

Deployment Modes

Enterprise Suricata Deployment



Review - Suricata Capabilities

- Standards based formats (YAML, JSON) ease integrations with SIEM or other analysis tools
- Multithreaded, hardware acceleration available
- Native IPv6
- Auto protocol detection
- Advanced HTTP/HTTP2, DNS, SMTP and TLS support
- File extraction - FTP/SMTP/HTTP/HTTP2/NFS/SMBv1/2/3
- File MD5, SHA1 and SHA256 checksum support
- Netflow output available

Review - Suricata Capabilities Con't

- Lua scripting
- IP lists, IP reputation and GeoIP
- Bypass (deals with Elephant flows)
- Community ID
- JA3/JA3S
- SCADA protocols - DNP3, ENIP, CIP and Modbus
- Full Packet Capture (FPC)

Network Security Monitoring

Network security monitoring (NSM) is network data collection & analysis.

- Entire ecosystem of bundled tools/software:
 - Stand alone
 - Distributed (multiple servers/systems)
- Suricata plays a central role in generating valuable data:
 - Logs
 - Events
 - Metadata and protocols
 - Alerts

NSM Core Components

- Can be **proactive**:
 - Generating network data for threat hunting
- Can be **reactive**:
 - Incident response or network forensics due to an incident or alert
- What you need:
 - Full packet capture (FPC) +
 - Network monitoring (Suricata)
 - Endpoint detection and response (EDR)
 - Aggregation and correlation tools such as Kibana, Splunk and Grafana (SIEM)
 - Analysis tools such as EveBox or Arkime

Intrusion Detection Mode (IDS)

- Passive:
 - Almost no impact on information system
- No direct protection:
 - Increase your capacity of reaction and visibility
- Organizational impact:
 - Needs process to treat alerts/data
- Determine how exposed you are.
- Improve visibility on your network.
- Increase knowledge.
- Detect attacks:
 - Difficult for adversaries to discover or detect

Intrusion Prevention Mode (IPS)

- Detect attacks:
 - Increase speed of reaction
- Improve visibility on your network:
 - Increase knowledge
 - Detect attacks
- Hard to set up:
 - Requires thorough understanding and knowledge of the internal infrastructure
 - Requires much more testing/verification

