ARL-SR-0473 ● JUNE 2023

Hands-on Cybersecurity Studies: Network Diversification Through Dynamic Honeypots

by Valeria Duron and Jaime C Acosta

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-SR-0473 ● JUNE 2023

Hands-on Cybersecurity Studies: Network Diversification Through Dynamic Honeypots

Valeria Duron
University of Texas at El Paso

Jaime C Acosta
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
June 2023	Special Report	8/01/2022	4/30/2023

4. TITLE AND SUBTITLE
Hands-on Cybersecurity Studies: Network Diversification Through Dynamic Honeypots

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Valeria Duron and Jaime C Acosta

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLC-ND Adelphi, MD 20783	ARL-SR-0473

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: Jaime C Acosta, 0000-0003-2555-9989

14. ABSTRACT
This report describes a hands-on exercise that guides participants in learning about the functionalities of the Common Open Research Emulator and how it can be used along with the Cybersecurity Deception Experimentation System to create dynamic honeypots. Both of these software tools are available publicly. The participant will take the role of defender, as a bank administrator responsible for protecting information stored in databases.

15. SUBJECT TERMS
cybersecurity, honeypots, autonomous active defense, Collaborative Innovation Testbed, CyberRIG, Network, Cyber, and Computational Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	30

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Jaime C Acosta	(575) 993-2375

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

Cyber defense through the use of honeypots is a common and effective strategy to divert malicious traffic away from important systems by emulating real systems and luring attackers to decoys. In the domain of cybersecurity, there is currently an asymmetry between the attacker and the defender due to the fact the defender must protect the entirety of a network or system, while the attacker must only find one weakness to carry out an exploit. Furthermore, as new defenses are developed, attackers continually find ways to bypass them. There are two types of honeypots, static and dynamic, with the main difference between them being that static honeypots are fixed and do not change, while dynamic honeypots are designed to evolve and adapt to more closely mimic real systems, making them more effective. For this reason, dynamic honeypots are very useful, as they not only redirect adversaries from legitimate resources, but also help gain insight into attacker techniques through observation and analysis in order to better react and adapt to their tactics.

1.1 Exercise Overview

This exercise is a continuation of the exercise presented in the previous Hands-on Cybersecurity Studies report,[1] which focuses on publicly available tools that can be used to emulate fictitious networks and experiment with defense technologies through the creation and use of honeypots. In this exercise, participants will learn about the functionalities of the Common Open Research Emulator (CORE) and how it can be used along with the Cybersecurity Deception Experimentation System (CDES) to create dynamic honeypots. Both of these software tools are available publicly. The participant will take the role of defender, as a bank administrator responsible for protecting sensitive information stored in databases using a publicly documented and available version of CouchDB. The exercise focuses on creating a dynamic honeypot using CORE and CDES to detect potential attackers scanning the CouchDB port (5984) for weaknesses. Participants will learn how to configure CORE, establish connections between nodes, and configure runtime hooks/scripts using Suricata for intrusion detection. The participant will be given the preconfigured environments established in the previous exercise from the Hands-on Cybersecurity Studies.[1] The ultimate goal of this exercise is to create a dynamic honeypot that can detect a potential attack and divert this potential malicious traffic away from the legitimate network.

1.2 Common Open Research Emulator (CORE)

CORE[2] is an open-source tool for building virtual networks. CORE has the functionality of emulating real computer networks and runs in real time. It has an easy-to-use graphical user interface (GUI) as well as an application program interface (API).

CORE uses nodes to represent machines and systems in a network and uses a link tool to create connections between the nodes, while auto-assigning IP addresses. The CORE RJ45 node acts as a physical interface tool, allowing other networks and devices to be connected to the running scenario through the node. CORE has numerous complex functionalities but remains an overall lightweight software, making it an extremely useful tool.

When used in conjunction with CDES,[3] CORE can create controlled environments for deploying cyber defense technologies, such as honeypots. This is very promising as it can help to simulate cyber incidents for many purposes such as studying attacker behavior, experimenting with different defense techniques, and testing honeypot effectivity, all in a controlled and safe environment.

1.3 Cybersecurity Deception Experimentation System (CDES)

CDES is an extension to CORE that allows one to design and implement cyber defense techniques in the form of dynamic honeypots. CDES uses a Conditional Connection Decision Node (CC_Decision Node) that runs logic for determining which CORE node connections should be enabled or disabled based on the Monitor, Trigger, and Swapper components. The Monitor engine monitors the activity and traffic within the environment, feeding it to the Trigger mechanism. The Trigger mechanism consists of a python file that is configured with rules and algorithms that will "trigger" a command to the Swapper component to activate or deactivate network connections. The Trigger and Swapper components are meant to be customized based on the specific project and/or services in order to trigger the swap when specific data is generated by the monitor. These functionalities are beneficial since one of the issues with honeypots is that it is not feasible for them to always be on or running. With CDES, these problems are avoided by having the honeypots instantiated with the trigger mechanism only when a potential attack is detected. These components, along with CORE's extensiveness, result in high capabilities for experimentation and testing. The combination of CORE and CDES creates a valuable experimentation platform and tool for proactively defending against cyber attacks, as well as improving strategies.

1.4 Gaining a Deeper Understanding

During the exercise, the participant will learn how to leverage the capabilities of CDES, along with CORE, to create a simple network environment that includes a dynamic honeypot. The participant will be provided with the same virtual machines (VMs) used in the first Hands-on Cybersecurity Studies exercise, which include an attacker VM, legitimate VM, and honeypot VM. Both the legitimate and honeypot systems will be running a well-known and publicly documented weak version of CouchDB. The participant will also be given the CORE scenario partially set up, and the participant will use CORE and CDES to configure and emulate the fictitious bank's network, using a decision node that will be triggered to close traffic to the legitimate node and instead redirect it to the honeypot. The participant will be in charge of configuring the nodes, runtime hooks and scripts, and Suricata rules that will trigger the network swapping. Once the participant has completed the configuration and network emulation portion of the exercise, they will then use a set of tools, including Metasploit, to test the network and trigger the redirection. Through this experimentation, the participants will learn and experience how dynamic honeypots are capable of adapting and in turn can be more effective than static honeypots at achieving higher levels of protection. Furthermore, the participant will experience first-hand how an attacker's resources and time can be exhausted exploiting a decoy system instead of the legitimate one.

2. Setup and Configuration

The setup of the exercise includes four VMs and several open-source software tools. These technologies are:

- Oracle VirtualBox 6.1.30 64-bit[4]
- Ubuntu 18 64-bit VM[5]
- Common Open Research Emulator (CORE) 7.1[2]
- Kali Linux 2022.1 64-bit VM[6]
- Metasploit Framework version 6[7]
- Alpine Linux 3.11 64-bit VMs[8]

The US Army Combat Capabilities Development Command Army Research Laboratory South Cyber Rapid Innovation Group (CyberRIG) Collaborative Innovation Testbed[9] (CIT) is used to host the four VMs. The Ubuntu VM includes the CORE software already installed and ready to use, including the CORE scenario file the participant will be given. The two Alpine VMs are already preconfigured

to be connected to the CORE scenario through their network interfaces. These two Alpine VMs will not be used by the participant but must be running for the exercise to be successful, as they contain the CouchDB services. The Kali VM will be used by the participant as the attacker machine during the exercise. It contains the Metasploit framework ready to use during the exercise.

