SIEM Use Case: Detecting Administrative Activity with a Custom Elastic Rule

Ziad Mahmoud Ahmed Abdelgwad

Objective

This document outlines the end-to-end process for proactively detecting a specific administrative activity on a Windows host using an Elastic SIEM. The workflow involves three main phases:

Generating Test Data: Using a PowerShell script to create specific Windows Security Event Logs.

Creating a Detection Rule: Configuring a custom rule within the Kibana Security application to look for the generated events.

Verifying Alerts: Confirming that the rule successfully triggers and generates alerts based on the test data.

Phase 1: Generating Test Events

To test the detection capabilities of the SIEM, a PowerShell script named `Generate-Test-Logs.ps1` was utilized. This script is designed to programmatically perform a series of administrative actions on a local Windows machine, each corresponding to a specific, high-value Event ID.

The script checks for Administrator privileges before executing. It then proceeds through a series of test cases, pausing after each one to allow for verification in Kibana.

The activities generated by the script include:

Test Case 1: User Creation (Event ID 4720)

Test Case 2: Disable User (Event ID 4725)

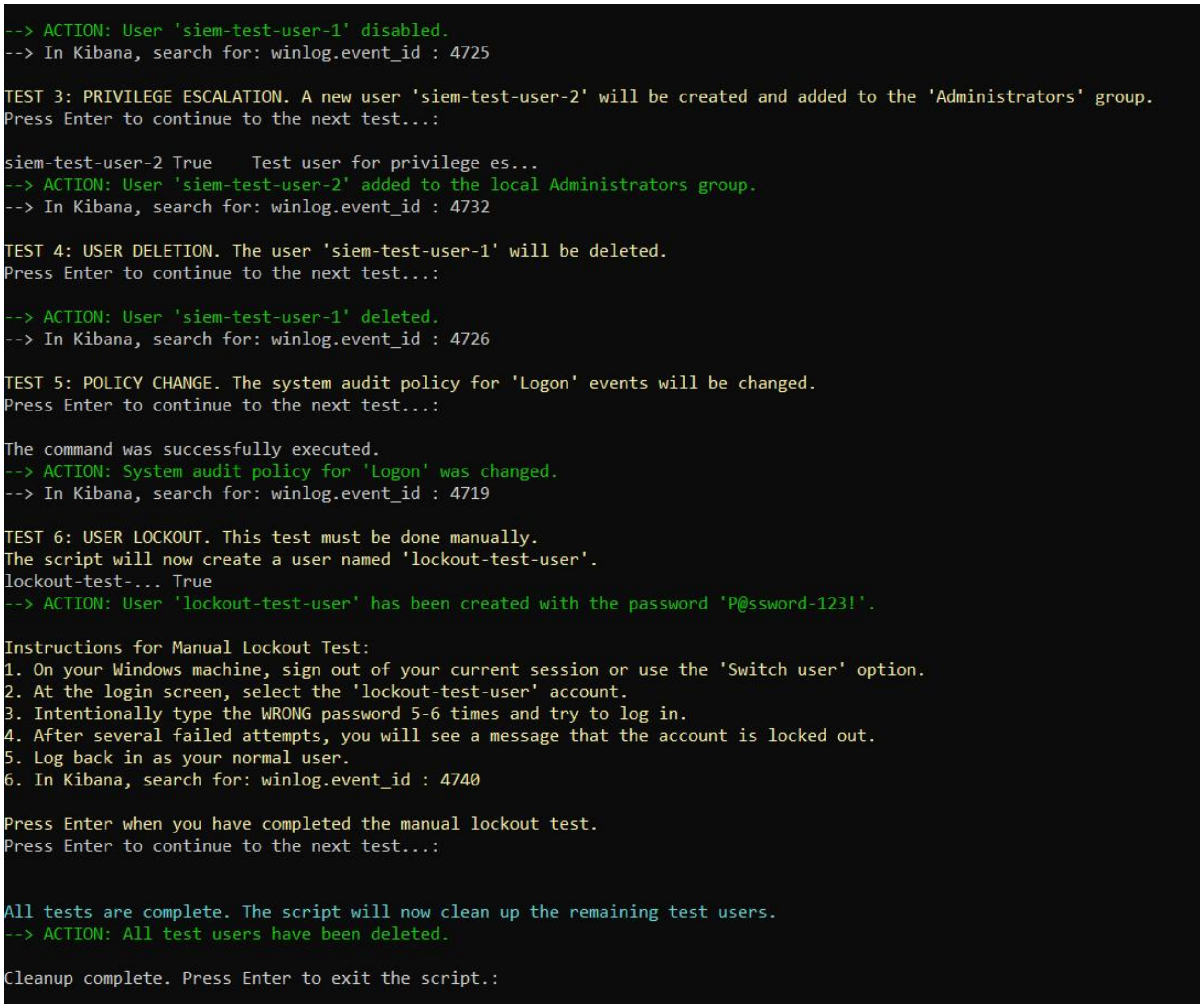
Test Case 3: User Privilege Escalation (Event ID 4732)

