Set Up

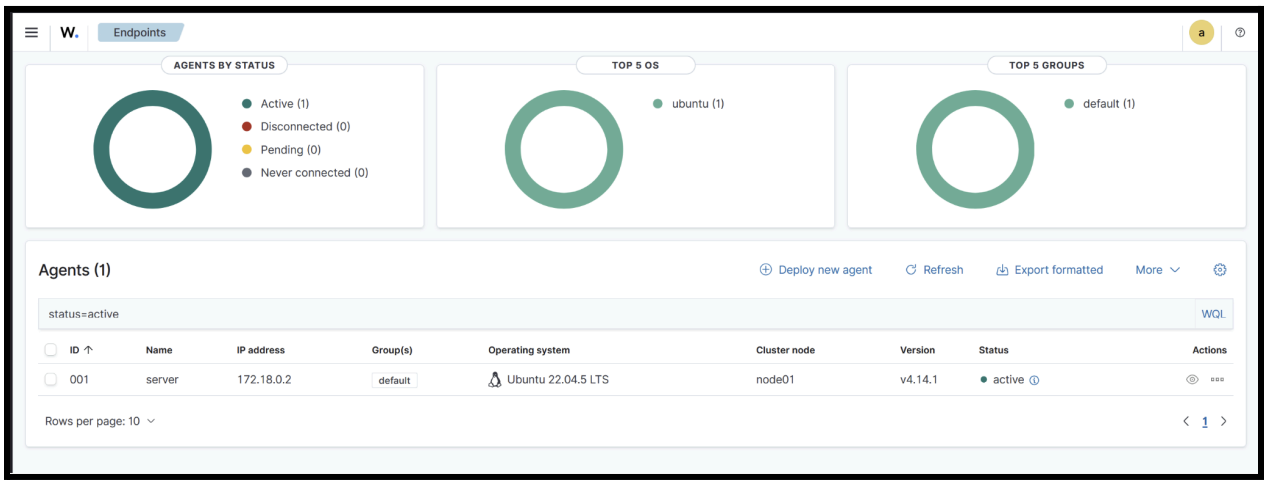
1. So, we had to set up Wazuh. I downloaded it using the guide linked in the guide: https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-linux.html
2. One thing that I had to check was my Linux distro and version. This is because older distros downloaded differently according the guide
    a. cat /etc/os-release
    b.
```
root@server:/opt/node_server# cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.5 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.5 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
root@server:/opt/node_server#
```
3. After running the command above, I see that the server is running Ubuntu 22 so we should be good to run the up to date commands. The command below are how I got the most up to date packages for wazuh by adding the Wazuh repository
    a. docker exec -it server /bin/bash
    b. apt update
    c. apt-get install gnupg apt-transport-https
    d. curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644 /usr/share/keyrings/wazuh.gpg
    e. echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list
    f. apt-get update
4. Now, I have to actually install wazuh and they provided a command for that
    a. WAZUH_MANAGER="**wazuh.manager**" apt-get install wazuh-agent
    b. To find the hostname for the manager, I looked at the Docker Compose

i.

```
wazuh.manager:
  image: wazuh/wazuh-manager:4.14.1
  hostname: wazuh.manager
  restart: always
  networks:
    - my_network
  ulimits:
    memlock:
      soft: -1
      hard: -1
    nofile:
      soft: 655360
      hard: 655360
  ports:
    - "1514:1514"
    - "1515:1515"
    - "514:514/udp"
    - "55000:55000"
  environment:
    - INDEXER_URL=https://wazuh.indexer:9200
    - INDEXER_USERNAME=admin
```