Any configuration needed to be done by the participant for the Kali VM or CORE software will be explained and instructed in the exercise.

3. Learning Objectives

The purpose of the exercise is to gain an understanding of available cyber security defense technologies in the area of network diversification through the use of honeypots. The participant will gain an understanding of how honeypots can be used to divert malicious traffic away from important systems by emulating real systems and luring attackers to the honeypot. Furthermore, they will experience how honeypots can be effective in shifting costs to the attacker, by wasting attacker resources and time.

The learning objectives associated with the exercise are as follows:

- **Use and functionality of CDES within CORE:** Participants will be tasked with creating and configuring a network scenario emulating that of a fictitious bank network, while incorporating a CDES decision node within it. Not only will they learn how to model a realistic network, but they will also experience first-hand how the decision node can be used to trigger a switch between the network connection, closing traffic to the legitimate system and instead redirecting it to the honeypot. They will learn how to set up virtual networks, deploy dynamic honeypots, and configure their characteristics, configurations, and responses. The participant will gain insight on how promising these tools can be in cyber defense experimentation.
- **The capabilities of dynamic honeypots in cyber defense:** Participants will realize the limitations of static honeypots and how dynamic honeypots can more successfully protect from cyber attacks and increase security. They will recognize how much more effective dynamic honeypots can be due to their adaptive nature as well as how they can be used to analyze attacker behavior, tactics, or techniques. Furthermore, the participant will experience first-hand how dynamic honeypots, through the generation of fake data, can result in attackers exhausting their resources, time, and even money on exploiting a honeypot system.

4. Exercise

The following exercise is presented to participants in a step-by-step fashion. Participants will complete several tasks through the CIT system. All the data and systems in the exercise are **fictional** and **simulated**.

This mission briefing is provided to participants:

You are an administrator for the El Paso Bank. All of El Paso Bank customers' information is stored in databases using CouchDB. One of your tasks as administrator consists of taking a defender role and protecting sensitive information. You have recently discovered there was a vulnerability found in CouchDB. It is your job to discover any vulnerabilities in the company's networks and find solutions to better secure the system.

You now know both the Legitimate and Honeypot networks have a CouchDB vulnerability. Your job is to detect whenever the CouchDB port (5984) is scanned, which can signify a potential attacker looking for vulnerabilities. If the CouchDB port is scanned, you will close traffic to the Legitimate network, and instead redirect the traffic to the Honeypot network.

1) To open CORE you will need to start the CORE daemon. Open a terminal window by right-clicking on the desktop and selecting Open Terminal. Afterward, input and run:

sudo core-daemon

*The **sudo keyword is used** before commands to run programs as a root user. When prompted for a password, use **toor***

2) Once the CORE daemon has been restarted, we will open the CORE gui. In a new, separate, terminal window as shown in Fig. 1, input and run:

sudo core-gui

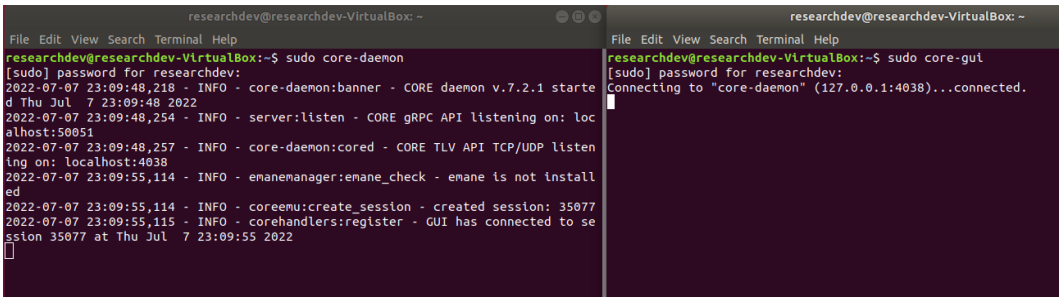

Fig. 1 Initiating CORE

4.1 Step 1: Creating the Dynamic Scenario

1) Navigate to /home/researchdev/Desktop/HoneypotWorkshop/Part2 and open the .imn file named Part2.imn

This file is the same as your Static scenario file, with the links for the Legitimate and Honeypot nodes removed. See Fig. 2.

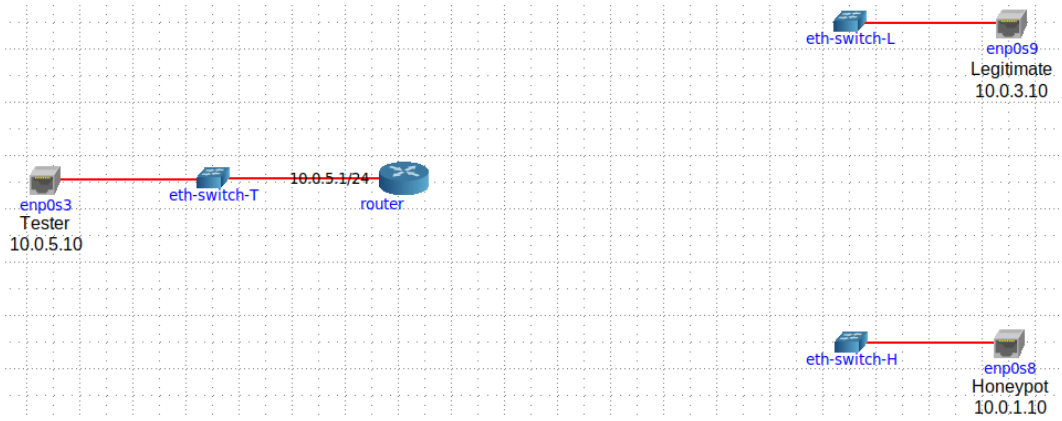

Fig. 2 Opening scenario file

2) Use the left-hand sidebar on CORE and add the following nodes to the canvas: one **cc_dec_node_ovs** and two **routers** as shown in Fig. 3 and Fig. 4.

