Equivalent: Jurnal Ilmiah Sosial Teknik
Volume 8, Issue 4, 1501-1513
e_ISSN: 2775-0833
https://jurnalequivalent.id/index.php/jequi/index
DOI: doi.org/10.59261/jequi.v8i4.408

Construction of Early Notification Framework to Anticipate Clone Phishing using GoPhish and Wazuh to Increase Blue Team's Detection Speed

Jeremy Pierre Tumbio* Universitas Bina Nusantara, Indonesia
Benfano Soewito Universitas Bina Nusantara, Indonesia

***Corresponding author:**
Jeremy Pierre Tumbio,
Universitas Bina Nusantara, Indonesia
✉jeremy.tumbio@binus.ac.id

Article Info:
Article history:
Received: July 06, 2026
Revised: August 27, 2026
Accepted: September 02, 2026
Available Online: September 10, 2026

Keywords:
API Polling; Clone Phishing; GoPhish; Mean Time to Detect; Wazuh SIEM.

Abstract

Background: Clone phishing can exploit trusted message formats and user behavior, while delayed event ingestion into a SIEM can prolong the interval during which a Blue Team remains unaware of a credential-compromise attempt. This study addresses the operational gap between phishing simulation events and automated security monitoring.
Objective: This study develops and evaluates an early-notification framework that integrates Gophish, Wazuh SIEM, API-based event forwarding, and Telegram notifications, with the measurable objective of reducing phishing-event detection latency relative to manual event injection.
Methods: A controlled comparative experiment was conducted in a Docker-based simulation environment using Gophish, MailHog, a Python API-polling engine, Wazuh SIEM, a Python integrator, and a Telegram bot. Manual Injection served as the baseline, while API Polling Forwarding served as the proposed mechanism. Detection latency was operationalized as the interval from the Gophish click timestamp to the Wazuh detection timestamp; Telegram delivery occurred after detection and was therefore not included in the mean time to detection (MTTD).
Results: The aggregate comparison reported in the experiment showed an average detection latency of 72.60 seconds for Manual Injection and 3.73 seconds for API Polling Forwarding, corresponding to a 94.86% reduction in mean detection latency and a mean-detection-latency ratio of 19.46. These figures describe detection latency, not notification-delivery latency.
Conclusion: Automated API-based forwarding substantially reduced the observed detection latency between Gophish and Wazuh and enabled automated post-detection notification to the Blue Team. Because polling introduces an interval-dependent delay, the framework is appropriately characterized as near-real-time rather than real-time.

To cite this article: Tumbio, J. P., & Soewito, B. (2026). Construction of Early Notification Framework to Anticipate Clone Phishing using GoPhish and Wazuh to Increase Blue Team's Detection Speed. *Equivalent: Jurnal Ilmiah Sosial Teknik*, *8*(4), 1501-1513. https://doi.org/10.59261/jequi.v8i4.408

INTRODUCTION

Phishing remains a persistent organizational cybersecurity problem because successful attacks can create a gap between user interaction and security-team awareness. In Security Operations Center (SOC) environments, that gap matters operationally: delayed ingestion of phishing events increases analyst response times, extends the credential-exposure window, adds to the manual monitoring workload, and may postpone incident containment. The growing scale of Internet use in Indonesia further increases the number of digital interactions in which such delays can become consequential (Cybersecurity, 2025; Group, 2025; Kemp, 2025).

The success of a cyberattack is generally influenced by several factors, including

weaknesses in applications, operating systems, network configurations, and human factors involving system users. Vulnerabilities in applications and operating systems can be relatively well controlled through system updates, security patches, hardening, and security testing, such as penetration testing. Nevertheless, human factors are often a weak point that is more difficult to control because they are related to behavior, security literacy, risk perception, and user responses to convincing-looking messages (Business, 2024; Chen et al., 2025; Jayatilaka et al., 2021).

Verizon Business (2024) reports that 68% of data breach incidents involve human elements, including phishing, social engineering, user error, and credential misuse. One of the most commonly used forms of social engineering is phishing, which is a fraudulent technique that involves impersonating a trusted party to obtain sensitive information from victims. Phishing attacks utilize psychological manipulation, false urgency, email messages that resemble official communications, malicious links, and fake pages to encourage users to click links or submit credentials (Hillman et al., 2023; Lain et al., 2022; Rozema & Davis, 2025). Previous research has shown that even when organizations have implemented technical detection systems and security training, users can still be deceived by phishing because email-related decision-making is influenced by context, habits, trust in the sender, and the level of understanding of threat indicators (Hillman et al., 2023; Jayatilaka et al., 2021). In this study, clone phishing is operationally defined as a phishing attempt that reproduces the structure and appearance of a legitimate or previously trusted message while substituting a malicious link or credential-capture destination. The simulated messages therefore qualify as clone-phishing scenarios because GoPhish reproduces controlled email content and records user interactions, while MailHog provides a safe SMTP sandbox for delivery. GoPhish is appropriate for this purpose because its campaign and event APIs expose click and credential-submission timestamps that can be forwarded to a SIEM.

To reduce these risks, organizations generally implement security awareness training programs to increase users' understanding of information security threats. In addition to training, a testing mechanism based on phishing simulations is also needed to measure user alertness to attack scenarios that resemble real-world conditions. Research by Lain et al. (2022) shows that phishing simulations in organizations can be used to measure click rates, risky actions, and suspicious-email reporting behavior. Meanwhile, Sirawongphatsara et al. (2025) show that phishing simulations in critical infrastructure organizations can help evaluate the effectiveness of security training and user responses to different phishing scenarios (Srivastava & Mukhopadhyay, 2026).

In addition to the organizational context, low information security awareness has also been identified among nontechnical user groups. Research by Parhusip et al. (2026) shows that the level of student awareness of personal data security still varies, particularly between informatics engineering and non-informatics engineering students. This suggests that differences in technological knowledge backgrounds can affect users' ability to recognize digital security risks. Therefore, education, simulation, monitoring, and early detection mechanisms are important for supporting faster and measurable phishing attack mitigation.