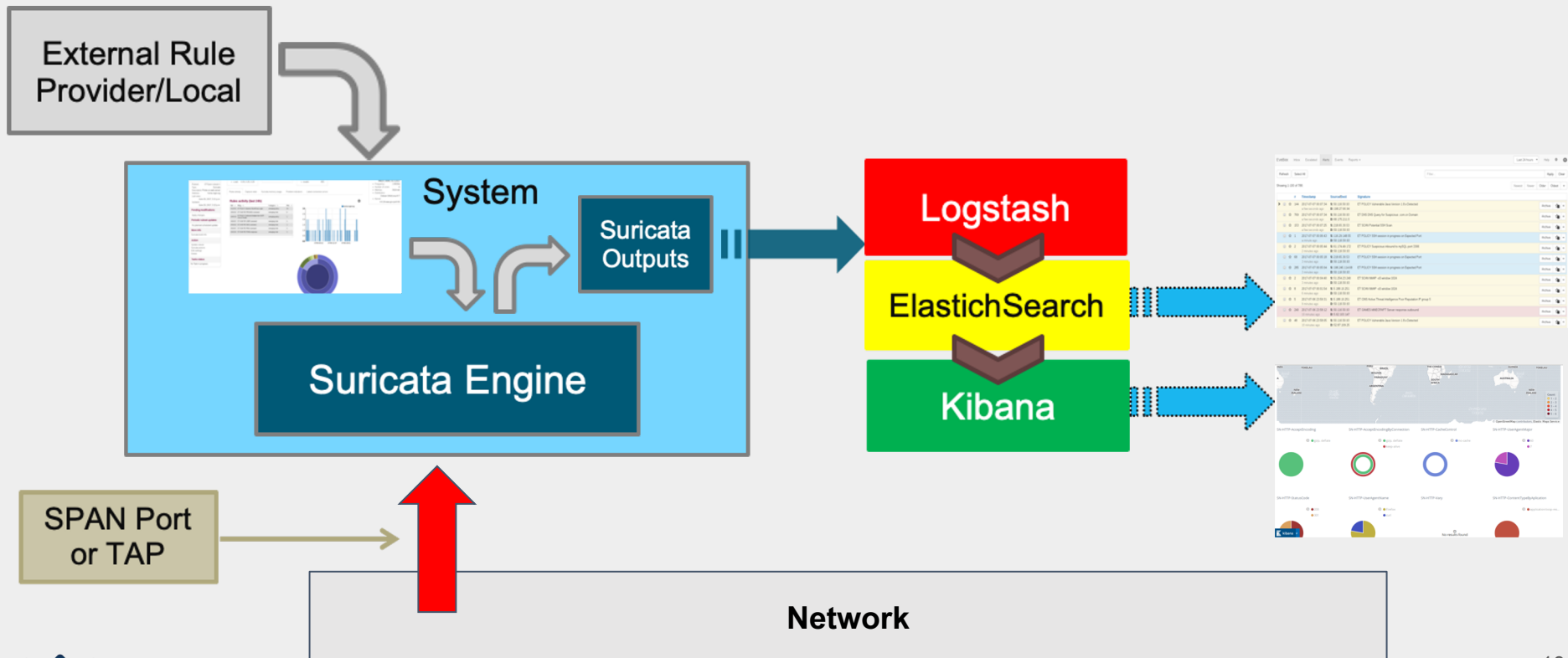
Hybrid Mode - IDPS

- Hybrid mode
- Suricata is deployed as an IDS.
- Has the capability to reset connections:
 - Via sending RST packets
- Low latency is critical for usability.
- In order for this mode to be engaged rules actions need to be adjusted from:
 - “alert”
 - To-
 - “reject...”

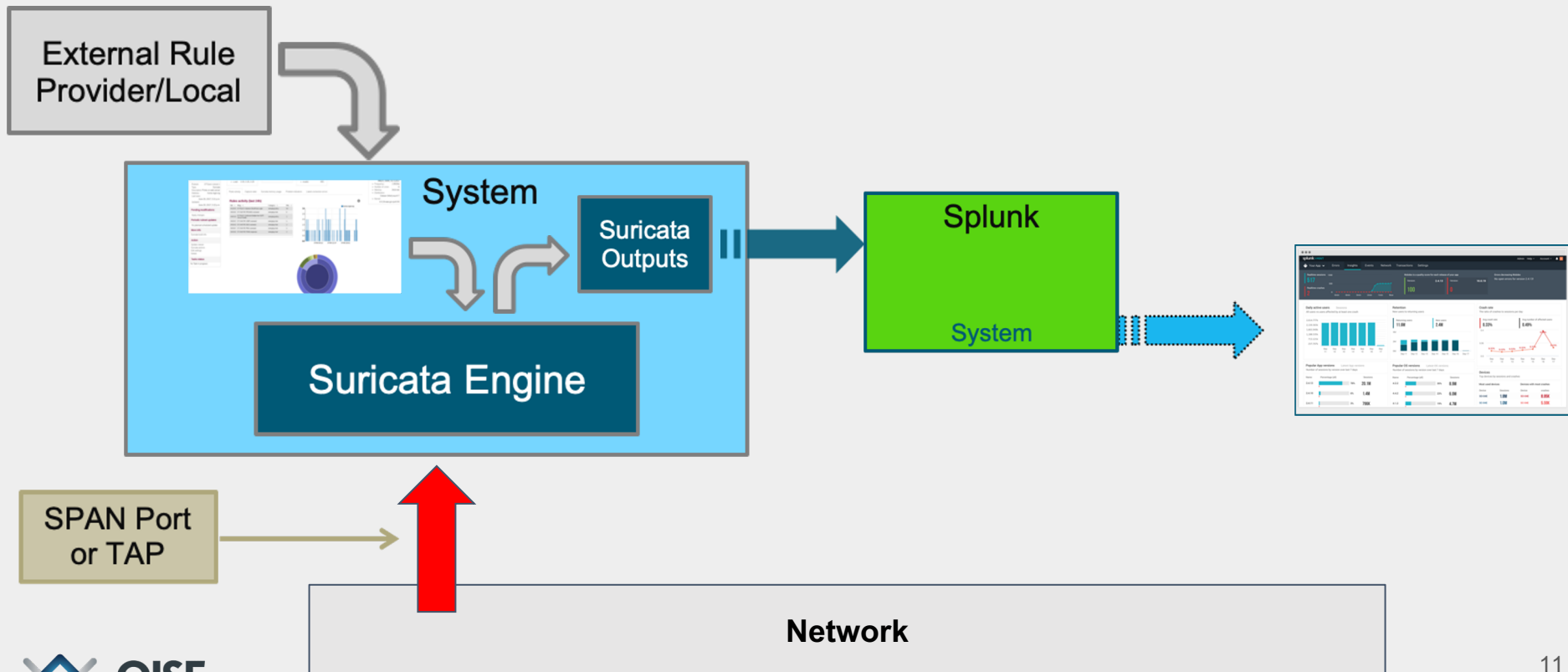
Other Hybrid Modes

- **Hybrid mode** is when one or more functions of the Suricata Capabilities are combined.
- Those combinations can be many (besides the standard IDS/IPS):
 - IDS+NSM
 - IDPS+NSM
 - IDS+NSM+Full Packet Capture