Test Case 4: User Deletion (Event ID 4726)

Test Case 5: Audit Policy Change (Event ID 4719)

Test Case 6: User Lockout (Event ID 4740), which requires manual user interaction to trigger.

The script was executed successfully, generating the intended event logs on the Windows host machine.



```
--> ACTION: User 'siem-test-user-1' disabled.
--> In Kibana, search for: winlog.event_id : 4725

TEST 3: PRIVILEGE ESCALATION. A new user 'siem-test-user-2' will be created and added to the 'Administrators' group.
Press Enter to continue to the next test...:

siem-test-user-2 True    Test user for privilege es...
--> ACTION: User 'siem-test-user-2' added to the local Administrators group.
--> In Kibana, search for: winlog.event_id : 4732

TEST 4: USER DELETION. The user 'siem-test-user-1' will be deleted.
Press Enter to continue to the next test...:

--> ACTION: User 'siem-test-user-1' deleted.
--> In Kibana, search for: winlog.event_id : 4726

TEST 5: POLICY CHANGE. The system audit policy for 'Logon' events will be changed.
Press Enter to continue to the next test...:

The command was successfully executed.
--> ACTION: System audit policy for 'Logon' was changed.
--> In Kibana, search for: winlog.event_id : 4719

TEST 6: USER LOCKOUT. This test must be done manually.
The script will now create a user named 'lockout-test-user'.
lockout-test-... True
--> ACTION: User 'lockout-test-user' has been created with the password 'P@ssword-123!'.

Instructions for Manual Lockout Test:
1. On your Windows machine, sign out of your current session or use the 'Switch user' option.
2. At the login screen, select the 'lockout-test-user' account.
3. Intentionally type the WRONG password 5-6 times and try to log in.
4. After several failed attempts, you will see a message that the account is locked out.
5. Log back in as your normal user.
6. In Kibana, search for: winlog.event_id : 4740

Press Enter when you have completed the manual lockout test.
Press Enter to continue to the next test...:


All tests are complete. The script will now clean up the remaining test users.
--> ACTION: All test users have been deleted.

Cleanup complete. Press Enter to exit the script.:
```

Phase 2: Creating a Custom Detection Rule

After generating the events, the next step is to create a rule in the SIEM to automate the detection of this activity in the future.

This process was initiated from the **Rules** page within the Kibana **Security** application by clicking the **"Create new rule"** button.

Figure 2: Navigating to the Rules page in the Kibana Security app to begin rule creation.

A "Custom query" rule was configured with specific parameters to detect the user creation events generated by the script. The key configuration details are as follows:

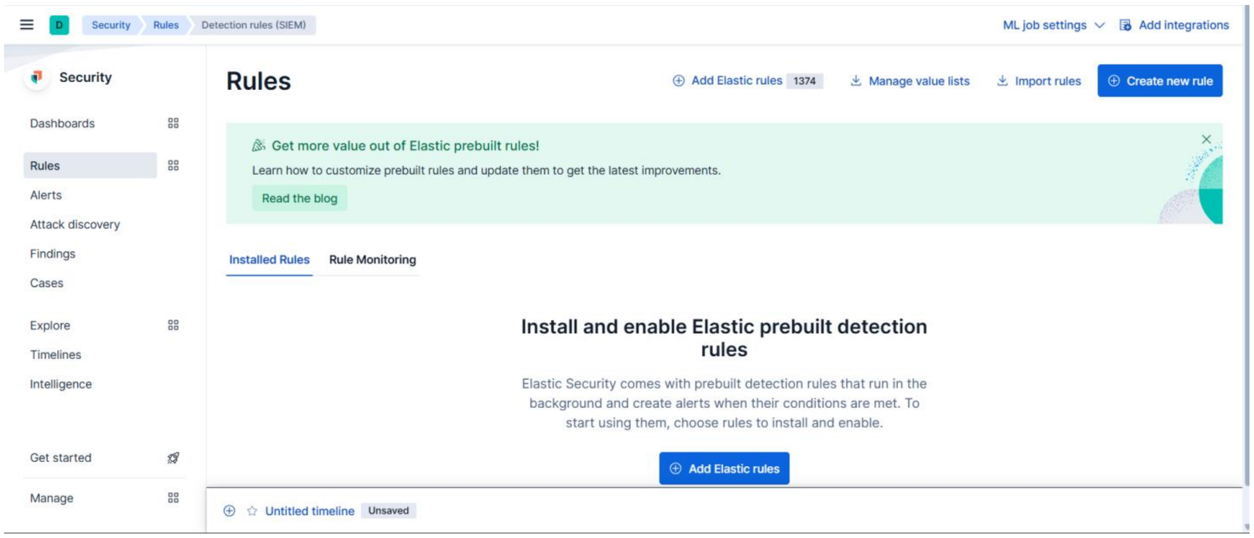
Name: New User Account Created

Description: This rule generates an alert when a new user account is created (Windows Event ID 4720) or an existing account is enabled (Event ID 4722).

Severity: Medium

Index Patterns: The rule was set to run against multiple index patterns, including winlogbeat-*, ensuring it would find the logs from the Windows host.

Custom Query: The core logic of the rule was defined with the KQL query winlog.event_id : (4720 or 4722).



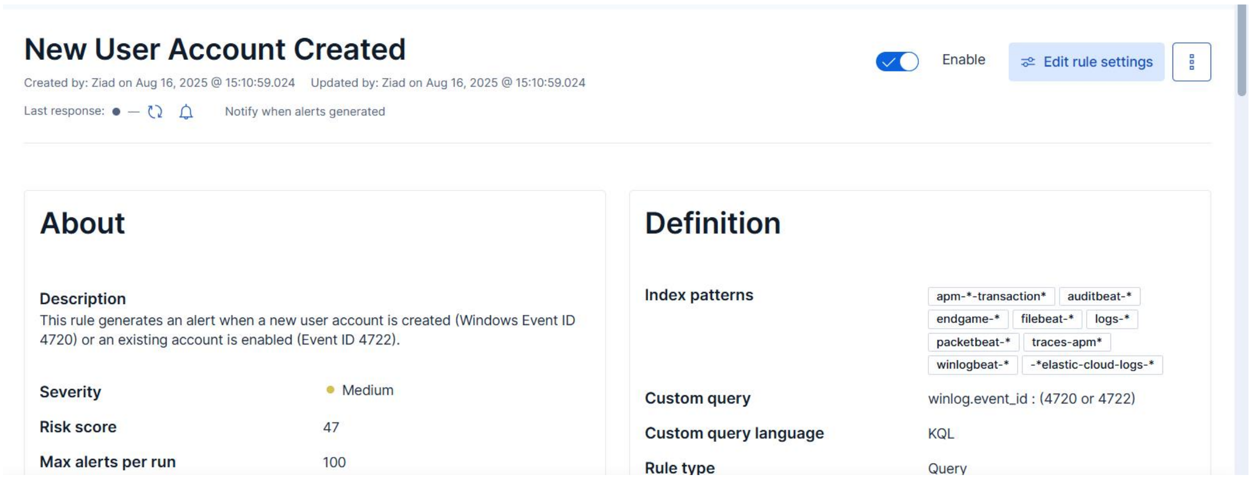
Rules

Get more value out of Elastic prebuilt rules!

Learn how to customize prebuilt rules and update them to get the latest improvements.

Install and enable Elastic prebuilt detection rules

Elastic Security comes with prebuilt detection rules that run in the background and create alerts when their conditions are met. To start using them, choose rules to install and enable.



New User Account Created

Created by: Ziad on Aug 16, 2025 @ 15:10:59.024   Updated by: Ziad on Aug 16, 2025 @ 15:10:59.024

Last response: ● —   Notify when alerts generated

About

Description
This rule generates an alert when a new user account is created (Windows Event ID 4720) or an existing account is enabled (Event ID 4722).

Severity	● Medium
Risk score	47
Max alerts per run	100

Definition

Index patterns	apm-*-transaction* auditbeat-* endgame-* filebeat-* logs-* packetbeat-* traces-apm* winlogbeat-* -*elastic-cloud-logs-*
Custom query	winlog.event_id : (4720 or 4722)
Custom query language	KQL
Rule type	Query

Phase 3: Verifying the Results

After the rule was created and activated, it successfully analyzed the incoming log data from the test script and began generating alerts.

