Suricata Rule Development Part 1

At its core, a rule in Suricata serves as a directive, instructing the engine to actively watch for certain markers in the network traffic. When such specific markers appear, we will receive a notification.

Suricata rules are not exclusively focused on the detection of nefarious activities or potentially harmful traffic. In many instances, rules can be designed to furnish network defenders or blue team members with critical insights or contextual data regarding ongoing network activity.

The specificity or generality of the rules is in our hands. Striking a balance is paramount to, say, identify variations of a certain malware strain while evading false positives.

The development of these rules often leverages crucial information provided by the infosec communities and threat intelligence. However, it's worth noting that each rule we deploy consumes a portion of the host's CPU and memory resources. Hence, Suricata provides specific guidelines for writing effective rules.

Suricata Rule Anatomy

A sample Suricata rule can be found below. Let's break it down.

Suricata Rule Development Part 1

```
action protocol from_ip port -> to_ip port (msg:"Known malicious behavior, possible X malware infec
```

- **Header** (`action protocol from_ip port -> to_ip port` part): The `header` section of a rule encapsulates the intended action of the rule, along with the protocol where the rule is expected to be applied. Additionally, it includes IP addresses, port information, and an arrow indicating traffic directionality.
  - `action` instructs Suricata on what steps to take if the contents match. This could range from generating an alert (`alert`), logging the traffic without an alert (`log`), ignoring the packet (`pass`), dropping the packet in IPS mode (`drop`), or sending TCP RST packets (`reject`).
  - `protocol` can vary, including `tcp`, `udp`, `icmp`, `ip`, `http`, `tls`, `smb`, or `dns`.
  - Traffic directionality is declared using `rule host variables` (such as `$HOME_NET`, `$EXTERNAL_NET`, etc. that we saw inside `suricata.yaml`) and `rule direction`. The direction arrow between the two IP-Port pairs informs Suricata about the traffic flow.
    - Examples:
      - Outbound: `$HOME_NET any -> $EXTERNAL_NET 9090`
      - Inbound: `$EXTERNAL_NET any -> $HOME_NET 8443`
      - Bidirectional: `$EXTERNAL_NET any <> $HOME_NET any`
  - `Rule ports` define the ports at which the traffic for this rule will be evaluated.
    - Examples:
      - `alert tcp $HOME_NET any -> $EXTERNAL_NET 9443`
      - `alert tcp $HOME_NET any -> $EXTERNAL_NET $UNCOMMON_PORTS`
        - $UNCOMMON_PORTS can be defined inside `suricata.yaml`
      - `alert tcp $HOME_NET any -> $EXTERNAL_NET [8443,8080,7001:7002,!8443]`
- **Rule message & content** (`(msg:"Known malicious behavior, possible X malware infection"; content:"some thing"; content:"some other thing";` part): The `rule message & content` section contains the message we wish to be displayed to the analysts or ourselves when an activity we want to be notified about is

? 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

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

detected. content are the segments of the traffic that we deem essential for such detections.