Deployment Example - Elastic Stack



Deployment Example - Splunk



Network Placement and Capture Methods

Know Your Turf

- Who:
 - are you protecting? (type/nature of users/environment)
- What:
 - are the existing applications/servers/services?
- Where:
 - are those residing? (geoip-localization,server farms/network zones)
- When:
 - are these applications/servers/services mostly used?
- How:
 - are they used?

Placement In The Network

- Tradeoffs between:
 - Resources to protect
 - Treatment capability of the security team
- Standard placement:
 - On internet link
 - On path from servers to other networks
 - On path from clients to the outside

Capture Methods Per Mode

- IDS mode
 - AF_Packet
 - PF_Ring
 - NETMAP
- Cross platform capture
 - Libpcap
- IPS mode
 - Netfilter (nfqueue)
 - IPFW
 - AF_Packet
 - NETMAP
- Specialized Capture
 - Endace
 - Napatech
 - Mellanox
 - Netronome

Virtual versus Hardware Deployments - Virtual

- Virtual (Pros)
 - Very flexible
 - Easy to manage, back-up, and restore
- Virtual (Cons)
 - No full control over the VM host
 - Virtualization settings on the host can interfere with the capture/sniffing traffic

Virtual versus Hardware Deployments - Hardware

- HW (Pros)
 - Full control of the settings on the box/server
 - Dedicated for the deployment
 - Best for Performance
- HW (Cons)
 - Not fast and flexible to order and rack
 - Not as flexible as a VM to resize in terms of adjusting CPU/RAM/HDD

What Operating System Is Best?

- Suricata is natively developed on Linux and mostly tested on Linux
- OS for prod deployment can depend on:
 - Corporate OS policy
 - In-house OS experience and expertise

Monitoring Encrypted Traffic

- Encryption is more and more common
- Naturally, the possibilities to inspect encrypted traffic are also more and more common
- Commercial vendors offer possibilities to decrypt traffic and mirror to monitoring device
- Positioning of Suricata is a key
- Even in encrypted traffic, Suricata would still inspect TLS (produce logs) and the flow (logs) and produce the relevant data

Keeping Suricata Up-To-Date

- Suricata:
 - A simple package upgrade is enough (ex Ubuntu/Debian)
- `apt-get update && apt-get upgrade suricata`
 - You should always use with the latest stable version available
 - You should always use the latest `suricata.yaml`
- OS:
 - Usually an OS upgrade will upgrade Suricata as well – ex:
- `apt-get update && apt-get upgrade`
 - You should always have your OS upgraded with the latest stable packages and drivers

Note: An OS upgrade can result in NIC driver and kernel upgrade as well

Caveat Emptor

- Before each OS/Suricata production upgrade
 - Test and verify the upgrade on a test/QA system
- After each OS/Suricata upgrade
 - Confirm and verify proper OS operation
 - Confirm and verify proper Suricata operation
 - Confirm and verify log management and handling
- Dashboards/visualizations

Modifying Rules with Suricata-Update

Rule Modification

- It is common to need to modify rules in your environment.
- First, you must figure out how to match on the rule(s) you want to modify.
- This can be done by:
 - Signature ID (SID)
 - Regular expression
 - Rule group
 - Filename

Matching Rules

- Signature ID:
 - Example: 1034
- Regular expression:
 - Example: re:heartbleed
- Group matching:
 - Example: group: emerging-icmp.rules
- Filename matching:
 - Example: filename: rules/*deleted*

Modifying Rules

- Modifications can be done via a regular expression search and replace
 - Basic pattern: `<match> <from> <to>`
- `<match>` = one of the rule matchers from above
- `<from>` = the text to be replaced
- `<to>` = the replacement text

Example:

- Convert all alert rules to drop:
 - `Re:. ^alert drop`

Order of Application of Config Files

- The order the modification configuration files will be applied to the rules:
 - Disable.conf
 - Enable.conf
 - Drop.conf
 - Modify.conf

