Suricata Fundamentals

Regarded as a potent instrument for Network Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), and Network Security Monitoring (NSM), Suricata represents a cornerstone of network security. This open-source powerhouse, managed and developed by the Open Information Security Foundation (OISF), is a testament to the strength of a community-led, non-profit initiative.

The objective of Suricata is to dissect every iota of network traffic, seeking potential signs of malicious activities. Its strength lies in the ability to not only conduct a sweeping evaluation of our network's condition but also delve into the details of individual application-layer transactions. The key to Suricata's successful operation lies in an intricately designed set of rules. These guidelines direct Suricata's analysis process, identifying potential threats and areas of interest. Equipped to perform at high velocities on both off-the-shelf and specifically designed hardware, Suricata's efficiency is second to none.

Suricata Operation Modes

Suricata operates in four (4) distinct modes:

1. The `Intrusion Detection System (IDS) mode` positions Suricata as a silent observer. In this capacity, Suricata meticulously examines traffic, flagging potential attacks but refraining from any form of intervention. By providing an in-depth view of network activities and accelerating response times, this mode augments network visibility, albeit without offering direct protection.

2. In the `Intrusion Prevention System (IPS) mode`, Suricata adopts a proactive stance. All network traffic must pass through Suricata's stringent checks and is only granted access to the internal network upon Suricata's approval. This mode bolsters security by proactively thwarting attacks before they penetrate our internal network. Deploying Suricata in IPS mode demands an intimate understanding of the network landscape to prevent the inadvertent blocking of legitimate traffic. Furthermore, each rule activation necessitates rigorous testing and validation. While this mode enhances security, the inspection process may introduce latency.

3. The `Intrusion Detection Prevention System (IDPS) mode` brings together the best of both IDS and IPS. While Suricata continues to passively monitor traffic, it possesses the ability to actively transmit RST packets in response to abnormal activities. This mode strikes a balance between active protection and maintaining low latency, crucial for seamless network operations.

4. In its `Network Security Monitoring (NSM) mode`, Suricata transitions into a dedicated logging mechanism, eschewing active or passive traffic analysis or prevention capabilities. It meticulously logs every piece of network information it encounters, providing a valuable wealth of data for retrospective security incident investigations, despite the high volume of data generated.

Suricata Inputs

Regarding Suricata inputs, there are two main categories:

1. `Offline Input:` This involves reading PCAP files for processing previously captured packets in the `LibPCAP` file format. It is not only advantageous for conducting post-mortem data examination but also instrumental when experimenting with various rule sets and configurations.

2. `Live Input`: Live input can be facilitated via `LibPCAP`, where packets are read directly from network interfaces. However, `LibPCAP` is somewhat hamstrung by its performance limitations and lack of load-balancing capabilities. For inline operations, `NFQ` and `AF_PACKET` options are available. `NFQ`, a Linux-specific inline IPS mode, collaborates

? Go to Questions

Table of Contents

Introduction To IDS/IPS

Suricata

Suricata Fundamentals

Suricata Rule Development Part 1

Suricata Rule Development Part 2 (Encrypted Traffic)

Snort

Snort Fundamentals

Snort Rule Development

Zeek

Zeek Fundamentals

Intrusion Detection With Zeek

Skills Assessment

Skills Assessment - Suricata

Skills Assessment - Snort

Skills Assessment - Zeek

with IPTables to divert packets from the kernel space into Suricata for detailed scrutiny. Commonly used inline, `NFQ` necessitates drop rules for Suricata to effectively obstruct packets. Conversely, `AF_PACKET` provides a performance improvement over `LibPCAP` and supports multi-threading. Nevertheless, it's not compatible with older Linux distributions and can't be employed inline if the machine is also tasked with routing packets.

Please note that there are other, less commonly used or more advanced inputs available.

Suricata Outputs

Suricata creates multiple outputs, including logs, alerts, and additional network-related data such as DNS requests and network flows. One of the most critical outputs is `EVE`, a JSON formatted log that records a wide range of event types including alerts, HTTP, DNS, TLS metadata, drop, SMTP metadata, flow, netflow, and more. Tools such as Logstash can easily consume this output, facilitating data analysis.

We might encounter `Unified2` Suricata output, which is essentially a Snort binary alert format, enabling integration with other software that leverages Unified2. Any Unified2 output can be read using Snort's `u2spewfoo` tool, which is a straightforward and effective method to gain insight into the alert data.

> Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, let's SSH into the Target IP using the provided credentials. The vast majority of the commands covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

