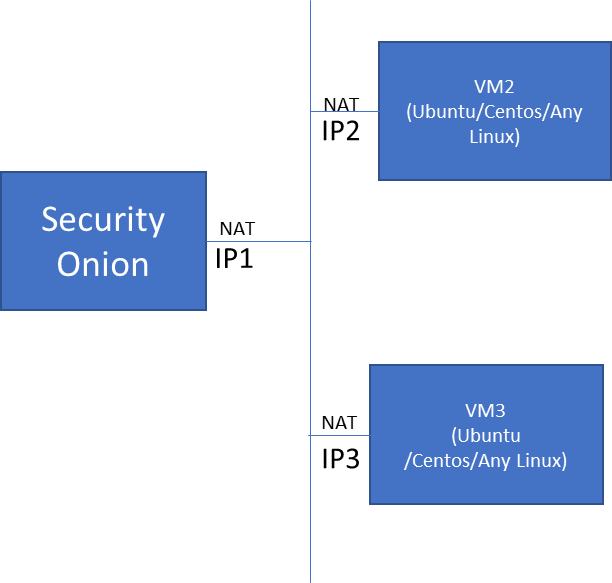
**Lab Introduction:**

An **intrusion detection system** (**IDS**) is a device or software application that monitors a network or systems for malicious activity or policy violations. Any malicious activity or violation is typically reported either to an administrator or collected centrally using a security information and event management **(SIEM)** system. A SIEM system combines outputs from multiple sources, and uses alarm filtering techniques to distinguish malicious activity from false alarms. IDS types range in scope from single computers to large networks. The most common classifications are **network intrusion detection systems** (**NIDS**) and **host-based intrusion detection systems** (**HIDS**). A system that monitors important operating system files is an example of an HIDS, while a system that analyzes incoming network traffic is an example of a NIDS.

It is also possible to classify IDS by detection approach: the most well-known variants are **Signature-based detection** (recognizing bad patterns, such as malware); and **Anomaly-based detection** (detecting deviations from a model of "good" traffic, which often relies on machine learning), another is **Reputationbased detection** (recognizing the potential threat according to the reputation scores).

|  |  |
| --- | --- |
| **Suricata** is an open-source intrusion detection and prevention system (IDPS) that can generate alerts when it detects suspicious network traffic. Suricata is a free and open source, mature, fast and robust network threat detection engine. Suricata inspects the network traffic using a powerful and extensive rules and signature language, and has powerful Lua scripting support for detection of complex threats. Suricata NIDS alerts can be found in Alerts, Dashboards, Hunt, and Kibana. Some examples of Suricata alerts include: | |
|  | * A host on the network is scanning for vulnerabilities * An attempted network intrusion * A malicious file or payload being transmitted over the network * A denial-of-service (DoS) attack * Traffic from a known malicious IP address * A suspicious protocol or application being used * A violation of a security policy |

This lab consists of two scenarios of Signature-Based detection using Suricata in Security Onion:



**Lab Requirements:**

1. Security Onion Virtual Machine
2. Two Virtual machines having either Ubuntu/CentOS/Linux based OS
3. Scapy tool for generation of traffic **Scenario 1:**
4. One Virtual machine IP2 generate traffic through Scapy towards Security Onion machine IP1 with destination port 9595 and send Payload Content “Onionattack”
5. The Security Onion web interface detects the attack and show the alert “Security Onion under Attack”.

**Scenario 2:**

1. One Virtual machine IP2 generate traffic through Scapy towards other virtual machine IP3 with source port 1236 and destination port 5002 and send Payload Content “Malicious”
2. The Security Onion web interface detects the attack and show the alert “Your Client VM under attack”

You are required to have screenshots wherever they are asked and answer questions. Please see submission requirements below in this document for the lab.

# Scenario I – Detecting an attack on Security Onion VM(port 9595) by one linux based machine [50 Marks]

|  |  |  |
| --- | --- | --- |
| Move to | /opt/so/saltstack/local/salt/idstools/ | on your manager  . |

1. Configuring Local Rules on Security Onion Machine (Switch to super user) **a)** 
   1. What is the command to find the number of local rules?**[1 Mark]**

The command to fid the number of local rules is “vi local.rules”

* 1. How many rules are present in local rules? **Attach the screenshot**. **[1 + 2 Marks]**

There are no defined rules

A screenshot of a computer

Description automatically generated

1. Add NIDS rule for the attack
   1. Enter local rule in local.rules by using vi editor

**The rule is** : alert tcp any any -> $HOME\_NET 9595 (msg: "Security Onion under Attack!"; content: "Onionattack"; flow:to\_server; nocase; sid:your7digitid; metadata:policy securityips; rev:1;)

* 1. Save the rule- :wq [**Attach the screenshot** with rule] and press enter. **[3 Marks]**

A computer screen shot of a black screen

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| The | next run | of | idstools | | should then | |
|  |  |  | | | | |
| merge | /opt/so/rules/nids/local.rules | into | | /opt/so/rules/nids/all.rules | | which is |

**3.**

what Suricata reads from.