5. And then to start the Wazuh agent, the guide showed us how to do that using systemctl but it didn't work, as hinted by the CCDC training material. So I did some research, and I found out how to start is using the "service" command just like I did with SSH. service is an older tool.
    a. update-rc.d wazuh-agent defaults 95 10
    b. service wazuh-agent start
6. Ok. And then around here, is when my Wazuh Manager Container continued to restart every like 10 seconds, which made the dashboard unusable since the API was going down every 10 seconds. I had zero clue what was going on, and tried so many different things. I tried re-downloading everything, giving more resources to WSL, giving more resource to each container by editing the docker compose file, and so much more. I tried making new SSL certificates as the instructions hints, but that didn't work.
7. After **2 hours** of continuous debugging and no progress, I decided to try and delete all the old certificates. I deleted all of them, and tried making certificates fresh.
    a. docker-compose -f generate-indexer-certs.yml run --rm generator
8. And then it worked!! Well, the API worked. Now I had to connect it to the agent. Since it said that the agent was connected, even though the console made it seem like the Agent was deployed
9. So, after debugging that problem for a while, my best guess was that possibly the value used in the WAZAH_MANAGER value was wrong. As a reminder, I did WAZUH_MANAGER="**wazuh.manager**". So after some digging online, I found that we should find the appropriate WAZUH_MANAGER value by using this command:
    a. grep "<address>" /var/ossec/etc/ossec.conf
    b.

```
root@server:/# grep "<address>" /var/ossec/etc/ossec.conf
        <address>wazuh-manager</address>
```

10. And so as you can see, the hostname wasn't wazuh.manager but rather wazuh-manager! So, I changed that line in the address field and restarted the agent
    a. service wazuh-agent restart
11. 

    a. My agent is finally up!

```
bash-5.2# cat /etc/os-release
NAME="Amazon Linux"
VERSION="2023"
ID="amzn"
ID_LIKE="fedora"
VERSION_ID="2023"
PLATFORM_ID="platform:al2023"
PRETTY_NAME="Amazon Linux 2023.9.20251110"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2023"
HOME_URL="https://aws.amazon.com/linux/amazon-linux-2023/"
DOCUMENTATION_URL="https://docs.aws.amazon.com/linux/"
SUPPORT_URL="https://aws.amazon.com/premiumsupport/"
BUG_REPORT_URL="https://github.com/amazonlinux/amazon-linux-2023"
VENDOR_NAME="AWS"
VENDOR_URL="https://aws.amazon.com/"
SUPPORT_END="2029-06-30"
bash-5.2#
```

Implementing Rules

12. Now it's time to do the extra credit, and implement some rules. This one actually took a while, since I was having problems actually triggering the rules. And then I did some research, and there's a custom tool that they have to simulate whether an event was triggered based on logs.

13. So, the rule I created was to see if the useradd command used. This is because as a penetration tester, I know one common strategy is persistence. This means after getting initial access (ex. Reverse shell), you will want to create a permanent back door, and one method is to create a new user so you can continuously log back instead of having re-do the exploit again to get a reverse shell.
14. So to create a new rule, I had to log in to this:
    a. docker exec -it docker_deets-wazuh.manager-1 /bin/bash
    b. nano /var/ossec/etc/rules/local_rules.xml
    i.
```
</group>

<group name="my_custom_rules,">
  <rule id="100013" level="12">
    <program_name>useradd</program_name>
    <description>User Creation Detected: useradd command executed!</description>
  </rule>

</group>
```
15. And then to test it out, you have to run this custom script (/var/ossec/bin/wazuh-logtest) and it will ask for one line as input and you put in an example log to see if it would trigger an event and mine does:
    a.
```
bash-5.2# /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.14.1
Type one log per line

Nov 27 10:00:00 server useradd[1234]: new user: name=hacker_steve

**Phase 1: Completed pre-decoding.
        full event: 'Nov 27 10:00:00 server useradd[1234]: new user: name=hacker_steve'
        timestamp: 'Nov 27 10:00:00'
        hostname: 'server'
        program_name: 'useradd'

**Phase 2: Completed decoding.
        name: 'useradd'
        parent: 'useradd'

**Phase 3: Completed filtering (rules).
        id: '100013'
        level: '12'
        description: 'User Creation Detected: useradd command executed!'
        groups: '['my_custom_rules']'
        firedtimes: '1'
        mail: 'True'
**Alert to be generated.
```
16. And as you can see from the bottom, an event was triggered!
17. The resource I used was this:
    a. https://documentation.wazuh.com/current/user-manual/ruleset/rules/custom.html#custom-rules
    b. https://documentation.wazuh.com/current/user-manual/ruleset/decoders/custom.html

Note: if I want to create a new rule folder do this:

- # 1. Give ownership to the Wazuh user
- chown wazuh:wazuh /var/ossec/etc/rules/**competition_rules.xml**

