Efficient Packet Capture Creation and Testing on Suricata and Snort

David Wharton

Senior Security Researcher,

Counter Threat Unit

Will Urbanski

Principal Consultant, Office of the CTO

SuriCon, 16 Nov 2017

A Tale of Two Tools

This talk is about tools!

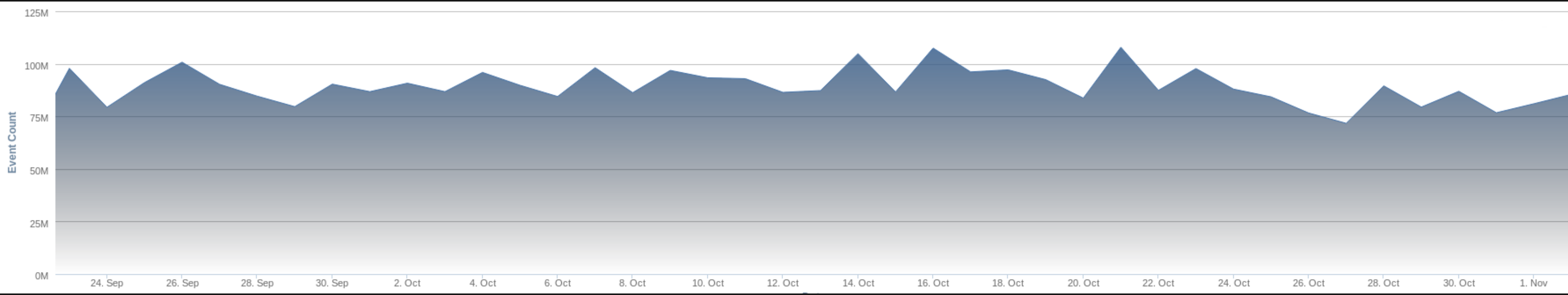
- A tool for creating packet captures!
- A tool for testing rules with packet captures!

We're releasing these tools after the talk!

We hope you find them useful!

Background

- **60 to 125+ million IPS/IDS events a day**
  - From some sets of sensors; the total number is higher
- **Create rulesets for and actively manage and monitor over 6K IPS/IDS sensors**
  - Suricata and Snort-based (also Palo-Alto NGFW….)
  - Over 20K rules
- **Mature rule creation & ruleset release process**
- **Ruleset releases at least once a day (sometimes more)**



Problem Definitions

Rule Creation - Requirements

- Rules must work as expected
  - “Why doesn’t my rule work?”
- All rules must have passing true positive test case
- Rules tuned for false positives must have a false positive test case
- Multi-platform / sensor support

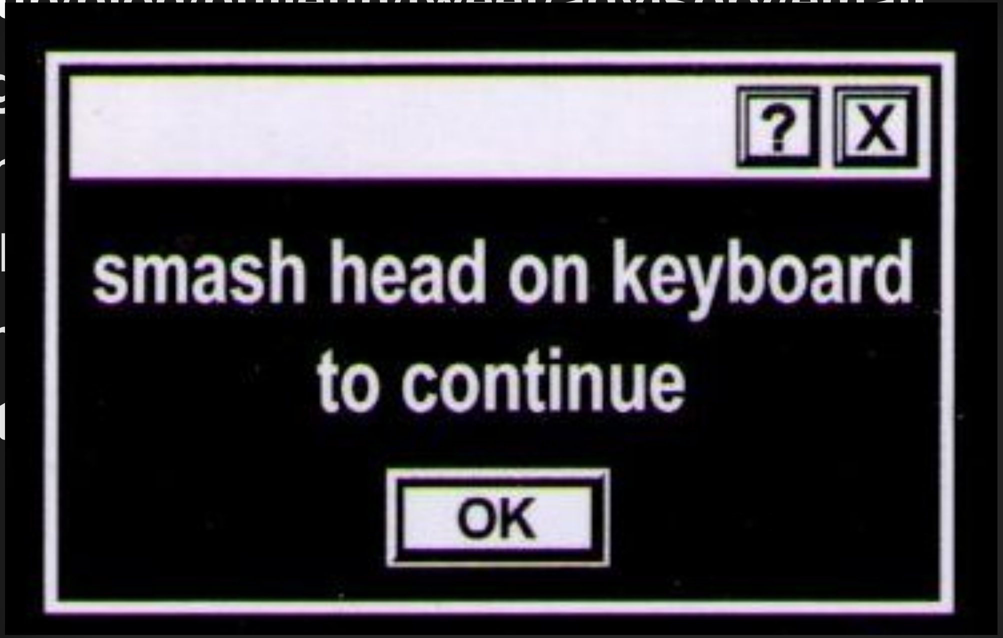
Coverage – Questions

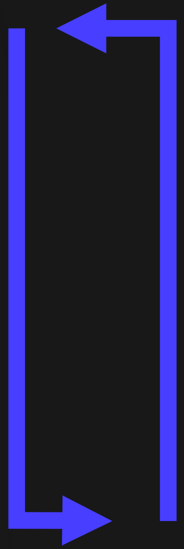
- Do we have detection for <insert malware here>?
- Do we have coverage for <insert vuln here>?
- Will this pcap trigger any rules on our sensor(s)?
- Does this unknown malware traffic match any known malware signatures?