Disable Rules

- `Disable.conf`
- Use rule matchers to disable rules from rule sources.
- Example:
 - Disable a rule by SID:

```
# suricata-update - disable.conf

# Example of disabling a rule by signature ID (gid is optional).
# 1:2019401
# 2019401
```

Enable Rules

- `Enable.conf`
- Example:

```
# suricata-update - enable.conf

# Example of enabling a rule by signature ID (gid is optional).
# 1:2019401
# 2019401

# Example of enabling a rule by regular expression.
# - All regular expression matches are case insensitive.
# re:heartbleed
# re:MS(0[7-9]|10)-\d+
```

Converting Rules to Drop

- Drop.conf
- Example:

```
# suricata-update - drop.conf
#
# Rules matching specifiers in this file will be converted to drop rules.
#
# Examples:
#
# 1:2019401
# 2019401
#
# re:heartbleed
# re:MS(0[7-9]|10)-\d+
```

Modifying Rules

- Modify.conf
- Examples:

```
# suricata-update - modify.conf

# Format: <sid> "<from>" "<to>"

# Example changing the seconds for rule 2019401 to 3600.
# 2019401 "seconds \d+" "seconds 3600"
#
# Example converting all alert rules to drop:
# re:. ^alert drop
#
# Example converting all drop rules with noalert back to alert:
# re:. "^drop(.*)noalert(.*)" "alert\\1noalert\\2"

# Change all trojan-activity rules to drop. Its better to setup a
# drop.conf for this, but this does show the use of back references.
# re:classtype:trojan-activity "(alert)(.*)" "drop\\2"
```

Ready for Production

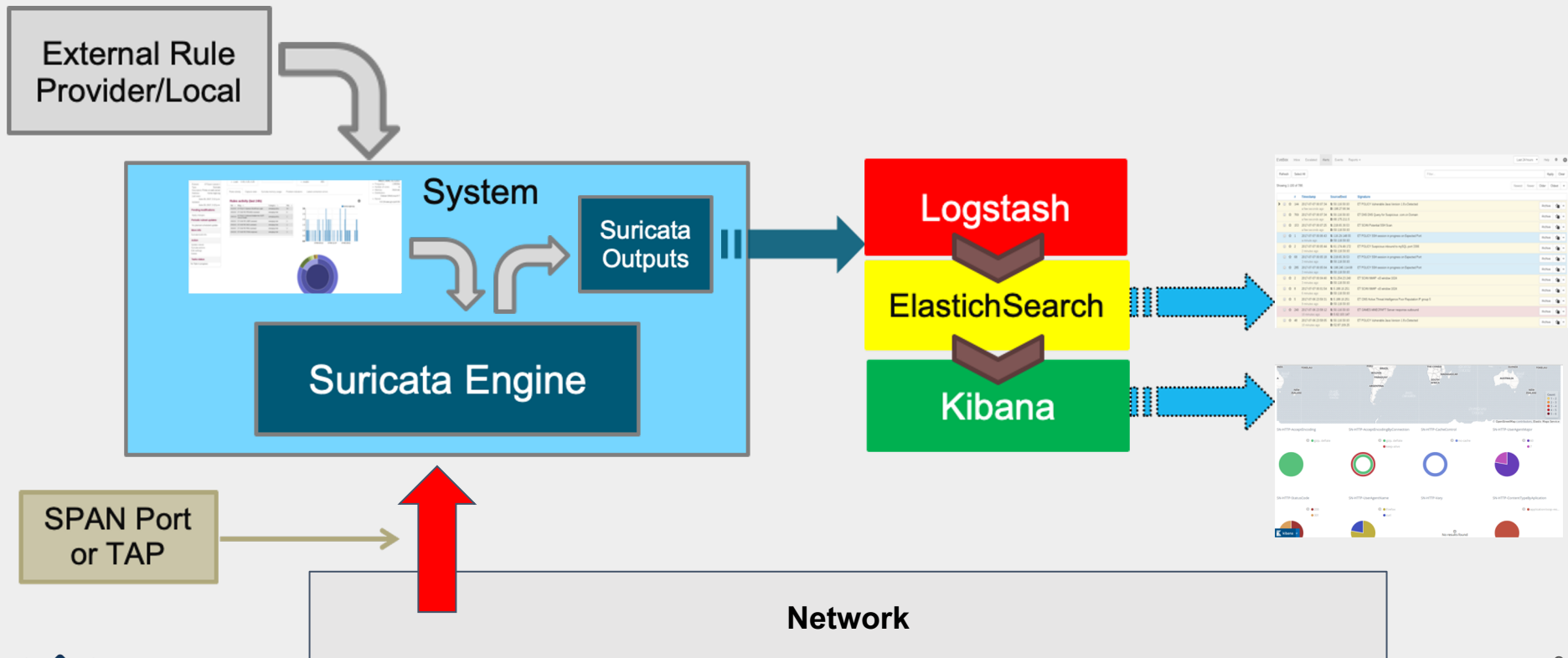
- Once your modification files are ready, simply run `suricata-update`.
 - Don't forget to consider testing your changes first