Fig. 3 Decision node

**** Take note of the node number assigned to the cc_dec_node_ovs when you place it ****

node number: _____

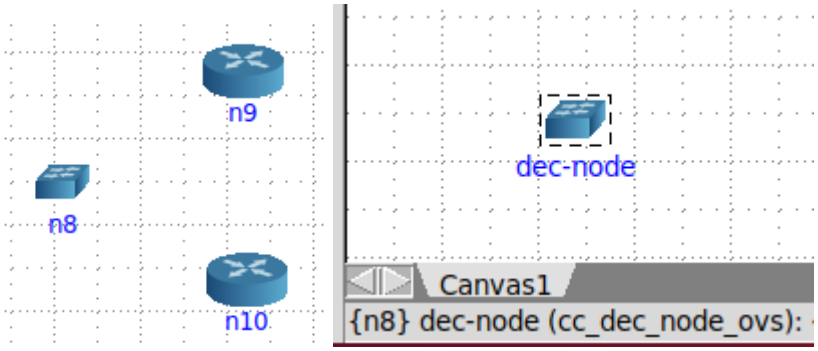

Fig. 4 Decision node number

**** You can also find this number on the bottom left of the CORE screen when hovering over a specific node ****

3) Double click the nodes to rename them. For the router nodes, name one **router-L** and another **router-H** (for Legitimate and Honeypot). Name the cc_dec_node_ovs, **dec-node** (see Fig. 5).

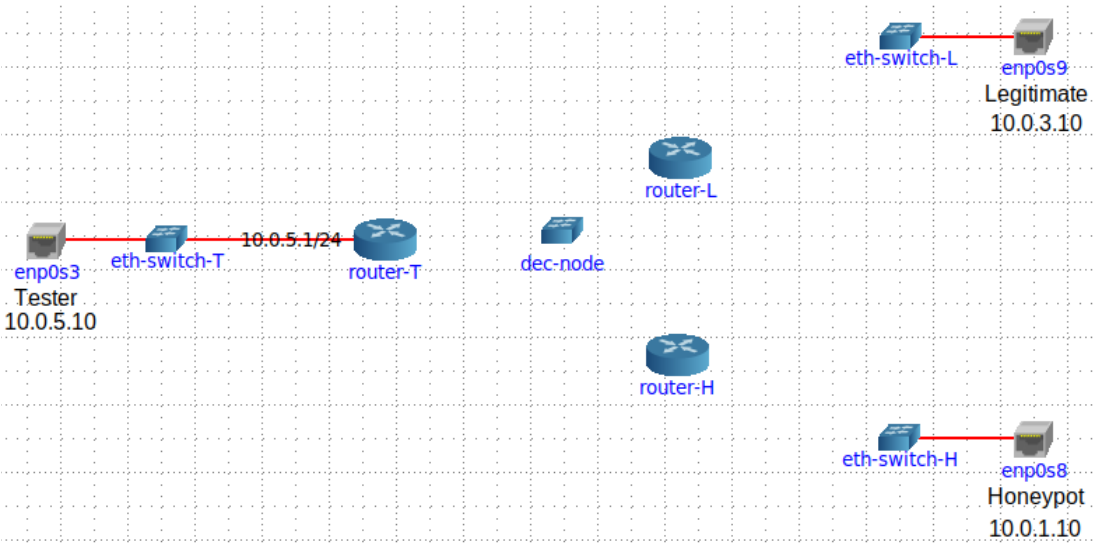

Fig. 5 Node name convention

**** Make sure to use a hyphen and not an underscore, CORE is very particular and will not run if any node name has a character other than those allowed, such as underscore ****

4) Now that the nodes have been added and named, connect them as shown in Fig. 6.

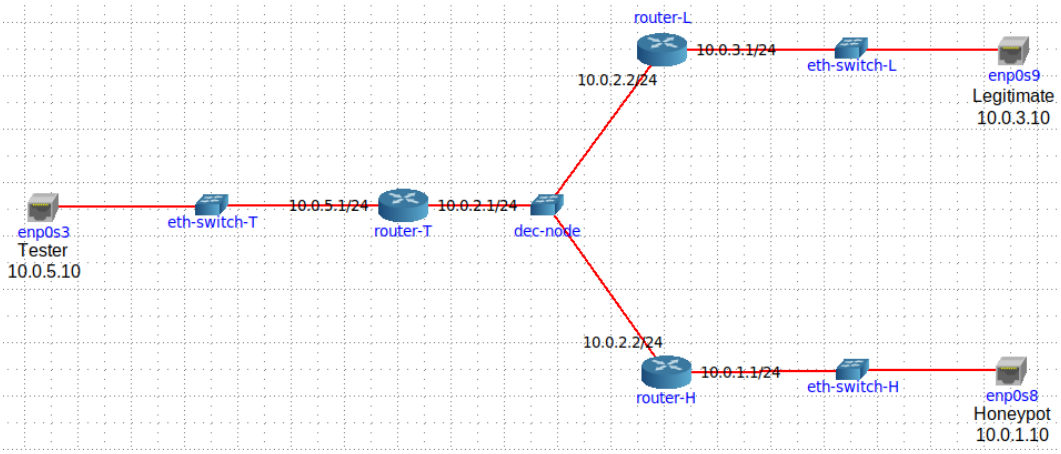

Fig. 6 Node connection

5) After linking the nodes, an IP address will be assigned to each. You will now configure the two new router IP addresses. To configure the router node settings, double click on each router node and set the IPv4 addresses to match those in Fig. 7.

Legitimate router to Legitimate node – 10.0.3.1

Legitimate router to dec-node – 10.0.2.2

Honeypot router to Legitimate node – 10.0.1.1

Honeypot router to dec-node – 10.0.2.2

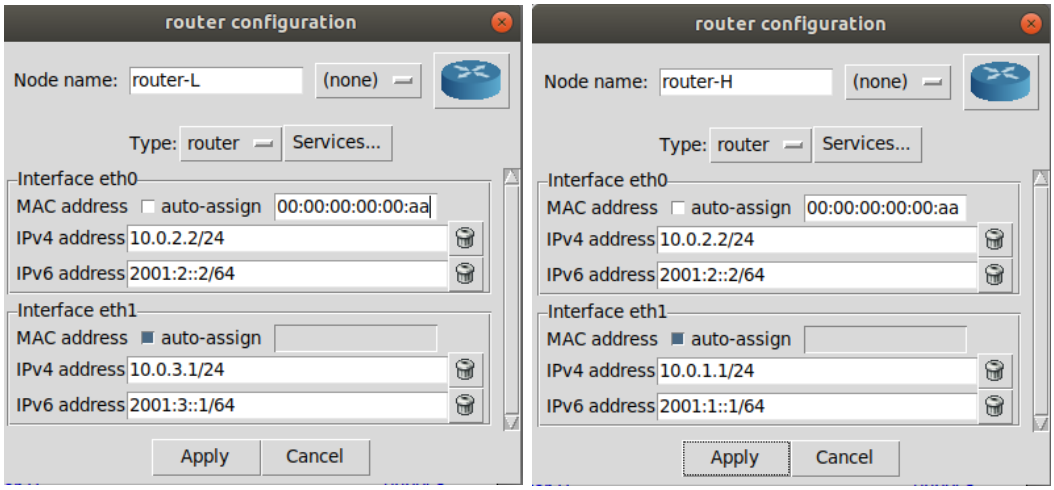

Fig. 7 Legitimate and Honeypot IP address configuration

****

We would want the attacker (or tester in our case) to not know the difference between the Legitimate and Honeypot networks. For this reason, in this scenario, the Legitimate and Honeypot routers have the same IP address and MAC address in the connection to the decision node.

