Josh Murdock

Enhancing Threat Detection with Wazuh: Custom Rules, YARA Integration, and AI-Assisted Log Analysis

Abstract

The FBI reports that in 2024 alone the United States lost $16.6 billion to cybercrime—a 33% increase from 2023 (Federal Bureau of Investigation, 2025). This is a result of not only the increasing number of malicious actors but also the world's reliance on digital systems. The need for secure digital systems is at an all-time high and isn't going to go down anytime soon. Large organizations and industries with large, interconnected infrastructures are arguably the most vulnerable. This is because of their numerous access points creating a broad attack surface for cyber threats. To combat these threats, organizations must implement proactive security strategies and leverage technologies like a Security Information and Event manager or SIEM for short. SIEMs are used to detect and respond to potential intrusions quickly. It does this by collecting, analyzing, and responding to security data from across an organization's IT infrastructure. With a SIEM you can protect yourself from attacks like SSH brute force attacks, malware, and lateral movement just to name a few. A good SIEM option is a free and open-source platform called Wazuh.

Introduction

Cybersecurity threats such as brute-force attacks and malware infections continue to pose significant risks to organizations of all sizes. To address these challenges, Security Information and Event Management (SIEM) platforms like Wazuh offer powerful tools for log analysis, intrusion detection, and automated response. This paper presents a highly customized, open-source SIEM solution built on Wazuh, enhanced with YARA pattern matching and an AI-powered analysis layer. The SIEM detects and blocks SSH brute-force attacks using Wazuh's Active Response mechanism and custom rules. The SIEM also leverages YARA to identify malware signatures and integrates ChatGPT-4o to provide real-time, human-readable explanations of YARA detections. This combination of technologies improves threat visibility, reduces analyst workload, and accelerates incident response. Wazuh is completely open source, an essential feature in the context of security because of the auditability of the code that Wazuh runs on.

Background

SIEM platforms are essential tools in modern cybersecurity and are used daily by organizations to protect their data. One of the most popular SIEMs used by large enterprises is a proprietary platform called Splunk. It is a good choice but only if you have the budget for it because it is quite expensive to come in at around $1,800-$18,000 per year for 1-10 GB/day ingestion (Uptrace Team). Another popular SIEM is Microsoft's proprietary Sentinel. This SIEM is cloud native and runs inside Microsoft's cloud computing platform azure. Like Splunk's pricing Sentinel comes in at $1,900-$18,960 per year for 1-10 GB/day (Kibana).

The components of my SIEM consists of a manager, indexer, dashboard, and the agents it monitors. The manager acts like the brain of SIEM. It receives logs and events from agents, processes detection rules, and triggers active responses if something is found that matches a detection rule. The indexer stores all alerts, events, logs, and metadata in a searchable format. The dashboard is a web interface that connects to the indexer to display alerts (See Figure 1). In an enterprise environment, it is best to deploy each component on a separate machine to ensure performance, scalability, fault tolerance, and security. Encrypted communication using SSL between components is available and necessary.

Wazuh includes an Active Response framework that allows for automatic execution of predefined scripts in reaction to specific security events. These responses are triggered by matching rules within Wazuhs detection engine. This is helpful for preventing an attack like SSH brute force. If a machine tries to login to it via SSH and enters an incorrect password more than a few attempts, it's likely an unauthorized access attempt. In such cases, Wazuh can automatically trigger a response such as blocking the attacking IP address using a firewall rule effectively stopping the brute-force attack in real time without human intervention. Wazuh by itself comes with many built-in frameworks that are ready to be configured out of the box with things like file detector monitoring, rootkit detection, etc. But because Wazuh is open source you are easily able to build on top of what it comes with. An example of this would be YARA. YARA is another open-source tool used to identify and classify malware (or other suspicious files) based on patterns. A pattern can be identifiable byte patterns, strings, or file characteristics. Rules can be as simple or as complicated as you want, and a rule can be triggered when 1 or more patterns are detected. YARA was created in 2009 and since then cyber security specialists have created and shared thousands of rules that accurately detect many different types of malwares. YARA is multiplatform and easily integrated into Wazuh. YARA can be enhanced and built on top of with AI models for further inspection of detected malware. There are many different types of malwares out there and with the help of AI models like Chat-GPT, they can be fed the YARA rule that was triggered and sent to Chat-GPT for further explanation. To come full circle Chat-GPT can then send its detailed analysis back to the Wazuh manager to be displayed on the log dashboard.