Cyberattacks that exploit human weaknesses are generally carried out through social engineering techniques, such as the manipulation of malicious links, credential theft, identity impersonation, and misuse of users' personal information. Hartanto et al. (2025) explained that digital security education can increase public awareness of phishing links because users need to understand the characteristics of suspicious links, the risks of entering personal data on fake pages, and the importance of exercising caution before clicking. In the context of e-commerce, Fitrian et al. (2024) emphasized that phishing prevention needs to be carried out by strengthening network security, using more secure authentication, and educating users to recognize fake websites and malicious links.

Previous research has also addressed phishing prevention efforts through social and institutional approaches. Syah (2023) discusses the police's strategy for preventing phishing crimes through social media in cyberspace by emphasizing the importance of public education, digital security awareness campaigns, and supervision of online fraud activities (Batu & Hasan, 2026). Meanwhile, Gumay et al. (2024) analyzed the impact of cybercrime threats on student data in the SIAK UIKA phishing web attack and found that some students still accessed phishing sites and entered credential data. These findings reinforce the urgency of increasing awareness and

implementing detection mechanisms that can help prevent the misuse of personal data.

Although previous studies have discussed phishing prevention in terms of education, awareness, network security, and institutional strategies, research remains limited in terms of near-real-time monitoring and detection of phishing events. Most studies focus primarily on preventive efforts and increasing user awareness, but there has been limited discussion of automated phishing-event forwarding mechanisms to support the detection process in Security Information and Event Management (SIEM) environments. In fact, the integration of SIEM with notification systems can increase threat visibility and accelerate the cybersecurity monitoring process (Farrel et al., 2024; Jumiaty & Soewito, 2024). Previous phishing studies have demonstrated the value of awareness programs, simulation, user-behavior analysis, and SIEM-based alerting, but these strands are usually evaluated separately. Simulation studies commonly measure click or reporting behavior (Hillman et al., 2023; Lain et al., 2022; Wang et al., 2026), whereas SIEM studies emphasize centralized monitoring and notification (Farrel et al., 2024; Jumiaty & Soewito, 2024). What remains insufficiently demonstrated is an automated pathway that transfers phishing-simulation events into SIEM detection with a measurable latency benchmark against a manual forwarding process. This unresolved integration and measurement problem constitutes the specific research gap addressed in this study.

In addition, previous research has not discussed the process of automatically collecting phishing events to support the measurement of Mean Time to Detect (MTTD) in a Blue Team monitoring environment. In the initial implementation of the GoPhish and Wazuh integration, phishing events could not be automatically forwarded to the SIEM system, so a semi-manual approach using an appended JSON log was needed to validate the process of parsing events and generating alerts. This condition meant that the event-forwarding process still relied on manual intervention and did not support optimal near-real-time detection. The novelty of this study is threefold: technically, the framework automates API-based forwarding of GoPhish events; operationally, it links Wazuh detection to automated Telegram notification for the Blue Team; and evaluatively, it compares the mean detection latency of API forwarding with manual injection using MTTD as the performance indicator.

Based on these conditions, this study builds an automated mechanism for forwarding phishing events through API polling so that phishing events from GoPhish can be automatically forwarded to and detected by Wazuh in near-real-time. This research aims to build a Python-based system framework that integrates GoPhish with a SIEM to improve Blue Team detection speed. The benefit of this study is that it provides an implementable overview of GoPhish-Wazuh integration for accelerating phishing attack detection, allowing the approach to be adapted by organizations to improve the effectiveness of their cyber defense systems. Accordingly, the study has three operational objectives: (1) to implement automated forwarding of clone-phishing events from GoPhish to Wazuh through a Python-based polling API; (2) to verify automated post-detection notification from Wazuh to Telegram; and (3) to quantify the difference in detection latency between Manual Injection and API Polling Forwarding using MTTD. The study addresses the following questions: How reliably can GoPhish events be ingested and detected by Wazuh through polling-based forwarding? How much does API forwarding reduce mean detection latency relative to manual injection? How does automated notification support Blue Team monitoring after SIEM detection?

METHOD

This study used a controlled comparative experiment with repeated system testing to evaluate detection latency under two forwarding conditions within the same Docker-based simulation environment. GoPhish generated clone-phishing interactions, MailHog served as the SMTP sandbox, the Python API polling engine forwarded events, Wazuh SIEM performed detection, the Python integrator processed alerts, and the Telegram Bot delivered post-detection notifications. All test data were synthetic, and no production or personally identifiable data were used.

The unit of analysis was a phishing-event test run rather than a human population. Each run began with a controlled GoPhish interaction and ended when the corresponding event was detected by Wazuh. The experimental workflow comprised scenario preparation, simulated email delivery, controlled click or credential-submission events, event forwarding to Wazuh, alert

generation, Telegram notification, and timestamp comparison. The same event logic was used for the baseline and proposed forwarding mechanisms so that detection latency could be compared on an equivalent basis.

The evaluation was carried out by comparing two test scenarios, namely Manual Injection as the baseline and API Polling Forwarding as the proposed mechanism. In the Manual Injection scenario, the phishing event was passed to Wazuh manually, whereas in the API Polling Forwarding scenario, the phishing event was periodically retrieved from the GoPhish API and automatically forwarded to Wazuh in JSON format. Both scenarios used the same test data so that the measurement results could be objectively compared. The manuscript reports aggregate results across 30 test scenarios. The two methods were evaluated using the same controlled event logic; consequently, the reported MTTD comparison should be interpreted as a within-system comparison of forwarding conditions rather than as an analysis involving human participants.