****

6) You will now configure the tester router IP address. To configure the router node settings, double click on the router node and set the IPv4 addresses to match those in Fig. 8.

   Tester router to dec-node – 10.0.2.1

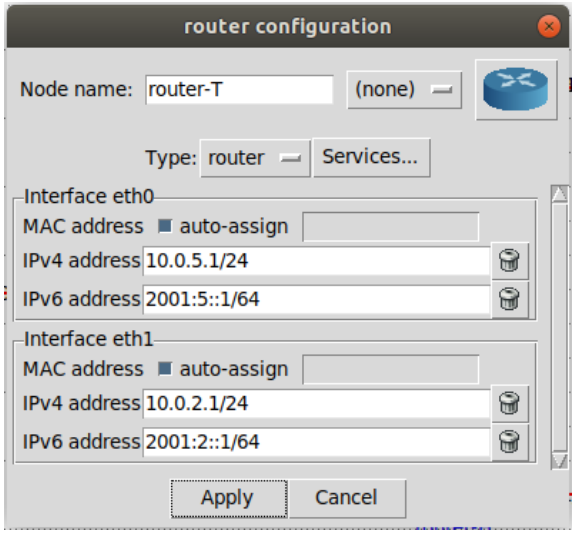

Fig. 8 Tester IP address configuration

7) Lastly, for **each** router node double click to configure and select **services**. Under **Utility** ensure **CC_Node** is selected.

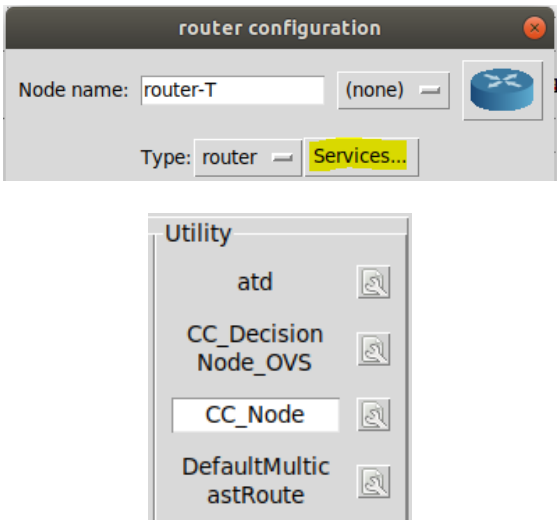

Fig. 9 Router node configuration

4.2 Step 2: Session Hooks

We will be using Suricata for intrusion detection so we will need to set some runtime hooks/scripts.

1) Click the **Session** tab on CORE and select **Hooks**. You should see a window popup as in Fig. 10. Click on the New Hook icon to create a new script.

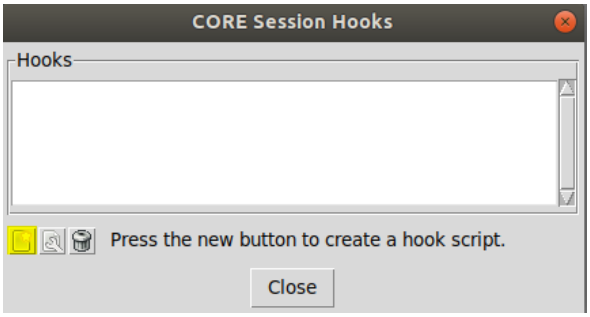

Fig. 10 Session hooks

2) Select the Load File icon to load a preconfigured script file. You will see a window like that in Fig. 11.

   Navigate to **home/researchdev/Desktop/HoneypotWorkshop/Part2** and select the **suricatarule_instantiation_hook**

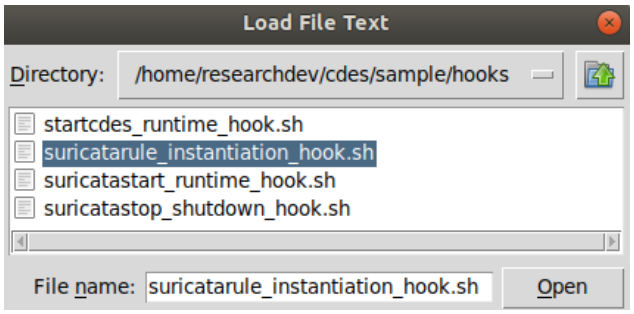

Fig. 11 Load instantiation hook

Click **open** to load the file and on the right drop-down bar select **instantiation** as shown in Fig. 12.

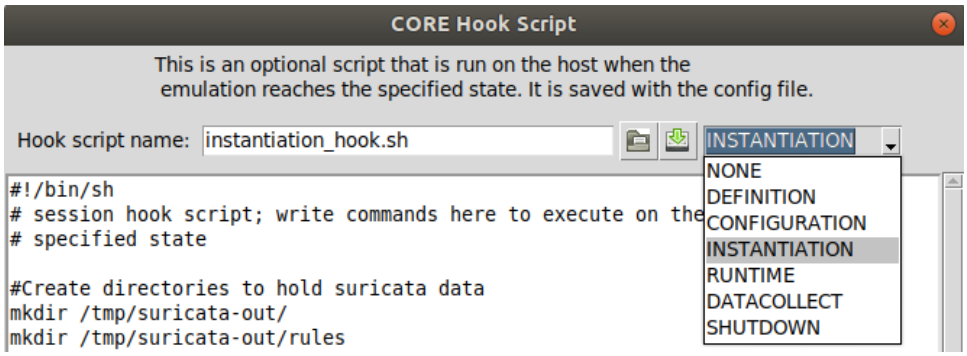

Fig. 12 Instantiation hook configuration

This instantiation script includes the rules that will be used by Suricata and will trigger alerts.

3) As shown in Fig. 13, add the following rule to the script. **alert tcp any any -> any 5984 (msg:"TCP - Port 5984 scan detected"; classtype:network-scan; sid:2005001; rev:1;)**

4) Select **apply** to save the script.

```
#Create directories to hold suricata data
mkdir /tmp/suricata-out/
mkdir /tmp/suricata-out/rules

#Create suricata rules -- note that if too long, this will get truncated when using .imn files
cat << EOF > /tmp/suricata-out/rules/custom.rules

alert tcp any any -> any 5984 (msg:"TCP - Port 5984 scan detected"; classtype:network-scan; sid:
2005001; rev:1;)

alert udp any any -> any 5984 (msg:"UDP - Port 5984 scan detected"; classtype:network-scan; sid:
2005002; rev:1;)
EOF
```

Fig. 13 Suricata rules

****

The rule you created has several configuration attributes as stated below.

tcp/udp is specifying the protocol to detect.

