Security Testing of XYZ Website Application Using ISSAF and OWASP WSTG v4.2 Methods

Muhammad Firdaus Yusuf[1*], Ira Rosianal Hikmah[2], Amiruddin[3], Septia Ulfa Sunaringtyas[4]

[1,2,3,4]Cybersecurity Engineering, Politeknik Siber dan Sandi Negara, Bogor, West Java, Indonesia
E-mail: [1*]m.firdaus@student.poltekssn.ac.id, [2]ira.rosianal@poltekssn.ac.id, [3]amir@poltekssn.ac.id, [4]septia.ulfa@poltekssn.ac.id

(Received: 25 Nov 2024, revised: 10 Dec 2024, accepted: 11 Dec 2024)

Abstract

The research focuses on improving the security of information systems in ABC City, specifically on the XYZ website application developed by the Communication and Informatics Office ABC to assist in governmental administration and manage various critical data. This study is motivated by the high incidence of cybersecurity threats in the governmental administration sector, as reported by Badan Siber dan Sandi Negara in November 2023. The primary objective of this research is to identify security vulnerabilities within the XYZ website application. The research employs the Information Systems Security Assessment Framework (ISSAF) as the primary security testing framework and the OWASP Web Security Testing Guide (WSTG) version 4.2 as the guide for the penetration testing phase, one of the stages in ISSAF for validating vulnerabilities. Validated vulnerabilities are further assessed for severity using the OWASP Risk Rating guidelines to estimate the risk and impact of potential attacks on the Communication and Informatics Office ABC. The research methodology uses a black-box testing approach. To ensure a structured approach, it provides security recommendations using the SMAACT method. This research includes a report on the identified vulnerabilities and recommendations that the Communication and Informatics Office ABC can implement to address these vulnerabilities. The findings of this study are expected to provide insights into existing security vulnerabilities within the website application and practical recommendations for improvement, benefiting both the practical context of enhancing information security at the Communication and Informatics Office ABC and the theoretical context as a reference for similar future research.

Keywords: ISSAF, Penetration, Vulnerability, Website, WSTG V4.2.

I. INTRODUCTION

A smart city is defined as a city that effectively and efficiently manages all resource potential to address challenges and meet various needs through integrated and sustainable innovation management. The smart city concept is implemented in ABC City to maximize technology utilization and enhance public services. The city government has undertaken various innovations to improve public services in ABC City. The Department of Communication and Informatics (Diskominfo) of ABC City developed the XYZ website application as an innovation to increase the effectiveness and efficiency of public service delivery.

The XYZ website application is an integrated government administration system for civil servants (Aparatur Sipil Negara or ASN) to manage all personnel services within the government of ABC City in a single system. The integrated services of the XYZ application are accessible from civil servants at the city level to those at the village level, thereby facilitating the work of civil servants in performing their official duties to serve the citizens of ABC City. The XYZ website application also serves as a medium for document exchange, often containing information of a sensitive and confidential nature within an organization. Additionally, the development of the XYZ website application aligns with the implementation of Sistem Pemerintahan Berbasis Elektronik (SPBE) as stipulated in Presidential Regulation Number 95 of 2018.

SPBE is the implementation of governance leveraging information and communication technology to provide services to SPBE users [1]. SPBE shifts service media from physical interaction to digital platforms like websites [2]. Websites' utilization positively impacts the quality of public services [3]. However, despite these advantages, website usage poses certain risks, such as cyberattacks [4]. Cyberattacks have the potential to threaten government data security in three primary aspects: confidentiality, integrity, and availability [5]. Cyberattacks like SQL Injection impact confidentiality and integrity, leading to the theft of sensitive information that unauthorized parties can modify or misuse [6]. Furthermore, other cyberattacks, such as Distributed Denial of Service (DDoS), impact availability by rendering the administrative website inaccessible, disrupting governmental operations and public services [7].

This is evidenced by a report from the National Cyber and Crypto Agency (BSSN) in November 2023, stating that the government administration sector was the most affected by cybersecurity incidents [8]. Such incidents occur due to security gaps in websites exploited by attackers [9]. Therefore, ensuring the security of web-based applications, particularly in the government sector, is crucial to protecting confidentiality, integrity, and availability as an initial step in mitigating data breaches, unauthorized access, or actions by irresponsible individuals.

To mitigate the security risks of the XYZ website application, which is part of the government administration sector and contains sensitive data, a framework is required to analyze system vulnerabilities based on security standards [10]. One such framework for website application security testing is the Information Systems Security Assessment Framework (ISSAF) [11]. ISSAF is a Penetration Testing Framework (PTF) that provides comprehensive technical security testing guidance and categorizes the security testing process into specific categories [12]. Its purpose is to offer recommendations and feedback on security testing based on actual workflows and to serve as a reference for ensuring the security of information systems [12]. ISSAF offers several advantages compared to existing security controls against threats and vulnerabilities for testing the resilience of website applications [13]. This is demonstrated in a study conducted by Guntoro et al. (2020), which analyzed the security of the Open Journal System web server at a university using ISSAF and OWASP version 4 testing methods [14]. The study found vulnerabilities to DoS attacks on the web server using ISSAF, while the OWASP version 4 method indicated that the web server was secure [14].

ISSAF is also suitable for organizations requiring customized security testing tailored to their specific scope and risks [15]. Specific risks refer to threats relevant to an organization, which can vary based on technology [15]. ISSAF consists of three phases: Planning and Preparation, Assessment, and Reporting, Clean Up, and Destroy Artifacts. The core phase in ISSAF is the Assessment phase, which includes nine structured tests: Information Gathering, Network Mapping, Vulnerability Identification, Penetration, Gaining Access and Privilege Escalation, Enumerating Further, Compromising Remote Users/Sites, Maintaining Access, and Covering Tracks [11]. However, this study is limited to the Enumerating Further stage for several reasons. First, the Compromising Remote Users/Sites and Maintaining Access stages risk undermining the integrity of the tested system, mainly through the implantation of backdoors that unauthorized parties could exploit. Second, the Covering Tracks and Clean Up and Destroy Artifacts stages could hinder the identification and recovery process from unintended incidents by concealing system changes and deleting traces during the study.

In ISSAF, the Vulnerability Identification stage is used to identify and analyze potential vulnerabilities in the website application. Vulnerability Identification involves collecting data related to potential vulnerabilities in the website application. Tools for performing Vulnerability Identification include OWASP ZAP, Burp Suite Professional, Qualys WAS, Arachni, Wapiti3, and Fortify WebInspect [16]. According to a study by M. Albahar, D. Alansari, and A. Jurcut [16], Burp Suite Professional achieved the highest scores among paid tools, while OWASP ZAP ranked highest among free tools. Based on this finding, OWASP ZAP and Burp Suite Professional were utilized for vulnerability scanning during the Vulnerability Identification stage. The Penetration stage validates vulnerabilities identified in the Vulnerability Identification stage. However, the Penetration stage lacks detailed guidance for vulnerability validation, so the study employed guidance from the Open Web Application Security Project (OWASP) for vulnerability validation.

OWASP is a nonprofit organization dedicated to improving software security [17]. As a nonprofit, OWASP has produced freely accessible articles, methodologies, documentation, tools, and technologies [18]. The Web Security Testing Guide (WSTG) is an OWASP guide for security testing on website applications that involves active analysis of the application for any weaknesses, whether technical flaws or vulnerabilities [19]. The WSTG has various versions updated annually, with the latest being WSTG v4.2, released in December 2020 [20]. To achieve comprehensive results, this study conducted security testing on the XYZ website application using the ISSAF framework and the OWASP WSTG as a guideline for the Penetration stage in ISSAF.

The vulnerability level was then assessed using the OWASP Risk Assessment Calculator, guided by the OWASP Risk Rating methodology, to estimate the risks and impacts of attacks on the organization, facilitating organizational decision-making regarding the identified vulnerabilities [21]. Finally, the results of the security testing using these frameworks were used to provide recommendations for the ABC City Government to improve information system security and support the implementation of the Smart City concept.

The SMAACT (Specific, Measurable, Act small, Achievable, Controlling, and Time Bound) method was employed to structure these recommendations. The SMAACT method is the result of a comparative study of HARD (Heartfelt, Animated, Required, Difficult), OKR (Objectives and Key Results), BSQ (Think Big, Act Small, and Move Quick), and SMART (Specific, Measurable, Achievable, Relevant, Time Bound) methods by Hilario et al. [22]. Compared to the other four methods, SMAACT adds two additional criteria: Act small and Controlling. SMAACT enables organizations to approach goal achievement in manageable small steps (Act small) while ensuring continuous control over the process (Controlling) [22].

