FIRST, "FIRST - Improving Security Together," 2023. [Online]. Available: https://www.first.org/cvss/.