Methodology

To implement the SIEM solution I created a virtual lab environment consisting of a Wazuh manager, dashboard, indexer, and a Linux agent. Because I only had 24GB of RAM and 16 cores at my disposal between my two machines the SIEM components ran on the same machine in a virtual environment using Virtual Box that had Ubuntu as the operating system. According to Wazuh documentation, Wazuh requires you have at least 8GB of ram and 4 Cores available to run it and that's exactly what I gave my SIEM. Leaving the rest of my resources for the agent and attacker. One of the agents acted as a victim, while another Linux machine running on Kali Linux simulated attacker behavior.

I ran three attack simulations: SSH brute force, NMAP port scanning, and malware delivery using ELF binaries. The SSH brute force detection was based on rule 5763 (Figure 2), which is part of the default Wazuh ruleset. I linked this rule to a custom active response script that triggers when more than six failed login attempts occur within a 60-second time frame. When Wazuh detects the event and matches it to rule 5763, it raises an alert and automatically executes a Bash script (Figure 3) that blocks the offending IP address after multiple failed login attempts.

To detect port scanning a custom rule was created (Figure 6) to detect signs of a port scan by matching log entries generated by the system firewall. The rule was configured to match patterns associated with Nmap scanning behavior. When Wazuh detects a log entry that matches the custom Nmap rule, it raises an alert and classifies the event as a potential port scanning attack.

For malware detection YARA was configured on the agent to scan a designated directory. A YARA rule (Figure 4) was used detect specific signatures inside known malware samples like Xbash. The “strings” within the rule are used to be compared to the elf file that is being scanned. If two of these strings are inside the elf file being analyzed match, then it triggers a match. The most important strings are s3 which check if the file targets a log directory commonly deleted by Xbash to cover its tracks and s4 which flags a specific API endpoint used by the malware for command and control (C2) or data exfiltration. If a match is found a bash script (Figure 5) sends the matched rule content to ChatGPT using the OpenAI API. The response from ChatGPT, which includes a human-readable explanation of the potential threat, is then injected into the Wazuh alert logs. This gives the victim of the attack to understand the nature of the threat immediately.

Results

The customized Wazuh setup successfully detected SSH brute force attempts in real time. During simulation, the attacker machine ran an automated login script using incorrect credentials. After 6 failed attempts, Wazuh matched the activity with rule 5763 and effectively blocked the attacker’s IP and added to the log dashboard. After Nmap scans were ran against the target system Wazuh matched the resulting firewall logs to the custom port scan detection rule and generated high-priority alerts. During malware testing I downloaded a version of the Xbash malware from https://wazuh-demo.s3-us-west-1.amazonaws.com/xbash and my YARA rule was triggered when added to the monitored directory. Alerts containing rule names and descriptions were created instantly. When passed through the ChatGPT integration, the returned analysis clearly described the purpose of the malware. These enriched logs proved valuable in not just identifying threats but understanding their behavior, significantly reducing investigation time.

Conclusion

This project demonstrates the effectiveness of combining open-source tools like Wazuh and YARA with the intelligence of AI models like ChatGPT to create a powerful, customizable SIEM. The integration of ChatGPT bridges the gap between raw alerts and contextual understanding. Since Wazuh is open source, organizations can tailor it to their specific needs without incurring the high costs associated with proprietary SIEM platforms. As cybersecurity threats continue to grow in both complexity and volume solutions like this offer an affordable and scalable path forward for threat detection and response.

Figure 1

[Agent] --(logs, FIM, YARA matches)--> [Manager] --(parsed alerts)--> [Indexer]

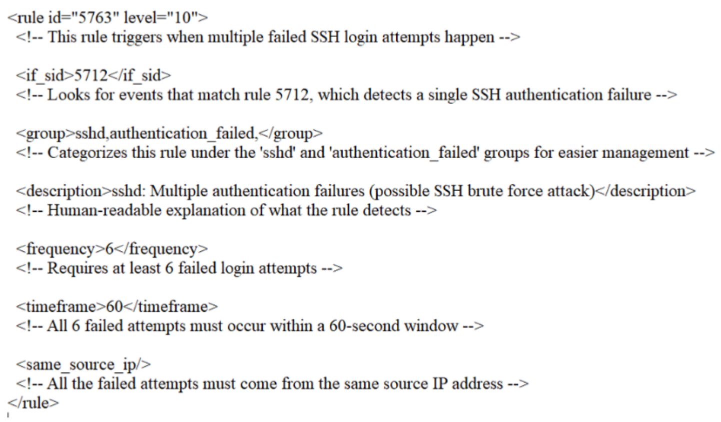
|

[Active Response: GPT + Block IP]