Based on this background, this study investigates "Security Testing of the XYZ Website Application Owned by the Department of Communication and Informatics, City of ABC, Using the Information Systems Security Assessment Framework (ISSAF) and OWASP Web Security Testing Guide (WSTG) v4.2."

II. LITERATURE REVIEW

A literature review is a section that provides a critical review and summary of research, theories, and findings relevant to the study's topic.

A. Sistem Pemerintah Berbasis Elektronik (SPBE)

Sistem Pemerintah Berbasis Elektronik (SPBE) refers to the administration of government that utilizes information and communication technology to deliver services to SPBE users. SPBE services are the outputs produced by one or more SPBE application functions and have tangible benefits. SPBE applications are divided into two categories: general applications and specific applications [1].

The XYZ website application is an example of implementing a specific application in SPBE. Specific applications are SPBE applications developed, used, and managed by specific central or regional government agencies to meet unique needs that are not shared with other central or regional government agencies. The development of specific applications must ensure the implementation of SPBE security principles. This includes the physical and logical security of information assets, the application of user access management, and regular security monitoring and evaluation. Moreover, the application owner must conduct information security risk assessments and implement appropriate mitigation measures to protect the data and information managed by the website application owner [1].

B. Website Application Security Testing Techniques

Website application security testing techniques are categorized into three approaches: black box, grey box, and white box. Black-box testing is a software testing technique conducted without knowledge of the software's internal structure, design, or code. Grey-box testing combines black-box and white-box techniques, where testing is performed with partial knowledge of the software's internal structure, design, or code. White-box testing, on the other hand, involves testing with full knowledge of the website application's internal structure, design, or code [23].

C. Information Systems Security Assessment Framework (ISSAF)

The Information Systems Security Assessment Framework (ISSAF) is a Penetration Testing Framework (PTF) developed by the Open Information Systems Security Group [11]. ISSAF is a structured framework that categorizes information system security assessments into various domains with specific evaluation criteria for each domain. The purpose of categorizing security assessments is to provide input for security evaluations based on real-world scenarios, enabling the development of effective mitigation strategies. Additionally, ISSAF ensures that the testing process is comprehensive and directed [24]. This is essential for organizations to safeguard against potential financial losses, maintain compliance with industry standards, and enhance their overall information security posture [25].

ISSAF is particularly suitable for testing the resilience of website applications as it offers a systematic approach to identifying and exploiting vulnerabilities in such applications [26]. By following the guidelines and phases outlined in ISSAF, organizations can effectively evaluate the resilience of their web applications and systems, identify vulnerabilities, and implement necessary controls to enhance their overall security posture. ISSAF consists of three phases and nine assessment steps [11], which are:

1. Phase I: Planning and Preparation

The first phase involves planning and preparation before conducting penetration testing. This phase includes creating a formal agreement signed by both parties that specifies the testing details, such as testing boundaries, timelines, and other arrangements. This agreement serves as a basis for security testing and provides mutual legal protection. This phase includes identifying contact individuals from both sides, holding an initial meeting to confirm the scope, methodology, and test cases, and finalizing agreements. The outputs of this phase include a Non-Disclosure Agreement (NDA) accompanied by a mutually agreed-upon testing methodology, including dates, objects, and testing boundaries.

2. Phase II: Assessment

The second phase, Assessment, is the core stage of ISSAF. This phase involves conducting security tests on the information system agreed upon during the Planning and Preparation Phase [27]. The Assessment phase consists of nine steps [11]:

a. Information Gathering
   Information Gathering is the initial step in security testing. It aims to collect as much information as possible about the target (company and/or individual) using technical methods (DNS/Whois) and non-technical methods (search engines, news groups, etc.), either through the Internet or non-internet sources. The objective is to explore all potential attack vectors, making this a critical step for subsequent phases. Tools such as Whois, IP lookup scanner, and SSL Labs can be used [14][13].
b. Network Mapping
   After gathering information about the target, a more technical approach is taken to map the network and associated resources. Specific network information gathered in the previous step is expanded upon. Various tools and applications can assist in discovering technical details about the hosts and networks involved in the testing. Activities include identifying active hosts, port and service scanning, OS fingerprinting, route identification using the Management Information Base (MIB), and service

fingerprinting. Tools such as Firewalk, Nmap, and Netcat are used.

c. Vulnerability Identification
   In this step, the website application is scanned to determine its vulnerabilities. Tools like Burp Suite [28], Acunetix [14], Vega [12], and OWASP ZAP [13] can be utilized for vulnerability scanning. The vulnerability scanning results inform the next steps in the testing process, such as Penetration.
d. Penetration
   The Penetration step demonstrates attacks based on vulnerabilities identified in the previous step. This involves the following:
   1) Finding Code/Proof of Concept (PoC) Tools: Searching for existing PoC code from trusted repositories or public sources.
   2) Testing Code/PoC Tools: Adjusting and testing the PoC in an isolated environment.
   3) Writing Custom PoC Tools: Creating PoC tools if none exist for the identified vulnerabilities.
   4) Using PoC Tools on the Target: Deploying the PoC tools against the target to exploit vulnerabilities.

   This step generates a validation report on the vulnerabilities identified in the Vulnerability Identification phase.
e. Gaining Access & Privilege Escalation
   This step attempts to gain access to the system and escalate privileges to obtain broader control. Privilege escalation involves exploiting misconfigurations or vulnerabilities to gain elevated access. Tools like Hydra [29] and Metasploit [30] can be used.
f. Enumerating Further
   This step involves deeper probing into the network protocols or systems of the website application. Activities include sniffing network traffic for man-in-the-middle attacks, exploiting session cookies, and attempting password attacks. Tools like Burp Suite and Wireshark are used.
g. Compromise Remote Users/Sites
   Remote access to the target website application is attempted. Methods include uploading shell files and exploiting Remote File Inclusion (RFI) vulnerabilities.
h. Maintaining Access
   This step ensures that testers maintain control over the system. Persistent access can demonstrate the extent of system exposure if compromised.
i. Covering Tracks
   In this step, security testers attempt to remain undetected unless instructed otherwise.

3. Phase III: Reporting, Clean-Up, and Destroying Artifacts

a. Reporting
   Critical findings during testing must be communicated verbally as soon as they are discovered. A final comprehensive report detailing the findings, vulnerabilities, and recommended mitigation actions is prepared.
b. Clean-up and Destroying Artifacts
   All files created or stored on the tested system must be deleted. If deletion is not possible remotely, the details of these files are included in the report for client staff to remove.

D. OWASP WSTG v4.2

The OWASP Web Security Testing Guide (WSTG) is a website application security testing methodology developed by the Open Web Application Security Project (OWASP). It provides a structured framework for evaluating web application security, focusing on validating and verifying the effectiveness of security controls [31]. The latest version, WSTG v4.2, was released on December 3, 2020, as an enhancement to the previous version, WSTG v4.1, released on April 22, 2020 [31].

E. OWASP Risk Rating

The OWASP Risk Rating methodology assesses security risks in website applications. It evaluates risks based on four factors: Threat Agent, Vulnerability Factor, Technical Impact, and Business Impact [21]. Guided by this methodology, the OWASP Risk Assessment Calculator estimates the likelihood and impact of security threats, aiding organizations in proactive risk management.

F. SMAACT Recommendation Model

The SMAACT (Specific, Measurable, Act small, Achievable, Controlling, Time Bound) model supports structured and measurable recommendations for problem-solving. It emphasizes specific goal-setting, measurable progress, incremental steps, feasibility, constant monitoring, and time-bound planning, ensuring actionable and practical solutions [22].

III. RESEARCH METHOD

The research design carried out in this study can be seen in Figure 1.