*The **first any** is the source IP address (we use "any" since we want to be alerted of any address scanning port 5984).*

*The **second any** is the source port (we use "any" since we want to be alerted of any port scanning port 5984).*

*The **third any** is the destination IP address (we could modify this to the El Paso Bank's addresses but for now we will leave it as any).*

***5984** signifies the destination port (in this case the port being scanned).*

*Inside the parentheses we have the alert or message, which will be logged. The **classtype** refers to the rule classification. **sid** is a signature ID for the alert and **rev** is revision number for the alert.*

****

You will add another script for runtime using the same steps.

5) Click the **Session** tab on CORE and select **Hooks.** Click on the New Hook icon to create a new script. Select the Load File icon to load a preconfigured script file.

   From the same folder **home/researchdev/Desktop/HoneypotWorkshop/Part2** select **suricatastart_runtime_hook** and set the drop-down to **runtime.** Make sure your screen matches that shown in Fig. 14.

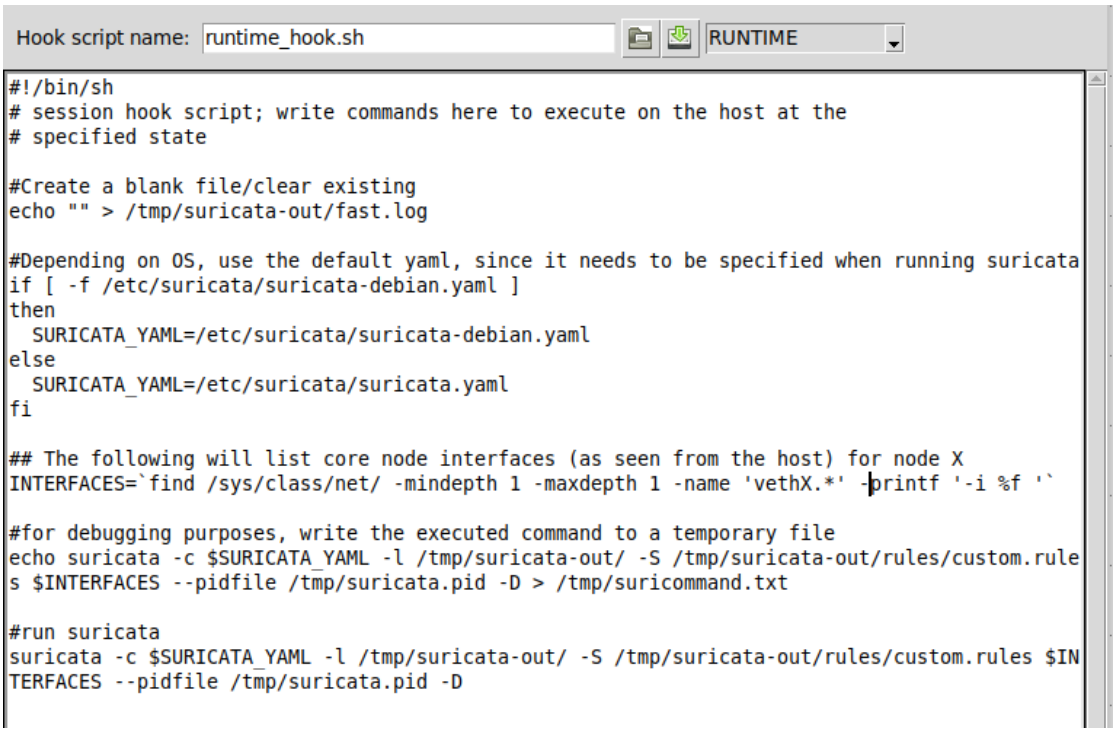

Fig. 14 Runtime hook configuration

6) There is a portion of the script in which we set the interfaces that interact with the decision node. To do so we need to set the node number for the decision node. The portion of the script is shown in Fig. 15.

```
## The following will list core node interfaces (as seen from the host) for node X
INTERFACES=`find /sys/class/net/ -mindepth 1 -maxdepth 1 -name 'vethX.*' -printf '-i %f '`
```

Fig. 15 Setting decision node number

7) Replace the **X** in **vethX** with your decision node's number. (Note that this number should be input as a hexadecimal, if greater than 9.)

   Select **apply** to save.

****

If you forgot to take note of the node number, you can find this number by hovering over the decision node. The number will be displayed at the bottom of the CORE screen, under ***Canvas 1***

In the script we can see a command ***echo "" > /tmp/suricata-out/fast.log*** *This tells us that the alerts and messages will be logged at this location*

****

You will add another script for shutdown using the same steps.

8) Click the **Session** tab on CORE and select **Hooks.** Click on the New Hook icon to create a new script. Select the Load File icon to load a preconfigured script file.

   From the same folder **home/researchdev/Desktop/HoneypotWorkshop/Part2** select **suricatastop_shutdown_hook** and set the drop-down to **shutdown.** Your screen should reflect that shown in Fig. 16.

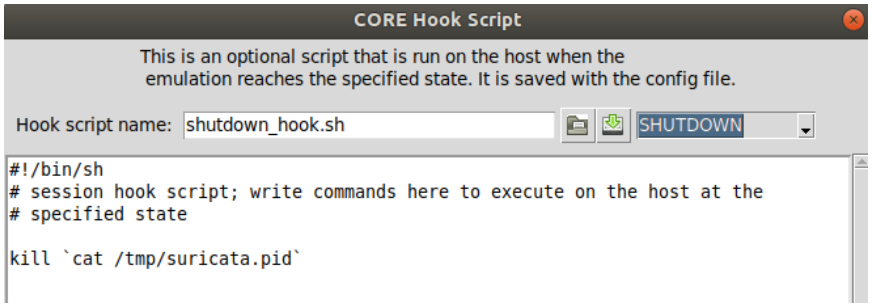

Fig. 16 Shutdown hook configuration

9) Select **apply** to save.

These three hooks are for the Suricata intrusion detection.

Lastly, we will add another hook for the CDES functionality.

10) Click the **Session** tab on CORE and select **Hooks.** Click on the New Hook icon to create a new script. Select the Load File icon to load a preconfigured script file.

    From the same folder **home/researchdev/Desktop/HoneypotWorkshop/Part2** select **cdes_runtime_hook** and set the drop-down to **runtime**. Your screen should reflect that shown in Fig. 17.