Configuring Suricata & Custom Rules

Once we've accessed the deployed Suricata instance over SSH, we can get an overview of all the rule files with a simple execution command.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ ls -lah /etc/suricata/rules/
total 27M
drwxr-xr-x 2 root root 4.0K Jun 28 12:10 .
drwxr-xr-x 3 root root 4.0K Jul  4 14:44 ..
-rw-r--r-- 1 root root  31K Jun 27 20:55 3coresec.rules
-rw-r--r-- 1 root root 1.9K Jun 15 05:51 app-layer-events.rules
-rw-r--r-- 1 root root 2.1K Jun 27 20:55 botcc.portgrouped.rules
-rw-r--r-- 1 root root  27K Jun 27 20:55 botcc.rules
-rw-r--r-- 1 root root 109K Jun 27 20:55 ciarmy.rules
-rw-r--r-- 1 root root  12K Jun 27 20:55 compromised.rules
-rw-r--r-- 1 root root  21K Jun 15 05:51 decoder-events.rules
-rw-r--r-- 1 root root  468 Jun 15 05:51 dhcp-events.rules
-rw-r--r-- 1 root root 1.2K Jun 15 05:51 dnp3-events.rules
-rw-r--r-- 1 root root 1.2K Jun 15 05:51 dns-events.rules
-rw-r--r-- 1 root root  32K Jun 27 20:55 drop.rules
-rw-r--r-- 1 root root 2.7K Jun 27 20:55 dshield.rules
-rw-r--r-- 1 root root 365K Jun 27 20:55 emerging-activex.rules
-rw-r--r-- 1 root root 613K Jun 27 20:55 emerging-adware_pup.rules
-rw-r--r-- 1 root root 650K Jun 27 20:55 emerging-attack_response.rules
-rw-r--r-- 1 root root  33K Jun 27 20:55 emerging-chat.rules
-rw-r--r-- 1 root root  19K Jun 27 20:55 emerging-coinminer.rules
-rw-r--r-- 1 root root 119K Jun 27 20:55 emerging-current_events.rules
-rw-r--r-- 1 root root 1.7M Jun 27 20:55 emerging-deleted.rules
-rw-r--r-- 1 root root  20K Jun 27 20:55 emerging-dns.rules
-rw-r--r-- 1 root root  62K Jun 27 20:55 emerging-dos.rules
-rw-r--r-- 1 root root 606K Jun 27 20:55 emerging-exploit_kit.rules
-rw-r--r-- 1 root root 1.1M Jun 27 20:55 emerging-exploit.rules
-rw-r--r-- 1 root root  45K Jun 27 20:55 emerging-ftp.rules
-rw-r--r-- 1 root root  37K Jun 27 20:55 emerging-games.rules
-rw-r--r-- 1 root root 572K Jun 27 20:55 emerging-hunting.rules
-rw-r--r-- 1 root root  18K Jun 27 20:55 emerging-icmp_info.rules
-rw-r--r-- 1 root root  11K Jun 27 20:55 emerging-icmp.rules
-rw-r--r-- 1 root root  15K Jun 27 20:55 emerging-imap.rules
-rw-r--r-- 1 root root  11K Jun 27 20:55 emerging-inappropriate.rules
-rw-r--r-- 1 root root 2.9M Jun 27 20:55 emerging-info.rules
-rw-r--r-- 1 root root  48K Jun 27 20:55 emerging-ja3.rules
-rw-r--r-- 1 root root 8.2M Jun 27 20:55 emerging-malware.rules
---SNIP---
```

The rules can be seen in a straightforward list format and be inspected to understand their functionality, as follows.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ more /etc/suricata/rules/emerging-malware.rules
# Emerging Threats
#
# This distribution may contain rules under two different licenses.
#
#  Rules with sids 1 through 3464, and 100000000 through 100000908 are under the GPLv2.
#  A copy of that license is available at http://www.gnu.org/licenses/gpl-2.0.html
#
#  Rules with sids 2000000 through 2799999 are from Emerging Threats and are covered under the BSD
#  as follows:
#
#*************************************************************
#  Copyright (c) 2003-2022, Emerging Threats
#  All rights reserved.
#
#  Redistribution and use in source and binary forms, with or without modification, are permitted p
#  following conditions are met:
#
#  * Redistributions of source code must retain the above copyright notice, this list of conditions
#    disclaimer.
#  * Redistributions in binary form must reproduce the above copyright notice, this list of conditi
#    following disclaimer in the documentation and/or other materials provided with the distributio
#  * Neither the name of the nor the names of its contributors may be used to endorse or promote pr
#    from this software without specific prior written permission.
#
#  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMP
#  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTI
#  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDI
#  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBS
#  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THE
#  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN AN
#  USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#*************************************************************
#
#
#
#

# This Ruleset is EmergingThreats Open optimized for suricata-5.0-enhanced.

#alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET MALWARE Psyb0t joining an IRC Channel"; flow
JOIN #mipsel"; reference:url,www.adam.com.au/bogaurd/PSYB0T.pdf; reference:url,doc.emergingthreats.
tadata:created_at 2010_07_30, updated_at 2010_07_30;)

alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"ET MALWARE SC-KeyLog Keylogger Installed - Sendin
Installation of SC-KeyLog on host "; nocase; content:"<p>You will receive a log report every "; noc
url,doc.emergingthreats.net/2002979; classtype:trojan-activity; sid:2002979; rev:4; metadata:create

alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"ET MALWARE SC-KeyLog Keylogger Installed - Sendin
eyLog log report"; nocase; content:"See attached file"; nocase; content:".log"; nocase; reference:u
threats.net/2008348; classtype:trojan-activity; sid:2008348; rev:2; metadata:created_at 2010_07_30,
---SNIP---
```

Rules might be commented out, meaning they aren't loaded and don't affect the system. This usually happens when a new version of the rule comes into play or if the threat associated with the rule becomes outdated or irrelevant.

Each rule usually involves specific variables, such as `$HOME_NET` and `$EXTERNAL_NET`. The rule examines traffic from the IP addresses specified in the `$HOME_NET` variable heading towards the IP addresses in the `$EXTERNAL_NET` variable.

These variables can be defined in the `suricata.yaml` configuration file.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ more /etc/suricata/suricata.yaml
%YAML 1.1
---

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://suricata.readthedocs.io/en/latest/configuration/suricata-yaml.html

# This configuration file generated by Suricata 6.0.13.
suricata-version: "6.0"
```

```
##
## Step 1: Inform Suricata about your network
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"

    HTTP_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    TELNET_SERVERS: "$HOME_NET"
    AIM_SERVERS: "$EXTERNAL_NET"
    DC_SERVERS: "$HOME_NET"
    DNP3_SERVER: "$HOME_NET"
---SNIP---
```

This allows us to customize these variables according to our specific network environment and even define our own variables.

Finally, to configure Suricata to load signatures from a custom rules file, such as `local.rules` in the `/home/htb-student` directory, we would execute the below.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo vim /etc/suricata/suricata.yaml
```

1. Add `/home/htb-student/local.rules` to `rule-files:`
2. Press the `Esc` key
3. Enter `:wq` and then, press the `Enter` key

The local.rules file that resides in the /home/htb-student directory of this section's target already contains a Suricata rule. This rule is adequate for this section's learning objectives. We will elaborate more on Suricata rule development in the next section.