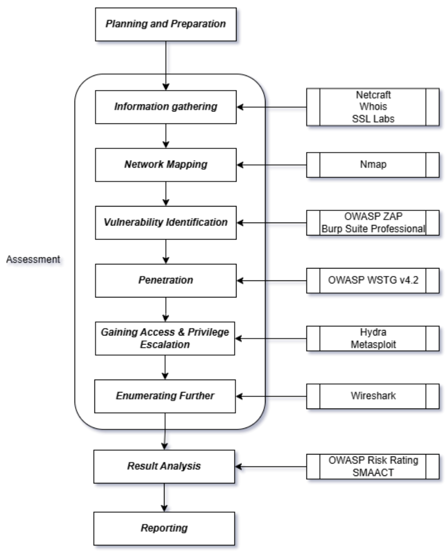


Figure 1. Research Method

Figure 1 illustrates that this research begins with the Planning and Preparation phase. In the Planning and Preparation phase, security testing preparations are carried out by drafting a Non-Disclosure Agreement (NDA) between the researcher and the Diskominfo ABC, followed by preparing tools for security testing. The next phase, Information Gathering, involves the comprehensive collection of information related to the XYZ website application, limited by the agreed-upon availability and ethical conduct between the security testers and Diskominfo ABC. The Information Gathering phase uses tools such as Whois, SSL Labs, and Netcraft.

The subsequent phase is Network Mapping, which involves collecting specific information about the network of the XYZ website application by performing port scanning using the decoy scan method on the Nmap tool, resulting in open ports and services on the XYZ website application. After completing the information collection, the process continues to the Vulnerability Identification phase. In this phase, a vulnerability scan of the XYZ website application uses OWASP Zed Attack Proxy (ZAP) and Burp Suite Professional as Vulnerability Scanning tools. Vulnerabilities identified during the Vulnerability Identification phase using these tools are then validated using the OWASP WSTG v4.2 guidelines in the Penetration phase.

After identifying vulnerabilities with Vulnerability Scanning tools and validating them with OWASP WSTG v4.2, the next step involves Gaining Access & Privilege Escalation using Hydra and Metasploit tools to test the website application's security against attacks that could enable unauthorized users to access the XYZ website application system. This is followed by the Enumerating Further phase, where Wireshark tests the website application's security, specifically concerning the transmission of data packets within the network.

After completing a series of security testing processes on the XYZ website application, the process continues to the Result Analysis phase. In the Result Analysis phase, the vulnerability levels are assessed using the OWASP Risk Assessment Calculator tool based on the OWASP Risk Rating methodology, and improvement recommendations are prepared using the SMAACT model. Vulnerability assessments are conducted on validated vulnerabilities. This is followed by drafting recommendations using the SMAACT model (Specific, Measurable, Achievable, Relevant, Time-Bound). The final phase is Reporting, which contains a summary of the testing results from the Information Gathering phase to the Result Analysis phase. This is compiled into a security testing report document submitted to Diskominfo ABC.

IV. RESULT AND DISCUSSION

A. Planning and Preparation

This phase begins with planning the testing process through interviews with Diskominfo ABC, resulting in a Non-Disclosure Agreement (NDA). The agreement covers the testing timeline, the testing type, which employs Black Box Testing in this study, and the testing methodology using the ISSAF framework and OWASP WSTG v4.2 guidelines. After that, the tools used to test the security of the XYZ website application are prepared. A summary of the tools used can be seen in Table 1.

Table 1. Summary of the Tools Used in Security Testing

No	Tool Name	Version	Function
1.	Kali Linux	2023	Operating System for the Virtual Machine being run.
2.	Virtual Box	7.0.14	Creator of an isolated Virtual Machine from the host system.
3.	Netcraft	-	Information Gathering Tool.
4.	Whois	-	Information Gathering Tool.
5.	Nmap	7.94SVN	Network Mapping Tool.
6.	SSL Labs	-	Information Gathering Tool.
7.	OWASP ZAP	2.14.0	Vulnerability Identification Tool.
8.	Burp Suite Professional	2024.1.1.5	Vulnerability Identification and Penetration Tool.
9.	Visual Studio Code	1.85.2	Penetration Tool.
10.	Mozilla Observatory	-	Penetration Tool.
11.	Clickjacker.io	-	Penetration Tool.
12.	Cookie Editor	-	Penetration Tool.
13.	Hydra	V9.5	Gaining Access & Privilege Escalation Tool.
14.	Metasploit	-	Gaining Access & Privilege Escalation Tool.
15.	Wireshark	4.2.3	Enumerating Further Tool.
16.	OWASP Risk Rating Calculator	-	Result Analysis Tool.

The results of this planning and preparation phase ensure that the testing is conducted using appropriate tools and methodologies, setting up a systematic testing framework.

B. Information Gathering

This phase involves collecting basic information about the XYZ website application using tools such as Netcraft, Whois, and SSL Labs to support the information gathering process. The testing results are as follows in Table 2.

Table 2. Results of Information Gathering

Tool Name	Results Obtained
Netcraft	Provides critical information such as IP addresses, nameservers, and application frameworks.
Whois	Provides administrative details, including technical contacts and the physical location of the application administrator, in this case, Diskominfo ABC.
SSL Labs	Security rating of B, indicating the need to improve SSL/TLS configurations, as outdated protocols like TLS 1.0 and 1.1 are still supported.

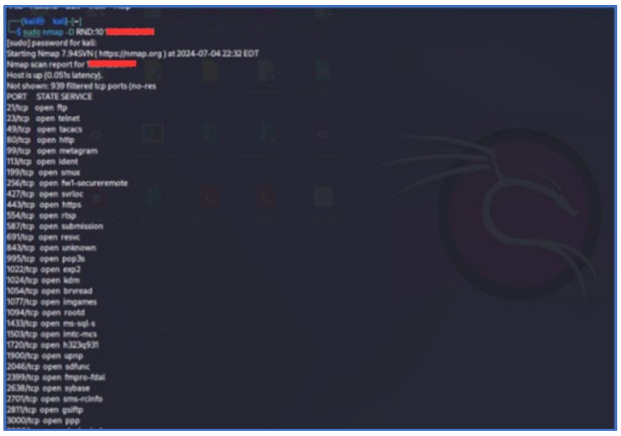
The results of this information-gathering phase provide a comprehensive overview of the XYZ application's structure, administration, and security, forming the basis for subsequent testing phases.

C. Network Mapping

The Network Mapping phase continues with the Information Gathering phase. Its objective is to expand upon the network-related information obtained during the Information Gathering phase. In this phase, port scanning is performed using the Nmap tool to identify open ports and running services on the XYZ website application. Port scanning is conducted using the decoy scan technique available in the Nmap tool.

The decoy scan technique obfuscates the user's identity when performing port scanning. This technique allows the user to make the remote host believe that other specified hosts, acting as decoys, are conducting the scanning of the XYZ website application network. Consequently, the website application's Intrusion Detection System (IDS) reports 5–10 port scanning attempts from unique IP addresses but cannot discern which IP is performing the scan and which are merely decoys.

The command to perform a decoy scan with the Nmap tool is entered as Nmap -D RND:10 <IP of the XYZ application> into the Nmap tool terminal. The results of the port scanning conducted using the Nmap tool are available in Figure 2.



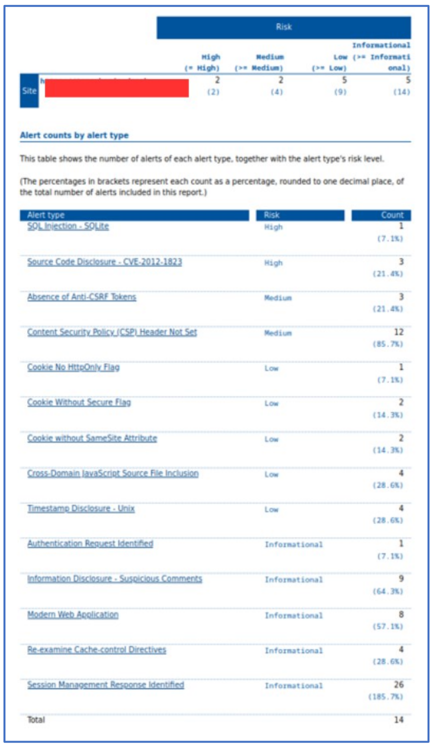
```
┌──(kali㉿ kali)-[~]
└─$ sudo nmap -D RND:10 [redacted]
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-04 22:32 EDT
Nmap scan report for [redacted]
Host is up (0.051s latency).
Not shown: 939 filtered tcp ports (no-res
PORT     STATE SERVICE
21/tcp   open  ftp
23/tcp   open  telnet
49/tcp   open  tacacs
80/tcp   open  http
99/tcp   open  metagram
113/tcp  open  ident
199/tcp  open  smux
256/tcp  open  fw1-secureremote
427/tcp  open  svrloc
443/tcp  open  https
554/tcp  open  rtsp
587/tcp  open  submission
691/tcp  open  resvc
843/tcp  open  unknown
995/tcp  open  pop3s
1022/tcp open  exp2
1024/tcp open  kdm
1054/tcp open  brvread
1077/tcp open  imgames
1094/tcp open  rootd
1433/tcp open  ms-sql-s
1503/tcp open  imtc-mcs
1720/tcp open  h323q931
1900/tcp open  upnp
2046/tcp open  sdfunc
2399/tcp open  fmpro-fdal
2638/tcp open  sybase
2701/tcp open  sms-rcinfo
2811/tcp open  gsiftp
3000/tcp open  ppp
```

Figure 2. Nmap Tool Result

Based on the testing conducted, the results show that numerous ports are open. Many open ports may indicate that the system or network has more potential entry points for exploitation. Ports commonly known to be vulnerable to attacks include port 21 (FTP), port 23 (Telnet), port 1443 (MS-SQL-S), and port 3389 (RDP). These ports may serve as entry points for attackers to attempt various attacks. Thus, the more open ports there are, the greater the likelihood of vulnerabilities existing, necessitating the subsequent Vulnerability Identification phase.

