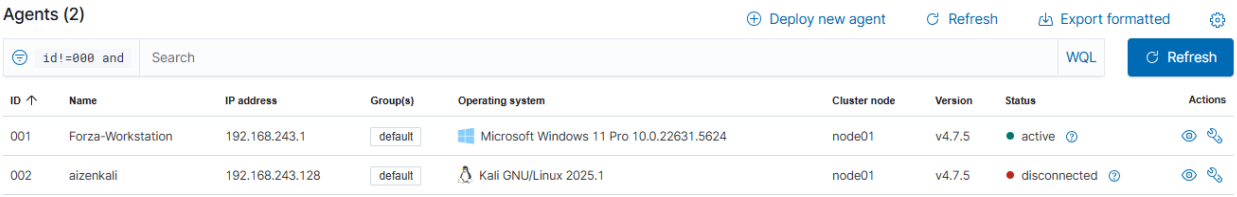
Wazuh SIEM Testing & Alert Validation

Introduction

In this project I decided to test the alert and detection capabilities of the wazuh system by simulating different scenarios that should trigger alerts and generate event logs.

At this point, wazuh is successfully set up and is monitoring two agents:

- Windows computer
- Kali Linux Virtual machine



The tests in this project were conducted both the Kali and windows machines.

Test 1: Event log creation

This test involved simulation an application error to check whether the wazuh system would detect and log the event. This test involved using the command:

eventcreate /ID 0001 /L APPLICATION /T ERROR /SO wazuhtest /D “wazuh application error test”

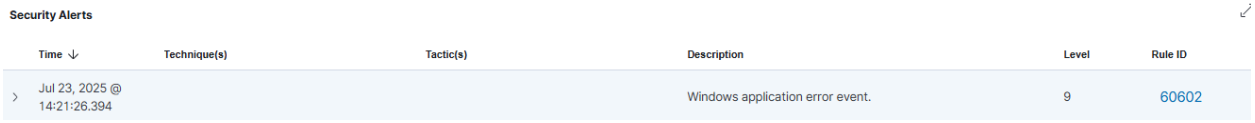
- Eventcreate – the command name
- /ID – any number between 1-1000
- /L – Log name
- /T – type of event
- /SO – source of event (can be any string)
- /D – description

```
C:\Windows\System32> eventcreate /ID 0001 /L APPLICATION /T ERROR /SO wazuhtest /D "wazuh application error test"

SUCCESS: An event of type 'ERROR' was created in the 'APPLICATION' log with 'wazuhtest' as the source.

C:\Windows\System32>
```

On the wazuh dashboard, the log is created successfully hence validating the error logging capability.



Test 2: Failed authentication attempts

This test involved checking whether wazuh would detect and log any failed authentication attempts made on a device. This is very important in order to detect unauthorized access attempts or brute-force attempts. This was an easy test, I turned off the screen and intentionally inserted wrong passwords. The wazuh system was able to detect this successfully and create event logs.

Total	Level 12 or above alerts	Authentication failure	Authentication success
2644	23	2	102

Test 3: starting and stopping of services

This test involved simulating the starting and stopping of services to find out if wazuh will detect and create event logs. The service used in this test was the Background download service and the commands used are:

Net start BITS – this is to start the service.

Net stop BITS – this is to stop the service.

```
C:\Windows\System32>net start bits
The Background Intelligent Transfer Service service is starting..
The Background Intelligent Transfer Service service was started successfully.


C:\Windows\System32>net stop bits
The Background Intelligent Transfer Service service is stopping..
The Background Intelligent Transfer Service service was stopped successfully.
```

The event was successfully detected and recorded.

Jul 23, 2025 @ 16:02:39.674		Service startup type was changed	3	61104

Test 4: File manipulation

This test involved testing whether wazuh would detect and log any actions performed on files in a specific folder. In this test, I used the desktop folder. This required additional configuration as wazuh does not monitor files by default. The following line was added in the **ossec.conf** file:

<directories realtime="yes" report_changes="yes" check_all="yes">"C:\Users\Mike Some\Desktop"</directories>

This line enabled wazuh to monitor file actions that occur in the desktop folder. With that set up, all I had to do was to create a file on the desktop folder and modify its contents to find out what wazuh can detect.

Time	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
Jul 25, 2025 @ 13:27:13.213	c:\users\mike some\desktop\wazuh test1.txt	added	File added to the system.	5	554
Jul 25, 2025 @ 13:24:38.827	c:\users\mike some\desktop\new text document.txt	modified	Integrity checksum changed.	7	550

This test was successful as wazuh was able to detect these changes and create logs.

Test 5: malware test

This test is to check the malware detection abilities of wazuh. This was done by using a benign file that acts as a malware called EICAR. EICAR is an anti-malware testfile to test the response of computer anti-virus software. This test was conducted on the Kali Linux machine as the windows machine already has Defender to deal with such threats.

Additional configuration

This test required some additional configuration as wazuh can only detect file changes but an antivirus software would be required to actually eliminate the threat. I decided to use VIrustotal as it was widely used by other wazuh users.

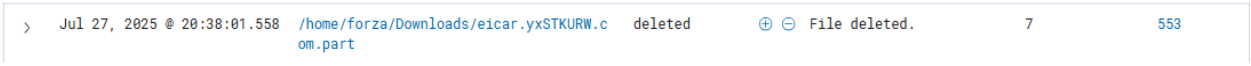
Configuration steps

1. Firstly, I needed a free virustotal account. This was simple as all I did was to register and that was it.
2. After creating an account, an API key is generated that is required in the configuration of the wazuh manager to use virustotal. The free versio is limited to 4 lookups per minute.
3. Having the API key, I opened the ossec.conf file for the wazuh manager and added a new intergration block containing the API key.

```
406  <integration>
407    <name>virustotal</name>
408    <api_key>15f6502cf4accc07409d5c7a4f3ad776ea46553d6b72b4d746c3acecb634e6e5</api_key>
409    <group>syscheck</group>
410  </integration>
```

4. The last step is to restart the manager either via the GUI or by using the command: sudo systemctl restart wazuh-manager

The EICAR file was detected and deleted immediately I downloaded it. This shows that virusTotal was working properly.



This test was conducted on the Kali Linux machine as the windows machine already has Defender to deal with such threats.

Conclusion

This project was an extremely nice learning experience as I was able to discover more about Wazuh and SIEMs in general. I encountered a few problems relating to the configuration as some changes required an hour or more to be relected. The biggest problem, however, was that this SIEM is hosted on an Ubuntu server installed on a virtual machine on my computer. This VM requires 4 GB RAM minimum hence it was very taxing on my computer. Additionally, the Kali Linux machine was also on a VM taking up an additional 2 GB RAM. This caused a significant RAM consumption especially when both VMs were powered on simultaneously.

Future tweaks

To solve the performance issues related to RAM usage, the SIEM should be hosted on the cloud as well as other VMs. This is a costly endeavor as cloud services are not free but it would be the best option. This would also solve the issue of having to power off the SIEM server when the host machine

is also powered off. This would ensure the SIEM runs continuously regardless of the status of the host machine.