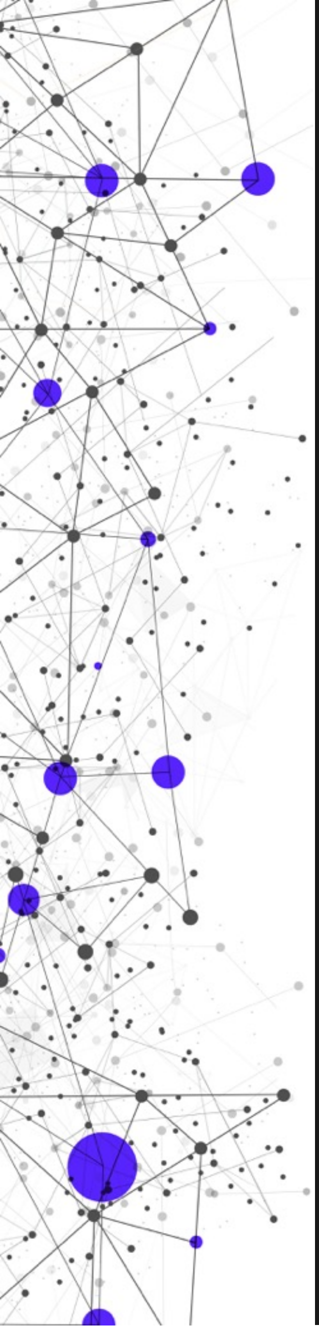
IDS/IPS Testing Paradigm

"Do we have coverage for X"

1. Read write-up/blog/bulletin/tweet/advisory/email
2. Spin up targ
3. Start Wiresh
4. Run nc, pytl
5. Stop Wiresh
6. Run against





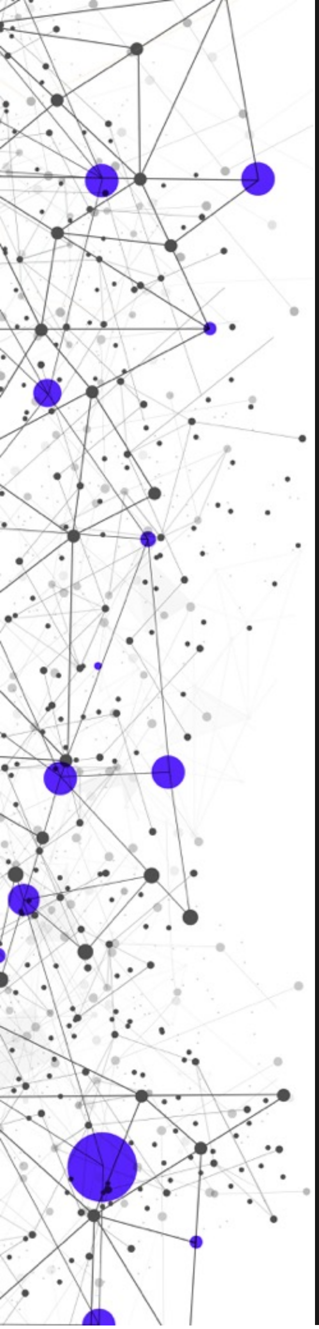


If only we had a tool…

…and it had the following properties

- Things it should do:
  - Make it simple to model two-way communication.
  - Support common transport protocols (Sorry, SCTP)
  - Easily express text and binary protocols.
  - Support comments, so I don’t have to re-read the RFC when working with a binary protocol.
  - Easily chained with other tools.

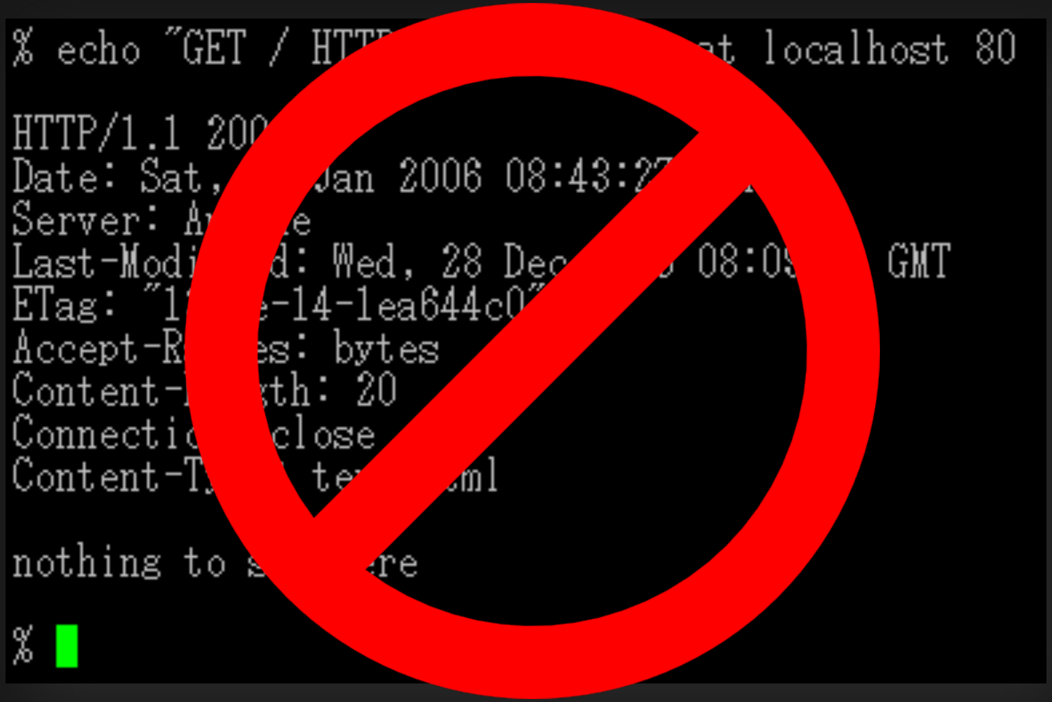
- Things I ***really*** don’t want to think about:
  - The TCP three-way handshake
  - Computing TCP SEQ and ACK numbers
  - Maximum Segment Size, VLAN tags, etc..



