JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 9 Number 2 | December 2024: 276-285
DOI: 10.15575/join.v9i2.1326

File Integrity Monitoring as a Method for Detecting and Preventing Web Defacement Attacks

Candra Kurniawan[1], Agung Triayudi[2]
[1,2]Department of Communication and Information Technology, Universitas Nasional, Indonesia

Article Info

Article history:

Received April 04, 2024
Revised August 27, 2024
Accepted October 25, 2024
Published December 31, 2024

Keywords:

Brute Force Attack
File Integrity Monitoring
Wazuh
Web Defacement

ABSTRACT

The cybersecurity landscape in Indonesia recorded an increase in cyberattacks in 2022. One of the types of attacks observed was web defacement attacks targeting government websites. In 2022, there were a total of 2,348 web defacement attacks in Indonesia, with the majority occurring in the governmental sector. In proactive efforts to monitor and prevent web defacement attacks, this study implemented the open-source tool Wazuh and activated the file integrity monitoring module to detect file changes in the system. Testing was conducted with two types of attacks: brute force attacks to gain system access and web defacement attacks involving script insertion to trigger alerts from the file integrity monitoring. The results of the testing show that the implementation of Wazuh and the file integrity monitoring module can real-time detect malicious activities and file additions, so that it can be used to mitigate cyberattacks.

Corresponding Author:

Agung Triayudi
Department of Communication and Information Technology
Jl. Sawo Manila No.61, Pejaten Barat, Pasar Minggu, Kota Jakarta Selatan, DKI Jakarta
Email: agungtriayudi@civitas.unas.ac.id

1. INTRODUCTION

The rapid advancement of technology brings various impacts, including negative consequences. Additionality this technological advancement can be exploited by irresponsible parties to launch cyberattacks. Cyberattacks have been continuously increasing from year to year. Cybersecurity researchers from SonicWall reported in their cyber threat report that malware attacks, crypto jacking incidents, and IoT attacks increased by 2%, 43%, and 87% per year, respectively. Additionally, attempted attacks have increased by 19%, with the targets based on continents, namely Asia (129%) and Europe (548%) [1].

Researchers from Trend Micro released a report titled "Calibrating Expansion: 2023 Annual Cybersecurity Report," which discusses the cybersecurity landscape in 2023. The report highlights that the highest attacks occurred in cloud environments, specifically brute-force login attacks reaching 1,234,502,439 activities. These brute-force attacks targeted ports 3389 (RDP), 21 (FTP), and 22 (SSH) and were conducted using common usernames and passwords. Additionally, the report discusses malware campaign attacks during 2023, with the governmental sector being the most affected, totaling 302,600 malware attacks. The highest malware attack in 2022 was web shell activities, totaling 303,399, whereas in 2023, coin miner malware took the first position with 74,933 attacks, followed by web shell with 59,788 attacks. Successful infiltration of malware attacks into systems poses a challenge for early detection efforts to prevent the expansion of attack impacts [2].

Cyberattacks nationally in Indonesia have also experienced an increase from year to year. According to the cybersecurity landscape report of Indonesia in 2022 issued by the National Cyber and Crypto Agency in 2023, it was stated that cyberattacks targeting Indonesia had increased. In 2022, there were a total of 976,429,996 anomalous traffic attacks targeting Indonesia. Most detected attacks were malware attacks, including Mining Pool, Generic Trojan RAT, and Phishing. Apart from malware attacks, the report also elaborated on defacement attack data in Indonesia. In 2022, a total of 2,348 web defacement attacks were detected across several sectors such as Government Administration, Defense, Health, ICT, Food, Finance, Energy and Mineral Resources, Transportation, and other sectors beyond the scope of Presidential Regulation No. 82 of 2022 concerning the Protection of Vital Information Infrastructure. Among these, the highest number of cases occurred in the Government Administration sector, covering both Central and Regional Governments, totaling 885 cases [3].

Web defacement attacks have become a serious threat to the government sector at all levels, encompassing both central and regional governments. This phenomenon is often attributed to suboptimal management by IT teams, as it closely relates to procurement and financial matters. The lack of attention from administrators towards web defacement attacks can have adverse effects on the reputation of government agencies and erode public trust in the services provided by these institutions.

Web defacement attacks can be mitigated through the implementation of file integrity monitoring (FIM) on servers. Several FIM solutions are available for free or opensource, thereby avoiding budgetary concerns for individual agencies. FIM represents one of the approaches that can be utilized to detect activities such as file modifications, deletions, or additions within a folder [4]. The implementation of FIM on Industrial Control Systems (ICS) has proven to be effective in detecting suspicious activities, file violations, including data tampering, and data destruction [5]. FIM can serve as a proactive monitoring method for detecting activities based on incoming logs on the FIM dashboard, triggering alerts accordingly [6]. As an effort towards prevention, this research implements FIM to detect and prevent web defacement attacks.

2. METHOD

A Security Operations Center (SOC) is a cybersecurity operations center with the goal of detecting and responding to security incidents occurring within an organization's network infrastructure. SOC implements a system for monitoring the organization's systems and networks to early threat detection. In the process of detection and response, SOC employs various combinations of people, processes, and technologies to provide swift responses to security incidents. [7].

File Integrity Monitoring (FIM) is a system task that monitors activities occurring within a system. This monitoring is conducted to identify and detect potential changes or modifications that occur in files within the system [8]. FIM operates by periodically checking files based on their file checksums (hash values of files [9], [10].

FIM has several fundamental criteria for examining files within the system. Below are some of the basic criteria of FIM [11]:

a. Checking metadata information
b. Performing processes automatically
c. Self-protected to secure FIM configurations
d. Conducting periodic checks
e. Easily updated to protect against new vulnerabilities.

FIM becomes a compliance requirement in several industry standards, one of which is the compliance standard in the industrial and financial sectors known as the Payment Card Industry Digital Security Standards (PCI DSS). PCI DSS is a standard that regulates security related to companies dealing with payment cards. By implementing FIM in PCI DSS, it is expected to detect unwanted change activities. [12].