- Rule message (msg) is an arbitrary text displayed when the rule is triggered. Ideally, the rule messages we create should contain details about malware architecture, family, and action.
  - flow identifies the originator and responder. Always remember, when crafting rules, to have the engine monitor "established" tcp sessions.
    - Examples:
      - alert tcp any any -> 192.168.1.0/24 22 (msg:"SSH connection attempt"; flow:to_server; sid:1001;)
      - alert udp 10.0.0.0/24 any -> any 53 (msg:"DNS query"; flow:from_client; sid:1002;)
      - alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"Potential HTTP-based attack"; flow:established,to_server; sid:1003;)
  - dsize matches using the payload size of the packet. It relies on TCP segment length, not the total packet length.
    - Example: alert http any any -> any any (msg:"Large HTTP response"; dsize:>10000; content:"HTTP/1.1 200 OK"; sid:2003;)
- Rule content comprises unique values that help identify specific network traffic or activities. Suricata matches these unique content values in packets for detection.
  - Example: content:"User-Agent|3a 20|Go-http-client/1.1|0d 0a|Accept-Encoding|3a 20|gzip";
    - |3a 20|: This represents the hexadecimal representation of the characters ":", followed by a space character. It is used to match the exact byte sequence in the packet payload.
    - |0d 0a|: This represents the hexadecimal representation of the characters "\r\n", which signifies the end of a line in HTTP headers.
  - By using Rule Buffers, we don't have to search the entire packet for every content match. This saves time and resources. More details can be found here: https://suricata.readthedocs.io/en/latest/rules/http-keywords.html
    - Example: alert http any any -> any any (http.accept; content:"image/gif"; sid:1;)
      - http.accept: Sticky buffer to match on the HTTP Accept header. Only contains the header value. The \r\n after the header are not part of the buffer.
  - Rule options act as additional modifiers to aid detection, helping Suricata locate the exact location of contents.
    - nocase ensures rules are not bypassed through case changes.
      - Example: alert tcp any any -> any any (msg:"Detect HTTP traffic with user agent Mozilla"; content:"User-Agent: Mozilla"; nocase; sid:8001;)
    - offset informs Suricata about the start position inside the packet for matching.
      - Example: alert tcp any any -> any any (msg:"Detect specific protocol command"; content:"|01 02 03|"; offset:0; depth:5; sid:3003;)
        - This rule triggers an alert when Suricata detects a specific protocol command represented by the byte sequence |01 02 03| in the TCP payload. The offset:0 keyword sets the content match to start from the beginning of the payload, and depth:5 specifies a length of five bytes to be considered for matching.
    - distance tells Suricata to look for the specified content n bytes relative to the previous match.
      - Example: alert tcp any any -> any any (msg:"Detect suspicious URL path"; content:"/admin"; offset:4; depth:10; distance:20; within:50; sid:3001;)
        - This rule triggers an alert when Suricata detects the string /admin in the TCP payload, starting from the fifth byte (offset:4) and

in the TCP payload, starting from the fifth byte (`offset:4`) and considering a length of ten bytes (`depth:10`). The `distance:20` keyword specifies that subsequent matches of `/admin` should not occur within the next 20 bytes after a previous match. The `within:50` keyword ensures that the content match occurs within the next 50 bytes after a previous match.

- **Rule metadata** (`sid:10000001; rev:1;` part):
  - `reference` provides us with a lead, a trail that takes us back to the original source of information that inspired the creation of the rule.
  - `sid` (signature ID). The unique quality of this numeric identifier makes it essential for the rule writer to manage and distinguish between rules.
  - `revision` offers insights into the rule's version. It serves as an indicator of the evolution of the rule over time, highlighting modifications and enhancements made.

Having discussed the crux of Suricata rules, it's now time to shed light on a powerful tool in rule development: `PCRE` or `Pearl Compatible Regular Expression`. Utilizing PCRE can be a game-changer when crafting rules. To employ PCRE, we use the `pcre` statement, which is then followed by a regular expression. Keep in mind that the PCRE should be encased in leading and trailing forward slashes, with any flags positioned after the last slash.

Also, note that anchors are positioned after and before the encasing slashes, and certain characters demand escaping with a backslash. A piece of advice from the trenches - steer clear from authoring a rule that relies solely on PCRE.