↓

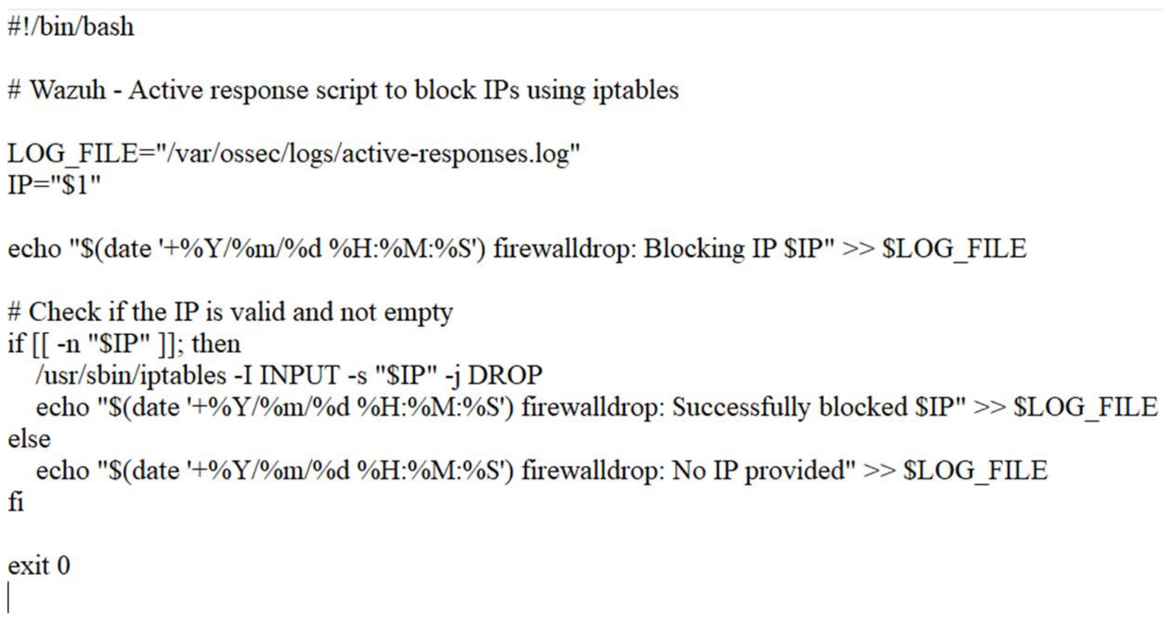
[Dashboard View]

Figure 2



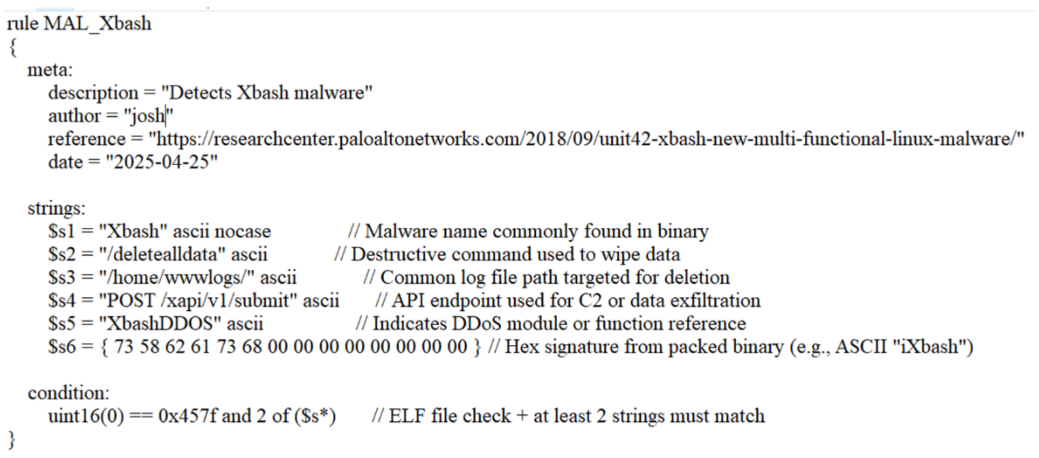
```
<rule id="5763" level="10">
  <!-- This rule triggers when multiple failed SSH login attempts happen -->

  <if_sid>5712</if_sid>
  <!-- Looks for events that match rule 5712, which detects a single SSH authentication failure -->

  <group>sshd,authentication_failed,</group>
  <!-- Categorizes this rule under the 'sshd' and 'authentication_failed' groups for easier management -->

  <description>sshd: Multiple authentication failures (possible SSH brute force attack)</description>
  <!-- Human-readable explanation of what the rule detects -->

  <frequency>6</frequency>
  <!-- Requires at least 6 failed login attempts -->

  <timeframe>60</timeframe>
  <!-- All 6 failed attempts must occur within a 60-second window -->

  <same_source_ip/>
  <!-- All the failed attempts must come from the same source IP address -->
</rule>
```