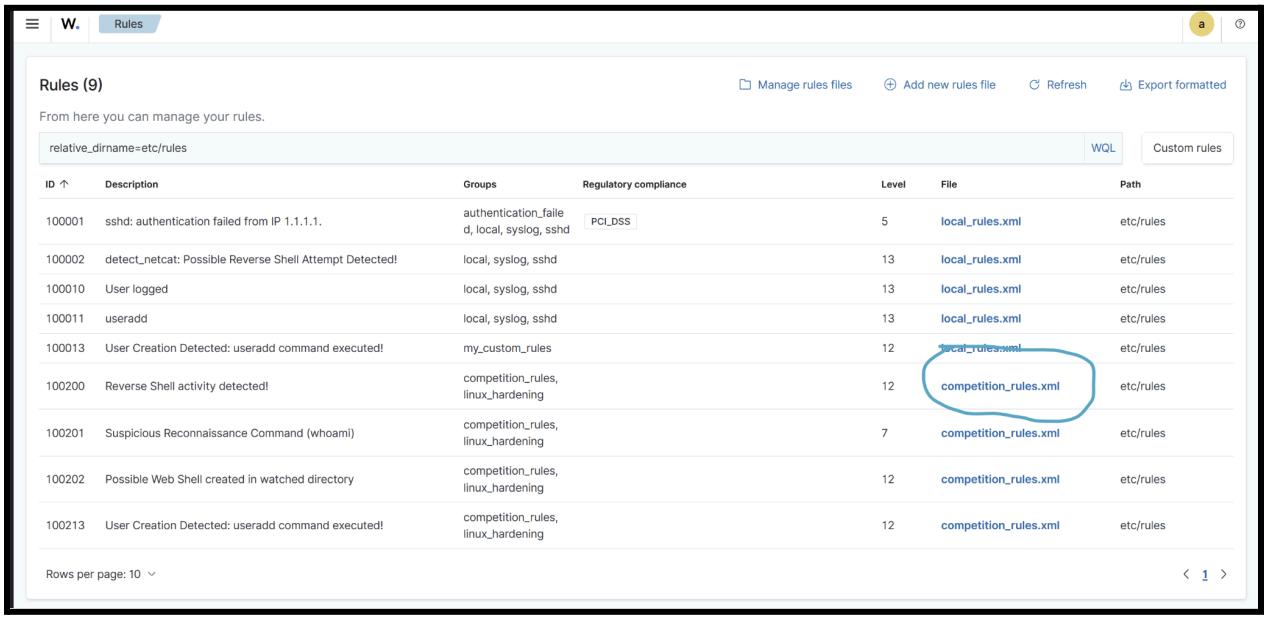
- # 2. Give read permissions
- chmod 660 /var/ossec/etc/rules/**competition_rules.xml**

Example rule set:

```
<group name="linux, competition,">

  <rule id="100100" level="12">
    <decoded_as>auditd</decoded_as>
    <match>nc -e|nc.traditional -e|ncat -e|bash -i >&|python -c 'import socket|socat exec</match>
    <description>Reverse Shell activity detected!</description>
  </rule>

  <rule id="100101" level="7">
    <decoded_as>auditd</decoded_as>
    <match>exe="/usr/bin/whoami"|exe="/usr/bin/id"</match>
    <description>Suspicious Reconnaissance Command (whoami/id)</description>
  </rule>

  <rule id="100102" level="12">
    <if_group>syscheck</if_group>
    <match>.php|.jsp|.phtml|.php5</match>
    <description>Possible Web Shell created in watched directory</description>
  </rule>

  <rule id="100103" level="10">
    <decoded_as>auditd</decoded_as>
    <match>rm .bash_history|ln -sf /dev/null .bash_history|truncate -s 0 .bash_history</match>
    <description>User attempted to clear command history</description>
  </rule>

  <rule id="100104" level="10">
    <if_group>syscheck</if_group>
    <match>/etc/cron</match>
    <description>Cron job configuration modified (Persistence risk)</description>
  </rule>
```

</group>

How to test for rules better



- Click on the file name on the right

- Click on "Ruleset Test" on the top right

- Then ask gemini to generate some examples of logs