**a)** From the manager, tell Salt to update: sudo salt-call state.highstate. **[1 + 1+ 2 Marks]**

* + - 1. How many total states run?
      2. What was the total run time taken?
      3. Justify your answer with **screenshot**.

A screenshot of a computer

Description automatically generated

**b)** Update rules: sudo so-rule-update **[1 + 1 +1 + 1 + 2 Marks]**

* + - 1. What is the version of Suricata?
      2. What is the location(complete path) of the loading of local file? **iii.** How many total rules loaded? **iv.** What is the location(complete path) where the local files are written or merged?
      3. Justify your answer with **screenshot**.

A screenshot of a computer

Description automatically generated

|  |
| --- |
| $SENSORNAME\_$ROLE |

**c)** Restart Suricata (replacing as necessary): sudo salt

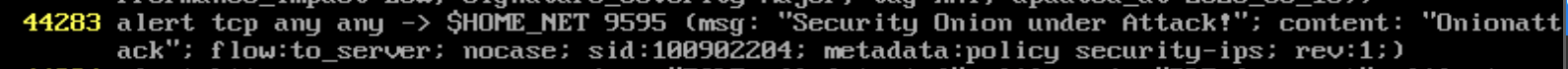
$SENSORNAME\_$ROLE state.apply suricata. **[1 + 1 +1 + 1 + 2 Marks]**

* + 1. What is the complete path to find the $SENSORNAME\_$ROLE?
    2. What is your $sensorname\_$role?
    3. How many total states run?
    4. What was the total run time taken?
    5. Justify your answer with **screenshot**.

A screen shot of a computer

Description automatically generated

* 1. Attach the screenshot showing the local rule added in all.rules in cd /opt/so/rules/nids. **[ 3 Marks]**



1. Perform the following steps in Ubuntu / Centos machine: **[1 + 1 + 2 Marks]**
   1. What is the version of python?
   2. What is the ip address of ens33 Interface?
   3. Install Scapy which is a Python program that enables the user to send, sniff and dissect and forge network packets by using command **pip install scapy**.
   4. If you don’t have pip installed in your VM, then you need to install pip by using apt install pythonversion-pip before step c)
   5. Attach the screenshot of successful installation of scapy.

A computer screen with white text

Description automatically generated

1. Packet generator/sniffer and network scanner/discovery (Python 3) Scapy is **a powerful interactive packet manipulation tool, packet generator, network scanner, network discovery, packet sniffer, etc**. It’s time to craft the script for sending message to Security Onion Machine.
   1. From root, go to scapy directory and type scapy and press enter. You will see prompt >>>
   2. Create the script as follows(Include the comments):

**# Craft the layer 2 information.**

**# The ip addresses can be random, but I would suggest sticking to RFC1918** ip = IP()

ip.dst = " Security Onion Machine IP address” ip.src = " Ubuntu VM IP2 address” **# Craft the layer 3 information.**

**# Since we specified port 9595 in our suricata rule,** tcp = TCP()

tcp.dport = 9595 ( This has to be 9595 because you mentioned in your suricata rule) tcp.sport = 1234 ( You can mention any source port other than assigned ports)

**# Set the playload** payload = " Onionattack"

**# Use the / operator to compose our packet and transfer it with the send() method.**

send(ip/tcp/payload) send(ip/tcp/payload)

**# you can send as many packets you want and once you are done, you can exit the Scapy tool** exit()

* 1. **Attach the screenshot** of your script. **[4 Marks]**

A screen shot of a computer

Description automatically generated

1. Go to web interface of Security onion . **Attach the screenshot** highlighting the attack. **[4 Marks]**
2. Answer the following: **[12 Marks]**
3. A screenshot of a computer

   Description automatically generated
   1. network.data.decoded : Onionattack
   2. observer.name: maisha
   3. rule.category : NIL
   4. rule.metadata,policy: [“security-ips”]
   5. rule.name: Security Onion under Attack!
   6. rule.rule : alert tcp any any -> $HOME\_NET 9595 (msg: "Security Onion under Attack!"; content: "Onionattack"; flow:to\_server; nocase; sid:100902204; metadata:policy security-ips; rev:1;)
   7. source.ip : 192.168.194.130
   8. source.port : 1234
   9. destination.ip : 192.168.44.100
   10. destination.port : 9595
   11. event.severity : 1
   12. event.digested : 2023-08-11T22:31:33.545Z

# Scenario II – Detecting an attack on one linux based machine from another linux based machine by Security Onion [36 Marks]

|  |  |  |
| --- | --- | --- |
| Move to | /opt/so/saltstack/local/salt/idstools/ | on your manager. |

**1.** Configuring Local Rules on Security Onion Machine (Switch to super user) **a)**

**b)** Enter local rule in local.rules by using vi editor. The rule should include:

* 1. Alert with tcp protocol
  2. Source IP as your Ubuntu IP2 VM(Scapy installed) and source port as 1236
  3. Destination IP as your second Ubuntu IP3 VM and destination port as 5002 **iv.** Msg as “Your client VM under attack!”
  4. Content could be anything- eg – badpackets, malicious, badtraffic,etc.
  5. Sid number within the sid range.
  6. Construct your payload with whatever content you have chosen
  7. The payload should use nocase keyword to accept the content with no restriction on

case of the payload.