Flowsynth

Network traffic modeling tool

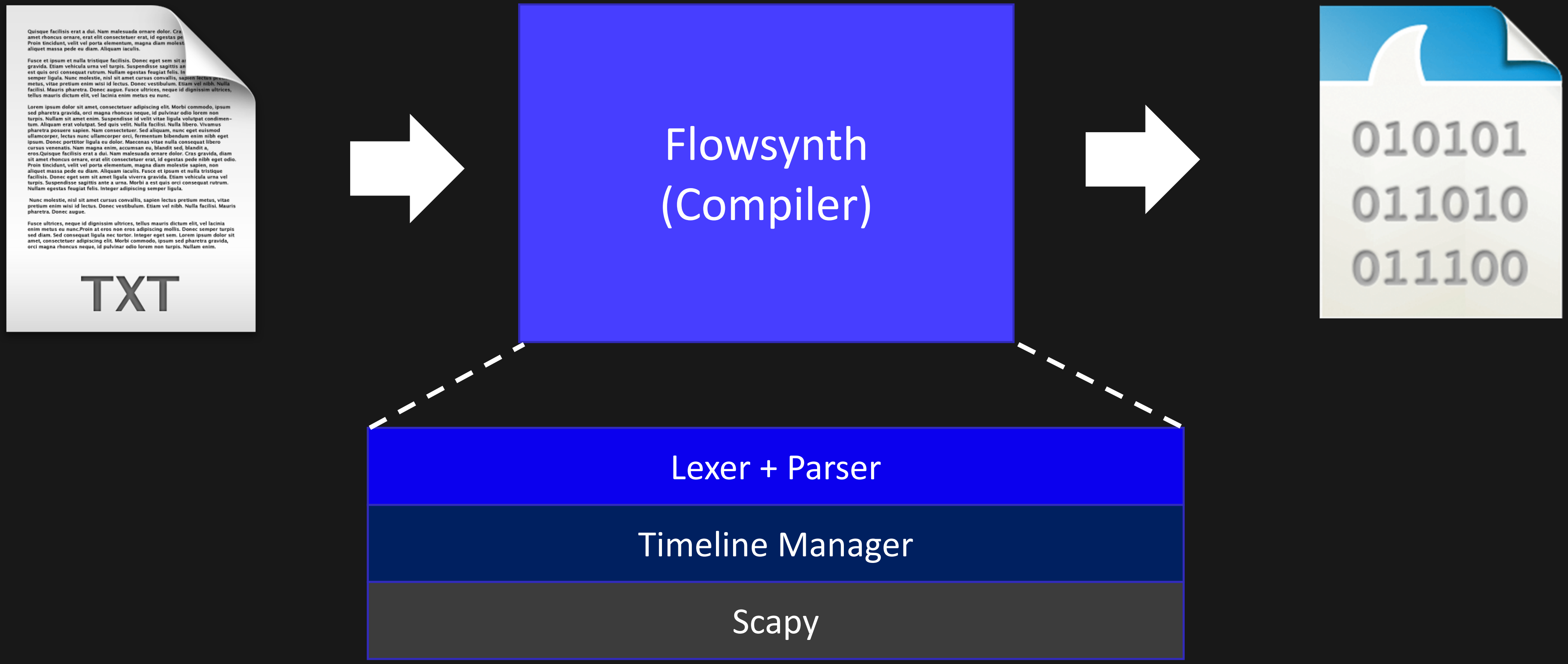
- **Enables a researcher to quickly generate network traffic.**
- **Painlessly generate:**
  - **Text-based hexdumps of packet captures**
  - **libpcap-format packet captures**
- **Reduces time and complexity of PCAP-creation workflows.**
- **PCAPs as code -- version-control friendly!**



```
% echo "GET / HTTP                    st localhost 80

HTTP/1.1 200
Date: Sat,   Jan 2006 08:43:2
Server: A    e
Last-Modi    d: Wed, 28 Dec      08:05    GMT
ETag: "1    -14-1ea644c0"
Accept-R    es: bytes
Content-    th: 20
Connecti    close
Content-T     te    ml

nothing to s      re

%
```

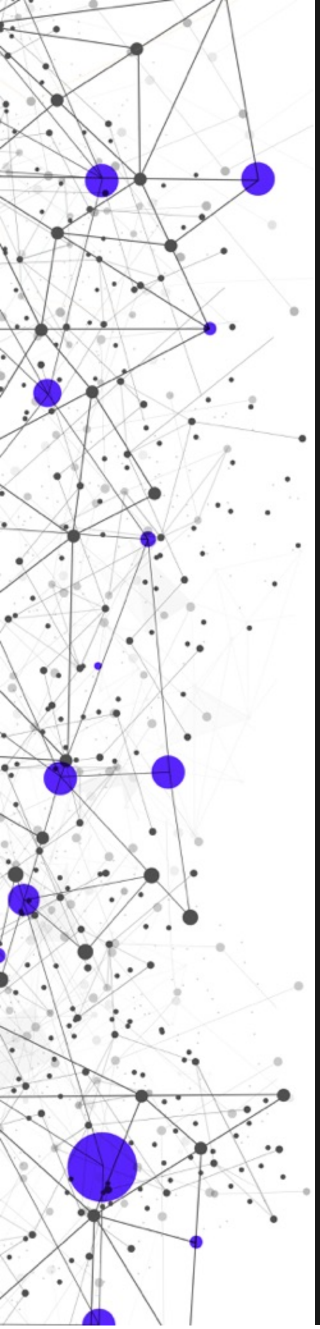
Flowsynth

How it works



Flowsynth Syntax Language

- Used to describe/define network flows and behavior
- Instruction types:
  - Flows
    - Define particular network flow and tuple (IPs, ports, protocol)
  - Events
    - Something happened! Like data transfer…or my flow was RST.
    - These reference and apply to defined flows.
  - Comments
    - Because everyone documents their PCAPs, right?
    - Start with ‘#’



Flowsynth Syntax Language

Flow Declarations

Structure

```
flow [flow_name] [proto] [src]:[srcport] > [dst]:[dstport] ([flow_attributes]);
```

Example

```
flow my_connection tcp 192.168.23.4:44123 > 172.16.14.189:80 (tcp.initialize;
mss:1460;);
```

Flowsynth Syntax Language

Event Declarations

Expresses data transferred between hosts in a flow.