The main parameter used in this study was Mean Time to Detect (MTTD), defined as the time difference between when a user clicked a phishing link in GoPhish and when the corresponding event was detected by Wazuh. The average MTTD was calculated using the following formula:

$$Average\ MTTD = \frac{\sum_{i=1}^{n}(T_{detect,i} - T_{click,i})}{n}$$

with the detection time of the i event by Wazuh, Tclick, iT_{click,i}Tclick, i is the time of the user's click on the phishing link, and nnn is the number of phishing events analyzed. The research was declared successful if the API Polling Forwarding mechanism produced a lower MTTD value than the Manual Injection scenario. The decrease in MTTD value shows that the integration of GoPhish and Wazuh through the Polling API is able to increase the detection speed and support near-real-time cybersecurity monitoring for Blue Teams. For n detected phishing events, MTTD = (1/n) Σ(i=1 to n) [T_detection,i - T_click,i], where T_detection,i is the Wazuh detection timestamp for event i and T_click,i is the corresponding GoPhish interaction timestamp. A lower MTTD indicates lower detection latency. The relative reduction in mean latency is calculated as [(MTTD_manual - MTTD_API) / MTTD_manual] × 100%, while the mean-time ratio is MTTD_manual / MTTD_API.

RESULTS AND DISCUSSION

Results

MTTD Evaluation

The aggregate latency results show a mean MTTD of 72.60 seconds for Manual Injection and 3.73 seconds for API Polling Forwarding. Based on these reported means, API forwarding reduced the mean detection latency by 94.86% [(72.60 − 3.73) / 72.60 × 100] and produced a mean-time ratio of 19.46 (72.60 / 3.73). These values quantify the detection stage only. Telegram notification is triggered after Wazuh detection and therefore represents a separate notification-delivery stage.

Designing an Automatic Notification Mechanism

Once the phishing event was successfully forwarded and detected by Wazuh, the system sent an automatic notification to the security analyst via the Telegram bot. Notifications were sent based on events identified as *clicked_link* and *submitted_credentials*.

The automated notification system was designed to speed up the delivery of information related to phishing activities to the security team after the events were successfully detected by Wazuh. This mechanism started with GoPhish, which was used to simulate phishing attacks and generate events based on user interactions. The events were then retrieved periodically using the polling API mechanism implemented in the Python Engine.

The Python Engine functioned to process events obtained from GoPhish, normalize the data, and pass the information to the Wazuh SIEM in JSON format. Furthermore, Wazuh performed the detection process based on preconfigured rules to identify phishing activities such as *clicked_link* and *submitted_credentials*.

If an event was successfully detected by Wazuh, the system automatically generated a

notification message containing important information such as the campaign name, target email address, event type, and event time. The information was then sent via the Telegram bot to security analysts. With this mechanism, the security team could obtain information quickly without having to continuously monitor the Wazuh dashboard, allowing phishing activities to be monitored more effectively.

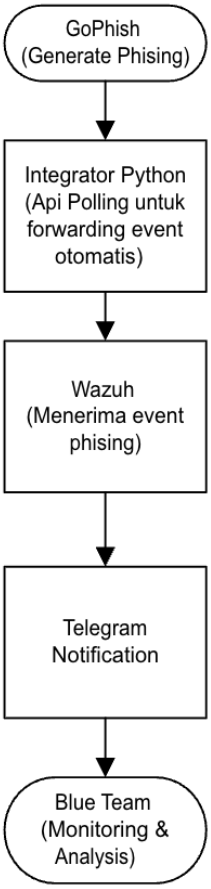

Figure1. Flow Notification flow implementation

Figure 1 shows the flow of the automated notification mechanism proposed in this study. The process started with GoPhish generating phishing events based on user activity. Next, the Python Engine retrieved the data through the polling API mechanism and forwarded the events to Wazuh in JSON format. Wazuh then performed detection based on the configured rules and generated alerts when phishing activity was identified. The alerts were subsequently sent through the Telegram bot as automatic notifications to security analysts. With this mechanism, the SOC team could obtain information quickly and monitor events based on the priority level of the detected activity.

Telegram Bot Implementation

In this study, a Telegram bot was used as a medium for delivering automatic notifications of phishing events detected by Wazuh. Telegram was selected based on its ease of integration through the Telegram Bot API and its ability to deliver near-real-time messages to security analysts. The integration process was carried out by creating a bot through the BotFather service to obtain a bot token and chat ID, which were used to authenticate and identify the destination for notification delivery.