D. Vulnerability Identification

The next phase is vulnerability identification, conducted using vulnerability scanning tools, namely OWASP ZAP and Burp Suite Professional. Based on the Vulnerability Identification, nine vulnerabilities were identified using OWASP ZAP and three vulnerabilities were identified using Burp Suite Professional, as shown in Figure 3 and Figure 4.



	Risk			
	High (= High)	Medium (>= Medium)	Low (>= Low)	Informational (>= Informational)
Site [redacted]	2 (2)	2 (4)	5 (9)	5 (14)

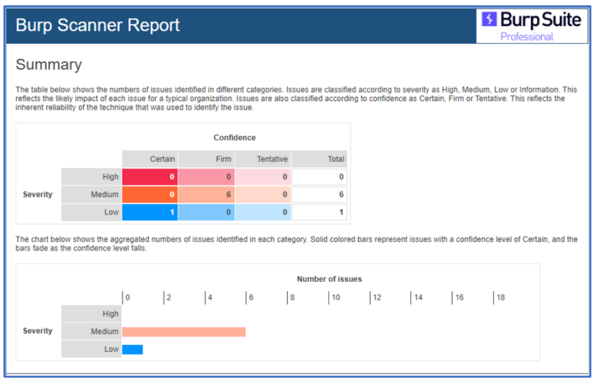
Alert counts by alert type

This table shows the number of alerts of each alert type, together with the alert type's risk level.

(The percentages in brackets represent each count as a percentage, rounded to one decimal place, of the total number of alerts included in this report.)

Alert type	Risk	Count
SQL Injection - SQLite	High	1 (7.1%)
Source Code Disclosure - CVE-2012-1823	High	3 (21.4%)
Absence of Anti-CSRF Tokens	Medium	3 (21.4%)
Content Security Policy (CSP) Header Not Set	Medium	12 (85.7%)
Cookie No HttpOnly Flag	Low	1 (7.1%)
Cookie Without Secure Flag	Low	2 (14.3%)
Cookie without SameSite Attribute	Low	2 (14.3%)
Cross-Domain JavaScript Source File Inclusion	Low	4 (28.6%)
Timestamp Disclosure - Unix	Low	4 (28.6%)
Authentication Request Identified	Informational	1 (7.1%)
Information Disclosure - Suspicious Comments	Informational	9 (64.3%)
Modern Web Application	Informational	8 (57.1%)
Re-examine Cache-control Directives	Informational	4 (28.6%)
Session Management Response Identified	Informational	26 (185.7%)
Total		14

Figure 3. OWASP ZAP Result



Burp Scanner Report

Summary

The table below shows the numbers of issues identified in different categories. Issues are classified according to severity as High, Medium, Low or Information. This reflects the likely impact of each issue for a typical organization. Issues are also classified according to confidence as Certain, Firm or Tentative. This reflects the inherent reliability of the technique that was used to identify the issue.

		Confidence			
		Certain	Firm	Tentative	Total
Severity	High	0	0	0	0
	Medium	0	6	0	6
	Low	1	0	0	1

The chart below shows the aggregated numbers of issues identified in each category. Solid colored bars represent issues with a confidence level of Certain, and the bars fade as the confidence level falls.

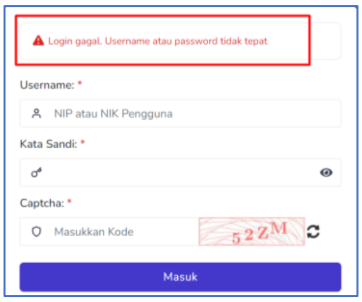
Figure 4. Burp Suite Professional Report

Based on Figure 3 and Figure 4, it can be concluded that a total of eleven vulnerability alerts were identified using OWASP ZAP and Burp Suite Professional. Of these eleven vulnerability alerts, nine were identified using the OWASP ZAP tool, and three were identified using the Burp Suite Professional tool, with one overlapping vulnerability found in both tools, namely Cookie Without Secure Flag. However, Table 3 also indicates that OWASP ZAP identified some vulnerabilities but not by Burp Suite Professional, and vice versa. Therefore, validation of the identified vulnerabilities is required in the Penetration phase.

E. Penetration

This phase validates vulnerabilities using OWASP WSTG v4.2 guidelines. The vulnerabilities found are explained in the validation process as follows:

1. Sql Injection
   Validation was performed based on WSTG-INPV-05 and evidence from OWASP ZAP. Validation was carried out in several ways, as follows:
   a.) First, validation was performed by conducting a classic SQL Injection test on the login page of the website application. The result showed that the classic SQL Injection attempt was unsuccessful, as shown in Figure 5 below.



Login gagal. Username atau password tidak tepat

Username: *

NIP atau NIK Pengguna

Kata Sandi: *

Captcha: *

Masukkan Kode

Figure 5. Results of the Classic SQL Injection Test

   b.) Second, validation was conducted using the Burp Suite tool to intercept requests to the XYZ application server. However, the XYZ website application did not display any information that could be exploited by an attacker, as shown in Figure 6 below.

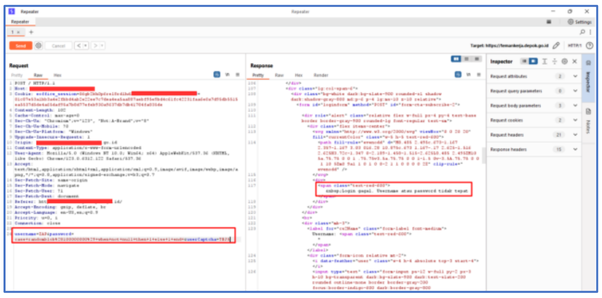
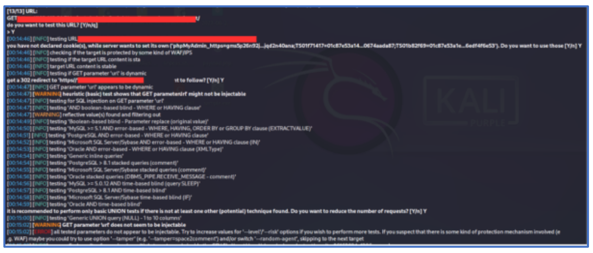


Figure 6. Result Test with Burp Suite

   c.) Third, validation was performed using the Sqlmap tool to search for parameters that could be used for SQL Injection testing through brute force attacks. However, it did not reveal any information that could be exploited by an attacker, as shown in Figure 7 below.