Structure

```
[flow name] [directionality] ([event_attributes]);

 directionality = “>” | “<“
 event_attributes is one or more of: content | filecontent | tcp_option
```

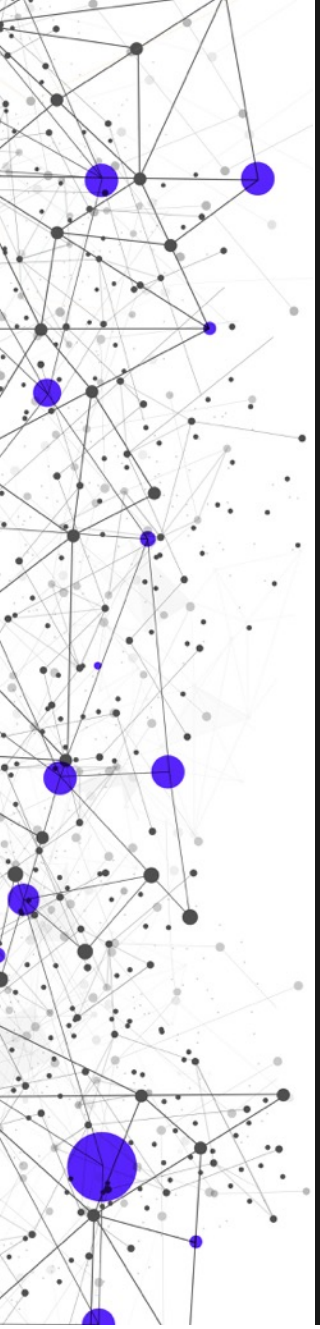
Example

```
 my_connection > (content:"GET / HTTP/1.1\x0D\x0A\x0D\x0a";);
 my_connection < (content:"HTTP/1.1 200 OK\x0D\x0A\x0D\x0A";
                   filecontent:"index.html";);
```

Flowsynth Syntax Language

A more complex example

```
flow myflow1 tcp 192.168.9.10:22301 > myifone.ru:80 (tcp.initialize;);


myflow1 >     (
              content:"POST /c2.php HTTP/1.1\x0D\x0A";
              content:"User-Agent: Internet Exploder\x0D\x0A";
              content:"Content-Length:16\x0D\x0a";
              content:"\x0D\x0a";
              );


myflow1 > (content:"password=letmein";);
```

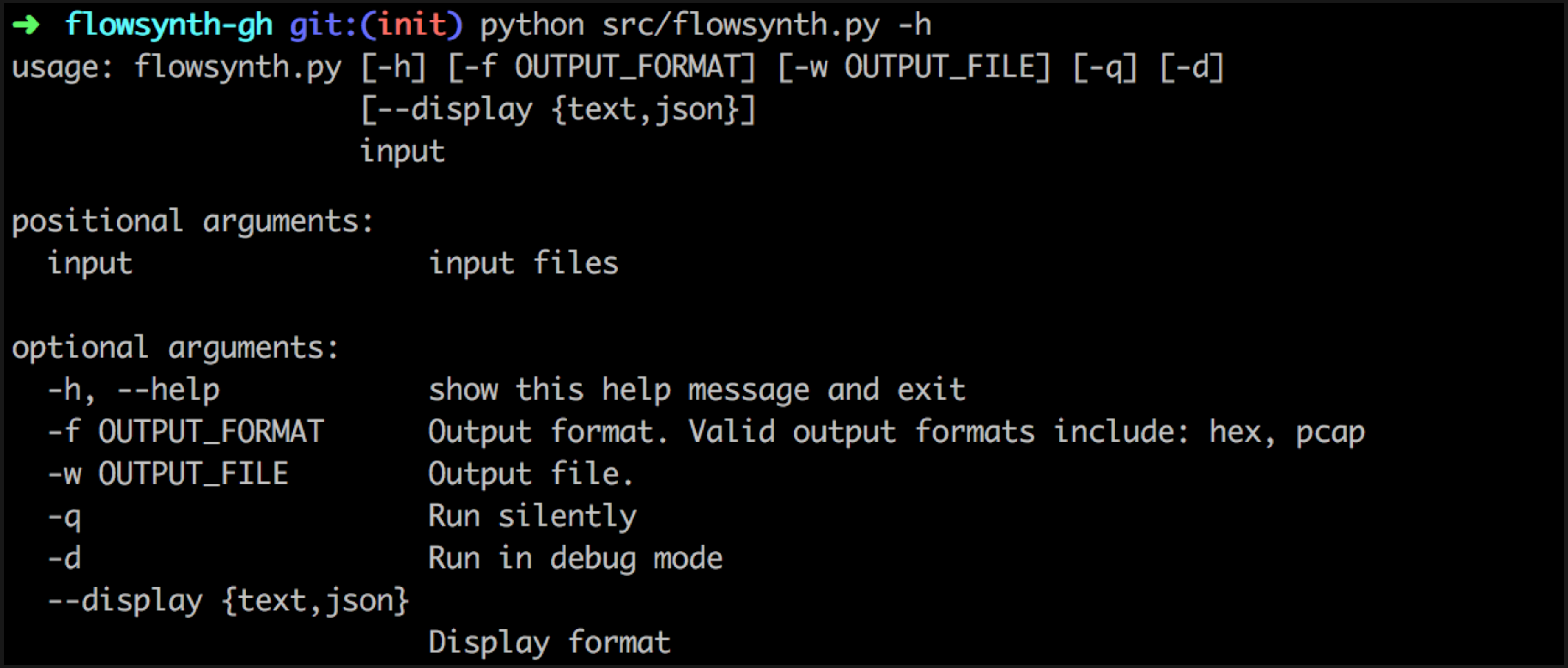


Some fancier things…

- filecontent: load content from file on disk
  `filecontent:”/tmp/ek-landingpage.html”`
- tcp.seq – sets the TCP sequence number
  `tcp.seq:150;`
- tcp.ack – sets the TCP acknowledgement number
  `tcp.ack:9000;`
- tcp.noack – tells Flowsynth not to create an ACK response for this data
- tcp.flags.syn – set the SYN flag
- tcp.flags.ack – set the ACK flag
- tcp.flags.rst – set the RST flag