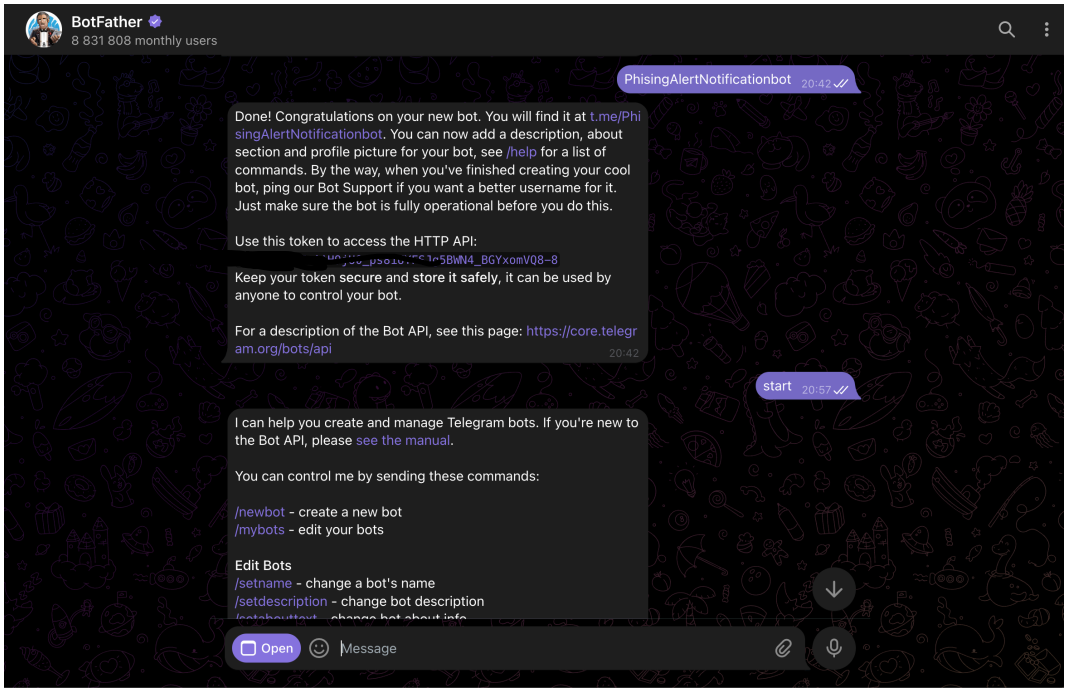

Figure 2. The process of getting the API from the telegram bot

The Telegram bot was created using the BotFather service provided by Telegram. BotFather was used to create and manage the bot that served as the medium for sending automatic notifications in this study. Once the creation process was completed successfully, the system obtained a bot token that served as the bot's authentication credential when communicating with the Telegram Bot API. The created bot was then used to send notifications related to phishing events detected by Wazuh.

Notification Delivery Implementation

After the phishing event was successfully forwarded to Wazuh and detected based on the configured rules, the Python Engine generated a notification message containing information related to the detected event. The information sent included the campaign name, target email address, event type, priority level, and event time. The system then sent the message to Telegram using the Telegram Bot API so that it could be automatically received by security analysts.

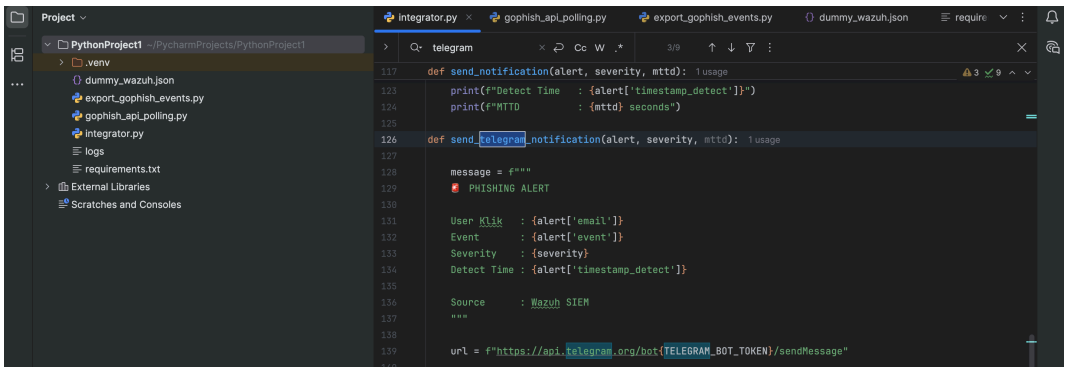

Figure 3. Implementation of the telegram notification sending function

Figure 3 shows the implementation of the *send_telegram_notification()* function in the Python Engine, which was used to send automated notifications to Telegram. This function was responsible for generating notification messages based on phishing event information detected by the Wazuh SIEM, such as the target email address, event type, severity level, detection time, and Mean Time to Detect (MTTD) value.

Once the message was successfully generated, the system used the Telegram Bot API to send the information to the predetermined Telegram account through the bot token and chat ID.

This implementation enabled the delivery of information related to phishing activities to security analysts in near-real-time without the need for continuous monitoring of the Wazuh dashboard.

Automated Notification Testing

Testing was conducted to ensure that phishing events detected by Wazuh could be automatically sent to Telegram. The testing was conducted using two scenarios, namely *clicked_link* and *submitted_credentials* events.

Scenario 1 – Clicked Link

In the first scenario, the user clicked on a phishing link sent through GoPhish. The event was successfully forwarded to Wazuh through the Python Engine and generated an alert that was subsequently sent to Telegram.

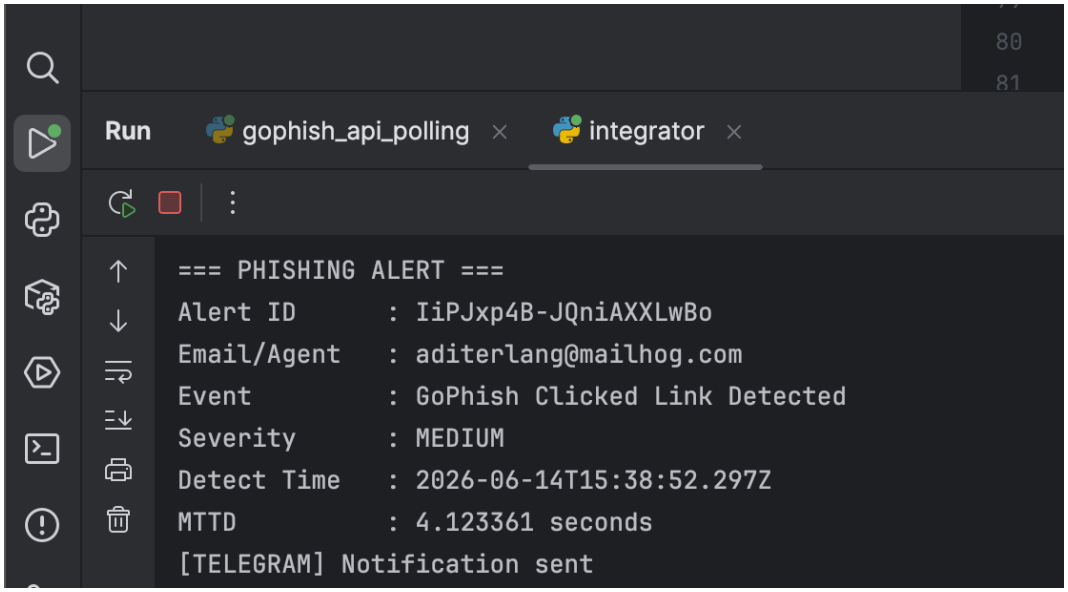

Figure 4.The results of sending the event clicked link to telegram

Figure 4 shows the results of the notification-sending process for *clicked_link* events that were successfully detected by Wazuh. The displayed information included the Alert ID, target email address, event type, severity level, detection time, and Mean Time to Detect (MTTD) value. In addition, the "[TELEGRAM] Notification sent" status indicated that the system successfully sent the notification to Telegram automatically without requiring manual intervention from security analysts.