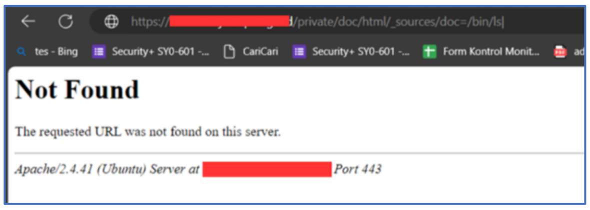


```
[13/13] URL:
GET [redacted]/
do you want to test this URL? [Y/n/q]
> Y
[00:14:46] [INFO] testing URL [redacted]
you have not declared cookie(s), while server wants to set its own ('phpMyAdmin_https=gms5p26n92j...jqd2n40ana;TS01f71417=01c87e53a14...0674aada87;TS01b82f69=01c87e53a1e...6edf4f6e53'). Do you want to use those [Y/n] Y
[00:14:46] [INFO] checking if the target is protected by some kind of WAF/IPS
[00:14:46] [INFO] testing if the target URL content is sta
[00:14:46] [INFO] target URL content is stable
[00:14:46] [INFO] testing if GET parameter 'url' is dynamic
got a 302 redirect to 'https://[redacted]                    it to follow? [Y/n] Y
[00:14:47] [INFO] GET parameter 'url' appears to be dynamic
[00:14:47] [WARNING] heuristic (basic) test shows that GET parameter'url' might not be injectable
[00:14:47] [INFO] testing for SQL injection on GET parameter 'url'
[00:14:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[00:14:47] [WARNING] reflective value(s) found and filtering out
[00:14:49] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[00:14:50] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[00:14:51] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[00:14:52] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[00:14:53] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[00:14:54] [INFO] testing 'Generic inline queries'
[00:14:54] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[00:14:55] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[00:14:56] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[00:14:56] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[00:14:57] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[00:14:58] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[00:14:59] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[00:15:00] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[00:15:02] [WARNING] GET parameter 'url' does not seem to be injectable
[00:15:02] [ERROR] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e
.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent', skipping to the next target
```

Figure 7. Result Test with Sqlmap

Based on the results of all the SQL Injection vulnerability tests above, it can be concluded that there are no SQL Injection vulnerabilities that can be exploited by an attacker on the XYZ website application, thus validating that it is not vulnerable(Invalid).

2. Source Code Disclosure (CVE-2012-1823)
   Validation of the Source Code Disclosure vulnerability (CVE-2012-1823) was conducted using the OWASP WSTG v4.2 guidelines, specifically the Testing for Command Injection subcategory (WSTG-INPV-12). Validation was performed in two scenarios for a single URL of the XYZ website application, one containing a file and one using the PHP programming language, in accordance with WSTG-INPV-12.
   The first validation was carried out by modifying the original URL from
   http://XYZ.go.id/private/doc/html/_sources/index.rst.txt
   to
   http://XYZ.go.id/private/doc/html/_sources/doc=/bin/ls|,
   as shown in Figure 8.



https://[redacted]/private/doc/html/_sources/doc=/bin/ls|

Not Found

The requested URL was not found on this server.

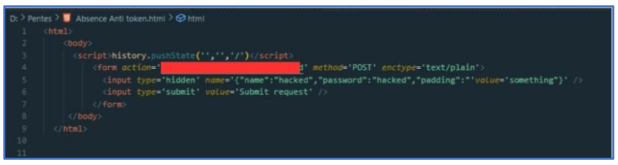
Apache/2.4.41 (Ubuntu) Server at [redacted] *Port 443*

Figure 8. Result Test of Modifying URL

   The second validation was performed by modifying the URL https://XYZ.go.id/index.php to https://XYZ.go.id/?dir=%3Bcat%20/etc/passwd. From the results of the vulnerability testing above, it can be concluded, thus validating that it is not vulnerable Invalid.

3. Absence of Anti-CSRF Tokens
   Absence of Anti-CSRF Tokens was tested by exploiting the vulnerability with validation referring to Testing for Cross-Site Request Forgery (WSTG-V42-SESS-05). Validation was carried out by creating an HTML page according to the script provided in OWASP WSTG-V42-SESS-05, as shown in Figure 9.

DOI: 10.34148/teknika.v14i1.1156

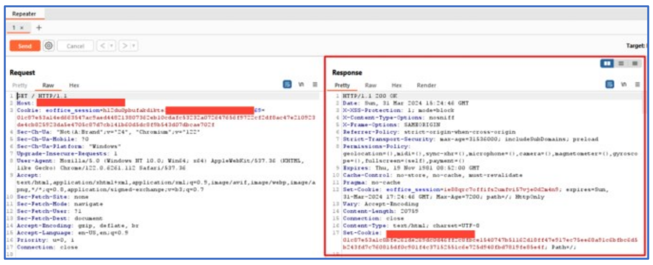


```
D: > Pentes > Absence Anti token.html > html
<html>
    <body>
      <script>history.pushState('','','/')</script>
        <form action='[redacted]d' method='POST' enctype='text/plain'>
          <input type='hidden' name='{"name":"hacked","password":"hacked","padding":"'value='something"}' />
          <input type='submit' value='Submit request' />
        </form>
    </body>
  </html>
```

Figure 9. Script CSRF Test

Subsequently, the request was successfully executed without an Anti-CSRF Token to prevent it. Based on the testing conducted above Cross-Site Request Forgery (CSRF) Token vulnerability is validated as valid.

4. Content Security Policy (CSP) Header Not Set
The validation was done using the Burp Suite Professional tool, following the Testing for Content Security Policy (WSTG-CONF-12) guideline, as shown in Figure 10.



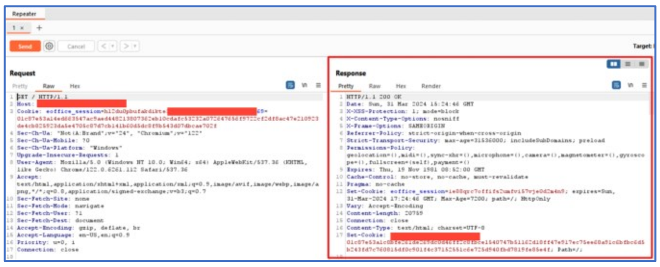
```
GET / HTTP/1.1
Host: [redacted]
Cookie: eoffice_session=hl2du0pbufakdikte[redacted]69=
01c07e53a14ed8d3547ac5aed448213807362eb10cdafc53232a072d4765df9722cf2df8ac47e210923
de4cb025923da5e4705c07d7cb141b60d5dc0f9b543d07dbcae702f
Sec-Ch-Ua: "Not(A:Brand";v="24", "Chromium";v="122"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/122.0.6261.112 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: close
```

```
HTTP/1.1 200 OK
Date: Sun, 31 Mar 2024 15:24:46 GMT
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Permissions-Policy:
geolocation=(),midi=(),sync-xhr=(),microphone=(),camera=(),magnetometer=(),gyrosco
pe=(),fullscreen=(self),payment=()
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: eoffice_session=ie08qrc7offifs2umfvi57vje0d2m4n9; expires=Sun,
31-Mar-2024 17:24:46 GMT; Max-Age=7200; path=/; HttpOnly
Vary: Accept-Encoding
Content-Length: 20759
Connection: close
Content-Type: text/html; charset=UTF-8
Set-Cookie: [redacted]
01c07e53a1c0bfe2c1de2c9dc0d4cff2c0fbce1540747b5116 2d10ff47e917ec75eed0a91c0bfbc6d5
b243fd7c760015df0c901f4c3715265 1c6e725d940fbd7819fe86e4f; Path=/;
```

Figure 10. CSP Validation

Based on Figure 10, the result indicates that the CSP header response does not have a Content Security Policy. It can be concluded that the vulnerability of the Content Security Policy (CSP) Header Not Set is validated as valid.

5. Cookie No HttpOnly Flag
The vulnerability of the Cookie No HttpOnly Flag was tested for exploitation based on validation referring to the testing guidelines for the subcategory Testing for Cookies Attributes (WSTG-SESS-02). Validation was done by inspecting the cookies sent by the browser used, as shown in Figure 11.