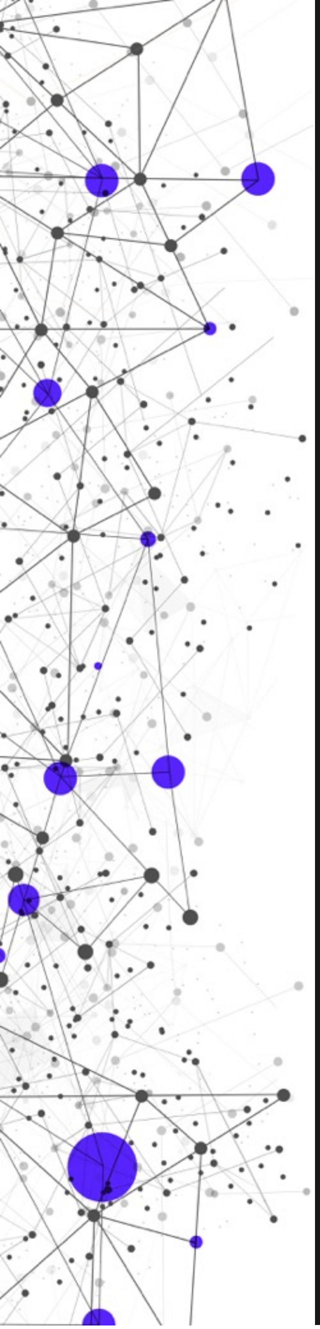
Using Flowsynth

- Python 2.x
- Scapy (and its dependencies)



```
➜  flowsynth-gh git:(init) python src/flowsynth.py -h
usage: flowsynth.py [-h] [-f OUTPUT_FORMAT] [-w OUTPUT_FILE] [-q] [-d]
                    [--display {text,json}]
                    input

positional arguments:
  input                 input files

optional arguments:
  -h, --help            show this help message and exit
  -f OUTPUT_FORMAT      Output format. Valid output formats include: hex, pcap
  -w OUTPUT_FILE        Output file.
  -q                    Run silently
  -d                    Run in debug mode
  --display {text,json}
                        Display format
```

DEMO



Dalton

Agenda

- Overview
- Demo
  - So you know what I’m talking about
- Details
- Demo
  - More examples
  - Flowsynth WebUI
- Questions

Common Use Cases

- Testing Rulesets/Ruleset Coverage
- Troubleshooting and Developing Signatures
  - Testing custom signatures
- Testing Variable Changes
- Testing Configuration Changes
- Testing specific IDS engine behavior
- Crafting custom packet captures
  - Flowsynth WebUI

Common questions I ask

- "Are we covered?"
- "Does my rule alert as expected on our sensor(s)?"
  - "Does this rule work?"
  - "Why not?"
  - "What about this other sensor / version?"

Secureworks®

Background

- A Brief History of Dalton
- Code Caveats

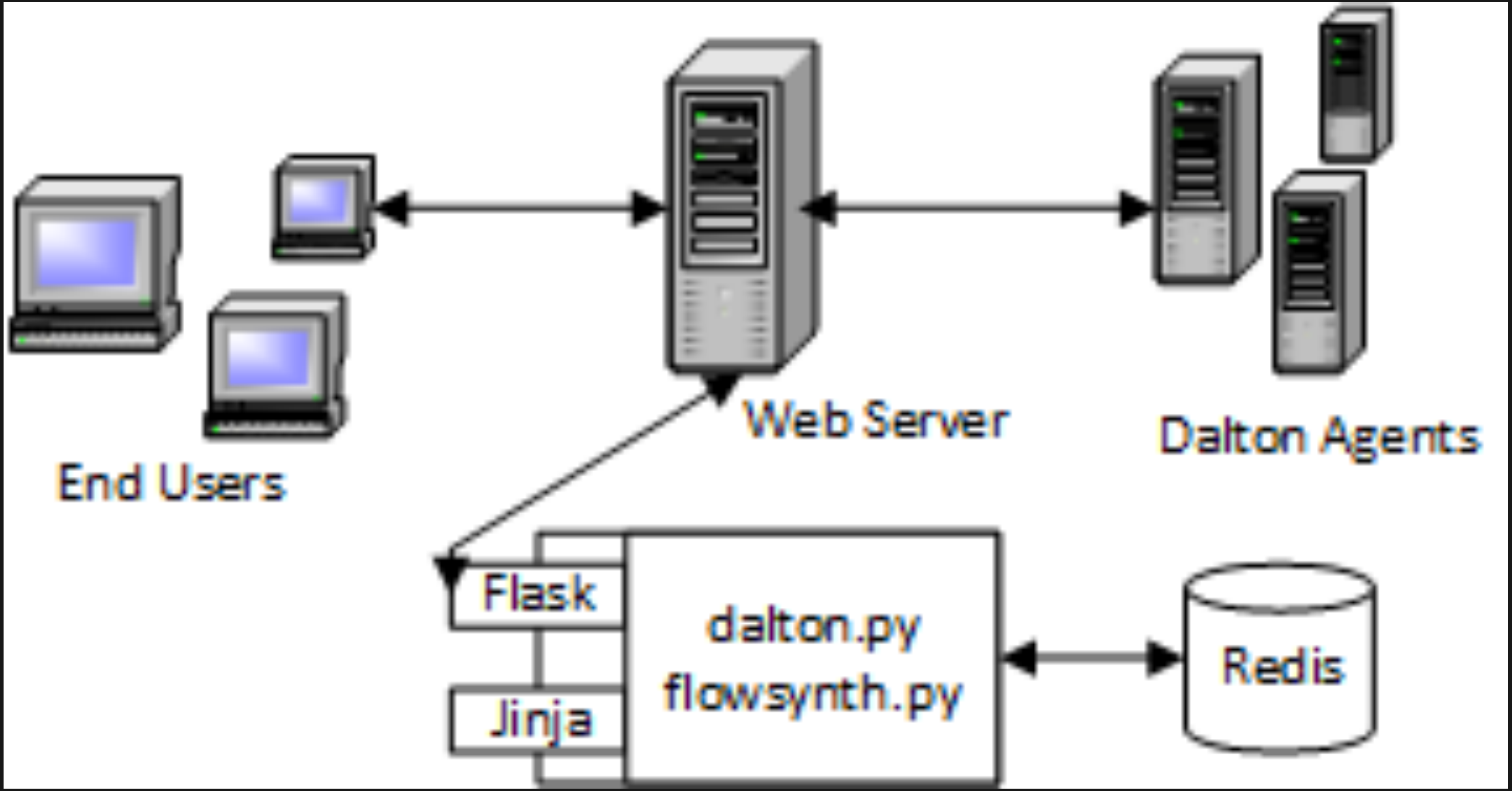
Functionality

Summary

- Test pcaps on sensors against rulesets
- Engines supported
  - Suricata
  - Snort
- Rulesets supported:
  - Pre-defined
  - Ad-hoc (custom)
- Inputs:
  - Pcap(s)
  - Rule(s)