Figure 3



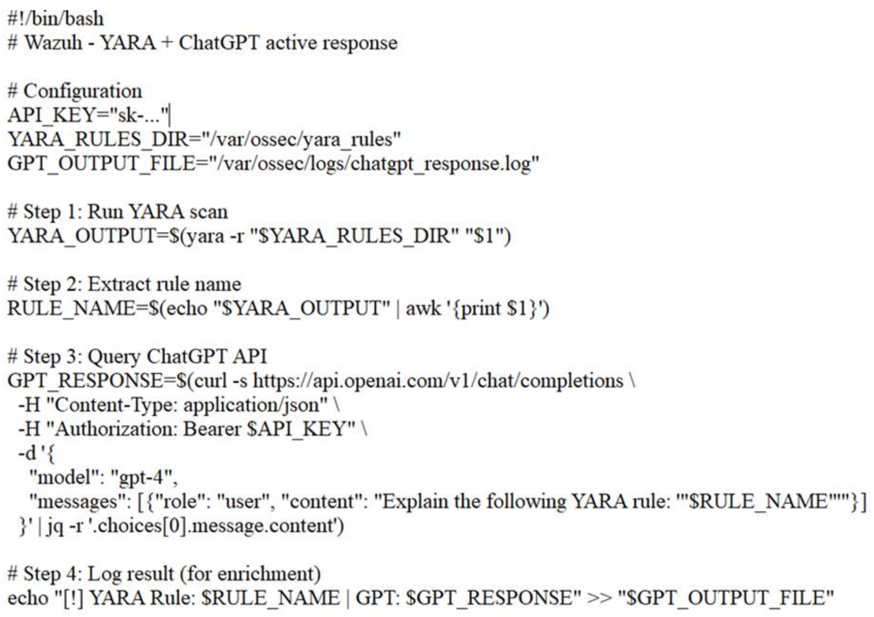
```
#!/bin/bash

# Wazuh - Active response script to block IPs using iptables

LOG_FILE="/var/ossec/logs/active-responses.log"
IP="$1"

echo "$(date '+%Y/%m/%d %H:%M:%S') firewalldrop: Blocking IP $IP" >> $LOG_FILE

# Check if the IP is valid and not empty
if [[ -n "$IP" ]]; then
  /usr/sbin/iptables -I INPUT -s "$IP" -j DROP
  echo "$(date '+%Y/%m/%d %H:%M:%S') firewalldrop: Successfully blocked $IP" >> $LOG_FILE
else
  echo "$(date '+%Y/%m/%d %H:%M:%S') firewalldrop: No IP provided" >> $LOG_FILE
fi

exit 0
```