Hands-on With Suricata Inputs

With Suricata inputs, we can experiment with both offline and live input:

1. For offline input (reading PCAP files - suspicious.pcap in this case), the following command needs to be executed, and Suricata will create various logs (mainly `eve.json`, `fast.log`, and `stats.log`).

```
MisaelMacias@htb[/htb]$ suricata -r /home/htb-student/pcaps/suspicious.pcap
5/7/2023 -- 13:35:51 - <Notice> - This is Suricata version 6.0.13 RELEASE running in USER mode
5/7/2023 -- 13:35:51 - <Notice> - all 3 packet processing threads, 4 management threads initia
5/7/2023 -- 13:35:51 - <Notice> - Signal Received.  Stopping engine.
5/7/2023 -- 13:35:51 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
```

An alternative command can be executed to bypass checksums (`-k` flag) and log in a different directory (`-l` flag).

```
MisaelMacias@htb[/htb]$ suricata -r /home/htb-student/pcaps/suspicious.pcap -k none -l .
5/7/2023 -- 13:37:43 - <Notice> - This is Suricata version 6.0.13 RELEASE running in USER mode
```

```
5/7/2023 -- 13:37:43 - <Notice> - all 3 packet processing threads, 4 management threads initia
5/7/2023 -- 13:37:43 - <Notice> - Signal Received.  Stopping engine.
5/7/2023 -- 13:37:43 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
```

2. For live input, we can try Suricata's (Live) `LibPCAP` mode as follows.

```
MisaelMacias@htb[/htb]$ ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.129.205.193  netmask 255.255.0.0  broadcast 10.129.255.255
    inet6 dead:beef::250:56ff:feb9:68dc  prefixlen 64  scopeid 0x0<global>
    inet6 fe80::250:56ff:feb9:68dc  prefixlen 64  scopeid 0x20<link>
    ether 00:50:56:b9:68:dc  txqueuelen 1000  (Ethernet)
    RX packets 281625  bytes 84557478 (84.5 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 62276  bytes 23518127 (23.5 MB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop  txqueuelen 1000  (Local Loopback)
    RX packets 888  bytes 64466 (64.4 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 888  bytes 64466 (64.4 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
MisaelMacias@htb[/htb]$ sudo suricata --pcap=ens160 -vv
[sudo] password for htb-student:
5/7/2023 -- 13:44:01 - <Notice> - This is Suricata version 6.0.13 RELEASE running in SYSTEM mo
5/7/2023 -- 13:44:01 - <Info> - CPUs/cores online: 2
5/7/2023 -- 13:44:01 - <Info> - Setting engine mode to IDS mode by default
5/7/2023 -- 13:44:01 - <Info> - Found an MTU of 1500 for 'ens160'
5/7/2023 -- 13:44:01 - <Info> - Found an MTU of 1500 for 'ens160'
5/7/2023 -- 13:44:01 - <Info> - fast output device (regular) initialized: fast.log
5/7/2023 -- 13:44:01 - <Info> - eve-log output device (regular) initialized: eve.json
5/7/2023 -- 13:44:01 - <Info> - stats output device (regular) initialized: stats.log
5/7/2023 -- 13:44:01 - <Info> - Running in live mode, activating unix socket
5/7/2023 -- 13:44:01 - <Perf> - using shared mpm ctx' for http_uri
5/7/2023 -- 13:44:01 - <Perf> - using shared mpm ctx' for http_uri
5/7/2023 -- 13:44:01 - <Perf> - using shared mpm ctx' for http_raw_uri
5/7/2023 -- 13:44:01 - <Perf> - using shared mpm ctx' for http_raw_uri
5/7/2023 -- 13:44:01 - <Info> - 1 rule files processed. 1 rules successfully loaded, 0 rules fa
5/7/2023 -- 13:44:01 - <Info> - Threshold config parsed: 0 rule(s) found
5/7/2023 -- 13:44:01 - <Perf> - using shared mpm ctx' for tcp-packet
5/7/2023 -- 13:44:01 - <Perf> - using shared mpm ctx' for tcp-stream
5/7/2023 -- 13:44:01 - <Perf> - using shared mpm ctx' for udp-packet
5/7/2023 -- 13:44:01 - <Perf> - using shared mpm ctx' for other-ip
5/7/2023 -- 13:44:01 - <Info> - 1 signatures processed. 0 are IP-only rules, 0 are inspecting |
5/7/2023 -- 13:44:01 - <Perf> - TCP toserver: 1 port groups, 1 unique SGH's, 0 copies
5/7/2023 -- 13:44:01 - <Perf> - TCP toclient: 0 port groups, 0 unique SGH's, 0 copies
5/7/2023 -- 13:44:01 - <Perf> - UDP toserver: 1 port groups, 1 unique SGH's, 0 copies
5/7/2023 -- 13:44:01 - <Perf> - UDP toclient: 0 port groups, 0 unique SGH's, 0 copies
5/7/2023 -- 13:44:01 - <Perf> - OTHER toserver: 0 proto groups, 0 unique SGH's, 0 copies
5/7/2023 -- 13:44:01 - <Perf> - OTHER toclient: 0 proto groups, 0 unique SGH's, 0 copies
5/7/2023 -- 13:44:01 - <Perf> - Unique rule groups: 2
5/7/2023 -- 13:44:01 - <Perf> - Builtin MPM "toserver TCP packet": 0
5/7/2023 -- 13:44:01 - <Perf> - Builtin MPM "toclient TCP packet": 0
5/7/2023 -- 13:44:01 - <Perf> - Builtin MPM "toserver TCP stream": 0
5/7/2023 -- 13:44:01 - <Perf> - Builtin MPM "toclient TCP stream": 0
5/7/2023 -- 13:44:01 - <Perf> - Builtin MPM "toserver UDP packet": 0
5/7/2023 -- 13:44:01 - <Perf> - Builtin MPM "toclient UDP packet": 0
5/7/2023 -- 13:44:01 - <Perf> - Builtin MPM "other IP packet": 0
5/7/2023 -- 13:44:01 - <Perf> - AppLayer MPM "toserver dns_query (dns)": 1
5/7/2023 -- 13:44:01 - <Info> - Using 1 live device(s).
5/7/2023 -- 13:44:01 - <Info> - using interface ens160
5/7/2023 -- 13:44:01 - <Perf> - ens160: disabling rxcsum offloading
5/7/2023 -- 13:44:01 - <Perf> - ens160: disabling txcsum offloading
5/7/2023 -- 13:44:01 - <Info> - running in 'auto' checksum mode. Detection of interface state
5/7/2023 -- 13:44:01 - <Info> - Found an MTU of 1500 for 'ens160'
5/7/2023 -- 13:44:01 - <Info> - Set snaplen to 1524 for 'ens160'
5/7/2023 -- 13:44:01 - <Perf> - NIC offloading on ens160: RX unset TX unset
5/7/2023 -- 13:44:01 - <Perf> - NIC offloading on ens160: SG: unset, GRO: unset, LRO: unset, T
5/7/2023 -- 13:44:01 - <Info> - RunModeIdsPcapAutoFp initialised
5/7/2023 -- 13:44:01 - <Info> - Running in live mode, activating unix socket
```

```
5/7/2023 -- 13:44:01 - <Info> - Using unix socket file '/var/run/suricata/suricata-command.soc
5/7/2023 -- 13:44:01 - <Notice> - all 3 packet processing threads, 4 management threads initia
```