Wazuh is one of the open-source applications or platforms available for free. Wazuh is a Host-based Intrusion Detection System (HIDS) with the capability to perform detection and monitoring based on a set of configured rules [13]. Wazuh operates in a client-server manner where the client sends detection data to the server. Some capabilities of Wazuh include File Integrity Monitoring, Malware Detection, and Active Response [14].

Web defacement attacks target vulnerabilities found on websites or web servers to deploy malicious code that alters website pages. These attacks typically occur due to poor system management and failure to implement necessary technology updates, resulting in outdated and exploitable technology. They can significantly impact an organization's reputation, making them a serious concern [10]. Based on research conducted by Romagna and Hout, several types of attacks used to exploit website vulnerabilities include web application bugs, SQL Injection, SSH and FTP server brute force, and File inclusion [15].

The research was conducted using a quantitative testing method consisting of preparation, implementation, testing, and analysis stages for drawing conclusions. The study employed a verificative element to verify and prove the correctness of the targeted object, which is the implementation of Wazuh FIM to detect and prevent web defacement attacks [16]. The research process was conducted in the Figure 1.

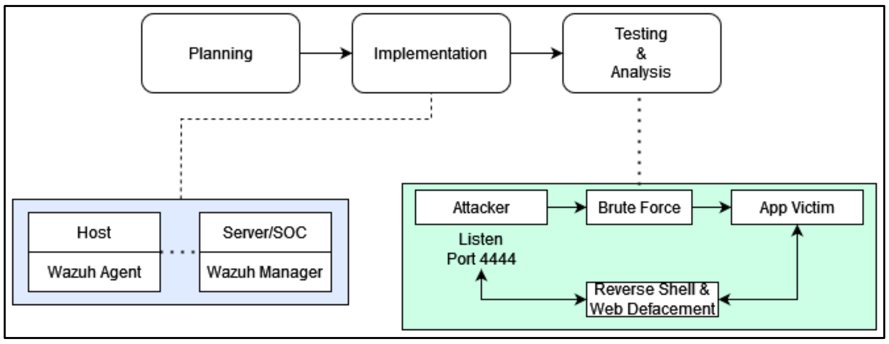


Figure 1. Research Methods

3.1. Planning

In the planning phase, a system development strategy will be developed, consisting of the target system, monitoring system, and attacker system. All systems will be developed in an isolated virtualization environment to prevent scope expansion.

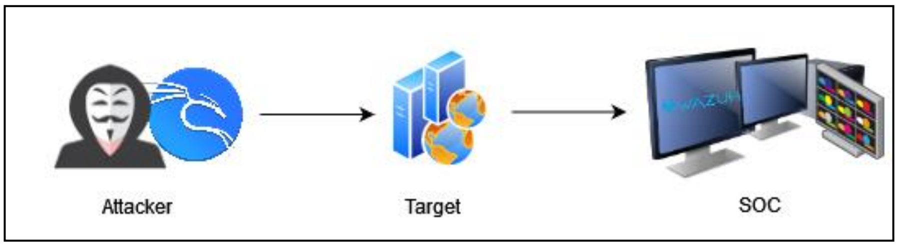


Figure 2. Planning Step

3.2. Implementation

The implementation phase is conducted in a virtual environment, where the attacker is developed using the Kali Linux 2023 operating system with the IP address 192.168.143.133, the target system is developed using Ubuntu Server 16.04 LTS operating system with WordPress 4.1.31 CMS, having the IP address 192.168.143.132, and the Security Operation Center (SOC) system is developed using Wazuh 7.3 with the IP address 192.168.143.130.

On the target device or WordPress, the Wazuh agent is installed to send logs to the Wazuh server. Configuration on the Wazuh agent is performed to detect file or folder changes on the server using FIM. Wazuh FIM works by periodically scanning specific folders or directories that experience real-time changes. FIM stores the hash value of a file or checksum in the local FIM database. Subsequently, the Wazuh agent will send the detected change results to the Wazuh server, and the Wazuh server through the Wazuh manager will compare the checksum with the checksum stored in the database. When there is a checksum change, it will trigger an alert on the dashboard [17].

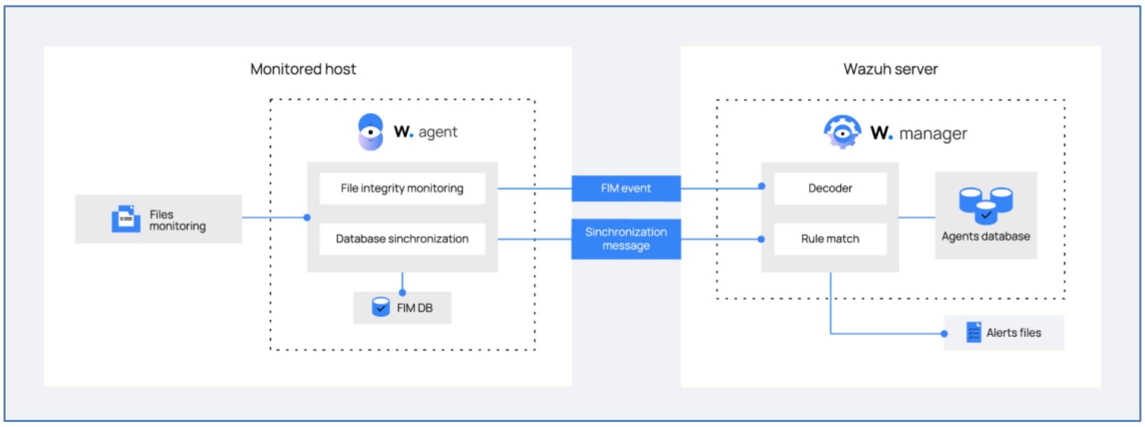


Figure 3. FIM Works

For FIM to function, configuration is performed in the ossec.conf file located at /var/ossec/etc/ossec.conf. The configuration is done by adding the following commands:

```
<syscheck>
   <directories realtime="yes" check_all="yes"
report_changes="yes">/var/www/html</directories>
</syscheck>

<localfile>
   <log_format>apache</log_format>
<location>/var/log/apache2/access.log</location>
</localfile>
```

3.3. Testing & Analysis

Testing and analysis constitute the final stage of the conducted research. In this phase, tests for brute force attacks and web defacement are carried out. Brute force attacks serve as the entry point for attackers to modify the appearance or perform web defacement and insert backdoors into the target system. Analysis is necessary to demonstrate the ability of Wazuh FIM in detecting and preventing web defacement attacks early. Wazuh is also expected to detect attack activities before attackers successfully infiltrate the target system.