Figure 5. The results of sending the event clicked link to telegram

Figure 5 shows the successful delivery of another *clicked_link* event notification to Telegram after the event was processed by the Python Engine and detected by Wazuh. The information displayed in the notification included the target identity, event type, priority level,

detection time, and MTTD value used to measure the speed of the detection process. These results demonstrate that the automated notification mechanism was able to operate consistently in delivering information related to phishing activities to security analysts in near-real-time.

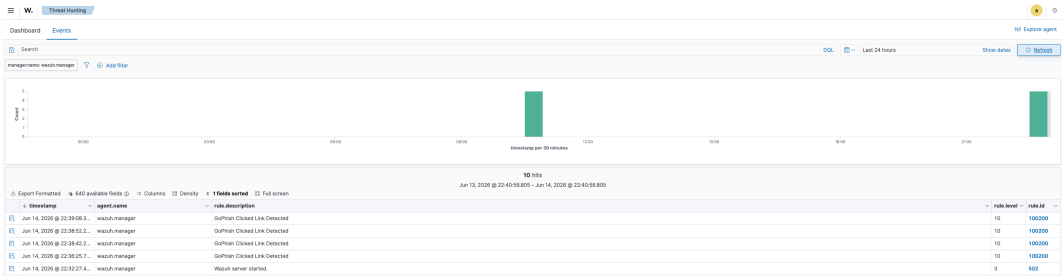

Figure 6. Wazuh Dashboard

Figure 6 shows a phishing event that was successfully received and detected by the Wazuh SIEM. Events originating from GoPhish were displayed on the Wazuh dashboard based on preconfigured rules. In this view, several phishing events, such as *GoPhish Clicked Link Detected* and *GoPhish Submitted Credentials Detected*, were successfully recognized by the system. These results demonstrate that the event-forwarding mechanism from GoPhish to Wazuh functioned properly, enabling phishing activities to be centrally monitored through the dashboard.

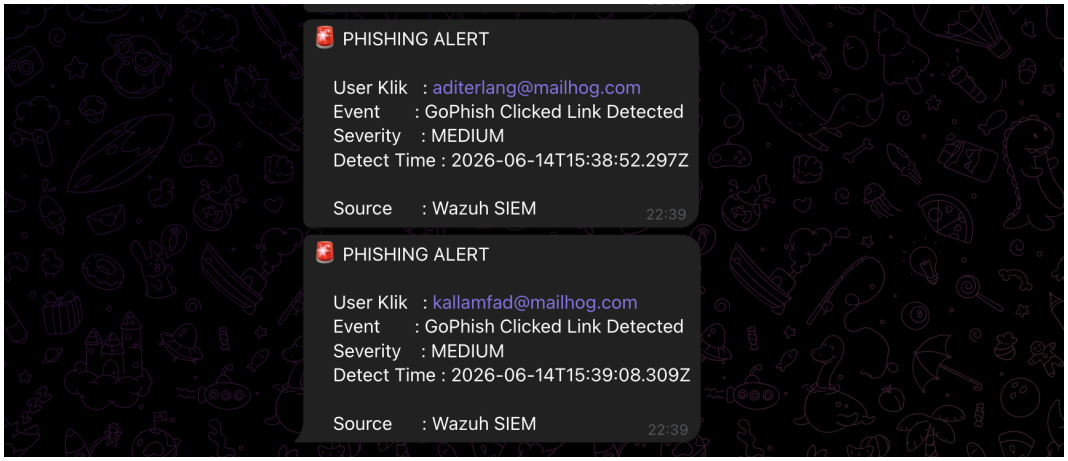

Figure 7. Notifications in Telegram

Figure 7 shows an automated notification that was successfully sent to Telegram after a phishing event was detected by the Wazuh SIEM. The information displayed in the notification included the target email address, event type, severity level, detection time, and the source of the alert from Wazuh. These results demonstrate that the automated notification mechanism proposed in this study was able to convey information related to phishing activities to security analysts in near-real-time without the need for continuous monitoring of the Wazuh dashboard.

Scenario 2 – Submitted Credentials

In the second scenario, the user filled in and submitted credentials on the provided phishing page. The *submitted_credentials* event was successfully forwarded to Wazuh and generated an alert with a higher priority level than the *clicked_link* event. The alert was then automatically sent to Telegram.

```
=== PHISHING ALERT ===
Alert ID      : zZrdxp4BcNUiDpfSCH7s
Email/Agent   : pierrelocal@mailho.com
Event         : GoPhish Submitted Credentials Detected
Severity      : HIGH
Detect Time   : 2026-06-14T16:00:32.936Z
MTTD          : 14.271229 seconds
[TELEGRAM] Notification sent
```

Figure 8. Send event submitted credentials to Telegram

Figure 8 shows the results of sending notifications for *submitted_credentials* events that were successfully detected by Wazuh. This event occurred when a user filled in and submitted credentials on the provided phishing page. The displayed information included the Alert ID, target email address, event type, severity level, detection time, and Mean Time to Detect (MTTD) value. The "[TELEGRAM] Notification sent" status indicated that the system successfully sent an automatic notification to Telegram after the event was detected.