3. For Suricata in Inline (`NFQ`) mode, the following command should be executed first.

```
MisaelMacias@htb[/htb]$ sudo iptables -I FORWARD -j NFQUEUE
```

Then, we should be able to execute the following.

```
MisaelMacias@htb[/htb]$ sudo suricata -q 0
5/7/2023 -- 13:52:38 - <Notice> - This is Suricata version 6.0.13 RELEASE running in SYSTEM mo
5/7/2023 -- 13:52:39 - <Notice> - all 4 packet processing threads, 4 management threads initia
```

Moreover, to try Suricata in `IDS` mode with `AF_PACKET` input, execute one of the below.

```
MisaelMacias@htb[/htb]$ sudo suricata -i ens160
5/7/2023 -- 13:53:35 - <Notice> - This is Suricata version 6.0.13 RELEASE running in SYSTEM mo
5/7/2023 -- 13:53:35 - <Notice> - all 1 packet processing threads, 4 management threads initia
```

```
MisaelMacias@htb[/htb]$ sudo suricata --af-packet=ens160
5/7/2023 -- 13:54:34 - <Notice> - This is Suricata version 6.0.13 RELEASE running in SYSTEM mo
5/7/2023 -- 13:54:34 - <Notice> - all 1 packet processing threads, 4 management threads initia
```

To observe Suricata dealing with "live" traffic, let's establish an additional SSH connection and utilize `tcpreplay` to replay network traffic from a PCAP file (`suspicious.pcap` in this case).

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo  tcpreplay -i ens160 /home/htb-student/pcaps/suspicious.pcap
^C User interrupt...
sendpacket_abort
Actual: 730 packets (663801 bytes) sent in 22.84 seconds
Rated: 29060.3 Bps, 0.232 Mbps, 31.95 pps
Statistics for network device: ens160
        Successful packets:        729
        Failed packets:            0
        Truncated packets:         0
        Retried packets (ENOBUFS): 0
        Retried packets (EAGAIN):  0
```

Then, feel free to terminate both tcpreplay and Suricata. The logs from the observed (replayed) traffic will be available at `/var/log/suricata`

> The -i option helps Suricata choose the best input option. In the case of Linux, the best input option is AF_PACKET. If pcap mode is needed, the --pcap option is recommended. Configuration of (Live) LibPCAP can be achieved via the suricata.yaml file, including settings for buffer size, BPF or tcpdump filters, checksum validation, threads, promiscuous mode, snap length, etc.

Hands-on With Suricata Outputs

Suricata records a variety of data into logs that reside in the `/var/log/suricata` directory by default. For us to access

and manipulate these logs, we require root-level access. Among these logs, we find the `eve.json`, `fast.log`, and `stats.log` files, which provide invaluable insight into the network activity. Let's delve into each:

1. `eve.json`: This file is Suricata's recommended output and contains JSON objects, each carrying diverse information such as timestamps, flow_id, event_type, and more. Try inspecting the content of `old_eve.json` residing at `/var/log/suricata` as follows.

```
MisaelMacias@htb[/htb]$ less /var/log/suricata/old_eve.json
{"timestamp":"2023-07-06T08:34:24.526482+0000","event_type":"stats","stats":{"uptime":8,"captu
---SNIP---
```

If we wish to filter out only alert events, for example, we can utilize the `jq` command-line JSON processor as follows.

```
MisaelMacias@htb[/htb]$ cat /var/log/suricata/old_eve.json | jq -c 'select(.event_type == "ale
{"timestamp":"2023-07-06T08:34:35.003163+0000","flow_id":1959965318909019,"in_iface":"ens160",
```

If we wish to identify the earliest DNS event, for example, we can utilize the `jq` command-line JSON processor as follows.