Architecture

Summary



Requirements

- Docker
- Docker Compose
- Internet Connection (to build)

Quickstart

- `./start-dalton.sh`

Or (same thing):

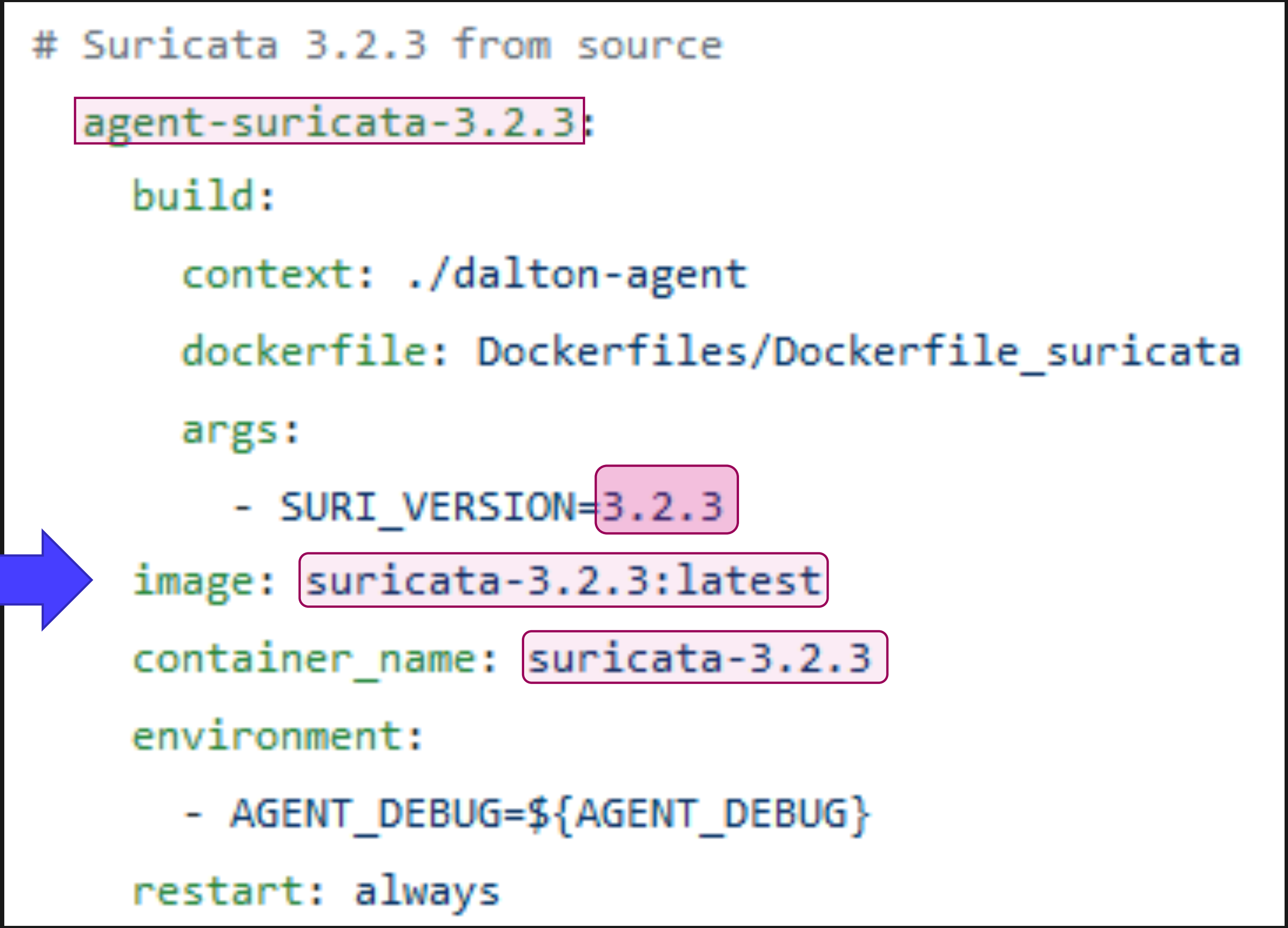
- `docker-compose build && docker-compose up -d`