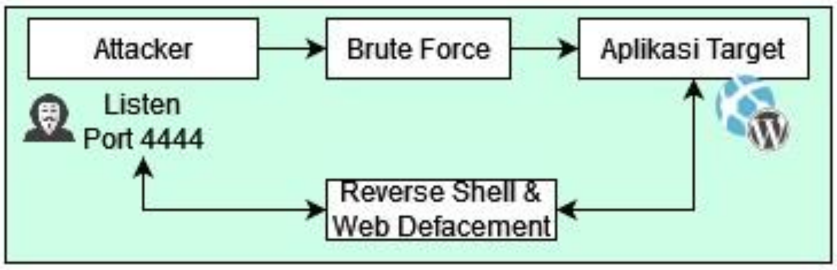


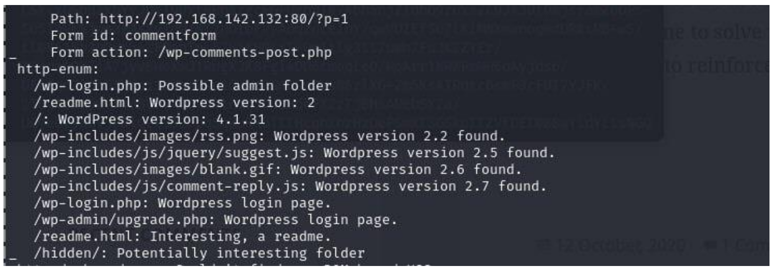
Figure 4. Attack Flow

3. RESULT AND DISCUSSION

During the testing phase, several attack activities are conducted to assess the detection capabilities of attacks and file changes based on the implementation of FIM using Wazuh. In conducting these tests, several attacks are carried out to trigger activities on the target system, thereby sending alerts to the SOC system. The attack activities are based on the initial access, which serves as the entry point for web defacement attacks, one of which is brute force attack [15][18]. The following are the two attacks conducted during testing:

3.1. Brute Force Attack

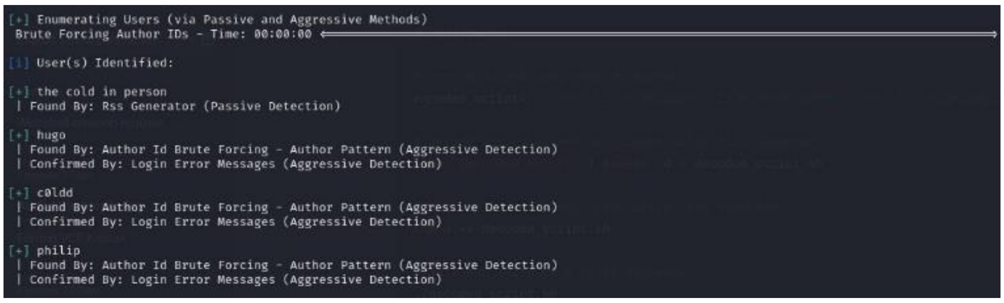
Brute force attack is an illegal activity that attempts to obtain username and password pairs by repeatedly attempting logins using a wordlist via the network [19]. Brute force activity is conducted to guess weak credentials and is carried out using automated tools. In Australia, brute force attacks increased by 160 percent by the end of 2021 [20]. Before conducting the brute force login attack, information gathering about the target system is performed. Based on the information collected, it is known that the system is using WordPress CMS.



```
    Path: http://192.168.142.132:80/?p=1
    Form id: commentform
    Form action: /wp-comments-post.php
http-enum:
  /wp-login.php: Possible admin folder
  /readme.html: Wordpress version: 2
  /: WordPress version: 4.1.31
  /wp-includes/images/rss.png: Wordpress version 2.2 found.
  /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
  /wp-includes/images/blank.gif: Wordpress version 2.6 found.
  /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
  /wp-login.php: Wordpress login page.
  /wp-admin/upgrade.php: Wordpress login page.
  /readme.html: Interesting, a readme.
  /hidden/: Potentially interesting folder
```

Figure 5. Information Gathering (1)

Based on this information, scanning is then conducted to obtain information about the list of usernames present in the system.

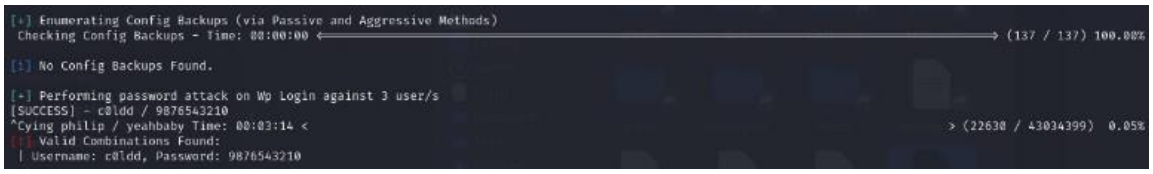


```
[+] Enumerating Users (via Passive and Aggressive Methods)
 Brute Forcing Author IDs - Time: 00:00:00 <==========================>

[i] User(s) Identified:

[+] the cold in person
 | Found By: Rss Generator (Passive Detection)

[+] hugo
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] c0ldd
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] philip
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)
```

Figure 6. Information Gathering (2)

The scanning results obtained 3 usernames: hugo, c0ldd, and Philip. These three usernames are then used as the username wordlist (user.txt), and for the password wordlist, the rockyou.txt file is used, which contains 14 million common passwords [20]. The brute force process is conducted using the scanning application Wpscan. Wpscan is a vulnerability scanning application for WordPress. The scanning process is carried out with the following command:

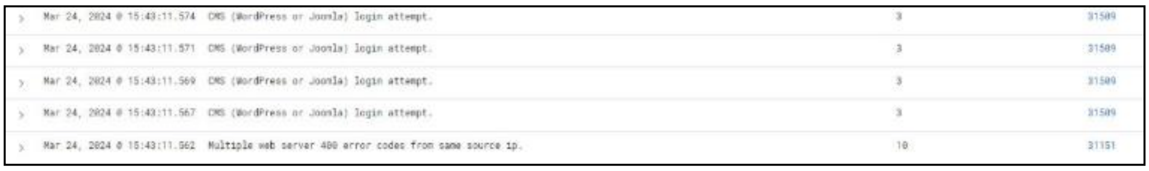
```
wpscan   -url   https://192.168.143.132   -usernames
user.txt -passwords rockyou.txt
```



```
[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:00 <==========================> (137 / 137) 100.00%

[i] No Config Backups Found.

[+] Performing password attack on Wp Login against 3 user/s
[SUCCESS] - c0ldd / 9876543210
^Cying philip / yeahbaby Time: 00:03:14 <                           > (22630 / 43034399)  0.05%
[!] Valid Combinations Found:
 | Username: c0ldd, Password: 9876543210
```

Figure 7. Valid Credential

The brute force result indicates that a valid username and password were successfully used to log in to the application, specifically the user "c0ldd" with the password "9876543210".

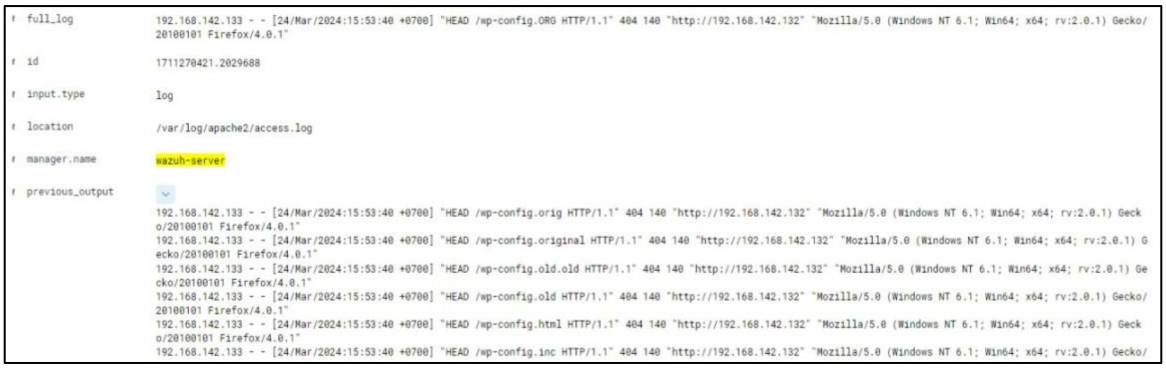


```
> Mar 24, 2024 @ 15:43:11.574  CMS (WordPress or Joomla) login attempt.                  3    31509
> Mar 24, 2024 @ 15:43:11.571  CMS (WordPress or Joomla) login attempt.                  3    31509
> Mar 24, 2024 @ 15:43:11.569  CMS (WordPress or Joomla) login attempt.                  3    31509
> Mar 24, 2024 @ 15:43:11.567  CMS (WordPress or Joomla) login attempt.                  3    31509
> Mar 24, 2024 @ 15:43:11.562  Multiple web server 400 error codes from same source ip.  10   31151
```

Figure 8. Multiple Web Server Attempts

The scanning activity conducted was detected by the SOC, where the SOC dashboard provided an alert with the description "Multiple web server 400 error codes from some source IP."

Before detecting the brute-force activity, Wazuh had already detected activity indicated as scanning activity, providing a rule description of "Multiple web server 400 error codes from the same source IP." This detection indicates that there has been activity from the same IP address in large quantities within a short period. The 400 code in the server indicates that the request sent by the client to the server cannot be understood by the server or is invalid. Upon checking the full_log field, it can be determined that attempts made by the IP address 192.168.142.133 were trying to access the directory "/wp-config.ORG" using the HEAD method, resulting in a response code of 404 or "Not Found." This indicates that the directory or file being targeted is not present on the server. Additionally, the previous_output field shows several similar activities, indicating that these activities occurred in less than 1 second to send multiple requests to the server. Therefore, this can trigger the alert "Multiple web server 400 error codes from the same source IP" or what is commonly known as vulnerability scanning activity based on the attack technique by Mittre Attack.



```
t  full_log          192.168.142.133 - - [24/Mar/2024:15:53:40 +0700] "HEAD /wp-config.ORG HTTP/1.1" 404 140 "http://192.168.142.132" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0.1) Gecko/
                     20100101 Firefox/4.0.1"

t  id                1711270421.2029688

t  input.type        log

t  location          /var/log/apache2/access.log

t  manager.name      wazuh-server

t  previous_output
                     192.168.142.133 - - [24/Mar/2024:15:53:40 +0700] "HEAD /wp-config.orig HTTP/1.1" 404 140 "http://192.168.142.132" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0.1) Geck
                     o/20100101 Firefox/4.0.1"
                     192.168.142.133 - - [24/Mar/2024:15:53:40 +0700] "HEAD /wp-config.original HTTP/1.1" 404 140 "http://192.168.142.132" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0.1) G
                     ecko/20100101 Firefox/4.0.1"
                     192.168.142.133 - - [24/Mar/2024:15:53:40 +0700] "HEAD /wp-config.old.old HTTP/1.1" 404 140 "http://192.168.142.132" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0.1) Ge
                     cko/20100101 Firefox/4.0.1"
                     192.168.142.133 - - [24/Mar/2024:15:53:40 +0700] "HEAD /wp-config.old HTTP/1.1" 404 140 "http://192.168.142.132" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0.1) Gecko/
                     20100101 Firefox/4.0.1"
                     192.168.142.133 - - [24/Mar/2024:15:53:40 +0700] "HEAD /wp-config.html HTTP/1.1" 404 140 "http://192.168.142.132" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0.1) Geck
                     o/20100101 Firefox/4.0.1"
                     192.168.142.133 - - [24/Mar/2024:15:53:40 +0700] "HEAD /wp-config.inc HTTP/1.1" 404 140 "http://192.168.142.132" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0.1) Gecko/
```