```
MisaelMacias@htb[/htb]$ cat /var/log/suricata/old_eve.json | jq -c 'select(.event_type == "dns
{
 "timestamp": "2023-07-06T08:34:35.003163+0000",
 "flow_id": 1959965318909019,
 "in_iface": "ens160",
  "event_type": "dns",
  "src_ip": "10.9.24.101",
 "src_port": 51833,
 "dest_ip": "10.9.24.1",
 "dest_port": 53,
 "proto": "UDP",
 "dns": {
  "type": "query",
  "id": 6430,
     "rrname": "adv.epostoday.uk",
     "rrtype": "A",
     "tx_id": 0,
    "opcode": 0
 }
}
```

We can also use similar commands to filter for other event types like TLS and SSH.

`flow_id`: This is a unique identifier assigned by Suricata to each network flow. A flow, in Suricata terms, is defined as a set of IP packets passing through a network interface in a specific direction and between a given pair of source and destination endpoints. Each of these flows gets a unique flow_id. This identifier helps us track and correlate various events related to the same network flow in the EVE JSON log. Using `flow_id`, we can associate different pieces of information related to the same flow, such as alerts, network transactions, and packets, providing a cohesive view of what is happening on a specific communication channel.

`pcap_cnt`: This is a counter that Suricata increments for each packet it processes from the network traffic or from a PCAP file (in offline mode). `pcap_cnt` allows us to trace a packet back to its original order in the PCAP file or network stream. This is beneficial in understanding the sequence of network events as they occurred. It can help to precisely pinpoint when an alert was triggered in relation to other packets, which can provide valuable context in an investigation.

2. `fast.log`: This is a text-based log format that records alerts only and is enabled by default. Try inspecting the

content of `old_fast.log` residing at `/var/log/suricata` as follows.

```
MisaelMacias@htb[/htb]$ cat /var/log/suricata/old_fast.log
07/06/2023-08:34:35.003163  [**] [1:1:0] Known bad DNS lookup, possible Dridex infection [**]
```

3. `stats.log`: This is a human-readable statistics log, which can be particularly useful while debugging Suricata deployments. Try inspecting the content of `old_stats.log` residing at `/var/log/suricata` as follows.

```
MisaelMacias@htb[/htb]$ cat /var/log/suricata/old_stats.log
-------------------------------------------------------------------------------
Date: 7/6/2023 -- 08:34:24 (uptime: 0d, 00h 00m 08s)
-------------------------------------------------------------------------------
Counter                                     | TM Name                   | Value
-------------------------------------------------------------------------------
capture.kernel_packets                      | Total                     | 4
decoder.pkts                                | Total                     | 3
decoder.bytes                               | Total                     | 212
decoder.ipv6                                | Total                     | 1
decoder.ethernet                            | Total                     | 3
decoder.icmpv6                              | Total                     | 1
decoder.avg_pkt_size                        | Total                     | 70
decoder.max_pkt_size                        | Total                     | 110
flow.icmpv6                                 | Total                     | 1
flow.wrk.spare_sync_avg                     | Total                     | 100
flow.wrk.spare_sync                         | Total                     | 1
flow.mgr.full_hash_pass                     | Total                     | 1
flow.spare                                  | Total                     | 9900
tcp.memuse                                  | Total                     | 606208
tcp.reassembly_memuse                       | Total                     | 98304
flow.memuse                                 | Total                     | 7394304
-------------------------------------------------------------------------------
---SNIP---
```

For those of us who want a more focused output strategy, there's an option to deactivate the comprehensive `EVE` output and activate particular outputs instead. Take `http-log`, for instance. By activating this, every time Suricata is run and encounters HTTP events, a fresh `http.log` file will be generated.

Hands-on With Suricata Outputs - File Extraction

Suricata, has an underused yet highly potent feature - file extraction. This feature allows us to capture and store files transferred over a number of different protocols, providing invaluable data for threat hunting, forensics, or simply data analysis.

Here's how we go about enabling file extraction in Suricata.

We start by making changes to the Suricata configuration file (`suricata.yaml`). In this file, we'll find a section named `file-store`. This is where we tell Suricata how to handle the files it extracts. Specifically, we need to set `version` to `2`, enabled to `yes`, and the `force-filestore` option also to `yes`. The resulting section should look something like this.

Suricata Fundamentals

```
file-store:
  version: 2
  enabled: yes
  force-filestore: yes
```

In the same `file-store` section, we define where Suricata stores the extracted files. We set the `dir` option to the directory of our choice.

As a quick exercise, let's enable file extraction and run Suricata on the `/home/htb-student/pcaps/vm-2.pcap` file from www.netresec.com.

In accordance with the guidelines put forth in Suricata's documentation, file extraction isn't an automatic process that occurs without our explicit instructions. It's fundamentally crucial for us to craft a specific rule that instructs Suricata when and what kind of files it should extract.

The simplest rule we can add to our `local.rules` file to experiment with file extraction is the following.

Suricata Fundamentals

```
alert http any any -> any any (msg:"FILE store all"; filestore; sid:2; rev:1;)
```

If we configured Suricata correctly, multiple files will be stored inside the `filestore` directory.

Let's run Suricata on the `/home/htb-student/pcaps/vm-2.pcap` file.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ suricata -r /home/htb-student/pcaps/vm-2.pcap
7/7/2023 -- 06:25:57 - <Notice> - This is Suricata version 6.0.13 RELEASE running in USER mode
7/7/2023 -- 06:25:57 - <Notice> - all 3 packet processing threads, 4 management threads initialized
7/7/2023 -- 06:25:57 - <Notice> - Signal Received.  Stopping engine.
7/7/2023 -- 06:25:57 - <Notice> - Pcap-file module read 1 files, 803 packets, 683915 bytes
```