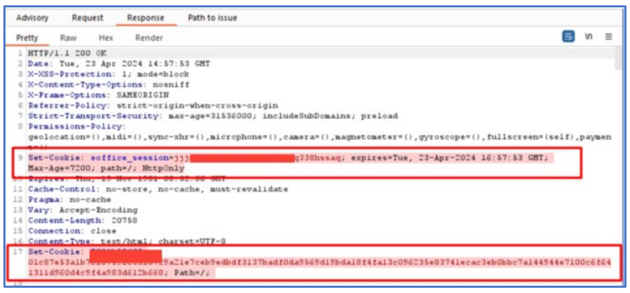
```
GET / HTTP/1.1
Host: [redacted]
Cookie: eoffice_session=hl2du0pbufakdikte[redacted]69=
01c07e53a14ed8d3547ac5aed448213807362eb10cdafc53232a072d4765df9722cf2df8ac47e210923
de4cb025923da5e4705c07d7cb141b60d5dc0f9b543d07dbcae702f
Sec-Ch-Ua: "Not(A:Brand";v="24", "Chromium";v="122"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/122.0.6261.112 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: close
```

```
HTTP/1.1 200 OK
Date: Sun, 31 Mar 2024 15:24:46 GMT
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Permissions-Policy:
geolocation=(),midi=(),sync-xhr=(),microphone=(),camera=(),magnetometer=(),gyrosco
pe=(),fullscreen=(self),payment=()
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: eoffice_session=ie08qrc7offifs2umfvi57vje0d2m4n9; expires=Sun,
31-Mar-2024 17:24:46 GMT; Max-Age=7200; path=/; HttpOnly
Vary: Accept-Encoding
Content-Length: 20759
Connection: close
Content-Type: text/html; charset=UTF-8
Set-Cookie: [redacted]
01c07e53a1c0bfe2c1de2c9dc0d4cff2c0fbce1540747b5116 2d10ff47e917ec75eed0a91c0bfbc6d5
b243fd7c760015df0c901f4c3715265 1c6e725d940fbd7819fe86e4f; Path=/;
```

Figure 11. Cookie No HttpOnly Flag Validation

Based on Figure 11, the HttpOnly flag was not found on the cookies of the XYZ website application. It can be concluded that the Cookie No HttpOnly Flag vulnerability has been validated as valid.

6. Cookie Without Secure Flag
Cookie Without Secure Flag was tested referring to the testing guidelines for the subcategory Testing for Cookies Attributes (WSTG-SESS-02). Validation was carried out by inspecting the cookies sent by the browser used, as shown in Figure 12.

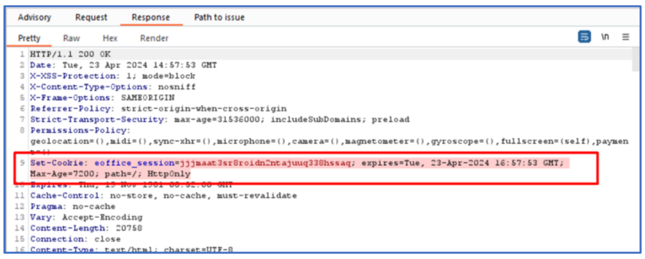


```
HTTP/1.1 200 OK
Date: Tue, 23 Apr 2024 14:57:53 GMT
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Permissions-Policy:
geolocation=(),midi=(),sync-xhr=(),microphone=(),camera=(),magnetometer=(),gyroscope=(),fullscreen=(self),paymen
t=()
Set-Cookie: eoffice_session=jjj[redacted]q338hssaq; expires=Tue, 23-Apr-2024 16:57:53 GMT;
Max-Age=7200; path=/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 20758
Connection: close
Content-Type: text/html; charset=UTF-8
Set-Cookie: [redacted]
01c87e53a1b7[redacted]9a21e7ceb9edbdf3137badf0da9569d19bda18f4fa13c096235e83741ecac3eb0bbc7a144944e7100c6f64
1311d960d4c9f4a983d612b668; Path=/;
```

Figure 12. Cookie Without Secure Flag Validation

Based on Figure 12, it is stated that the Secure flag was not found on the cookies of the XYZ website application. Therefore, it can be concluded that the Cookie Without Secure Flag vulnerability has been validated as valid.

7. Cookie without SameSite Attribute
Cookie Without SameSite Attribute was tested referring to the testing guidelines for the subcategory Testing for Cookies Attributes (WSTG-SESS-02). The validation was performed by intercepting with the Burp Suite tool and then moving the request to the repeater, thereby obtaining the response from the Burp Suite tool, as shown in Figure 13.

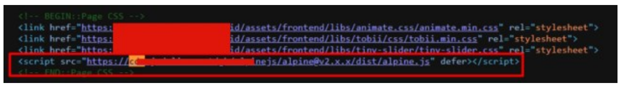


```
HTTP/1.1 200 OK
Date: Tue, 23 Apr 2024 14:57:53 GMT
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Permissions-Policy:
geolocation=(),midi=(),sync-xhr=(),microphone=(),camera=(),magnetometer=(),gyroscope=(),fullscreen=(self),paymen
t=()
Set-Cookie: eoffice_session=jjjmaat3sr8roidn2ntajuuq338hssaq; expires=Tue, 23-Apr-2024 16:57:53 GMT;
Max-Age=7200; path=/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 20758
Connection: close
Content-Type: text/html; charset=UTF-8
```

Figure 13. Cookie Without SameSite Attribute Validation

Based on Figure 13, the SameSite attribute has yet to be implemented on the XYZ website application. Therefore, it can be concluded that the Cookie Without SameSite Attribute vulnerability has been validated as valid.

8. Cross-Domain JavaScript Source File Inclusion
The Cross-Domain JavaScript Source File Inclusion vulnerability was validated based on the testing subcategory Testing for Cross-Site Script Inclusion (WSTG-CLNT-13). The vulnerability validation was performed using the inspect element feature on the Microsoft Edge browser, as shown in Figure 14.



```
<!-- BEGIN::Page CSS -->
<link href="https:[redacted]id/assets/frontend/libs/animate.css/animate.min.css" rel="stylesheet">
<link href="https:[redacted]id/assets/frontend/libs/tobii/css/tobii.min.css" rel="stylesheet">
<link href="https:[redacted]id/assets/frontend/libs/tiny-slider/tiny-slider.css" rel="stylesheet">
<script src="https://c[redacted]nejs/alpine@v2.x.x/dist/alpine.js" defer></script>
<!-- END::Page CSS -->
```

Figure 14. Cross-Domain JavaScript Source File Inclusion

Figure 14 illustrates how the XYZ website application sends a request URL from an external site that is not sufficiently trusted. Based on the exploitation test conducted, the Cross-Domain JavaScript Source File Inclusion alert has been validated as valid.

9. Timestamp Disclosure – Unix
   The Timestamp Disclosure – Unix vulnerability was validated by inspecting the source code of the XYZ website application based on the report from the OWASP ZAP tool. After inspecting the source code, it was found that the timestamp generated by OWASP ZAP corresponds to a CAPTCHA file and not sensitive information about the Unix or Linux server, as shown in Figure 15.

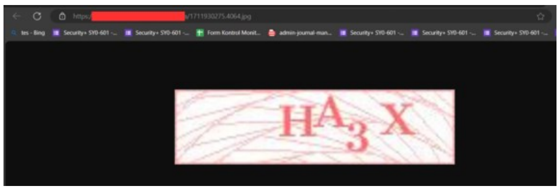

[redacted]

Figure 15. Timestamp Disclosure Validation

Based on this vulnerability validation, it can be concluded that there is no Timestamp Disclosure – Unix vulnerability in the XYZ website application, and therefore, it has been validated as not valid.

10. Session token in URL
    The Session Token in URL vulnerability was validated using the Burp Suite Professional tool. The validation was carried out by utilizing the evidence of the vulnerability found with the Burp Suite scanner tool, as shown in Figure 16.

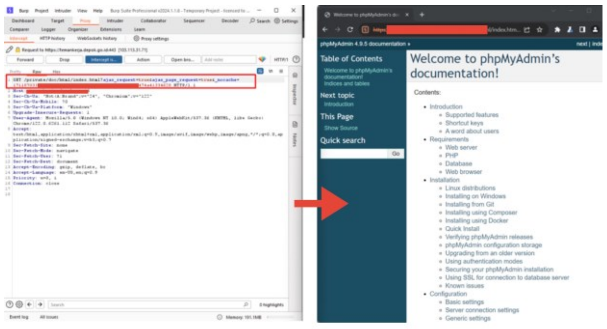
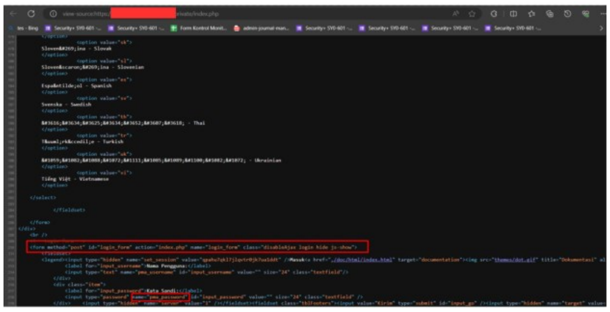

```
GET /private/doc/html/index.html?ajax_request=true&ajax_page_request=true&_nocache=
[redacted] HTTP/1.1
Host: [redacted]
Sec-Ch-Ua: "Not(A:Brand";v="24", "Chromium";v="122"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/122.0.6261.112 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
plication/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: close
```

Figure 16. Session Token in URL Test with Burp Suite

As seen in Figure 16, the XYZ website application server responds to the request generated by the Burp Suite tool and reveals the session token. This indicates that the vulnerability is indeed present. Based on the tests conducted, it can be concluded that the Session Token in URL vulnerability has been validated as valid.