- Example: `alert http any any -> $HOME_NET any (msg: "ATTACK [PTsecurity] Apache Continuum <= v1.4.2 CMD Injection"; content: "POST"; http_method; content: "/continuum/saveInstallation.action"; offset: 0; depth: 34; http_uri; content: "installation.varValue="; nocase; http_client_body; pcre: !"/^\$?[\sa-z\\_0-9.-]*(\&|$)/iRP"; flow: to_server, established;sid: 10000048; rev: 1;)`
  - Firstly, the rule triggers on HTTP traffic (`alert http`) from any source and destination to any port on the home network (`any any -> $HOME_NET any`).
  - The msg field gives a human-readable description of what the alert is for, namely `ATTACK [PTsecurity] Apache Continuum <= v1.4.2 CMD Injection`.
  - Next, the rule checks for the `POST` string in the HTTP method using the `content` and `http_method` keywords. The rule will match if the HTTP method used is a POST request.
  - The `content` keyword is then used with `http_uri` to match the URI `/continuum/saveInstallation.action`, starting at `offset 0` and going to a `depth` of `34` bytes. This specifies the targeted endpoint, which in this case is the `saveInstallation` action of the Apache Continuum application.
  - Following this, another content keyword searches for `installation.varValue=` in the HTTP client body, case insensitively (`nocase`). This string may be part of the command injection payload that the attacker is trying to deliver.
  - Next, we see a `pcre` keyword, which is used to implement Perl Compatible Regular Expressions.
    - `^` marks the start of the line.
    - `\$?` checks for an optional dollar sign at the start.
    - `[\sa-z\\_0-9.-]*` matches zero or more (`*`) of the characters in the set. The set includes:
      - `\s` a space
      - `a-z` any lowercase letter
      - `\\` a backslash
      - `_` an underscore
      - `0-9` any digit
      - `.` a period
      - `-` a hyphen
    - `(\&|$)` checks for either an ampersand or the end of the line.

        - `/iRP` at the end indicates this is an inverted match (meaning the rule triggers when the match does not occur), case insensitive (`i`), and relative to the buffer position (`RP`).
    - Finally, the `flow` keyword is used to specify that the rule triggers on `established`, inbound traffic towards the server.

For those who seek to broaden their understanding of Suricata rules and delve deeper into rule development, the following resource serves as a comprehensive guide: https://docs.suricata.io/en/latest/rules/index.html.

IDS/IPS Rule Development Approaches

When it comes to creating rules for Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), there's an art and a science behind it. It requires a comprehensive understanding of network protocols, malware behaviors, system vulnerabilities, and the threat landscape in general.

A key strategy that we employ while crafting these rules involves the detection of specific elements within network traffic that are unique to malware. This is often referred to as signature-based detection, and it's the classic approach that most IDS/IPS rely on. Signatures can range from simple patterns in packet payloads, such as the detection of a specific command or a distinctive string associated with a particular malware, to complex patterns that match a series of packets or packet characteristics. Signature-based detection is highly effective when dealing with known threats as it can identify these threats with high precision, however, it struggles to detect novel threats for which no signature exists yet.

Another approach focuses on identifying specific behaviors that are characteristic to malware. This is typically referred to as anomaly-based or behavior-based detection. For instance, a certain HTTP response size constantly appearing within a threshold, or a specific beaconing interval might be indicative of a malware communication pattern. Other behaviors can include unusually high volumes of data transfers and uncommon ports being used. The advantage of this approach is its ability to potentially identify zero-day attacks or novel threats that would not be detected by signature-based systems. However, it also tends to have higher false-positive rates due to the dynamic nature of network behaviors.

A third approach that we utilize in crafting IDS/IPS rules is stateful protocol analysis. This technique involves understanding and tracking the state of network protocols and comparing observed behaviors to the expected state transitions of these protocols. By keeping track of the state of each connection, we can identify deviations from expected behavior which might suggest a malicious activity.

Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, let's RDP 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.

Please wait for approximately 5-6 minutes before initiating a connection using Remote Desktop Protocol (RDP). You may have to try 2-3 times before a successful RDP connection is established!

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ xfreerdp /u:htb-student /p:'HTB_@cademy_stdnt!' /v:[Target IP] /dynamic-res
```

Now, we will explore several examples of Suricata rule development to gain a solid understanding of the different approaches we can take and the structure of a rule.

Suricata Rule Development Example 1: Detecting PowerShell Empire

Suricata Rule Development Part 1

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET MALWARE Possible PowerShell Empire Activity
```

The Suricata rule above is designed to detect possible outbound activity from PowerShell Empire, a common post-exploitation framework used by attackers. Let's break down the important parts of this rule to understand its workings.