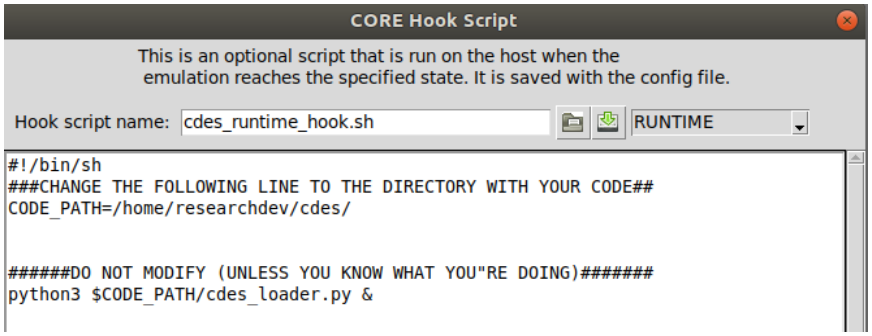

Fig. 17 CDES runtime hook configuration

11) Select **apply** to save.

4.3 Step 3: Decision Node

The CDES decision node has a monitor and trigger functionality. The trigger script reads input from the monitor and "triggers" an action.

1) Double click the **dec_node** to configure and select **services** as shown in Fig. 18.

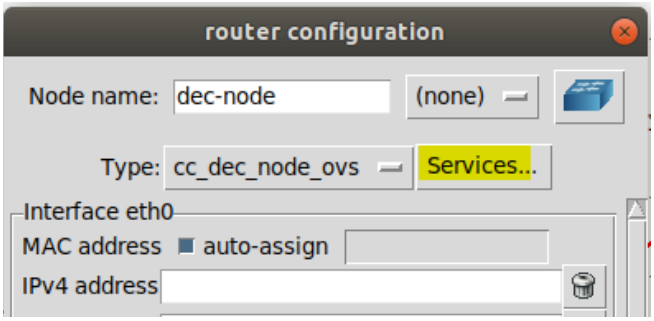

Fig. 18 Decision node configuration

2) Under **Utility** select the settings icon for **CC_Decision_Node_OVS** as shown in Fig. 19.

Fig. 19 Decision node configuration 2

On the filename you should see **MyMonitor.sh**. This signifies this is the CDES Monitor portion script.

3) Remove the current script and instead input **tail /tmp/suricata-out/fast.log -f**, which is where the Suricata alerts are logged. Your screen should reflect that shown in Fig. 20.

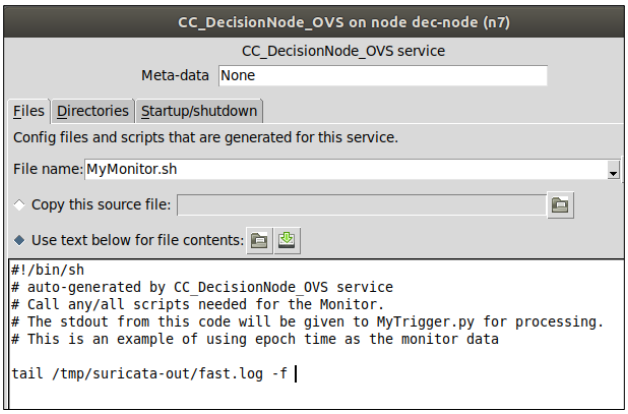

Fig. 20 Decision node monitor script

The alerts from the rules we set in the hook script will act as the CDES monitor.

3) Select **Apply.**

4) On the file name drop-down select **MyTrigger.py**. This is the script that will implement the trigger action. Your screen should reflect that shown in Fig. 21.

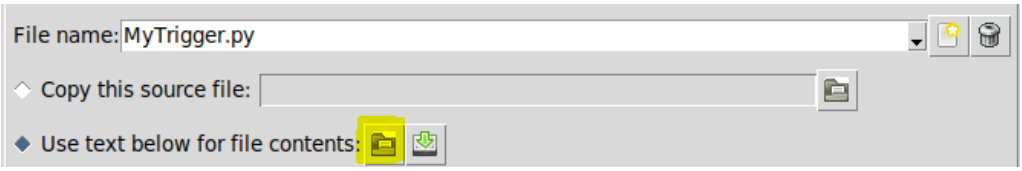

Fig. 21 Decision node trigger script

5) Click on the load file icon next to "Use text below for file contents." Navigate to home/researchdev/Desktop/HoneypotWorkshop/Part2 and open the WorkshopTrigger.py file as shown in Fig. 22.

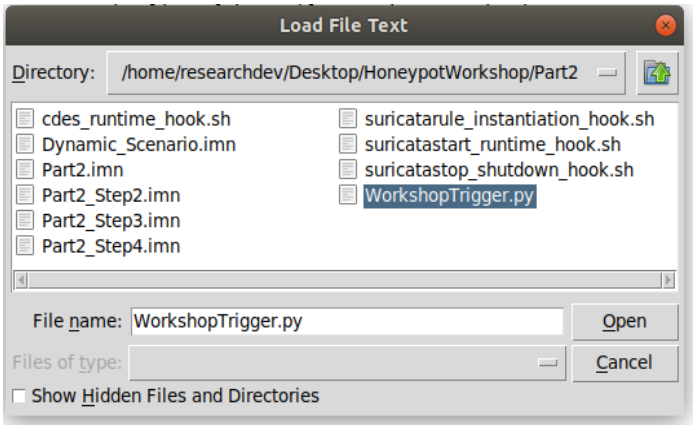

Fig. 22 Loading trigger script

6) The script is already set to process the Monitor's data, but we will add the script to trigger the switch. Open the file **triggerdata** from the HoneypotWorkshop/Part2 folder.

7) Copy the file contents onto the end of the **WorkshopTrigger.py** file. Ensure you copy all contents of the file, including the indentation/space before the text, or else you will run into errors. You can do so by using **CTRL+A** and **CTRL+C** to copy from the file, and **CTRL+V** to paste on the CORE MyTrigger.py script. Your screen should reflect that shown in Fig. 23.

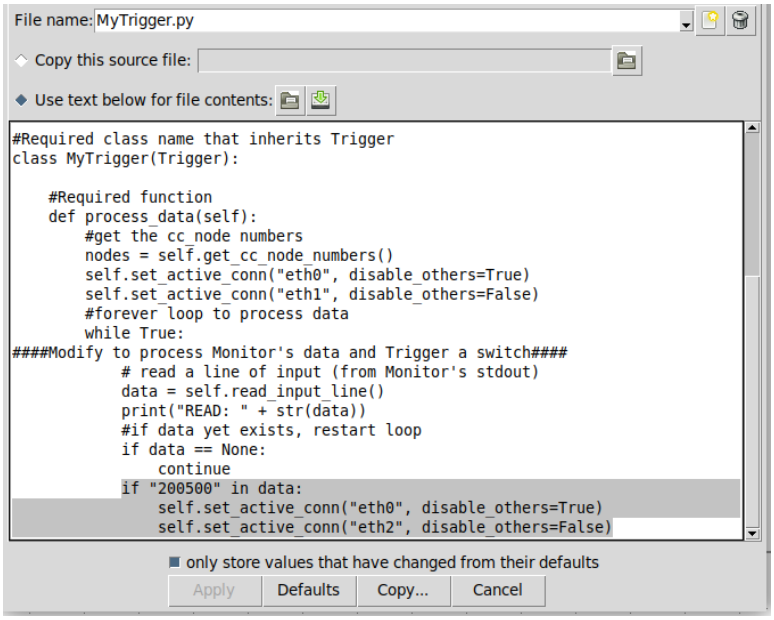

Fig. 23 Preconfigured trigger script

In this scenario, we have the same IP address for the routers of both the Legitimate and Honeypot. The initial portion of the script is already set up to activate only the legitimate connection.