- Then navigate to `http://<docker-host>/dalton/`

docker-compose.yml

Agent definition

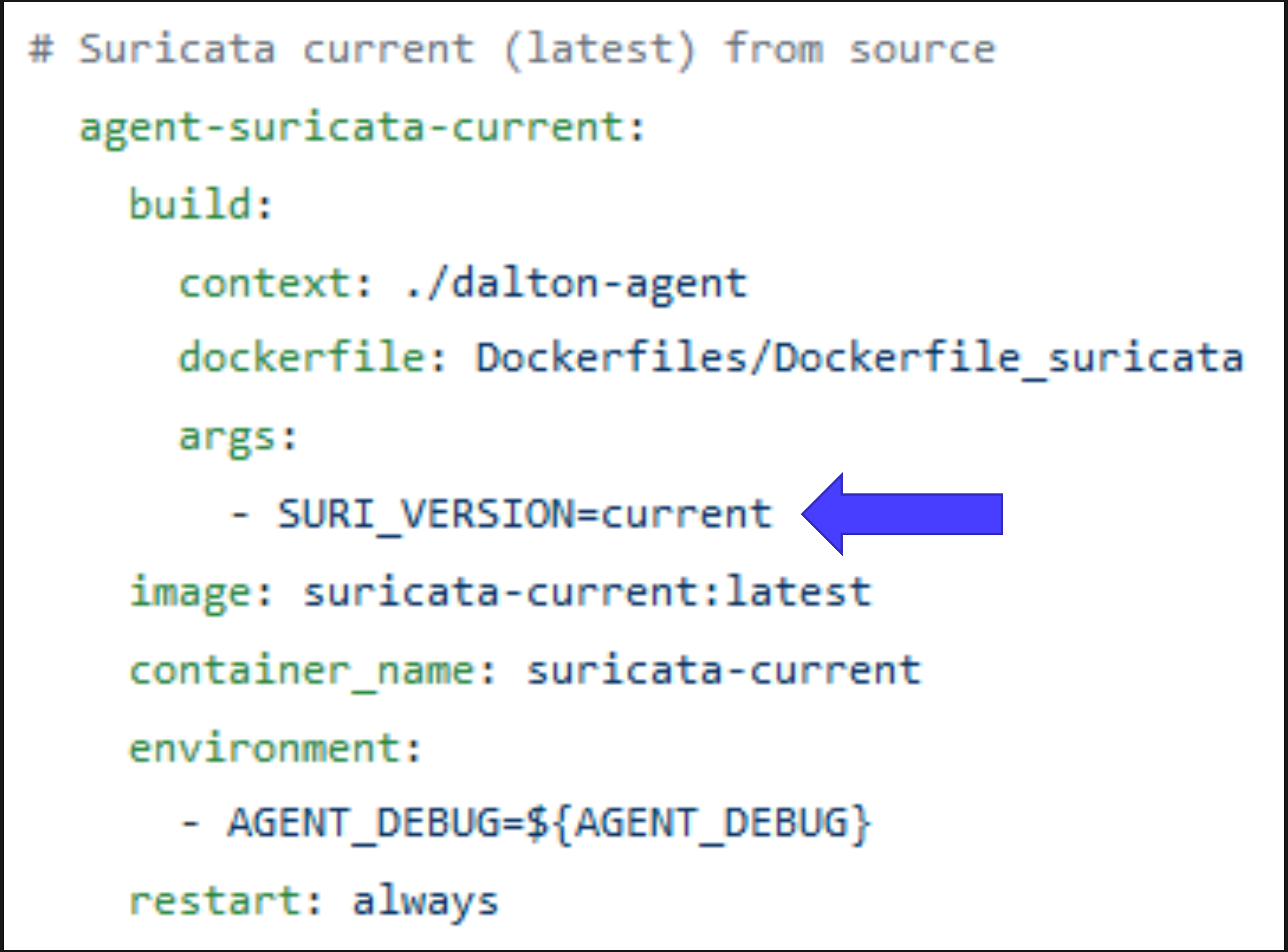
For multiple sensors of the same version, can use same image



```
# Suricata 3.2.3 from source
  agent-suricata-3.2.3:
    build:
      context: ./dalton-agent
      dockerfile: Dockerfiles/Dockerfile_suricata
      args:
        - SURI_VERSION=3.2.3
    image: suricata-3.2.3:latest
    container_name: suricata-3.2.3
    environment:
      - AGENT_DEBUG=${AGENT_DEBUG}
    restart: always
```

docker-compose.yml

Suricata current



```
# Suricata current (latest) from source
  agent-suricata-current:
    build:
      context: ./dalton-agent
      dockerfile: Dockerfiles/Dockerfile_suricata
      args:
        - SURI_VERSION=current
    image: suricata-current:latest
    container_name: suricata-current
    environment:
      - AGENT_DEBUG=${AGENT_DEBUG}
    restart: always
```

Non-Docker Agents

Build your own!

- Sensors don't have to be Docker containers
- Requirements:
  - Engine (Suricata or Snort)
  - Python
  - `dalton-agent.py`
  - `dalton-agent.conf`
  - (network connection between sensor and Dalton Controller)