Figure 4



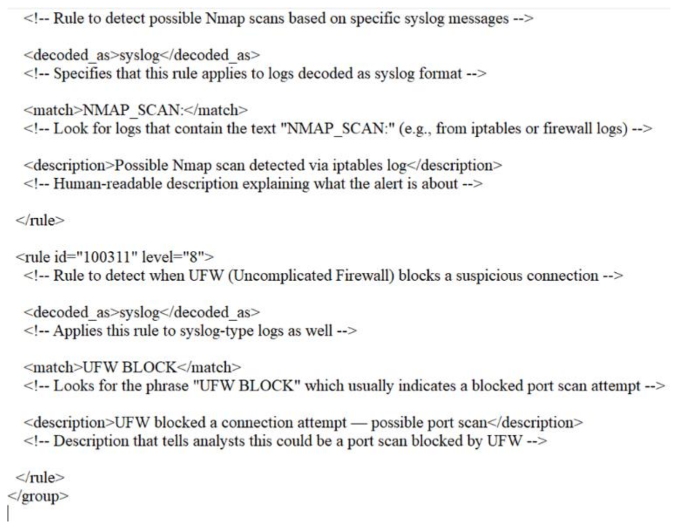
```
rule MAL_Xbash
{
   meta:
      description = "Detects Xbash malware"
      author = "josh"
      reference = "https://researchcenter.paloaltonetworks.com/2018/09/unit42-xbash-new-multi-functional-linux-malware/"
      date = "2025-04-25"

   strings:
      $s1 = "Xbash" ascii nocase              // Malware name commonly found in binary
      $s2 = "/deletealldata" ascii           // Destructive command used to wipe data
      $s3 = "/home/wwwlogs/" ascii            // Common log file path targeted for deletion
      $s4 = "POST /xapi/v1/submit" ascii       // API endpoint used for C2 or data exfiltration
      $s5 = "XbashDDOS" ascii                 // Indicates DDoS module or function reference
      $s6 = { 73 58 62 61 73 68 00 00 00 00 00 00 00 00 } // Hex signature from packed binary (e.g., ASCII "iXbash")

   condition:
      uint16(0) == 0x457f and 2 of ($s*)       // ELF file check + at least 2 strings must match
}
```

Figure 5



```
#!/bin/bash
# Wazuh - YARA + ChatGPT active response

# Configuration
API_KEY="sk-..."
YARA_RULES_DIR="/var/ossec/yara_rules"
GPT_OUTPUT_FILE="/var/ossec/logs/chatgpt_response.log"

# Step 1: Run YARA scan
YARA_OUTPUT=$(yara -r "$YARA_RULES_DIR" "$1")

# Step 2: Extract rule name
RULE_NAME=$(echo "$YARA_OUTPUT" | awk '{print $1}')

# Step 3: Query ChatGPT API
GPT_RESPONSE=$(curl -s https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $API_KEY" \
  -d '{
    "model": "gpt-4",
    "messages": [{"role": "user", "content": "Explain the following YARA rule: '"$RULE_NAME"'"}]
  }' | jq -r '.choices[0].message.content')

# Step 4: Log result (for enrichment)
echo "[!] YARA Rule: $RULE_NAME | GPT: $GPT_RESPONSE" >> "$GPT_OUTPUT_FILE"
```

Figure 6



```
  <!-- Rule to detect possible Nmap scans based on specific syslog messages -->

  <decoded_as>syslog</decoded_as>
  <!-- Specifies that this rule applies to logs decoded as syslog format -->

  <match>NMAP_SCAN:</match>
  <!-- Look for logs that contain the text "NMAP_SCAN:" (e.g., from iptables or firewall logs) -->

  <description>Possible Nmap scan detected via iptables log</description>
  <!-- Human-readable description explaining what the alert is about -->

 </rule>

 <rule id="100311" level="8">
  <!-- Rule to detect when UFW (Uncomplicated Firewall) blocks a suspicious connection -->

  <decoded_as>syslog</decoded_as>
  <!-- Applies this rule to syslog-type logs as well -->

  <match>UFW BLOCK</match>
  <!-- Looks for the phrase "UFW BLOCK" which usually indicates a blocked port scan attempt -->

  <description>UFW blocked a connection attempt — possible port scan</description>
  <!-- Description that tells analysts this could be a port scan blocked by UFW -->

 </rule>
</group>
```

Citations

Federal Bureau of Investigation. *2024 Internet Crime Report*. Internet Crime Complaint Center (IC3), 2025, https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf.

Uptrace Team. "Splunk Pricing: Complete Breakdown (2024)." Uptrace, 18 Jan. 2024, https://uptrace.dev/blog/splunk-pricing

Kibalna, Maryna. "Microsoft Sentinel: Pricing and Key Features in 2025." *UnderDefense*, 25 Feb. 2025, https://underdefense.com/industry-pricings/microsoft-sentinel-pricing/

Wazuh, Inc. Wazuh Documentation. 2025, https://documentation.wazuh.com.