- `alert`: This is the rule action, indicating that Suricata should generate an alert whenever the conditions specified in the rule options are met.
- `http`: This is the rule protocol. It specifies that the rule applies to HTTP traffic.
- `$HOME_NET any -> $EXTERNAL_NET any`: These are the source and destination IP address specifications. The rule will be triggered when HTTP traffic originates from any port (any) on a host within the `$HOME_NET` (internal network) and is destined to any port (any) on a host in the `$EXTERNAL_NET` (external network).
- `msg:"ET MALWARE Possible PowerShell Empire Activity Outbound"`: This is the message that will be included in the alert to describe what the rule is looking for.
- `flow:established,to_server`: This specifies the direction of the traffic. The rule is looking for established connections where data is flowing to the server.
- `content:"GET"; http_method;`: This matches the HTTP GET method in the HTTP request.
- `content:"/"; http_uri; depth:1;`: This matches the root directory ("/") in the URI.
- `pcre:"/^(?:login\/process|admin\/get|news)\.php$/RU";`: This Perl Compatible Regular Expression (PCRE) is looking for URIs that end with login/process.php, admin/get.php, or news.php.
  - `PowerShell Empire` is an open-source Command and Control (C2) framework. Its agent can be explored via the following repository.https://github.com/EmpireProject/Empire/blob/master/data/agent/agent.ps1#L78
  - Examine the `psempire.pcap` file which is located in the `/home/htb-student/pcaps` directory of this section's target using Wireshark to pinpoint the related requests.
- `content:"session="; http_cookie;`: This is looking for the string "session=" in the HTTP cookie.
- `pcre:"/^(?:[A-Z0-9+/]{4})*(?:[A-Z0-9+/]{2}==|[A-Z0-9+/]{3}=|[A-Z0-9+/]{4})$/CRi";`: This PCRE is checking for base64-encoded data in the Cookie.
  - A plethora of articles examining `PowerShell Empire` exist, here is one noting that the cookies utilized by PowerShell Empire adhere to the Base64 encoding standard. https://www.keysight.com/blogs/tech/nwvs/2021/06/16/empire-c2-networking-into-the-dark-side
- `content:"Mozilla|2f|5.0|20 28|Windows|20|NT|20|6.1"; http_user_agent; http_start;`: This matches a specific User-Agent string that includes "Mozilla/5.0 (Windows NT 6.1".
  - https://github.com/EmpireProject/Empire/blob/master/data/agent/agent.ps1#L78
- `content:".php|20|HTTP|2f|1.1|0d 0a|Cookie|3a 20|session="; fast_pattern; http_header_names;`: This matches a pattern in the HTTP headers that starts with ".php HTTP/1.1\r\nCookie: session=".
- `content:!"Referer"; content:!"Cache"; content:!"Accept";`: These are negative content matches. The rule will only trigger if the HTTP headers do not contain "Referer", "Cache", and "Accept".

This Suricata rule triggers an alert when it detects an established HTTP GET request from our network to an external network, with a specific pattern in the URI, cookie, and user-agent fields, and excluding certain headers.

The above rule is already incorporated in the `local.rules` file found in the `/home/htb-student` directory of this section's target. To test it, first, you need to uncomment the rule. Then, execute Suricata on the `psempire.pcap` file, which is located in the `/home/htb-student/pcaps` directory.