We will notice that `eve.json`, `fast.log`, `stats.log`, and `suricata.log` were created, alongside a new directory called `filestore`. `filestore`'s content in terms of the files it contains can be inspected as follows.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ cd filestore
MisaelMacias@htb[/htb]$ find . -type f
./fb/fb20d18d00c806deafe14859052072aecfb9f46be6210acfce80289740f2e20e
./21/214306c98a3483048d6a69eec6bf3b50497363bc2c98ed3cd954203ec52455e5
./21/21742fc621f83041db2e47b0899f5aea6caa00a4b67dbff0aae823e6817c5433
./26/2694f69c4abf2471e09f6263f66eb675a0ca6ce58050647dcdcfebaf69f11ff4
./2c/2ca1a0cd9d8727279f0ba99fd051e1c0acd621448ad4362e1c9fc78700015228
./7d/7d4c00f96f38e0ffd89bc2d69005c4212ef577354cc97d632a09f51b2d37f877
./6b/6b7fee8a4b813b6405361db2e70a4f5a213b34875dd2793667519117d8ca0e4e
./2e/2e2cb2cac099f08bc51abba263d9e3f8ac7176b54039cc30bbd4a45cfa769018
./50/508c47dd306da3084475faae17b3acd5ff2700d2cd85d71428cdfaae28c9fd41
./c2/c210f737f55716a089a33daf42658afe771cfb43228ffa405d338555a9918815
./ea/ea0936257b8d96ee6ae443adee0f3dacc3eff72b559cd5ee3f9d6763cf5ee2ab
./1a/1aab7d9c153887dfa63853534f684e5d46ecd17ba60cd3d61050f7f231c4babb
./c4/c4775e980c97b162fd15e0010663694c4e09f049ff701d9671e1578958388b9f
./63/63de4512dfbd0087f929b0e070cc90d534d6baabf2cdfbeaf76bee24ff9b1638
./48/482d9972c2152ca96616dc23bbaace55804c9d52f5d8b253b617919bb773d3bb
./8e/8ea3146c676ba436c0392c3ec26ee744155af4e4eca65f4e99ec68574a747a14
./8e/8e23160cc504b4551a94943e677f6985fa331659a1ba58ef01afb76574d2ad7c
./a5/a52dac473b33c22112a6f53c6a625f39fe0d6642eb436e5d125342a24de44581
```

Again in accordance with the guidelines put forth in Suricata's documentation the `file-store` module uses its own log directory (default: `filestore` in the default logging directory) and logs files using the SHA256 of the contents as the filename. Each file is then placed in a directory named 00 to ff where the directory shares the first 2 characters of the filename. For example, if the SHA256 hex string of an extracted file starts with `f9bc6d...` the file we be placed in the directory `filestore/f9`.

If we wanted to inspect, for example, the `/21/21742fc621f83041db2e47b0899f5aea6caa00a4b67dbff0aae823e6817c5433` file inside the `filestore` directory, we could use the `xxd` tool as follows.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ cd filestore
MisaelMacias@htb[/htb]$ xxd ./21/21742fc621f83041db2e47b0899f5aea6caa00a4b67dbff0aae823e6817c5433 |
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000  MZ..............
00000010: b800 0000 0000 0000 4000 0000 e907 0000  ........@.......
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 8000 0000  ................
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468  ........!..L.!Th
00000050: 6973 2070 726f 6772 616d 2063 616e 6e6f  is program canno
```

```
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320  t be run in DOS
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000  mode....$.......
00000080: 5045 0000 4c01 0300 fc90 8448 0000 0000  PE..L......H....
00000090: 0000 0000 e000 0f01 0b01 0600 00d0 0000  ................
```

In this case, the file was a Windows executable based on the file's header. More about the MS-DOS EXE format can be found in following resource MZ.

Live Rule Reloading Feature & Updating Suricata Rulesets

Live rule reloading is a crucial feature in Suricata that allows us to update our ruleset without interrupting ongoing traffic inspection. This feature provides continuous monitoring and minimizes the chances of missing any malicious activity.

To enable live rule reloading in Suricata, we need to configure our Suricata configuration file (`suricata.yaml`). In the `suricata.yaml` file, we should locate the `detect-engine` section and set the value of the `reload` parameter to `true`. It looks something like this:

Suricata Fundamentals

```
detect-engine:
  - reload: true
```

Proceed to execute the following `kill` command, which will signal the Suricata process (determined by `$(pidof suricata)`) to refresh its rule set without necessitating a complete restart.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo kill -usr2 $(pidof suricata)
```

This modification tells Suricata to check for changes in the ruleset periodically and apply them without needing to restart the service.

> Most of the commands below cannot be replicated inside this section's target since they require internet connectivity.

Updating Suricata's ruleset can be performed using the `suricata-update` tool. We can perform a simple update to the Suricata ruleset using the following command.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo suricata-update
6/7/2023 -- 06:46:44 - <Info> -- Using data-directory /var/lib/suricata.
6/7/2023 -- 06:46:44 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
6/7/2023 -- 06:46:44 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
6/7/2023 -- 06:46:44 - <Info> -- Found Suricata version 6.0.13 at /usr/bin/suricata.
6/7/2023 -- 06:46:44 - <Info> -- Loading /etc/suricata/suricata.yaml
6/7/2023 -- 06:46:44 - <Info> -- Disabling rules for protocol http2
6/7/2023 -- 06:46:44 - <Info> -- Disabling rules for protocol modbus
6/7/2023 -- 06:46:44 - <Info> -- Disabling rules for protocol dnp3
6/7/2023 -- 06:46:44 - <Info> -- Disabling rules for protocol enip
6/7/2023 -- 06:46:44 - <Info> -- No sources configured, will use Emerging Threats Open
6/7/2023 -- 06:46:44 - <Info> -- Fetching https://rules.emergingthreats.net/open/suricata-6.0.13/em
 100% - 3963342/3963342
6/7/2023 -- 06:46:45 - <Info> -- Done.
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/app-layer-event
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/decoder-events.
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/dhcp-events.rul
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/dnp3-events.rul
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/dns-events.rule
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/files.rules
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/http-events.rul
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/ipsec-events.ru
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/kerberos-events
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/modbus-events.r
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/nfs-events.rule
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/ntp-events.rule
```

```
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/smb-events.rule
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/smtp-events.rul
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/stream-events.r
6/7/2023 -- 06:46:45 - <Info> -- Loading distribution rule file /etc/suricata/rules/tls-events.rule
6/7/2023 -- 06:46:45 - <Info> -- Ignoring file rules/emerging-deleted.rules
6/7/2023 -- 06:46:48 - <Info> -- Loaded 43453 rules.
6/7/2023 -- 06:46:48 - <Info> -- Disabled 14 rules.
6/7/2023 -- 06:46:48 - <Info> -- Enabled 0 rules.
6/7/2023 -- 06:46:48 - <Info> -- Modified 0 rules.
6/7/2023 -- 06:46:48 - <Info> -- Dropped 0 rules.
6/7/2023 -- 06:46:48 - <Info> -- Enabled 131 rules for flowbit dependencies.
6/7/2023 -- 06:46:48 - <Info> -- Creating directory /var/lib/suricata/rules.
6/7/2023 -- 06:46:48 - <Info> -- Backing up current rules.
6/7/2023 -- 06:46:49 - <Info> -- Writing rules to /var/lib/suricata/rules/suricata.rules: total: 43
6/7/2023 -- 06:46:49 - <Info> -- Writing /var/lib/suricata/rules/classification.config
6/7/2023 -- 06:46:49 - <Info> -- Testing with suricata -T.
6/7/2023 -- 06:47:11 - <Info> -- Done.
```

