TDX ARENA

Jordan Knill
Final Assessment Report Submission

Case: Pigs Rules

1/19/2026

Executive Summary

This investigation was initiated to identify and alert on malicious network activity within a cloud environment. Using the Snort Intrusion Detection System (IDS), we monitored incoming traffic on the eth0 interface to detect potential threats that were bypassing standard security layers.
The key finding was the identification of a large-scale TCP port scan originating from the internal IP 172.29.0.1. By developing a custom Snort signature, we were able to successfully categorize this traffic as "Attempted Information Leak" and visualize the event in the Snorby dashboard. The outcome of the investigation was the successful capture of a hidden flag string, proving the detection capability was operational.

Findings and Analysis

Finding	Finding Details	Description
IP Address	172.29.0.1	The source IP address identified as the primary attacker. This host initiated thousands of connection attempts to the target.
Attack	Port scan	The adversary utilized a high-velocity port scan to map open services on the victim machine (172.29.0.3).hidden inside the internal structure of a .docx document

Finding	Finding Details	Description
Packet Attributes	TCP SYN Flags	The reconnaissance traffic consisted primarily of SYN packets directed at high-numbered ports to identify listening services.
Hash	1fdcf70d937c1d1796a53fb4fdb9e79	The unique MD5-style string discovered within the system's "Sensor" metadata, serving as the validation flag for the challenge.

Methodology

Tools and Technologies Used

The following tools were utilized to conduct the investigation.

- **tcpdump:** A command-line packet analyzer used to capture and inspect live network traffic. I used this to isolate the attacker's IP and verify the packet flags during the reconnaissance phase.
- **Snort:** An open-source network IDS. I used Snort to apply custom rules to the live traffic stream and generate automated alerts for malicious patterns.
- **Snorby:** A web-based front-end for Snort logs. I used Snorby to visualize the alerts, categorize their severity, and extract the final challenge flag.

Investigation Process

The investigation was conducted as follows.

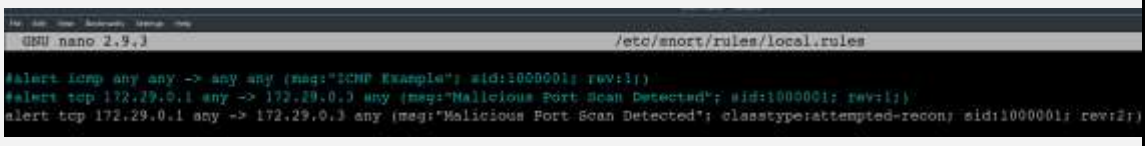
1. **Initial Traffic Capture:** I began by identifying the active ports and then running tcpdump -i eth0 -A not port 3389 to filter out management noise and observe raw traffic. This revealed a massive volume of SYN packets from 172.29.0.1.

```
snort@snort:~$ nano /etc/snort/rules/local.rules
snort@snort:~$ ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.33  netmask 255.255.0.0  broadcast 1
72.17.255.255
        ether 02:42:ac:11:00:21  txqueuelen 0  (Ethernet)
        RX packets 12902  bytes 13114297 (13.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11522  bytes 19098815 (19.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  coll
isions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 539  bytes 196874 (196.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 539  bytes 196874 (196.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  coll
isions 0

snort@snort:~$
```

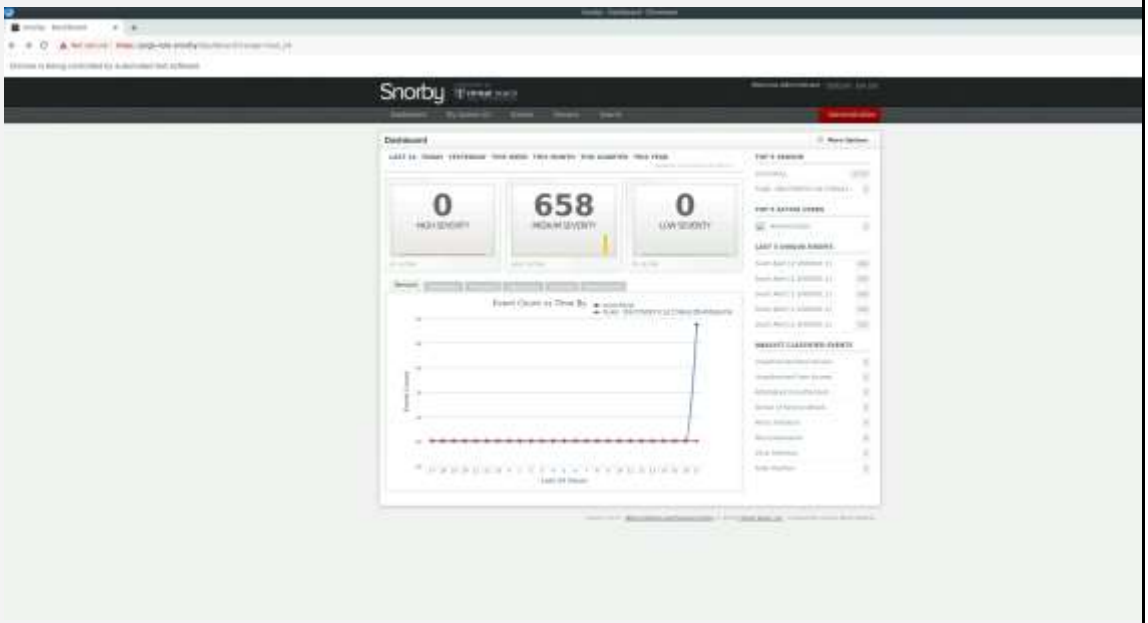
2. **Rule Development:** I created a custom local rule in /etc/snort/rules/local.rules: alert tcp 172.29.0.1 any -> 172.29.0.3 any (msg:"Malicious Port Scan Detected"; classtype:attempted-recon; sid:1000001; rev:2;). This allowed Snort to specifically track the attacker.



3. **IDS Implementation:** I launched Snort with the configuration file linked to the database: sudo snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0.

```
snort@snort:~$ nano /etc/snort/rules/local.rules
snort@snort:~$ sudo snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
^C*** Caught Int-Signal
snort@snort:~$
```

4. **Data Visualization:** I monitored the Snorby dashboard to ensure the alerts were being correctly logged into the database.



5. **Flag Extraction:** After the dashboard populated, I inspected the sensor metadata and unique events to recover the MD5 flag string

Recommendations

Based on the findings, I am proposing the following recommendations to mitigate the identified risks, secure the systems, and prevent future incidents.

1. **Implement IP Shunning:** Automatically block the IP 172.29.0.1 at the firewall level after a certain number of failed connection attempts is reached.
2. **Harden Security Groups:** Close all unused high-numbered ports on the victim machine (172.29.0.3) to reduce the attack surface.
3. **Enable Persistent Monitoring:** Ensure Barnyard2 and Snort are configured as system services to provide 24/7 visibility into network reconnaissance attempts. Barnyard2 is a dedicated "spooler" that takes the raw binary log files generated by Snort and translates logs into a format that a database (like the one Snorby uses) can understand.