11. Password field with autocomplete enabled
    The vulnerability of the Password Field with Autocomplete Enabled was validated using the inspect element feature on the Microsoft Edge browser, as shown in Figure 17. Figure 17 shows that the login form of the XYZ website application does not set autocomplete to disabled.


[redacted]
```
                <option value="sk">
        Sloven&#269;ina - Slovak
        </option>
                <option value="sl">
        Sloven&scaron;&#269;ina - Slovenian
        </option>
                <option value="es">
        Espa&ntilde;ol - Spanish
        </option>
                <option value="sv">
        Svenska - Swedish
        </option>
                <option value="th">
        &#3616;&#3634;&#3625;&#3634;&#3652;&#3607;&#3618; - Thai
        </option>
                <option value="tr">
        T&uuml;rk&ccedil;e - Turkish
        </option>
                <option value="uk">
        &#1059;&#1082;&#1088;&#1072;&#1111;&#1085;&#1089;&#1100;&#1082;&#1072; - Ukrainian
        </option>
                <option value="vi">
        Tiếng Việt - Vietnamese
        </option>

    </select>

        </fieldset>

    </form>
</div>
    <br />

    <form method="post" id="login_form" action="index.php" name="login_form" class="disableAjax login hide js-show">
        <fieldset>
        <legend><input type="hidden" name="set_session" value="[illegible]" />Masuk<a href="./doc/html/index.html" target="documentation"><img src="themes/dot.gif" title="Dokumentasi" al
            <label for="input_username">Nama Pengguna:</label>
            <input type="text" name="pma_username" id="input_username" value="" size="24" class="textfield"/>
        </div>
        <div class="item">
            <label for="input_password">Kata Sandi:</label>
            <input type="password" name="pma_password" id="input_password" value="" size="24" class="textfield" />
        </div>    <input type="hidden" name="server" value="1" /></fieldset><fieldset class="tblFooters"><input value="Kirim" type="submit" id="input_go" /><input type="hidden" name="target" value=
```

Figure 17. Inspection Source Code

The Password Field with Autocomplete Enabled alert has been validated based on the vulnerability validation as valid.

F. Gaining Access & Privilege Escalation

This phase tests unauthorized access and privilege escalation capabilities through vulnerable ports. The testing was conducted using Hydra and Metasploit on ports 21 (FTP), 23 (Telnet), and 1443 (MS-SQL). No credentials or exploitable vulnerabilities were found to gain unauthorized access. The system's security configuration proved effective.

The results of this gaining access phase demonstrate that the XYZ system is well-configured to prevent brute force attacks and exploitation of open ports.

G. Enumerating Further

The Enumerating Further phase involved conducting network sniffing tests on data packets transmitted during user login attempts to the XYZ website application. The testing in the Enumerating Further phase utilized the Wireshark tool. Wireshark was used by leveraging the Capturing Live Network Data feature available in the tool.

Enumerating Further testing using Wireshark analyzed login data packets on the XYZ application. The results stated that the XYZ Application has used the TLSv1.2 Protocol to encrypt data, which prevents network snooping. The analysis showed that the login data was safe from exploitation via network packets.

Based on the testing conducted, it is stated that the XYZ website application system has been well-configured against network sniffing attacks in this study. This conclusion is supported by the results of network sniffing tests using the Wireshark tool, which did not indicate any information that could be exploited to compromise the XYZ website application system.

H. Result Analysis

This phase evaluates the risk levels of identified vulnerabilities using OWASP Risk Rating. The assessment results show in Table 3.

Table 3. Summary of Vulnerability Level Assessment Results

No.	Validated Vulnerability	Likelihood	Impact	Result Level (Risk Rating)
1.	Absence of Anti-CSRF Tokens	High	Low	Medium
2.	Content Security Policy (CSP) Header Not Set	High	Low	Medium
3.	Cookie No HttpOnly Flag	Medium	Low	Low
4.	Cookie Without Secure Flag	High	Low	Medium
5.	Cookie Without SameSite Attribute	Medium	Low	Low
6.	Cross-Domain JavaScript Source File Inclusion	Medium	Low	Low
7.	Session Token in URL	Medium	Medium	Medium
8.	Password Field with Autocomplete Enabled	Medium	Low	Low

The results of this result analysis phase indicate that most validated vulnerabilities pose potential exploitation risks, requiring mitigation measures to reduce these risks.

Mitigation recommendations were prepared using the SMAACT model to ensure structured and measurable improvement steps. Recommendations for improvements were provided for the eight identified vulnerabilities in the XYZ website application. Table 4 presents the recommended improvements for each validated vulnerability.

Table 4. Improvement Recommendations for the XYZ Website Application Vulnerabilities

No.	Recommendation	Details
1	Anti-CSRF Tokens for Absence of Anti-CSRF Tokens	Implement anti-CSRF tokens for critical requests such as login or sensitive data submission using libraries like OWASP CSRFGuard.
2	CSP Header Addition for CSP Header Not Set	Add the Content-Security-Policy header to the server configuration to restrict script sources and frame embedding.
3	Secure Flag Configuration for Cookie Without Secure Flag	Add the Secure attribute to cookies to ensure they are only transmitted over HTTPS.
4	HttpOnly Flag for Cookie No HttpOnly Flag	Add the HttpOnly attribute to cookies to prevent access via JavaScript.
5	SameSite Attribute for Cookie Without SameSite Attribute	Add the SameSite attribute to cookies to restrict cross-site request inclusion.
6	Subresource Integrity (SRI) for Cross-Domain JavaScript Inclusion	Use the integrity attribute in <script> tags to ensure the integrity of external scripts.
7	POST Method for Session Token in URL	Replace all session token transmissions with the GET method and the POST method.
8	Autocomplete Off for Password Field with Autocomplete Enabled	Add the autocomplete="off" attribute to password fields in forms.

I. Reporting

This phase produces a report in the form of an executive summary and complete technical documentation. The report includes a list of vulnerabilities, validations, risk levels, and mitigation recommendations. The report is submitted to Diskominfo ABC as the basis for mitigation actions.

The results of this reporting phase provide strategic and technical guidance to Diskominfo ABC for addressing vulnerabilities and enhancing the security of the XYZ application.

V. CONCLUSION

Security testing of the XYZ website application was conducted using the ISSAF framework across nine phases. A total of 11 vulnerabilities were identified, with nine from OWASP ZAP and three from Burp Suite Professional validated using OWASP WSTG v4.2. The validation confirmed eight valid vulnerabilities, including four rated as medium risk (e.g., Absence of Anti-CSRF Tokens, CSP Header Not Set, Cookie Without Secure Flag, Session Token in URL) and four as low risk (e.g., Cookie No HttpOnly Flag, SameSite Attribute Missing).

Recommendations based on the SMAACT methodology included implementing Anti-CSRF Tokens, CSP headers, HttpOnly and Secure flags on cookies, SameSite attributes, Subresource Integrity (SRI), POST methods for sensitive data, and turning off autocomplete for login forms. Medium-risk vulnerabilities should be addressed within 1 month, while low-risk issues require resolution within 3 months. Periodic reviews using OWASP ZAP and Burp Suite Professional are advised to ensure vulnerabilities are resolved and do not reappear. Testing results and recommendations were documented and submitted to Diskominfo ABC.

REFERENCES

[1] Kementerian Pendayagunaan Aparatur Negara dan Reformasi Birokrasi (PANRB), "Peraturan Presiden Nomor 95 Tahun 2018 tentang Sistem Pemerintahan Berbasis Elektronik," *Menteri Huk. Dan Hak Asasi Mns. Republik Indones.*, p. 110, 2018.

[2] Karman, R. Deswanto, and S. A. Ningsih, "Implementasi E-Government Pada Pemerintah Daerah," *J. Stud. Ilmu Pemerintahan(JSIP)*, vol. 2, no. 2, pp. 43–50, Aug. 2021, doi: 10.35326/jsip.v2i2.1525.