As displayed in the example above, the output indicates that the `suricata-update` command has successfully retrieved the rules by establishing a connection with `https://rules.emergingthreats.net/open/`. Subsequently, the command saves the newly obtained rules to the `/var/lib/suricata/rules/` directory.

Moving forward, let's execute the command provided below to generate a comprehensive list of all ruleset providers.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo suricata-update list-sources
6/7/2023 -- 06:59:29 - <Info> -- Using data-directory /var/lib/suricata.
6/7/2023 -- 06:59:29 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
6/7/2023 -- 06:59:29 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
6/7/2023 -- 06:59:29 - <Info> -- Found Suricata version 6.0.13 at /usr/bin/suricata.
6/7/2023 -- 06:59:29 - <Info> -- No source index found, running update-sources
6/7/2023 -- 06:59:29 - <Info> -- Downloading https://www.openinfosecfoundation.org/rules/index.yaml
6/7/2023 -- 06:59:29 - <Info> -- Adding all sources
6/7/2023 -- 06:59:29 - <Info> -- Saved /var/lib/suricata/update/cache/index.yaml
Name: et/open
  Vendor: Proofpoint
  Summary: Emerging Threats Open Ruleset
  License: MIT
Name: et/pro
  Vendor: Proofpoint
  Summary: Emerging Threats Pro Ruleset
  License: Commercial
  Replaces: et/open
  Parameters: secret-code
  Subscription: https://www.proofpoint.com/us/threat-insight/et-pro-ruleset
Name: oisf/trafficid
  Vendor: OISF
  Summary: Suricata Traffic ID ruleset
  License: MIT
Name: scwx/enhanced
  Vendor: Secureworks
  Summary: Secureworks suricata-enhanced ruleset
  License: Commercial
  Parameters: secret-code
  Subscription: https://www.secureworks.com/contact/ (Please reference CTU Countermeasures)
Name: scwx/malware
  Vendor: Secureworks
  Summary: Secureworks suricata-malware ruleset
  License: Commercial
  Parameters: secret-code
  Subscription: https://www.secureworks.com/contact/ (Please reference CTU Countermeasures)
Name: scwx/security
  Vendor: Secureworks
  Summary: Secureworks suricata-security ruleset
  License: Commercial
  Parameters: secret-code
  Subscription: https://www.secureworks.com/contact/ (Please reference CTU Countermeasures)
Name: sslbl/ssl-fp-blacklist
  Vendor: Abuse.ch
  Summary: Abuse.ch SSL Blacklist
  License: Non-Commercial
Name: sslbl/ja3-fingerprints
  Vendor: Abuse.ch
  Summary: Abuse.ch Suricata JA3 Fingerprint Ruleset
  License: Non-Commercial
Name: etnetera/aggressive
  Vendor: Etnetera a.s.
  Summary: Etnetera aggressive IP blacklist
  License: MIT
```

```
  License: MIT
Name: tgreen/hunting
  Vendor: tgreen
  Summary: Threat hunting rules
  License: GPLv3
Name: malsilo/win-malware
  Vendor: malsilo
  Summary: Commodity malware rules
  License: MIT
Name: stamus/lateral
  Vendor: Stamus Networks
  Summary: Lateral movement rules
  License: GPL-3.0-only
```

Let's make a note from the output above of a specific ruleset name from which we want Suricata to fetch rulesets. The `et/open` rulesets serve as an excellent choice for demonstration purposes.

Next, let's proceed with executing the following command to retrieve and enable the `et/open` rulesets within our Suricata rules.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo suricata-update enable-source et/open
6/7/2023 -- 07:02:08 - <Info> -- Using data-directory /var/lib/suricata.
6/7/2023 -- 07:02:08 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
6/7/2023 -- 07:02:08 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
6/7/2023 -- 07:02:08 - <Info> -- Found Suricata version 6.0.13 at /usr/bin/suricata.
6/7/2023 -- 07:02:08 - <Info> -- Creating directory /var/lib/suricata/update/sources
6/7/2023 -- 07:02:08 - <Info> -- Source et/open enabled
```

Lastly, let's reissue the `suricata-update` command to load the newly acquired ruleset.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo suricata-update
```

A Suricata service restart may also be required.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo systemctl restart suricata
```

Validating Suricata's Configuration

Validation of Suricata's configuration is also an essential part of maintaining the robustness of our IDS/IPS setup. To validate the configuration, we can use the `-T` option provided by the Suricata command. This command runs a test to check if the configuration file is valid and all files referenced in the configuration are accessible.

Here is how we can do this.

