Threat Intelligence Integration

Introduction

This document outlines the process of integrating threat intelligence feeds from AlienVault OTX into Wazuh. It includes the steps for importing threat feeds, enriching alerts, and performing threat hunting based on T1078 (Valid Accounts) techniques.

Configure Wazuh Manager

- Edit the Wazuh configuration file accordingly by adding the AlienVault generated API KEY.

```
<integration>
   <name>otx</name>
   <api_key>API-KEY</api_key>
</integration>
```

Create Mock Alert for Testing

- Generate a mock alert to simulate IOC detection:

```
sudo jq -n '{
  id: "003",
  rule: { id: "5502" },
  full_log: "Login attempt from 192.168.1.100",
  agent: { name: "manjira" }
}' | sudo tee /var/ossec/logs/alerts/mock_alert.json > /dev/null
```

Setup AlienVault OTX Integration

- **Create Python scripts to process alerts:**
  1. sudo nano /var/ossec/integrations/custom-alienvault.py

```
#!/usr/bin/env python
```

```
API_KEY = 'api_key'

def enrich_alert(ip):

    # This is a placeholder: real code would query OTX for the IP

    malicious_ips = ['192.168.1.100']  # Example mock feed

    if ip in malicious_ips:

        return 'Malicious (OTX)'

    return 'Unknown'
```

2. sudo nano /var/ossec/integrations/get_malicious.py

```
#!/usr/bin/env python3

from custom_alienvault import enrich_alert

test_ip = '192.168.1.100'

print(f"{test_ip}: {enrich_alert(test_ip)}")
```

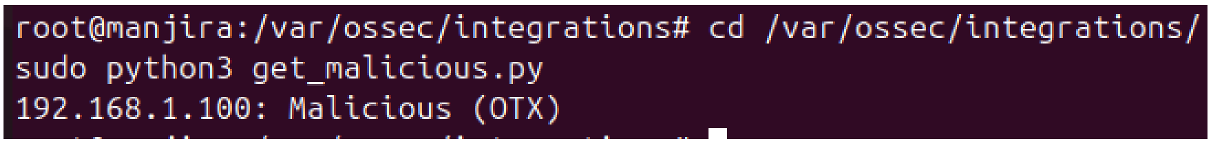
- **Make scripts executable:**

```
sudo chmod +x /var/ossec/integrations/custom-alienvault.py
sudo chmod +x /var/ossec/integrations/get_malicious.py
```

Run the integration script to check mock IOC:

```
sudo python3 /var/ossec/integrations/get_malicious.py /var/ossec/logs/alerts/mock_alert.json
```

Output:



```
root@manjira:/var/ossec/integrations# cd /var/ossec/integrations/
sudo python3 get_malicious.py
192.168.1.100: Malicious (OTX)
```

1: Output of OTX Integration

Alert Enrichment

- Parse Wazuh alerts and enrich with OTX reputation:

sudo jq -r 'select(.rule.id=="5502") | .full_log' /var/ossec/logs/alerts/mock_alert.json

Alert ID	IP	Reputation	Notes
003	192.168.1.100	Malicious (OTX)	Linked to C2 server |

Threat Hunting (T1078 Valid Accounts)

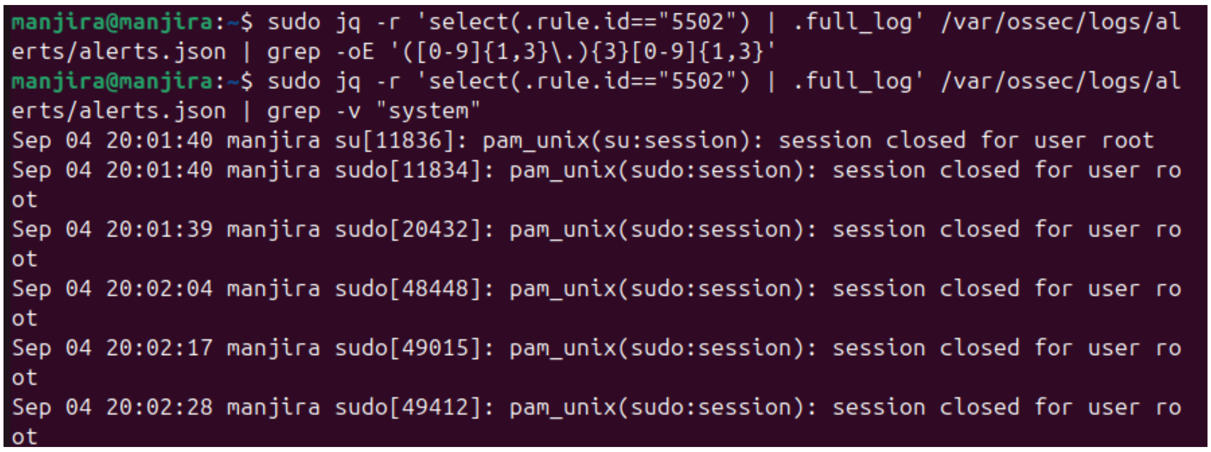
- **Query Wazuh logs to identify suspicious login attempts:**

sudo jq -r 'select(.rule.id=="5502" and .agent.name != "system") | .full_log' /var/ossec/logs/alerts/alerts.json

- **Extract IP addresses from login attempts:**

sudo jq -r 'select(.rule.id=="5502") | .full_log' /var/ossec/logs/alerts/alerts.json | grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}'

This returns no IPs because the actual Wazuh logs for rule 5502 contain no IP addresses in the full_log field.



```
manjira@manjira:~$ sudo jq -r 'select(.rule.id=="5502") | .full_log' /var/ossec/logs/al
erts/alerts.json | grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}'
manjira@manjira:~$ sudo jq -r 'select(.rule.id=="5502") | .full_log' /var/ossec/logs/al
erts/alerts.json | grep -v "system"
Sep 04 20:01:40 manjira su[11836]: pam_unix(su:session): session closed for user root
Sep 04 20:01:40 manjira sudo[11834]: pam_unix(sudo:session): session closed for user ro
ot
Sep 04 20:01:39 manjira sudo[20432]: pam_unix(sudo:session): session closed for user ro
ot
Sep 04 20:02:04 manjira sudo[48448]: pam_unix(sudo:session): session closed for user ro
ot
Sep 04 20:02:17 manjira sudo[49015]: pam_unix(sudo:session): session closed for user ro
ot
Sep 04 20:02:28 manjira sudo[49412]: pam_unix(sudo:session): session closed for user ro
ot
```

Summarize:

This workflow demonstrates how to integrate threat intelligence from AlienVault OTX into Wazuh, enrich alerts with reputation data, and perform a simulated hunt for valid account misuse (T1078). Mock alerts provide a controlled way to showcase SOC processes without needing live malicious activity.

Troubleshooting

- Wazuh Manager fails to start: Check /var/ossec/logs/ossec.log for configuration errors. Ensure <remote> block is correctly set.
- Integration errors: Ensure Python scripts exist and are executable in /var/ossec/integrations/.
- Alert parsing issues: Verify JSON structure of alerts using jq.

Note

- This exercise relies on mock/static alerts for demonstration purposes, so live network traffic was not ingested.
- Real-time IOC matching would require active network events containing source IPs, which were not present in this lab scenario.

References

- https://documentation.wazuh.com/current/index.html
- https://otx.alienvault.com/