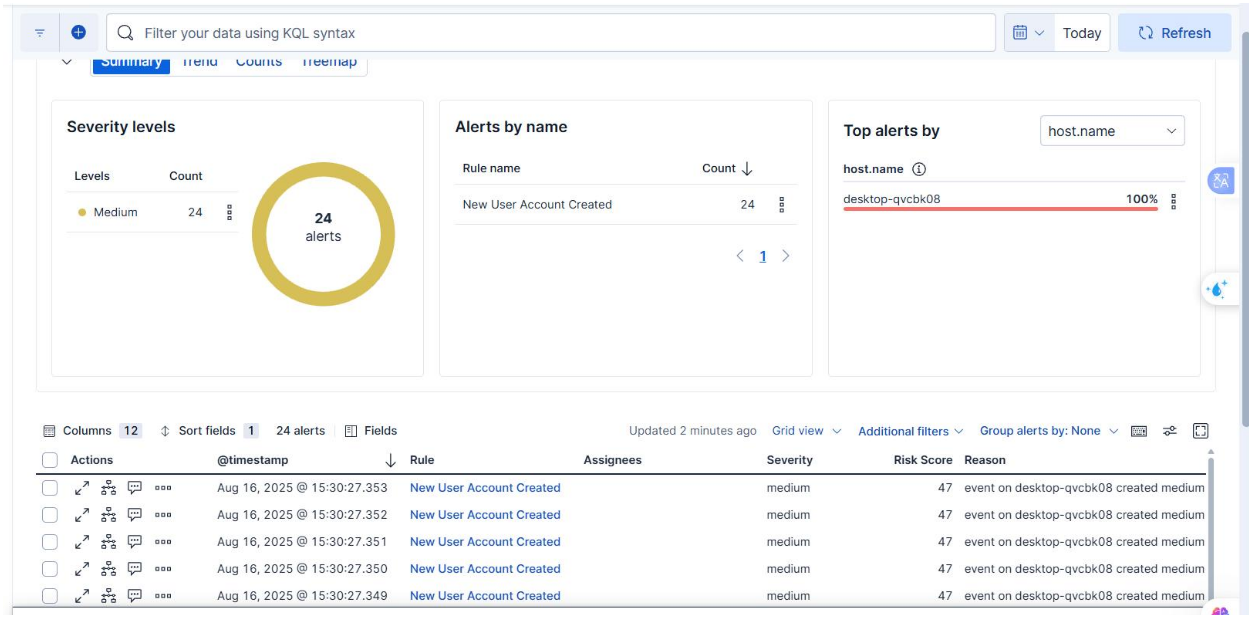
The final result, viewed on the Kibana Alerts dashboard, confirms the success of the entire process. The dashboard shows:

A total of **24 alerts** were generated with a "Medium" severity level.

The "Alerts by name" panel confirms that all 24 alerts originated from the **"New User Account Created"** rule.

The "Top alerts by" panel shows that 100% of these alerts came from the host named **"desktop-qvcbk08"**.

The table at the bottom lists the individual alerts, each with a timestamp, severity, and reason, confirming that the detection was successful and timely (e.g., an alert was generated on Aug 16, 2025 @ 15:30:27.353).


Filter your data using KQL syntax | Today | Refresh

Summary | Trend | Counts | Treemap

Severity levels

Levels	Count
● Medium	24

24 alerts

Alerts by name

Rule name	Count
New User Account Created	24

Top alerts by host.name

host.name	
desktop-qvcbk08	100%

Columns 12 | Sort fields 1 | 24 alerts | Fields | Updated 2 minutes ago | Grid view | Additional filters | Group alerts by: None

@timestamp	Rule	Assignees	Severity	Risk Score	Reason
Aug 16, 2025 @ 15:30:27.353	New User Account Created		medium	47	event on desktop-qvcbk08 created medium
Aug 16, 2025 @ 15:30:27.352	New User Account Created		medium	47	event on desktop-qvcbk08 created medium
Aug 16, 2025 @ 15:30:27.351	New User Account Created		medium	47	event on desktop-qvcbk08 created medium
Aug 16, 2025 @ 15:30:27.350	New User Account Created		medium	47	event on desktop-qvcbk08 created medium
Aug 16, 2025 @ 15:30:27.349	New User Account Created		medium	47	event on desktop-qvcbk08 created medium

Conclusion

The process was a complete success. Test events were programmatically generated on a Windows host using a PowerShell script. A custom detection rule was then configured in the Elastic SIEM, which successfully identified the specific activity and generated actionable alerts. This demonstrates a full, end-to-end workflow for identifying a security behavior of interest and implementing an automated monitoring and alerting capability to detect it in the future.