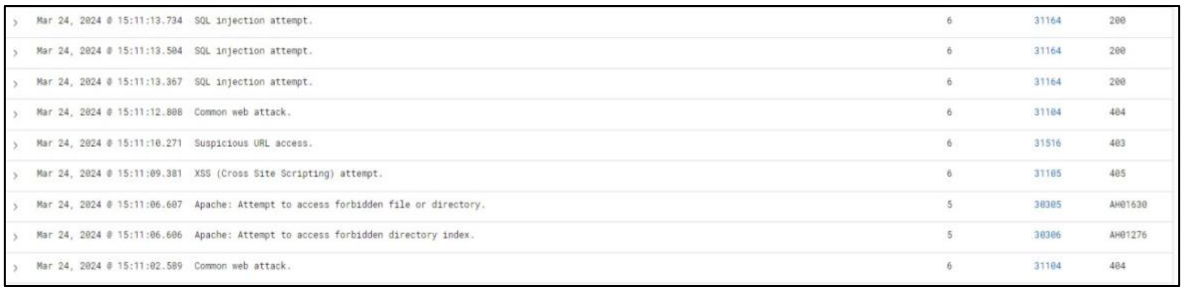
Figure 9. Multiple web server 400 error codes from same source ip

The activity was detected by rule_id 31151, which has the indicators of compromise (IoC) as follows:

– Activity from the same IP address results in response codes 400 or 404.
– Activity occurs at least 14 times within 90 seconds.

The vulnerability scanning activity is grouped by Wazuh into rule_level 10. Rule_level indicates the severity level, serving as a reference to determine the priority level for follow-up actions on detection results. The higher the rule_level, the more important it is to pay attention to and take follow-up actions. Wazuh divides rule_level from 0 to 14. Rule_level can be downgraded based on the validation results of detected alerts. For instance, if an alert with rule_level 14 is found to be a false positive after validation, its level can be downgraded and ignored.

Other vulnerability scanning activities are also detected, such as alerts for SQL Injection attempts, XSS (Cross-Site Scripting) attempts, Apache: attempts to access forbidden files or directories, and common web attacks such as directory traversal. The detected activities also provide information about the scanning application or tools used by the attacker, such as the Nmap Scripting Engine.

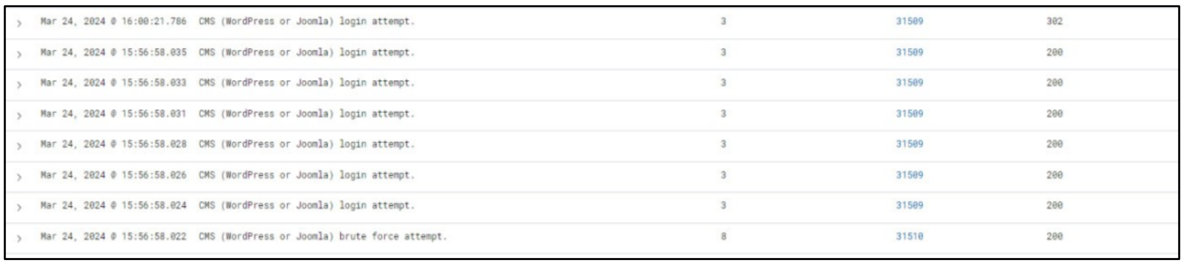


›	Mar 24, 2024 @ 15:11:13.734	SQL injection attempt.	6	31164	200
›	Mar 24, 2024 @ 15:11:13.504	SQL injection attempt.	6	31164	200
›	Mar 24, 2024 @ 15:11:13.367	SQL injection attempt.	6	31164	200
›	Mar 24, 2024 @ 15:11:12.808	Common web attack.	6	31104	404
›	Mar 24, 2024 @ 15:11:10.271	Suspicious URL access.	6	31516	403
›	Mar 24, 2024 @ 15:11:09.381	XSS (Cross Site Scripting) attempt.	6	31105	405
›	Mar 24, 2024 @ 15:11:06.607	Apache: Attempt to access forbidden file or directory.	5	30305	AH01630
›	Mar 24, 2024 @ 15:11:06.606	Apache: Attempt to access forbidden directory index.	5	30306	AH01276
›	Mar 24, 2024 @ 15:11:02.589	Common web attack.	6	31104	404

Figure 10. Common Attack

Meanwhile, the brute-force activity is detected by Wazuh as the alert "CMS (WordPress or Joomla) Brute Force attempt." This alert indicates that there have been repeated login attempts within a short period with many activities. In this activity, it is observed that multiple attempts on /wp-login.php occurred in less than 1 second. Brute-force activity, according to the Mitre Attack framework, falls under the Credential Access tactic with the Brute Force technique (Mitre ID T1110).

Wazuh detects this activity with rule ID 31510, indicating that login attempts have been detected in applications using CMS WordPress or Joomla within a 30-second timeframe with more than 8 attempts. This is considered an anomalous activity as it is improbable for a human to input usernames and passwords more than 8 times within 30 seconds. Based on this, the activity is suspected to be carried out by a machine or application aiming to brute-force usernames and passwords for application login. This activity has a rule_level of 8, indicating the need for immediate follow-up.



>	Mar 24, 2024 @ 16:00:21.786	CMS (WordPress or Joomla) login attempt.	3	31509	302
>	Mar 24, 2024 @ 15:56:58.035	CMS (WordPress or Joomla) login attempt.	3	31509	200
>	Mar 24, 2024 @ 15:56:58.033	CMS (WordPress or Joomla) login attempt.	3	31509	200
>	Mar 24, 2024 @ 15:56:58.031	CMS (WordPress or Joomla) login attempt.	3	31509	200
>	Mar 24, 2024 @ 15:56:58.028	CMS (WordPress or Joomla) login attempt.	3	31509	200
>	Mar 24, 2024 @ 15:56:58.026	CMS (WordPress or Joomla) login attempt.	3	31509	200
>	Mar 24, 2024 @ 15:56:58.024	CMS (WordPress or Joomla) login attempt.	3	31509	200
>	Mar 24, 2024 @ 15:56:58.022	CMS (WordPress or Joomla) brute force attempt.	8	31510	200

Figure 11. Rule_Level 8

Successful brute-force activities are typically followed by login attempt activities, which Wazuh categorizes under rule_level 3 with rule_id 31509, labeled as "CMS (WordPress or Joomla) login attempt." When attackers successfully obtain valid username and password pairs, they proceed to manually log in to the application. Therefore, in such situations, login attempt activities are detected. In WordPress, an activity is considered successful if it returns a status code of 302 or redirects. This occurs because during the login process, the application redirects to the admin page.