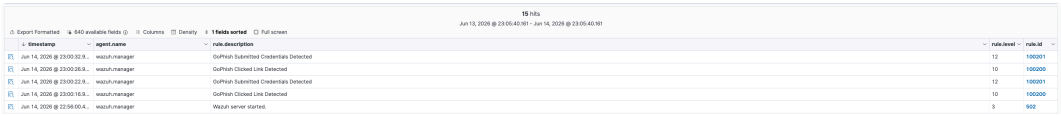

Figure 9. Wazuh dashboard

Figure 9 shows the *GoPhish Submitted Credentials Detected* event that was successfully detected on the Wazuh SIEM dashboard. The event originated from the data forwarded by the Python Engine after the user submitted credentials on a phishing page. The appearance of the alert on the Wazuh dashboard demonstrates that the event-forwarding mechanism and configured detection rules were functioning properly, enabling phishing activities to be automatically identified by the system.

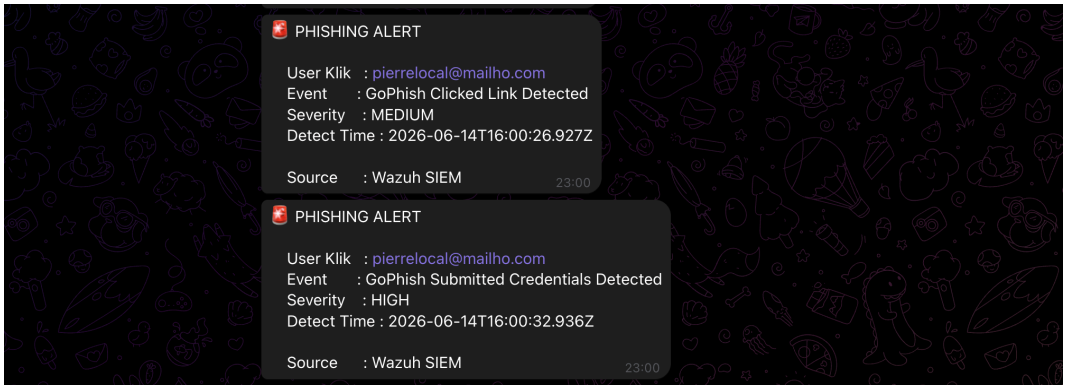

Figure 10. Notifications in Telegram

Figure 10 shows the automatic notification received via Telegram after a *submitted_credentials* event was successfully detected by Wazuh. The notification contained important information such as the target email address, event type, severity level, and detection time. In this implementation, the *submitted_credentials* event was categorized as high priority (HIGH) because it indicated that the user had entered credentials on the phishing page. The test results demonstrate that the system was able to deliver information related to phishing activities to security analysts in near-real-time, allowing response actions to be carried out more quickly.

Based on the test results in the second scenario, the *submitted_credentials* event was successfully forwarded from GoPhish to Wazuh through the Python Engine and generated an alert according to the configured rules. The alert was then successfully sent to Telegram automatically with a high priority level. These results demonstrate that the proposed automatic notification

mechanism was able to support the cybersecurity monitoring process for phishing activities that could potentially compromise user credentials.

Implementation Results

Based on the results of the tests conducted, the automatic notification system successfully sent information related to phishing events detected by Wazuh to Telegram in near-real-time. This mechanism allowed security analysts to obtain information more quickly than through manual dashboard monitoring. Thus, the implementation of automated notifications supported the cybersecurity monitoring process and helped increase the effectiveness of phishing incident detection.

Table 1

System implementation results

Event	Wazuh Alert	Telegram Notification
Clicked_link	Successful	Successful
Submitted _credentials	Successful	Successful

Based on the implementation results shown in Table 1, all tested phishing events were successfully processed by the system according to the design. The *clicked_link* and *submitted_credentials* events were successfully forwarded from GoPhish to Wazuh through the polling API mechanism and event-forwarding process implemented in the Python Engine. Furthermore, both events were successfully detected by Wazuh based on the configured rules and generated alerts according to the category of activity performed by the user.

In addition, the test results showed that all alerts successfully detected by Wazuh could be automatically sent to Telegram through the Telegram Bot API. Thus, the automatic notification mechanism proposed in this study was successfully implemented and was able to support near-real-time cybersecurity monitoring. The successful implementation demonstrated that the system could help security analysts obtain information related to phishing activities more quickly than through manual dashboard monitoring.

Thus, all stages of implementation and testing conducted in this study successfully addressed the second research problem concerning the implementation of an automatic notification mechanism for phishing events detected by the Wazuh SIEM. The system was able to automatically send notifications to Telegram after a phishing event was detected, thereby supporting the cybersecurity monitoring process and accelerating the delivery of information to security analysts.

Discussion

The lower MTTD under API Polling Forwarding is attributable primarily to the removal of the manual handoff between Gophish and Wazuh. The polling engine automatically retrieves, normalizes, and forwards event data, so detection latency is dominated by the polling interval, processing time, and SIEM rule evaluation rather than by analyst action. This mechanism explains why the observed mean decreased from 72.60 seconds to 3.73 seconds. The improvement should not be interpreted as zero-latency streaming: polling introduces a bounded waiting period, and higher event volumes could shift the bottleneck toward API retrieval, queue handling, parsing, or Wazuh rule processing.

From the operational perspective of the Security Operations Center (SOC), the system's ability to automatically forward events provides increased visibility into the activities of users who interact with phishing emails. Events such as clicked_link and submitted_credentials are not only successfully recorded as logs but also immediately processed into alerts that can be followed up by the Blue Team. This is important because one of the main challenges in handling cyber incidents is not only the ability to detect threats but also the speed with which information is delivered to security analysts so that an appropriate response can be initiated before the impact of an attack spreads further. This distinction also clarifies the practical contribution of the framework. Detection latency ends when Wazuh generates the alert; notification latency is the subsequent interval required to deliver that alert through Telegram. Prior work has shown that SIEM integration and messaging channels can improve threat visibility (Farrel et al., 2024; Jumiaty

& Soewito, 2024), while phishing-simulation research has primarily emphasized user behavior (Hillman et al., 2023; Lain et al., 2022). The present framework connects these two functions by measuring the upstream detection delay and automating the downstream notification path.