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ sudo suricata -r /home/htb-student/pcaps/psempire.pcap -l . -k none
 15/7/2023 -- 03:57:42 - <Notice> - This is Suricata version 4.0.0-beta1 RELEASE
```

```
15/7/2023 -- 03:57:42 - <Notice> - all 5 packet processing threads, 4 management threads initialize
15/7/2023 -- 03:57:42 - <Notice> - Signal Received.  Stopping engine.
15/7/2023 -- 03:57:42 - <Notice> - Pcap-file module read 511 packets, 101523 bytes
```

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ cat fast.log
11/21/2017-05:04:53.950737  [**] [1:2027512:1] ET MALWARE Possible PowerShell Empire Activity Outbo
11/21/2017-05:04:01.308390  [**] [1:2027512:1] ET MALWARE Possible PowerShell Empire Activity Outbo
11/21/2017-05:05:20.249515  [**] [1:2027512:1] ET MALWARE Possible PowerShell Empire Activity Outbo
11/21/2017-05:05:56.849190  [**] [1:2027512:1] ET MALWARE Possible PowerShell Empire Activity Outbo
11/21/2017-05:06:02.062235  [**] [1:2027512:1] ET MALWARE Possible PowerShell Empire Activity Outbo
11/21/2017-05:06:17.750895  [**] [1:2027512:1] ET MALWARE Possible PowerShell Empire Activity Outbo
11/21/2017-05:04:11.988856  [**] [1:2027512:1] ET MALWARE Possible PowerShell Empire Activity Outbo
---SNIP---
```

The `local.rules` file contains another rule for detecting `PowerShell Empire`, located directly below the rule we just examined. Invest some time in scrutinizing both the `psempire.pcap` file using `Wireshark` and this newly found rule to comprehend how it works.

Suricata Rule Development Example 2: Detecting Covenant

Suricata Rule Development Part 1

```
alert tcp any any -> $HOME_NET any (msg:"detected by body"; content:"<title>Hello World!</title>";
```

Rule source: Signature-based IDS for Encrypted C2 Traffic Detection - Eduardo Macedo

The (inefficient) Suricata rule above is designed to detect certain variations of Covenant, another common post-exploitation framework used by attackers. Let's break down the important parts of this rule to understand its workings.

- `alert`: This is the rule action. When the conditions in the rule options are met, Suricata will generate an alert.
- `tcp`: This is the rule protocol. The rule applies to TCP traffic.
- `any any -> $HOME_NET any`: These are the source and destination IP address and port specifications. The rule is watching for TCP traffic that originates from any IP and any port (`any any`) and is destined for any port (`any`) on a host in the `$HOME_NET` (internal network).
- `content:"<title>Hello World!</title>";`: This instructs Suricata to look for the string `<title>Hello World!</title>` in the TCP payload.
    - `Covenant` is an open-source Command and Control (C2) framework. Its underpinnings can be explored via the following repository.https://github.com/cobbr/Covenant/blob/master/Covenant/Data/Profiles/DefaultHttpProfile.yaml#L35
    - Examine the `covenant.pcap` file which is located in the `/home/htb-student/pcaps` directory of this section's target using Wireshark to pinpoint the related requests.
- `detection_filter: track by_src, count 4, seconds 10;`: This is a post-detection filter. It specifies that the rule should track the source IP address (`by_src`) and will only trigger an alert if this same detection happens at least 4 times (`count 4`) within a 10-second window (`seconds 10`).
    - Examine the `covenant.pcap` file which is located in the `/home/htb-student/pcaps` directory of this section's target using Wireshark to pinpoint the related requests.

This Suricata rule is designed to generate a high-priority alert if it detects at least four instances of TCP traffic within ten seconds that contain the string `<title>Hello World!</title>` in the payload, originating from the same source IP and headed towards any host on our internal network.

The above rule is already incorporated in the `local.rules` file found in the `/home/htb-student` directory of this section's target. To test it, first, you need to uncomment the rule. Then, execute Suricata on the `covenant.pcap` file,

which is located in the `/home/htb-student/pcaps` directory.