4.1. Web Defacement Attack

After successfully obtaining credentials to log in to the application, the next step is to inject a defacement script to test whether FIM can detect changes to files. Below is a display of the FIM Wazuh module dashboard based on the web defacement attack activity conducted.

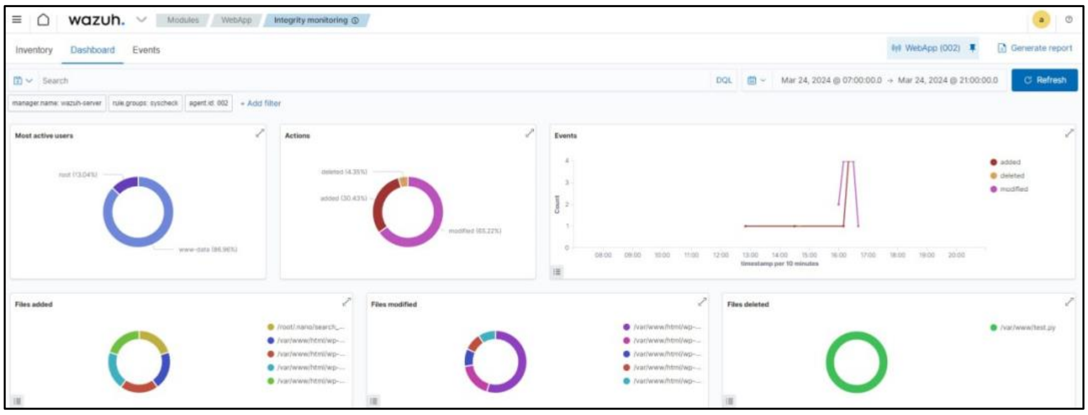


Figure 12. FIM Dashboard

In detecting file changes, Wazuh utilizes Syscheck, which is a module designed to check the integrity or authenticity of a file on the system, referred to as file integrity monitoring. This facilitates analysts in detecting suspicious or unauthorized changes. In this web defacement attack scenario, it is assumed that the attacker successfully logged into the WordPress application based on the brute-force activity. At this stage, the attacker made changes to the PHP script in the 404.php file, injecting defacement and web shell scripts.

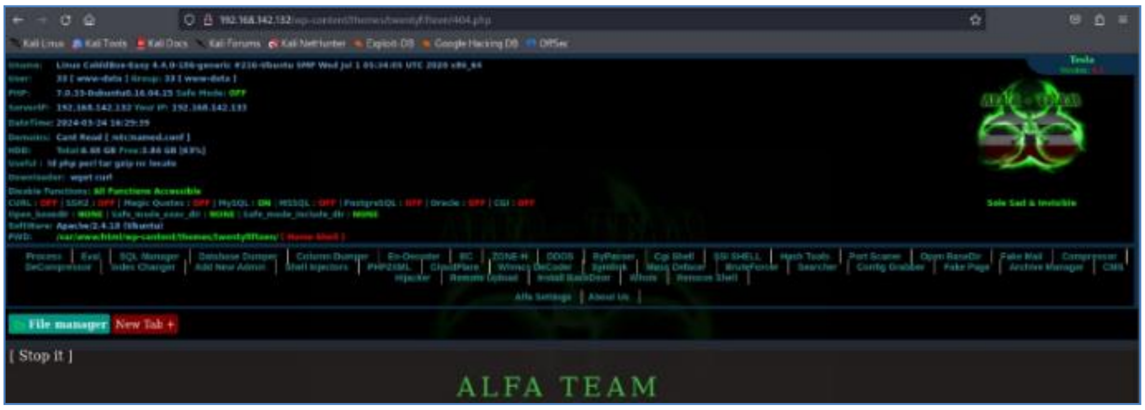
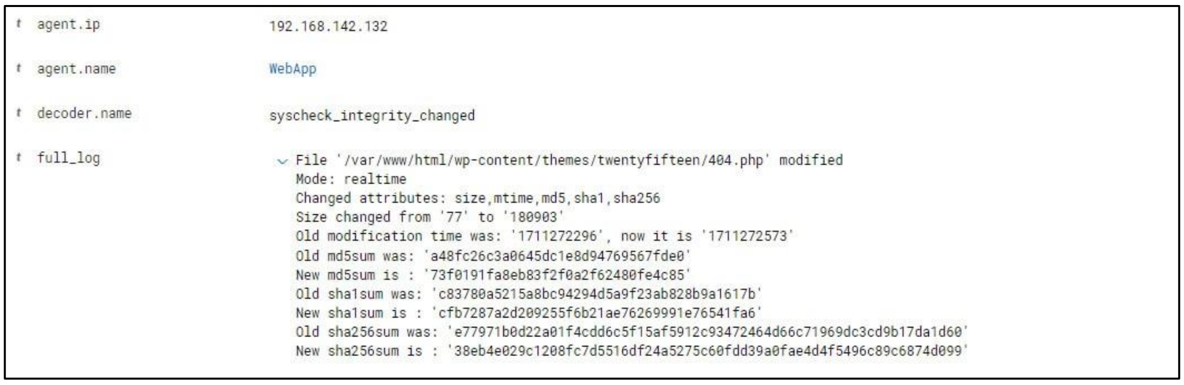