[3] O. T. Hutajulu, G. Argenti, and M. F. Rizki, "Implementasi Konsep Kebijakan Smart City Terhadap Efektivitas Mall Pelayanan Publik DKI Jakarta," *J. Pendidik. dan Konseling*, vol. 5, no. 1, pp. 5869–5879, 2023.

[4] Setiawan, F. Samopa, I. A. Akbar, N. A. Sani, B. C. Hidayanto, and Y. S. Dharmawan, "Pendampingan Analisis Vulnerability dan Hardening pada Website Pemerintah Kota Surabaya," *Sewagati*, vol. 7, no. 6, pp. 897–906, 2023, doi: 10.12962/j26139960.v7i6.624.

[5] A. H. Harahap, D. C. Andani, A. Christie, D. Nurhaliza, and A. Fauzi, "Pentingnya Peranan CIA Triad Dalam Keamanan Informasi dan Data Untuk Pemangku Kepentingan atau Stakholder," *J. Manaj. dan Pemasar. Digit.*, vol. 1, no. 2, pp. 73–83, Apr. 2023.

[6] A. Mutedi and B. Tjahjono, "Systematic Literature Review: Preventing SQL Injection Attacks Using Tools OWASP CSR Web Application Firewall," *J. Inform. Univ. Pamulang*, vol. 7, no. 1, pp. 151–156, 2022, doi: 10.32493/informatika.v7i1.17590.

[7] P. Simarmata, N. F. Saragih, and I. K. Jaya, "Deteksi Serangan DDOS Pada VPS Menggunakan Metode Deep Neural Network," *Methotika J. Ilm. Tek. Inform.*, vol. 3, no. 1, pp. 1–12, Apr. 2023, [Online]. Available:https://ejurnal.methodist.ac.id/index.php/methotika/article/view/2107

[8] Badan Siber dan Sandi Negara, "Laporan Bulanan Publik Agustus 2023," no. November, 2023, [Online]. Available: https://www.bssn.go.id/wp-content/uploads/2024/03/Lanskap-Keamanan-Siber-Indonesia-2023.pdf

[9] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review," *J. Cybersecurity Priv.*, vol. 2, pp. 764–777, 2022, doi: 10.3390/jcp2040039.

[10] A. S. Syahab, "Analisis Audit Keamanan Informasi Website Dari Drown Attack Menggunakan Network Mapper Dan Qualys SSL," *J. Manaj. Inform. Sist. Inf.*, vol. 6, no. 1, pp. 39–47, Jan. 2023, doi: 10.36595/misi.v5i2.

[11] OISSG, "Information Systems Security Assessment Framework (ISSAF)," *Open Inf. Syst. Secur. Gr.*, pp. 1–845, 2006, [Online]. Available: https://kr-labs.com.ua/books/oissg-pentest.pdf

[12] R. Umar, I. Riadi, and M. I. A. Elfatiha, "Analisis Keamanan Sistem Informasi Akademik Berbasis Web Menggunakan Framework ISSAF," *Jutisi J. Ilm. Tek. Inform. dan Sist. Inf.*, vol. 12, no. 1, pp. 280–292, Apr. 2023.

[13] I. G. A. S. Sanjaya, G. M. A. Sasmita, and D. M. S. Arsa, "Evaluasi Keamanan Website Lembaga X Melalui Penetration Testing Menggunakan Framework ISSAF," *J. Ilm. Merpati* , vol. 8, no. 2, pp. 113–124, 2020, doi: 10.24843/jim.2020.v08.i02.p05.

[14] Guntoro, L. Costaner, and Musfawati, "Analisis Keamanan Web Server Open Journal System (OJS) Menggunakan Metode ISSAF Dan OWASP (Studi Kasus OJS Universitas Lancang Kuning)," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 5, no. 1, pp. 45–55, Jun. 2020, doi: 10.29100/jipi.v5i1.1565.

[15] F. M. H. Falcón, M. D. L. Arévalo, G. S. Atuncar, and I. C. Sanchez, "Comparative Study of Computer Security Methodologies for Countering Cyber Attacks," *Res Mil.*, vol. 13, no. 3, pp. 448–457, 2023.

[16] M. Albahar, D. Alansari, and A. Jurcut, "An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities," *Electron.*, vol. 11, no. 19, pp. 1–25, 2022, doi: 10.3390/electronics11192991.

[17] H. Poston, "Mapping The OWASP Top Ten To Blockchain," *Procedia Comput. Sci.*, vol. 177, pp. 613–617, 2020, doi: 10.1016/j.procs.2020.10.087.

[18] D. S. Irawan, "Pengujian Keamanan Sistem Informasi Berbasis Web Berdasarkan Dokumen Owasp Wstg v4. 2 (Studi Kasus: Sistem Informatics Expo Universitas Islam Indonesia)," *Univ. Islam Indones.*, 2022, [Online]. Available: https://dspace.uii.ac.id/handle/123456789/40200

[19] A. I. Rafeli, H. B. Seta, and I. W. Widi, "Pengujian Celah Keamanan Menggunakan Metode OWASP Web Security Testing Guide (WSTG) pada Website XYZ," *IFTKJurnal Inform.*, vol. 18, no. 2, pp. 97–103, 2022, doi: 10.52958/iftk.v18i2.4632.

[20] M. K. Abdan, "Pengujian Keamanan Sistem Informasi Berbasis Web Berdasarkan Framework OWASP WSTG v4.2 (Studi Kasus: Sistem Sekawan v1 Universitas Islam Indonesia)," *Univ. Islam Indones.*, pp. 1–95, 2022, [Online]. Available: https://dspace.uii.ac.id/handle/123456789/40200

[21] OWASP, "OWASP Risk Rating Methodology." https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

[22] A. Suvaryan and A. Karapetyan, "Developing Organizational Goals In View Of SMAACT Goals Model Criteria," *E3S Web Conf.*, vol. 403, pp. 1–7, 2023, doi: 10.1051/e3sconf/202340308019.

[23] M. Albarka Umar, "A Study of Software Testing: Categories, Levels, Techniques, and Types Comprehensive Study of Software Testing: Categories, Levels, Techniques, and Types," vol. 5, no. 6, pp. 32–40, 2023, [Online]. Available:

https://doi.org/10.36227/techrxiv.12578714.v2

[24] M. A. Nabila, P. E. Mas'udia, and R. Saptono, "Analysis And Implementation Of The ISSAF Framework On OSSTMM On Website Security Vulnerabilities Testing In Polinema," *J. Telecommun. Netw.* , vol. 13, no. 1, 2023, doi: 10.33795/jartel.v13i1.511.

[25] A. Almaarif and M. Lubis, "Vulnerability Assessment and Penetration Testing (VAPT) Framework: Case Study of Government's Website," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 10, no. 5, pp. 1874–1880, 2020, doi: 10.18517/ijaseit.10.5.8862.

[26] I. G. A. S. Sanjaya, G. M. A. Sasmita, and D. M. S. Arsa, "Information Technology Risk Management Using ISO 31000 Based On ISSAF Framework Penetration Testing (Case study: Election Commission Of X City)," *Int. J. Comput. Netw. Inf. Secur.*, vol. 12, no. 4, pp. 30–40, 2020, doi: 10.5815/ijcnis.2020.04.03.

[27] E. P. Silmina, A. Firdonsyah, and R. A. A. Amanda, "Analisis Keamanan Jaringan Sistem Informasi Sekolah Menggunakan Penetration Test Dan ISSAF," *Transmisi*, vol. 24, no. 3, pp. 83–91, 2022, doi: 10.14710/transmisi.24.3.83-91.

[28] Fathurrachman, "Pengujian Kerentanan Log4Shell Pada Website E-Commerce Menggunakan Metode Vulnerability Assessment and Penetration Testing (VAPT) Life Cycle," 2023. [Online]. Available: https://repository.uinjkt.ac.id/dspace/bitstream/123456789/71211/1/FATHURRACHMAN-FST.pdf

[29] A. W. Wardhana and H. B. Seta, "Analisis Keamanan Sistem Pembelajaran Online Menggunakan Metode ISSAF pada Website Universitas XYZ," *Inform. J. Ilmu Komput.*, vol. 17, no. 3, p. 226, 2021, doi: 10.52958/iftk.v17i3.3653.

[30] M. A. Rabby and M. Sultana, "An Overview Of Metasploit Framework Supervisor," 2015.

[31] E. Saad and R. Mitchell, "OWASP Web Security Testing Guide v4-2," *OWASP Found.*, p. 465, 2020.