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ sudo suricata -r /home/htb-student/pcaps/covenant.pcap -l . -k none
15/7/2023 -- 04:47:15 - <Notice> - This is Suricata version 4.0.0-beta1 RELEASE
15/7/2023 -- 04:47:15 - <Notice> - all 5 packet processing threads, 4 management threads initialize
15/7/2023 -- 04:47:16 - <Notice> - Signal Received.  Stopping engine.
15/7/2023 -- 04:47:16 - <Notice> - Pcap-file module read 27384 packets, 3125549 bytes
```

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ cat fast.log
01/21/2021-06:38:51.250048  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-06:40:55.021993  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-06:36:21.280144  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-06:41:53.395248  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-06:42:21.582624  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-06:41:25.215525  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-07:17:01.778365  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-07:12:55.294094  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-07:14:27.846352  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
01/21/2021-07:17:29.981168  [**] [1:3000011:0] detected by body [**] [Classification: (null)] [Prio
---SNIP---
```

The `local.rules` file contains three (3) other rules for detecting `Covenant`, located directly below the rule we just examined. Invest some time in scrutinizing https://github.com/cobbr/Covenant/blob/master/Covenant/Data/Profiles/DefaultHttpProfile.yaml, the `covenant.pcap` file using `Wireshark`, and these newly found rule to comprehend how they work. These rules may yield false-positive results, and hence for optimal performance, it's advisable to integrate them with other detection rules.

Suricata Rule Development Example 3: Detecting Covenant (Using Analytics)

Suricata Rule Development Part 1

```
alert tcp $HOME_NET any -> any any (msg:"detected by size and counter"; dsize:312; detection_filter
```

The `local.rules` file also contains the above rule for detecting `Covenant`. Let's break down the important parts of this rule to understand its workings.

- `dsize:312;`: This instructs Suricata to look for TCP traffic with a data payload size of exactly 312 bytes.
- `detection_filter: track by_src, count 3 , seconds 10;`: This is a post-detection filter. It says that the rule should keep track of the source IP address (`by_src`), and it will only trigger an alert if it detects the same rule hit at least 3 times (`count 3`) within a 10-second window (`seconds 10`).

This Suricata rule is designed to generate a high-priority alert if it detects at least three instances of TCP traffic within ten seconds that each contain a data payload of exactly 312 bytes, all originating from the same source IP within our network and headed anywhere.

The above rule is already incorporated in the `local.rules` file found in the `/home/htb-student` directory of this section's target. To test it, first, you need to uncomment the rule. Then, execute Suricata on the `covenant.pcap` file, which is located in the `/home/htb-student/pcaps` directory.

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ sudo suricata -r /home/htb-student/pcaps/covenant.pcap -l . -k none
15/7/2023 -- 05:29:19 - <Notice> - This is Suricata version 4.0.0-beta1 RELEASE
15/7/2023 -- 05:29:19 - <Notice> - all 5 packet processing threads, 4 management threads initialize
15/7/2023 -- 05:29:20 - <Notice> - Signal Received.  Stopping engine.
15/7/2023 -- 05:29:20 - <Notice> - Pcap-file module read 27384 packets, 3125549 bytes
```

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ cat fast.log
01/21/2021-06:45:21.609212  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50386
01/21/2021-06:48:49.965761  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50395
01/21/2021-06:42:49.682887  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50380
01/21/2021-06:49:20.143398  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50396
01/21/2021-06:50:49.706170  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50400
01/21/2021-06:51:21.905950  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50401
01/21/2021-06:50:18.527587  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50399
01/21/2021-06:52:52.484676  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50406
01/21/2021-06:51:51.090923  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50404
01/21/2021-06:55:56.650678  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50413
01/21/2021-06:53:22.680676  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50407
01/21/2021-06:54:25.067327  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50409
01/21/2021-06:54:55.275951  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50410
01/21/2021-06:57:25.201284  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50416
01/21/2021-06:57:53.387489  [**] [1:3000001:0] detected by size and counter [**] [Classification: (
50417
---SNIP---
```

Invest some time in scrutinizing both the `covenant.pcap` file using `Wireshark` and this newly found rule to comprehend how it works.