The implementation of a Telegram bot as a notification medium also provides added value to the security monitoring process. In conventional SIEM implementations, security analysts generally have to monitor the dashboard continuously for new alerts. This study shows that the push-notification mechanism can change the monitoring approach from passive monitoring to active notification, allowing information about phishing activities to be received as soon as an alert is generated. This approach has the potential to reduce operational response delays, especially in organizations with a limited number of SOC personnel.

In addition to the successful implementation of the notification mechanism, the results of the study also show that the system is able to assign priority levels based on the type of user activity. The submitted_credentials event was assigned a higher severity level than clicked_link because it indicates that the user entered authentication information on the phishing page. This classification approach is consistent with Security Operations Center incident-triage practices, which prioritize handling based on the level of risk posed by a security event. Thus, security analysts can focus resources on incidents with the greatest potential impact. In Wazuh, rule levels determine the severity of generated alerts, making the term “severity level” appropriate when referring to the resulting alert classification.

The findings of this study reinforce the results of Farrel et al.'s (2024) research, which showed that the integration of Wazuh with Telegram can increase the effectiveness of automatically delivering security alerts. However, this study makes a different contribution because the source of the events used does not consist of brute-force activities or network attacks but rather of a phishing simulation using Gophish, with events forwarded through an API polling mechanism. In other words, this study expands the application of Wazuh integration into the domain of phishing detection, which has not been widely discussed in previous research.

The results of this study also complement the research of Jumiaty and Soewito (2024) regarding the use of Wazuh in the implementation of SIEM and threat intelligence. While previous research focused more on the integration of SIEM with threat intelligence platforms, this study shows that such integration can be expanded to include phishing-simulation platforms. This integration allows user activity during the phishing simulation to be treated as a security event that is then automatically analyzed by the SIEM to support the Blue Team monitoring process.

From the perspective of phishing simulation, this study also supports the findings of Lain et al. (2022) and Sirawongphatsara et al. (2025), who stated that phishing simulation is an effective means of evaluating an organization's readiness to face social engineering attacks (Bethany et al., 2025; Goldenits et al., 2026). However, both studies focus more on measuring user behavior, such as click rates and credential-submission rates, whereas this study adds a new aspect in the form of automated monitoring through SIEM integration. Thus, this study not only evaluates user behavior but also improves the operational readiness of Blue Teams through automation of the detection process.

The implementation of API polling also shows that this approach is effective when phishing-simulation platforms do not provide a mechanism for directly streaming events to a SIEM. By collecting data periodically through the API, the system is able to obtain the latest events without requiring modifications to the Gophish application. This approach is relatively simple to implement, easy to develop, and adaptable to various organizational environments that use Wazuh-based SIEM platforms.

However, this study still has some limitations. All testing was conducted in a Docker-based laboratory environment, so system performance does not fully represent the actual conditions of an organization's network. In addition, the phishing events used were limited to two types of activities, namely clicked_link and submitted_credentials. In production implementations, variations in user activity and substantially larger event volumes are likely to affect system performance and require further testing of the scalability of the forwarding mechanism and its ability to process alerts concurrently.

Overall, the results of the study show that the integration of Gophish, a Python-based engine, Wazuh SIEM, and a Telegram bot can establish an automated, integrated framework that supports the detection process in near-real-time. This approach makes a practical contribution to

organizations that seek to improve the operational effectiveness of Blue Teams without making major changes to their existing security infrastructure. In addition, this research opens opportunities for further development, such as integration with SOAR platforms for automated incident response, application of machine learning to the classification of phishing alerts, and evaluation of system performance in production environments with higher event volumes.

CONCLUSION

This study developed a Gophish-Wazuh early-notification framework and evaluated its detection latency against a manual-forwarding baseline. The aggregate results show that mean MTTD decreased from 72.60 seconds with Manual Injection to 3.73 seconds with API Polling Forwarding, equivalent to a 94.86% reduction in mean detection latency and a mean-time ratio of 19.46. The framework also automated Telegram notification after Wazuh detection, thereby reducing dependence on continuous dashboard monitoring.

ACKNOWLEDGEMENT

The authors thank the interview participants and the anonymized case organization (Company X) for their time, cooperation, and access to supporting materials. This study received no external funding. The views expressed in this paper are solely those of the authors.

AUTHOR CONTRIBUTION STATEMENT

Jeremy Pierre Tumbio contributed to the conceptualization, research design, development and implementation of the GoPhish-Wazuh integration framework, experimental setup, data collection, data analysis, interpretation of results, and preparation of the original manuscript draft. Benfano Soewito contributed to research supervision, methodological validation, theoretical framework development, critical evaluation of the findings, and manuscript refinement. Both authors contributed to the revision process, approved the final version of the manuscript, and agreed to be accountable for all aspects of the study.

REFERENCES

Batu, A. D. A. H., & Hasan, Y. A. (2026). Aspek Krimonologi Terhadap Penipuan Online Pada Transaksi Phishing Di Wilayah Hukum Kepolisian Resor Kota Besar Makassar. *Indonesian Journal Of Legality Of Law*, *8*(2), 229–235.

Bethany, M., Galiopoulos, A., Bethany, E., Karkevandi, M. B., Beebe, N., Vishwamitra, N., & Najafirad, P. (2025). Lateral Phishing With Large Language Models: A Large Organization Comparative Study. *IEEE Access*.