*The rules we set previously have **2005001** and **2005002** as their signature ID (one for TCP and the other for UDP protocols). Since we want both alerts to trigger a switch, we will use "look for **200500"** in the data. If this is found in the monitor data, we disable all connections through the decision node, except the Honeypot node.*

8) After completing the trigger script, select **apply** to save.

4.4 Step 4: Start CORE

Now that we have set up the Suricata and CDES settings, we will run the CORE scenario we created.

1) Click the green play button on the left-hand sidebar on CORE, to start the scenario. You should see red square outlines turn green, signifying the running state. Also, now that we have added a CDES runtime hook, you should see the link from **dec-node** to **router-L** turn blue and the link from **dec-node** to **router-H** turn yellow (blue signifying active and yellow inactive).

2) To test the connectivity once again, try pinging from **router-T** to each address (10.0.5.10 and 10.0.3.10 should ping; 10.0.1.10 should not). See Fig. 24 for reference.

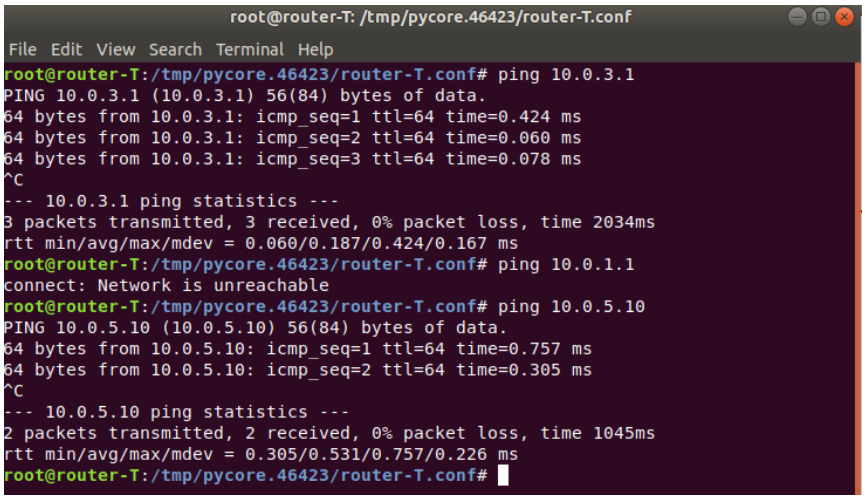

Fig. 24 Connectivity testing

3) Open a terminal within the CORE vm, not the CORE scenario, and input **tail -f /tmp/suricata-out/fast.log** as shown in Fig. 25 in order to see the data/alerts in the CDES monitor. Leave this terminal open.

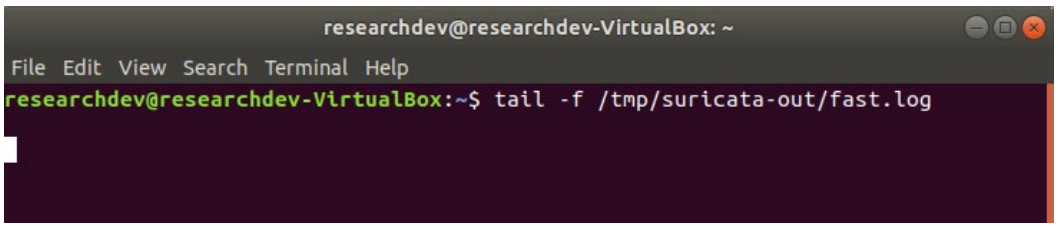

Fig. 25 Monitor data alerts

If everything is configured correctly, the scenario was created successfully. Switch to the Tester VM to start the attack.

4.5 Step 5: Testing

We will now attempt to scan port 5984 searching for the CouchDB vulnerability.

Press back on the browser and select the Tester VM named RJ45_A.

> If you are prompted for a login on the VM in the browser, use the following credentials:
>
> username: **researchdev**
>
> password: **toor**

1) Open two terminals in the Tester VM. From one **ping 10.0.3.10** and from the other **ping 10.0.1.10**. You should see only the ping to 10.0.3.10 go through and 10.0.1.10 destination be unreachable. Leave these commands running as shown in Figs. 26 and 27.

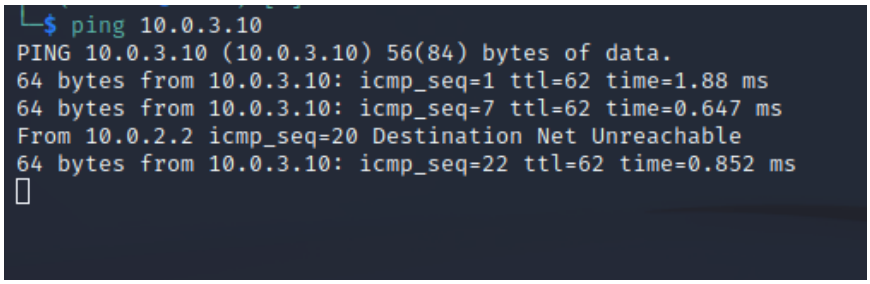

Fig. 26 Connectivity to Legitimate system

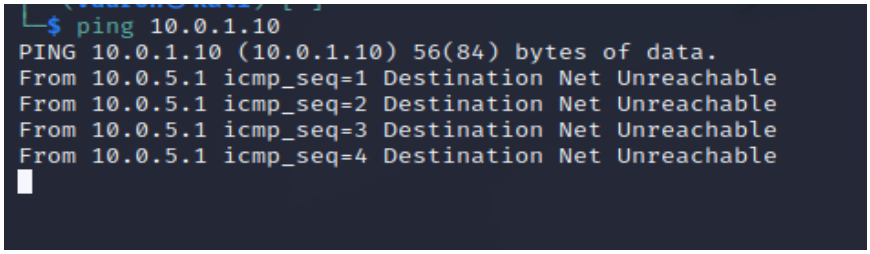

Fig. 27 Connectivity to Honeypot unreachable

2) Open a third terminal and input **msfconsole**.

If prompted for questions on Metasploit console, select no.
Startup should take approximately one minute.