Figure 13. Injection script deface and shell on 404.php and Script Display

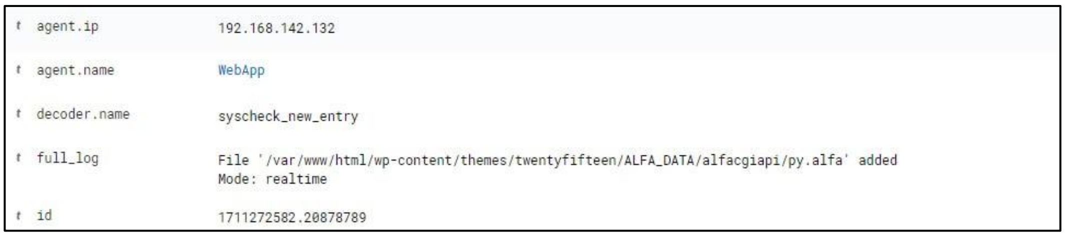
The web defacement activity conducted was successfully detected by FIM Wazuh. In real-time, the dashboard displayed an alert with the description "integrity checksum change," indicating that a change in the hash value of the file 404.php had occurred. A change in the hash value of the file indicates that there has been a modification to the file's content, resulting in a different hash value. The alert can be utilized by security analysts to conduct in-depth analysis and verification to determine whether the alert represents legitimate activity or not. Additionally, FIM Wazuh can also detect file addition activities.



```
t  agent.ip        192.168.142.132

t  agent.name      WebApp

t  decoder.name    syscheck_integrity_changed

t  full_log        ⌄ File '/var/www/html/wp-content/themes/twentyfifteen/404.php' modified
                     Mode: realtime
                     Changed attributes: size,mtime,md5,sha1,sha256
                     Size changed from '77' to '180903'
                     Old modification time was: '1711272296', now it is '1711272573'
                     Old md5sum was: 'a48fc26c3a0645dc1e8d94769567fde0'
                     New md5sum is : '73f0191fa8eb83f2f0a2f62480fe4c85'
                     Old sha1sum was: 'c83780a5215a8bc94294d5a9f23ab828b9a1617b'
                     New sha1sum is : 'cfb7287a2d209255f6b21ae76269991e76541fa6'
                     Old sha256sum was: 'e77971b0d22a01f4cdd6c5f15af5912c93472464d66c71969dc3cd9b17da1d60'
                     New sha256sum is : '38eb4e029c1208fc7d5516df24a5275c60fdd39a0fae4d4f5496c89c6874d099'
```

Figure 14. Detection of File Modification

The hash values detected by Wazuh can be further traced for information regarding the files that underwent changes. Several platforms support file tracing based on hash values, such as Virus Total. Regarding file addition, modification, and deletion activities, there are three important rule_ids in Wazuh: rule_id 550, 553, and 554. Rule_id 550 detects file modification activities, applying rules to periodically and in real-time check the hash values of every file in the defined folder. The detection of changes in the 404.php file is the result of rule_id 550, which provides data on the initial hash value and the new hash value, indicating that a change has occurred in the intended file. Meanwhile, rule_id 553 detects file or directory deletion activities. In contrast, rule_id 554 is designed to detect the addition of new files on the server. In the image below, you can see the information "added" with mode: "real-time," indicating the addition of the py.alfa file.

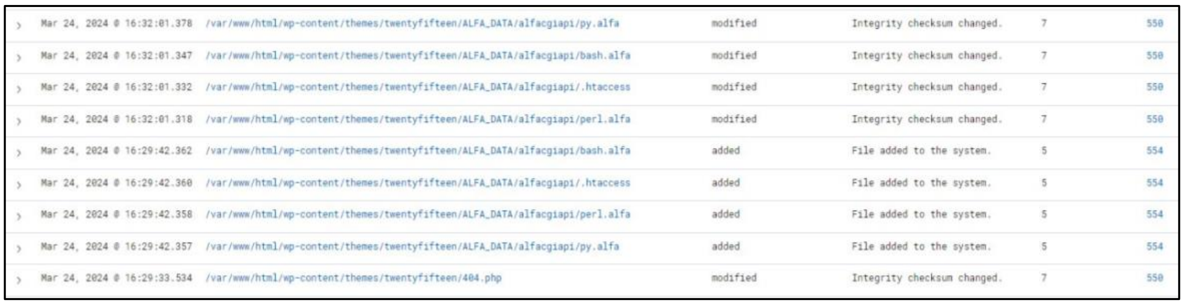


```
t  agent.ip        192.168.142.132

t  agent.name      WebApp

t  decoder.name    syscheck_new_entry

t  full_log        File '/var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/py.alfa' added
                   Mode: realtime

t  id              1711272582.20878789
```

Figure 15. Detection of File Addition

FIM operates by utilizing the Wazuh agent to gather information regarding specified files and folders, typically located in the /var/www/html directory. The Wazuh agent conducts hash value checks for each file in this directory, using MD5, SHA1, and SHA256 algorithms. Additionally, the Wazuh agent collects other information such as file names, locations, sizes, timestamps, file permissions, and file owners.

Subsequently, the gathered data is stored in a database, serving as a repository for comparing file integrity when new files are added. The comparison process is carried out by the Wazuh manager or server, which periodically compares the checksums of files stored in the database with those collected by the Wazuh agent. If a checksum discrepancy is detected, indicating a file change, an alert is generated on the Wazuh dashboard. Wazuh not only detects file changes or manipulations but also identifies the addition of new files and the deletion of files according to the folders configured for monitoring in the ossec.conf configuration file.



```
> Mar 24, 2024 @ 16:32:01.378  /var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/py.alfa       modified  Integrity checksum changed.  7  550
> Mar 24, 2024 @ 16:32:01.347  /var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/bash.alfa     modified  Integrity checksum changed.  7  550
> Mar 24, 2024 @ 16:32:01.332  /var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/.htaccess     modified  Integrity checksum changed.  7  550
> Mar 24, 2024 @ 16:32:01.318  /var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/perl.alfa     modified  Integrity checksum changed.  7  550
> Mar 24, 2024 @ 16:29:42.362  /var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/bash.alfa     added     File added to the system.    5  554
> Mar 24, 2024 @ 16:29:42.360  /var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/.htaccess     added     File added to the system.    5  554
> Mar 24, 2024 @ 16:29:42.358  /var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/perl.alfa     added     File added to the system.    5  554
> Mar 24, 2024 @ 16:29:42.357  /var/www/html/wp-content/themes/twentyfifteen/ALFA_DATA/alfacgiapi/py.alfa       added     File added to the system.    5  554
> Mar 24, 2024 @ 16:29:33.534  /var/www/html/wp-content/themes/twentyfifteen/404.php                            modified  Integrity checksum changed.  7  550
```

Figure 16. All FIM Detection on Scenario

4. CONCLUSION

Web defacement attacks represent a prevalent threat observed on government websites. One solution to mitigate such attacks is by leveraging open-source technologies that can be implemented at no cost. In this study, the utilization of Wazuh equipped with file integrity monitoring (FIM) modules for detecting web defacement attacks was explored. Through the tests conducted, including brute force attacks and web defacement scenarios, it was demonstrated that Wazuh effectively detects brute force activities. Furthermore, the Wazuh FIM module successfully identifies changes, additions, and deletions of files based on alterations in their hash values. This underscores the efficacy of employing Wazuh as a proactive defense mechanism against web defacement attacks. Further research can be done by testing the growing attack activities, such as ransomware and data breaches, and integrating Wazuh with MISP and OpenCTI.