Business, V. (2024). *2024 Data Breach Investigations Report*. Verizon Business. Https://Www.Verizon.Com/Business/Resources/Reports/Dbir/

Chen, L. C., Pardeshi, M. S., Liao, Y. X., & Pai, K. C. (2025). Application Of Retrieval-Augmented Generation For Interactive Industrial Knowledge Management Via A Large Language Model. *Computer Standards & Interfaces*, *94*, 103995.

Cybersecurity, E. U. A. For. (2025). *Enisa Threat Landscape 2025*. European Union Agency For Cybersecurity. https://www.Enisa.Europa.Eu/Publications/Enisa-Threat-Landscape-2025

Farrel, F. I., Mardianto, I., & Qamar, A. S. (2024). Implementation Of Security Information & Event Management (Siem) Wazuh With Active Response And Telegram Notification For Mitigating Brute Force Attacks On The Gt-I2ti Usakti Information System. *Intelmatics*, *4*(1), 1–7. https://doi.org/10.25105/Itm.V4i1.18529

Fitrian, H. P., Waryani, W., & Kirana, N. R. (2024). Analisis Keamanan Jaringan Komputer Terhadap Ancaman Phising Pada Pengguna E-Commerce. *Jurnal Nasional Komputasi Dan Teknologi Informasi*, *7*(6), 2347–2354. https://doi.org/10.32672/Jnkti.V7i6.8388

Goldenits, G., König, P., Raubitzek, S., & Ekelhart, A. (2026). Small Language Models For Phishing Website Detection: Cost, Performance, And Privacy Trade-Offs. *Journal Of Cybersecurity And Privacy*, *6*(2), 48.

Group, A.-P. W. (2025). *Phishing Activity Trends Report: 1st Quarter 2025*. Anti-Phishing Working Group. https://docs.Apwg.Org/Reports/Apwg_Trends_Report_Q1_2025.Pdf

Gumay, B., Hendrawan, A. H., & Kusumah, F. S. F. (2024). Analisis Dampak Ancaman Cybercrime Terhadap Data Mahasiswa Pada Serangan Web Phising Siak Uika. *Infotech Journal*, *10*(2),

297–305. https://doi.org/10.31949/Infotech.V10i2.11463
Hartanto, B. D., Nugraha, T. A., Ramadhan, B. R., Pratama, M. A., & Alamsyah, R. P. (2025). Edukasi Keamanan Digital Untuk Meningkatkan Kewaspadaan Masyarakat Terhadap Link Phising. *Jurnal Pengabdian Sosial*, *2*(9), 4341–4346. https://doi.org/10.59837/Nndaqp49
Hillman, D., Harel, Y., & Toch, E. (2023). Evaluating Organizational Phishing Awareness Training On An Enterprise Scale. *Computers & Security*, *132*, 103364. https://doi.org/10.1016/J.Cose.2023.103364
Jayatilaka, A., Arachchilage, N. A. G., & Babar, M. A. (2021). Falling For Phishing: An Empirical Investigation Into People's Email Response Behaviors. In *Proceedings Of The 42nd International Conference On Information Systems (Icis 2021)*. Association For Information Systems. https://Aisel.Aisnet.Org/Icis2021/Cyber_Security/Cyber_Security/1/
Jumiaty, & Soewito, B. (2024). Siem And Threat Intelligence: Protecting Applications With Wazuh And Thehive. *International Journal Of Advanced Computer Science And Applications*, *15*(9), 239–251. https://doi.org/10.14569/Ijacsa.2024.0150923
Kemp, S. (2025). *Digital 2025: Indonesia*. Datareportal. https://Datareportal.Com/Reports/Digital-2025-Indonesia
Lain, D., Kostiainen, K., & Capkun, S. (2022). Phishing In Organizations: Findings From A Large-Scale And Long-Term Study. In *2022 IEEE Symposium On Security And Privacy (Sp)* (Pp. 842–859). IEEE. https://doi.org/10.1109/Sp46214.2022.9833766
Parhusip, J., Zahra, F. U. H., Monalisa, A., Kurniawan, M. R., & Lowryanty, N. P. (2026). Analisis Tingkat Kesadaran Mahasiswa Terhadap Keamanan Data Pribadi Di Era Digital: Studi Komparatif Antara Mahasiswa Teknik Informatika Dan Non-Teknik Informatika. *Riggs: Journal Of Artificial Intelligence And Digital Business*, *4*(4), 8091–8101. https://doi.org/10.31004/Riggs.V4i4.4516
Rozema, A. T., & Davis, J. C. (2025). Anti-Phishing Training (Still) Does Not Work: A Large-Scale Reproduction Of Phishing Training Inefficacy Grounded In The Nist Phish Scale. *Arxiv Preprint Arxiv:2506.19899*.
Sirawongphatsara, P., Pornpongtechavanich, P., Phanthuna, N., & Daengsi, T. (2025). Comparative Simulation Of Phishing Attacks On A Critical Information Infrastructure Organization: An Empirical Study. *Bulletin Of Electrical Engineering And Informatics*, *14*(2), 1526–1534. https://doi.org/10.11591/Eei.V14i2.8020
Srivastava, P., & Mukhopadhyay, A. (2026). Cyber-Risk Assessment And Mitigation Framework For Critical Information Infrastructure: Mixed-Methods Approach. *Journal Of Enterprise Information Management*, 1–27.
Syah, R. (2023). Strategi Kepolisian Dalam Pencegahan Kejahatan Phising Melalui Media Sosial Di Ruang Siber. *Jurnal Impresi Indonesia*, *2*(9), 864–870. https://doi.org/10.58344/Jii.V2i9.3594
Wang, R. C., Chen, M. C., Chang, K. T., & Hsueh, T. Y. (2026). Spai Phishing: A Design And Evaluation Of Ai-Assisted Large Spear Phishing Cybersecurity Drill. *Systems And Soft Computing*, 200518.