* 1. Revision of the alert can be anything from 1 to 10

**c)** Save the rule- :wq and press enter

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| The | next run | of | idstools | | should then | |
|  |  |  | | | | |
| merge | /opt/so/rules/nids/local.rules | into | | /opt/so/rules/nids/all.rules | | which is |

**2.**

what Suricata reads from.

1. From the manager, tell Salt to update: sudo salt-call state.highstate **[1 + 1 + 2 Marks]**
   1. How many total states run?
   2. What was the total run time taken?
   3. Justify your answer with **screenshot**.

A screenshot of a computer

Description automatically generated

1. Update rules: sudo so-rule-update **[1 + 1 + 2 Marks]**
   1. How many total rules loaded?
   2. What is the location(complete path) where the local files are written or merged?
   3. Justify your answer with **screenshot**.

A screenshot of a computer program

Description automatically generated

|  |
| --- |
| $SENSORNAME\_$ROLE |

1. Restart Suricata (replacing as necessary): sudo salt

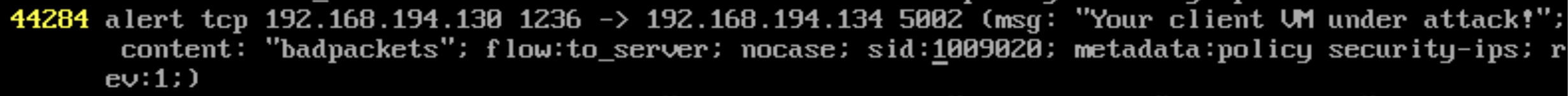
$SENSORNAME\_$ROLE state.apply suricata. **[1 + 1 + 2 Marks]**

* 1. How many total states run?
  2. What was the total run time taken? **iii.** Justify your answer with screenshot.

A screenshot of a computer program

Description automatically generated

1. **Attach the screenshot** showing the local rule added in all.rules in cd /opt/so/rules/nids. **[3 Marks]**



1. Perform the following steps in first Ubuntu / Centos machine(IP2)[Machine where Scapy is already installed from Scenario I]:
   * 1. What is the ip address of ens33 Interface? **[1 Mark]**
     2. From root, go to scapy directory and type scapy and press enter. You will see prompt >>>
     3. Generate the traffic from Ubuntu IP2 VM to Ubuntu IP3 VM. Create the script including comments as per the rule made in step 1b). Attach the **screenshot of the script** after sending 3-4 packets. **[4 Marks]**

A screen shot of a computer

Description automatically generated

1. Go to web interface of Security onion . **Attach the screenshot** highlighting the attack. **[4 Marks]**
2. Answer the following:**[12 Marks]**

A screenshot of a computer

Description automatically generated

* + 1. network.data.decoded : badpackets
    2. observer.name : maisha
    3. rule.category : NIL
    4. rule.metadata,policy: “security-ips”
    5. rule.name : Your client VM under attack!
    6. rule.rule : alert tcp 192.168.194.130 1236 -> 192.168.194.134 5002 (msg: "Your client VM under attack!"; content: "badpackets"; flow:to\_server; nocase; sid:1009020; metadata:policy security-ips; rev:1;)
    7. source.ip : 192.168.194.130
    8. source.port : 1236
    9. destination.ip : 192.168.44.100
    10. destination.port : 5002
    11. event.severity : 1
    12. event.digested : 2023-08-11T03:48:37.874Z

**Things to Explore:**

You are welcome to explore beyond the mandatory requirements if you wish.

**General Submission Requirements**

* + Include an opening comment with your full name and a short description of the lab. Please name your file as fistname\_Lab2.pdf.
  + The assignment asks you to provide response on the questions being asked. At times you have to also provide screenshot for the work you have done. Please make sure you use the **exact numbering scheme as used in the assignment** while reponding to questions. The assignment needs to be submitted in pdf with much clarity on the screenshot.
  + If a particular point does not ask any question and rather it’s a sequence step, please write N/A in the response of that question.
  + Please make sure all your screenshots should have your name as name of virtual machine and time stamp of your machine. This is primarily needed to make sure that each student should perform the lab on their own.
  + Academic Integrity violations would be treated severely.