Suricata Fundamentals

```
MisaelMacias@htb[/htb]$ sudo suricata -T -c /etc/suricata/suricata.yaml
6/7/2023 -- 07:13:29 - <Info> - Running suricata under test mode
6/7/2023 -- 07:13:29 - <Notice> - This is Suricata version 6.0.13 RELEASE running in SYSTEM mode
6/7/2023 -- 07:13:29 - <Notice> - Configuration provided was successfully loaded. Exiting.
```

Suricata Documentation

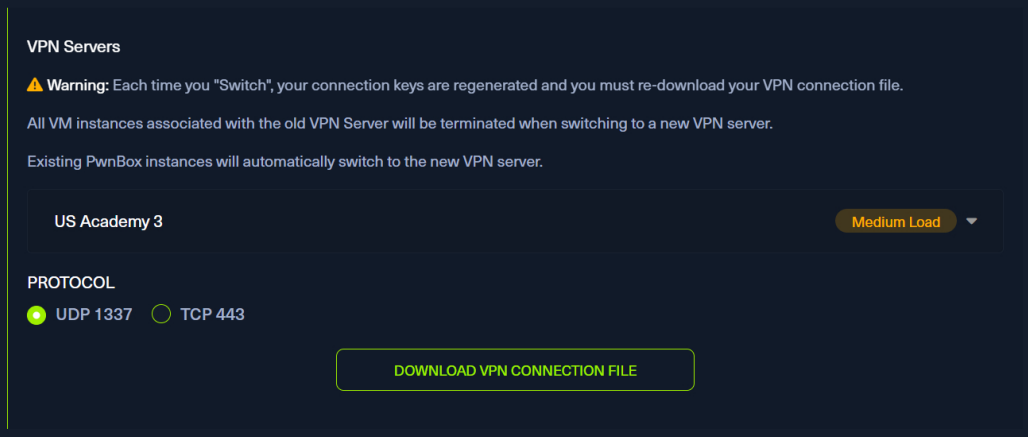
Suricata is an incredibly versatile tool with extensive functionality. We highly recommend exploring Suricata's documentation to gain a deeper understanding of its capabilities.

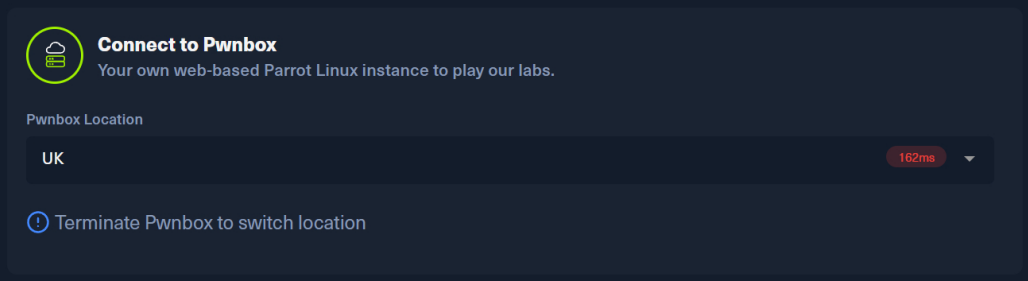
Suricata Key Features

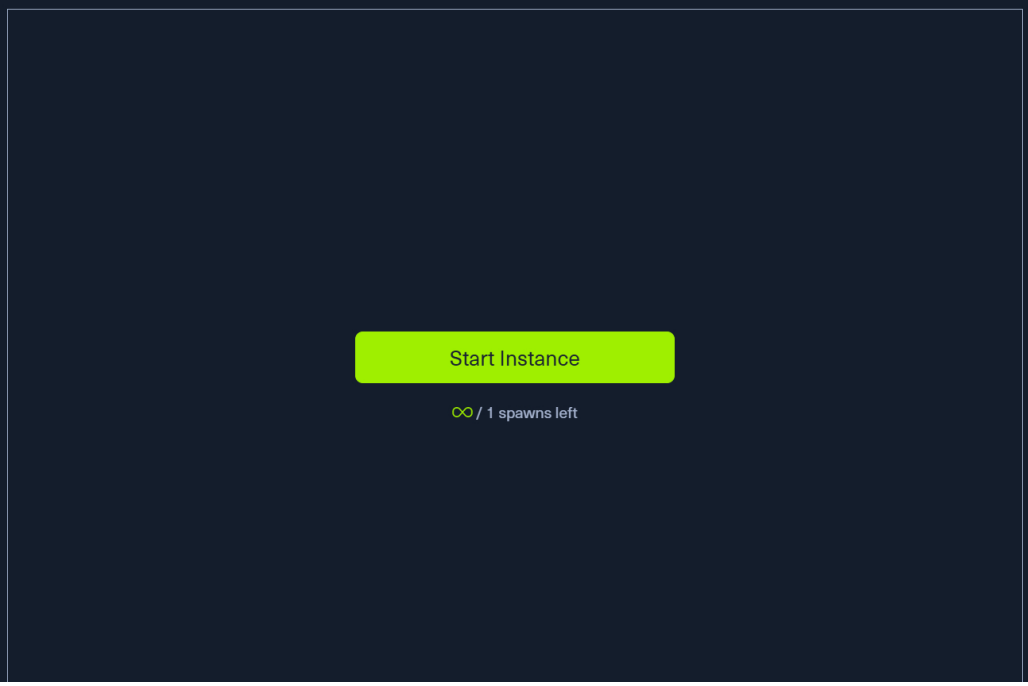
Key features that bolster Suricata's effectiveness include:

- Deep packet inspection and packet capture logging
- Anomaly detection and Network Security Monitoring
- Intrusion Detection and Prevention, with a hybrid mode available
- Lua scripting
- Geographic IP identification (GeoIP)
- Full IPv4 and IPv6 support
- IP reputation
- File extraction
- Advanced protocol inspection
- Multitenancy

Note: Suricata can also be used to detect "non-standard/anomalous" traffic. We can leverage strategies outlined in Suricata's Protocol Anomalies Detection page. This approach enhances our visibility into unusual or non-compliant behavior within our network, thus augmenting our security posture.







Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Download VPN Connection File

Target(s): Click here to spawn the target system!

SSH to with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 2 Filter out only HTTP events from /var/log/suricata/old_eve.json using the the jq command-line JSON processor. Enter the flow_id that you will come across as your answer.

1252204100696793

Submit

SSH to with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 2 Enable the http-log output in suricata.yaml and run Suricata against /home/htb-student/pcaps/suspicious.pcap. Enter the requested PHP page as your answer. Answer format: _.php

app.php

Submit

Previous Next

Mark Complete & Next