Integrating Into an Elastic Stack

You Have Suricata Data, Now What?

- By default, Suricata stores data on the server it is running on
- While you could log into the server to view this data, it's likely a better option to send your logs to a centralized server
- How you collect your data and where you send it to will vary
 - Utilizing an Elastic Stack we'll look at all of the key components needed to make this happen
- What is needed?
 - Something to collect the logs from the sensor
 - Something to normalize and possibly enrich the logs
 - Something to store the logs and make it searchable
 - Something to visualize the data

Deployment Example - Elastic Stack



Working with Files on the Wire

File Extraction with Suricata

- Suricata can extract files from the network stream while inspecting
- Works on top of protocol parsers, which in turn work on top of the stream reassembly engine
 - Settings in the stream engine, reassembly engine and the application layer parsers all affect the workings of the file extraction
- Configuration options are under “file-store” section in the YAML
- Which files are actually extracted and stored is controlled by rules

Suricata Also Generates FileInfo Events

- In EveBox, you can look under Events -> FileInfo
 - Select the record to view more information
- Can also go direct to EVE.JSON
 - `cat eve.json | jq -c 'select(.event_type=="fileinfo")'`

2019-10-13 21:37:23 2 months ago	FILEINFO	S: 160.153.74.197 D: 192.168.1.101	k795d7j035x.exe - Hostname: bluelionconflictsolutions.com; Content-Type: application/octet-stream
2019-10-13 21:36:20 2 months ago	FILEINFO	S: 104.100.93.16 D: 192.168.1.101	/ncsi.txt - Hostname: www.msftncsi.com; Content-Type: text/plain

FileInfo Caveats

- Magic tells us the file type – will not always match the file extension
 - Good IOC with the file hash, but...
- There is a problem here... yeah, it's TRUNCATED

Timestamp	2019-10-13T21:37:23.080585-0600
Sensor	SELKS
Protocol	TCP
Source	160.153.74.197:80 ▾
Destination	192.168.1.101:49295 ▾
Flow ID	1736197268429482

Filename	k795d7j035x.exe
Gaps	false
Magic	PE32 executable (GUI) Intel 80386, for MS Windows
Sha256	bafec0ecc524441f81b6da635d7c4def1614349d24a84cf59b9b103166514ec5
Size	108004
State	TRUNCATED
Stored	false
Tx ID	0
Type	PE32 executable (GUI) Intel 80386

Tweaking the Config

- Suricata only looks so far into the response stream
 - Which results in an inaccurate file hash
- Need to make some changes in our Suricata.yaml
 - -file-store -> enabled: yes
 - libhttp -> response-body-limit: 0
- And then we need to create some RULES

File Extraction Rules - Syntax

- filestore;

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

- fileext: "pdf";

```
alert http any any -> any any (msg:"FILE PDF file claimed"; fileext:"pdf"; filestore; sid:2; rev:1;)
```

- filemagic:"PDF Document";
 - Comes from libmagic

```
alert http any any -> any any (msg:"FILE pdf detected"; filemagic:"PDF document"; filestore; sid:3; rev:1;)
```


File Information and Extraction

- The state is CLOSED
- File hash is now accurate
- And if the rule matched, there will be a file at:
/var/log/suricata/filestore/

File ID	1
Filename	k795d7j035x.exe
Gaps	false
Magic	PE32 executable (GUI) Intel 80386, for MS Windows
Md5	f36d27c36ce258283a050db08051ddc3
Sha1	b15bb6f0892dc78e8cec312c97b78d00b59e60fd
Sha256	d7e48995f37ac2d3de583b3b9483d8f9a73180b01209a75b61f3b76777144bd5
Sid.0	3
Size	209920
State	CLOSED
Stored	true

1	2019-10-13 21:37:21	S: 160.153.74.197	FILE EXE Stored
	2 months ago	D: 192.168.1.101	