3) We will use nmap to scan for port 5984 (CouchDB). Due to the Suricata rule we created; it does not matter which IP address we scan. Input the following (also shown in Fig. 28).

db_nmap -p 5984 10.0.2.2

```
msf6 > db_nmap -p 5984 10.0.2.2 -sV -Pn
[*] Nmap: Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-05 16:46 MDT
[*] Nmap: Nmap scan report for 10.0.2.2
[*] Nmap: Host is up (0.00050s latency).
[*] Nmap: PORT     STATE  SERVICE VERSION
[*] Nmap: 5984/tcp closed couchdb
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 13.49 seconds
msf6 >
```

Fig. 28 Scanning through metasploit/Nmap

4) After performing the scan, you should see a change in the two terminals from which you pinged. Now you should see the 10.0.3.10 destination be unreachable and 10.0.1.10 go through. Furthermore, if you return to the CORE VM and scenario you should see there was also a change in the dec-node link colors (the link to router-L should be yellow/inactive and the link to router-H should be blue/active); see Fig. 29.

```
64 bytes from 10.0.3.10: icmp_seq=125 ttl=62 time=0.472 ms
64 bytes from 10.0.3.10: icmp_seq=126 ttl=62 time=0.415 ms
64 bytes from 10.0.3.10: icmp_seq=127 ttl=62 time=0.413 ms
64 bytes from 10.0.3.10: icmp_seq=128 ttl=62 time=0.357 ms
From 10.0.2.2 icmp_seq=136 Destination Net Unreachable
From 10.0.2.2 icmp_seq=137 Destination Net Unreachable
From 10.0.2.2 icmp_seq=138 Destination Net Unreachable
From 10.0.2.2 icmp_seq=139 Destination Net Unreachable
```

```
root@Tester2: /tmp/pycore.43145/Tester2.conf
File  Edit  View  Search  Terminal  Help
root@Tester2:/tmp/pycore.43145/Tester2.conf# ping 10.0.1.10
PING 10.0.1.10 (10.0.1.10) 56(84) bytes of data.
From 10.0.4.1 icmp_seq=1 Destination Net Unreachable
From 10.0.4.1 icmp_seq=2 Destination Net Unreachable
From 10.0.4.1 icmp_seq=3 Destination Net Unreachable
From 10.0.4.1 icmp_seq=4 Destination Net Unreachable
From 10.0.4.1 icmp_seq=44 Destination Net Unreachable
From 10.0.4.1 icmp_seq=86 Destination Net Unreachable
64 bytes from 10.0.1.10: icmp_seq=126 ttl=62 time=0.762 ms
64 bytes from 10.0.1.10: icmp_seq=127 ttl=62 time=0.359 ms
64 bytes from 10.0.1.10: icmp_seq=128 ttl=62 time=0.583 ms
64 bytes from 10.0.1.10: icmp_seq=129 ttl=62 time=0.479 ms
```

Fig. 29 Network connectivity swaps

The terminal from which you were tailing the log should also display the alert/message that triggered the change as shown in Fig. 30.

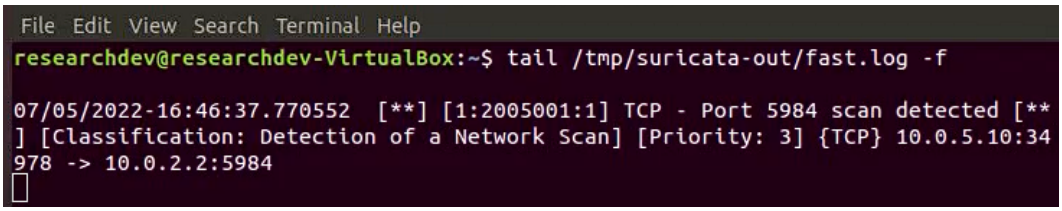

Fig. 30 Monitor data that triggered swap

Great job! Now you have successfully implemented a dynamic honeypot that will protect your network anytime your application is scanned.

The Suricata rules can be configured to detect multiple intrusion attempts and further protect vulnerable networks from various different attack attempts.

5. Conclusion

In this report, we provide a basic learning module that introduces participants to dynamic honeypots and how they can be used as a means of cybersecurity defense. The exercise offers the participant a unique opportunity of gaining practical experience in using dynamic honeypots within the context of CORE and CDES. This exercise equips participants with the knowledge and skills to effectively deploy, configure, and utilize dynamic honeypots in cyber defense scenarios. When preceded by the first Hands-on Cybersecurity Studies report, the two reports complement one another and give extensive insight on cyber defense through honeypots, as well as the capabilities and limitations of static and dynamic honeypots. In summary, the report provides a valuable opportunity to enhance cybersecurity skills, expand knowledge in cyber defense, and gain practical experience with honeypots for improving cyber defenses.

6. References

1. Duron V, Acosta JC. Hands-on cybersecurity studies: network diversification with honeypots. DEVCOM Army Research Laboratory (US); 2022 Sep. Report No.: ARL-SR-0462.

2. Ahrenholz J, Danilov C, Henderson TR, Kim JH. CORE: A real-time network emulator. Proceedings of the MILCOM 2008-2008 IEEE Military Communications Conference. 2008 Nov. IEEE. p. 1–70.

3. Acosta JC, Basak A, Kiekintveld C, Leslie N, Kamhoua C. Cybersecurity deception experimentation system. Proceedings of IEEE Secure Development (SecDev); 2020. p. 34–40. IEEE.

4. Oracle. Welcome to VirtualBox.org! 2023 [accessed 2023 15 Apr]. https://www.virtualbox.org/.

5. Canonical Ltd. Ubuntu 18.04.6 LTS (Bionic Beaver). 2018 [accessed 2023 15 Apr]. https://releases.ubuntu.com/18.04/

6. OffSec Services Limited. Kali Linux 2022.1 Release (Visual Updates, Kali Everything ISOs, Legacy SSH). 2022 Feb 14 [accessed 2022 15 Apr]. https://www.kali.org/blog/kali-linux-2022-1-release/

7. Rapid7. Metasploit. 2023 [accessed 2023 15 Apr]. https://www.metasploit.com/

8. Alpine Linux Development Team. Alpine Linux. 2023 09 May [accessed 2023 Apr]. https://www.alpinelinux.org/downloads/

9. Acosta C, Clarke L, Medina S, Akbar M, Hossain MS, Free-Nelson F. Repeatable experimentation for cybersecurity moving target defense. Proceedings of the International Conference on Security and Privacy in Communication Systems; 2021. p. 82–99. Springer, Cham.

List of Symbols, Abbreviations, and Acronyms

API	application program interface
ARL	Army Research Laboratory
CDES	Cybersecurity Deception Experimentation System
CIT	Collaborative Innovation Testbed
CORE	Common Open Research Emulator
CyberRIG	Cyber Rapid Innovation Group
DEVCOM	US Army Combat Capabilities Development Command
GUI	graphical user interface
VM	virtual machines

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
1 (PDF)	DEVCOM ARL FCDD RLA ND J ACOSTA