Suricata Rule Development Example 4: Detecting Sliver

Suricata Rule Development Part 1

```
alert tcp any any -> any any (msg:"Sliver C2 Implant Detected"; content:"POST"; pcre:"/\/(php|api|u
```

Rule source: https://www.bilibili.com/read/cv19510951/

The Suricata rule above is designed to detect certain variations of Sliver, yet another common post-exploitation framework used by attackers. Let's break down the important parts of this rule to understand its workings.

- `content:"POST";`: This option instructs Suricata to look for TCP traffic containing the string "POST".
- `pcre:"/\/(php|api|upload|actions|rest|v1|oauth2callback|authenticate|oauth2|oauth|auth|database|db|namespaces)(.*?)((login|signin|api|samples|rpc|index|admin|register|sign-up)\.php)\?[a-z_]{1,2}=[a-z0-9]{1,10}/i";`: This regular expression is utilized to identify specific URI patterns in the traffic. It will match URIs that contain particular directory names followed by file names ending with a PHP extension.
  - `Sliver` is an open-source Command and Control (C2) framework. Its underpinnings can be explored via the following repository.https://github.com/BishopFox/sliver/blob/master/server/configs/http-c2.go#L294
  - Examine the `sliver.pcap` file which is located in the `/home/htb-student/pcaps` directory of this section's target using Wireshark to pinpoint the related requests.

The above rule is already incorporated in the `local.rules` file found in the `/home/htb-student` directory of this section's target. To test it, first, you need to uncomment the rule. Then, execute Suricata on the `sliver.pcap` file, which

is located in the `/home/htb-student/pcaps` directory.

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ sudo suricata -r /home/htb-student/pcaps/sliver.pcap -l . -k none
16/7/2023 -- 02:27:50 - <Notice> - This is Suricata version 4.0.0-beta1 RELEASE
16/7/2023 -- 02:27:50 - <Notice> - all 5 packet processing threads, 4 management threads initialize
16/7/2023 -- 02:27:50 - <Notice> - Signal Received.  Stopping engine.
16/7/2023 -- 02:27:50 - <Notice> - Pcap-file module read 36 packets, 18851 bytes
```

Suricata Rule Development Part 1

```
MisaelMacias@htb[/htb]$ cat fast.log
01/23/2023-15:14:46.988537  [**] [1:1000002:1] Sliver C2 Implant Detected - POST [**] [Classificati
01/23/2023-15:14:47.321224  [**] [1:1000002:1] Sliver C2 Implant Detected - POST [**] [Classificati
01/23/2023-15:14:48.074797  [**] [1:1000002:1] Sliver C2 Implant Detected - POST [**] [Classificati
```

The `local.rules` file contains another rule for detecting `Sliver`, located directly below the rule we just examined.

Suricata Rule Development Part 1

```
alert tcp any any -> any any (msg:"Sliver C2 Implant Detected - Cookie"; content:"Set-Cookie"; pcre
```

Let's break down the important parts of this rule to understand its workings.

- `content:"Set-Cookie";`: This option instructs Suricata to look for TCP traffic containing the string `Set-Cookie`.
- `pcre:"/(PHPSESSID|SID|SSID|APISID|csrf-state|AWSALBCORS)\=[a-z0-9]{32}\;/";`: This is a regular expression used to identify specific cookie-setting patterns in the traffic. It matches the `Set-Cookie` header when it's setting specific cookie names (PHPSESSID, SID, SSID, APISID, csrf-state, AWSALBCORS) with a value that's a 32-character alphanumeric string.

Invest some time in scrutinizing the `sliver.pcap` file using `Wireshark` to identify the related requests.

Start Instance

∞ / 1 spawns left

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!"

+ 1 In the /home/htb-student directory of this section's target, there is a file called local.rules. Within this file, there is a rule with sid 2024217, which is associated with the MS17-010 exploit. Additionally, there is a PCAP file named eternalblue.pcap in the /home/htb-student/pcaps directory, which contains network traffic related to MS17-010. What is the minimum offset value that can be set to trigger an alert?

4

Submit

Previous Next

Mark Complete & Next