ACKNOWLEDGEMENTS

The author extends their gratitude to all parties who helped and supported during the research process, including the provision of research facilities and other forms of aid.

REFERENCES

[1] S. Inc, “2024 SonicWall Cyber Threat Report,” 2024.
[2] Trend Micro, “Calibrating Expansion: 2023 Annual Cybersecurity Report,” 2023.
[3] BSSN, “KEAMANAN SIBER INDONESIA 2022 T L P : C L E A R,” Jakarta, 2023.
[4] D. Zlatkovski, A. Mileva, K. Bogatinova, and I. Ampov, “A New Real-Time File Integrity Monitoring System for Windows-based Environments.”
[5] B. Al-Muntaser, M. Afendee Mohamed, A. Yaseen Tuama, U. Sultan Zainal Abidin, and K. Terengganu, “Real-Time Intrusion Detection of Insider Threats in Industrial Control System Workstations Through File Integrity Monitoring,” IJACSA) International Journal of Advanced Computer Science and Applications, vol. 14, no. 6, pp. 327–333, 2023, [Online]. Available: www.ijacsa.thesai.org
[6] S. Agarwal, A. Sable, D. Sawant, S. Kahalekar, and M. K. Hanawal, “Threat Detection and Response in Linux Endpoints,” in 2022 14th International Conference on COMmunication Systems and NETworkS, COMSNETS 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 447–449. doi: 10.1109/COMSNETS53615.2022.9668567.
[7] A. A. Mughal and A. A. Mughal, “Building and Securing the Modern Security Operations Center (SOC),” International Journal of Business Intelligence and Big Data Analytics, 2022, [Online]. Available: https://orcid.org/0009-0006-8460-8006
[8] B. Shi, B. Li, L. Cui, and L. Ouyang, “Vanguard: A cache-level sensitive file integrity monitoring system in virtual machine environment,” IEEE Access, vol. 6, pp. 38567–38577, Jun. 2018, doi: 10.1109/ACCESS.2018.2851192.
[9] S. K. Peddoju, H. Upadhyay, and L. Lagos, “File integrity monitoring tools: Issues, challenges, and solutions,” Concurr Comput, vol. 32, no. 22, Nov. 2020, doi: 10.1002/cpe.5825.

[10] M. Albalawi, R. Aloufi, N. Alamrani, N. Albalawi, A. Aljaedi, and A. R. Alharbi, “Website Defacement Detection and Monitoring Methods: A Review,” Electronics (Switzerland), vol. 11, no. 21. MDPI, Nov. 01, 2022. doi: 10.3390/electronics11213573.

[11] A. Salman, M. S. Khan, S. Idrees, F. Akram, M. Junaid, and A. L. Malik, “File Integrity Checkers: Functionality, Attacks, and Protection,” in 2022 2nd International Conference on Digital Futures and Transformative Technologies, ICoDT2 2022, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICoDT255437.2022.9787428.

[12] D. Laksmiati, “IMPLEMENTASI WAZUH 4.0 UNTUK PERLINDUNGAN KEAMANAN INTEGRITAS FILE,” Jurnal AKRAB JUARA, vol. 6, pp. 164–174, 2021.

[13] T. Suryantoro, B. D. P. Purnomosidi, and W. Andriyani, “The Analysis of Attacks Against Port 80 Webserver with SIEM Wazuh Using Detection and OSCAR Methods,” in 2022 5th International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1–6. doi: 10.1109/ISRITI56927.2022.10052950.

[14] S. Stankovic, S. Gajin, and R. Petrovic, “A Review of Wazuh Tool Capabilities for Detection Attack Based on Log Analysis,” Serbia: IX International Conference IcETRAN, Jun. 2022, pp. 1–5.

[15] M. Romagna and N. Jan van den Hout, “Hacktivism and Website Defacement: Motivations, Capabilites and Potential Threats,” 27th Virus Bulletin Conference, vol. 1, pp. 1–10, 2017, [Online]. Available: http://www.zone-h.org/.

[16] Anggrahito, R. Ibrahim, A. Fajri, and E. Murniyanti, “Implementasi Web Application Firewall Menggunakan ReverseProxy dan ModSecurity Sebagai Alternatif Pengamanan Aplikasi Web Pada Sektor Pemerintah,” CITEE2019, pp. 199–205, Jul. 2019, [Online]. Available: http://news.netcraft.com/archives/2018/02/13/february-2018-web-server-

[17] Wazuh, “Getting started with Wazuh,” https://documentation.wazuh.com/current/getting-started/index.html. Accessed: Nov. 17, 2023. [Online]. Available: https://documentation.wazuh.com/current/getting-started/index.html

[18] Incident Response Team, “WEB DEFACEMENT : JUDI ONLINE,” 2023.

[19] A. Nursetyo, D. R. I. M. Setiadi, C. A. Sari, and E. H. Rachmawanto, “Website and Network Security Techniques against Brute Force Attacks using Honeypot,” Fourth International Conference on Informatics and Computing (ICIC), pp. 1–6, Oct. 2019, doi: 10.1109/ICIC47613.2019.8985686.

[20] P. G. Shah and J. Ayoade, “An Empricial Study of Brute Force Attack on Wordpress Website,” in Proceedings - 5th International Conference on Smart Systems and Inventive Technology, ICSSIT 2023, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 659–662. doi: 10.1109/ICSSIT55814.2023.10060966.