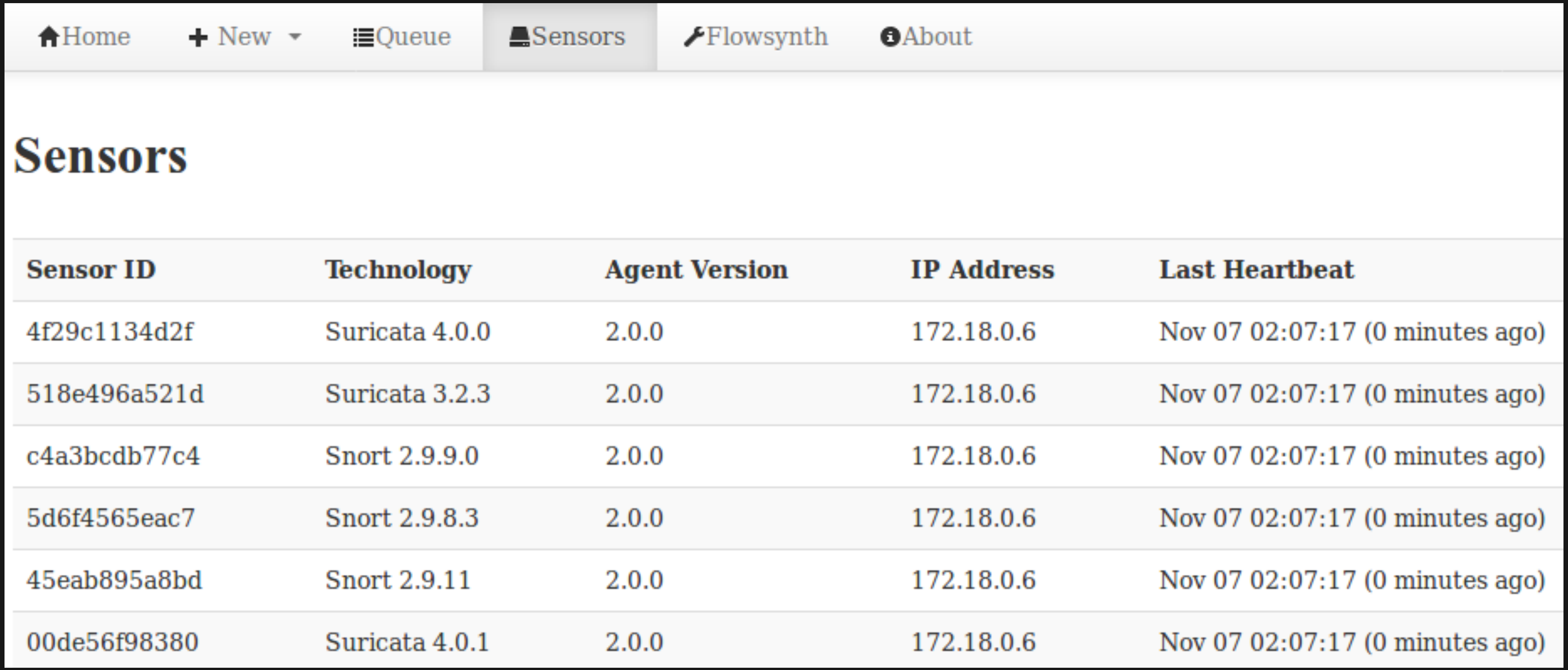
Agents

Sensors

- Poll Controller periodically to see if there are new jobs
  - Default: every 1 second
- If job for Agent, job is retrieved, run, and results reported back
- `dalton-agent.py`
- `dalton-agent.conf`
  - Technology supported
  - Address of Controller
  - Poll interval
  - See inline comments

Agents

continued…



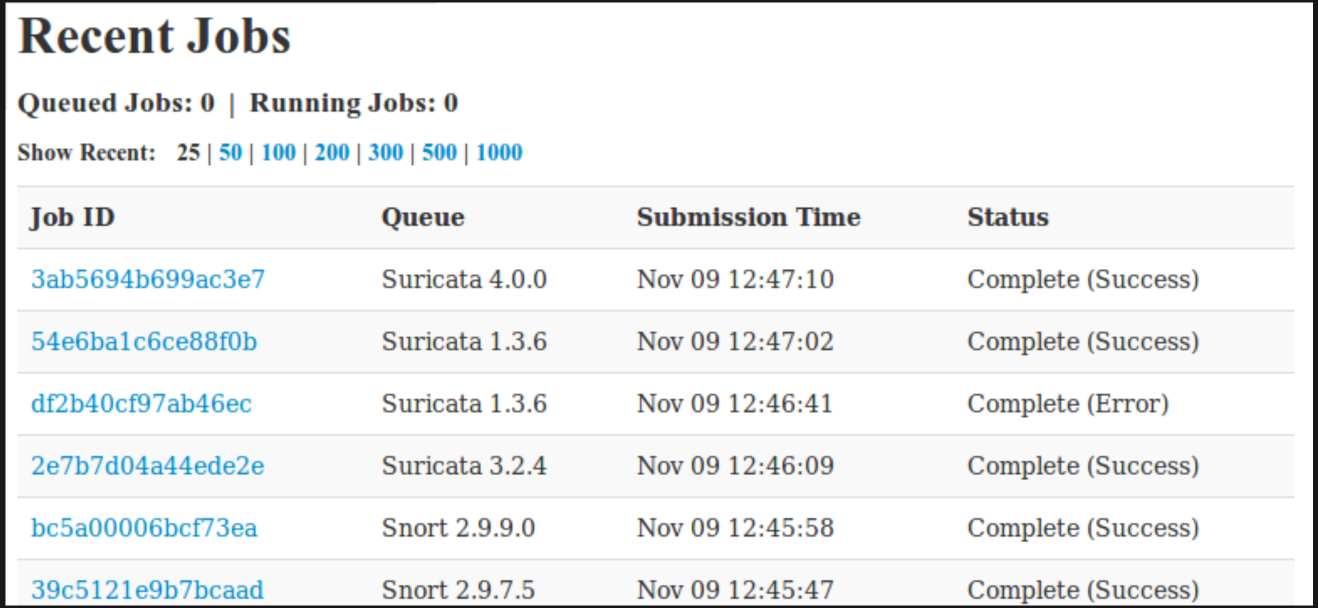
Sensors

Sensor ID	Technology	Agent Version	IP Address	Last Heartbeat
4f29c1134d2f	Suricata 4.0.0	2.0.0	172.18.0.6	Nov 07 02:07:17 (0 minutes ago)
518e496a521d	Suricata 3.2.3	2.0.0	172.18.0.6	Nov 07 02:07:17 (0 minutes ago)
c4a3bcdb77c4	Snort 2.9.9.0	2.0.0	172.18.0.6	Nov 07 02:07:17 (0 minutes ago)
5d6f4565eac7	Snort 2.9.8.3	2.0.0	172.18.0.6	Nov 07 02:07:17 (0 minutes ago)
45eab895a8bd	Snort 2.9.11	2.0.0	172.18.0.6	Nov 07 02:07:17 (0 minutes ago)
00de56f98380	Suricata 4.0.1	2.0.0	172.18.0.6	Nov 07 02:07:17 (0 minutes ago)

- **Automatically added to Controller when they first poll**
- **Removed after period of inactivity**
  - **Default: 20 mins (see `agent_purge_time` in `dalton.conf`)**

Secureworks®

Queue



Recent Jobs

Queued Jobs: 0 | Running Jobs: 0

Show Recent: 25 | 50 | 100 | 200 | 300 | 500 | 1000

Job ID	Queue	Submission Time	Status
3ab5694b699ac3e7	Suricata 4.0.0	Nov 09 12:47:10	Complete (Success)
54e6ba1c6ce88f0b	Suricata 1.3.6	Nov 09 12:47:02	Complete (Success)
df2b40cf97ab46ec	Suricata 1.3.6	Nov 09 12:46:41	Complete (Error)
2e7b7d04a44ede2e	Suricata 3.2.4	Nov 09 12:46:09	Complete (Success)
bc5a00006bcf73ea	Snort 2.9.9.0	Nov 09 12:45:58	Complete (Success)
39c5121e9b7bcaad	Snort 2.9.7.5	Nov 09 12:45:47	Complete (Success)

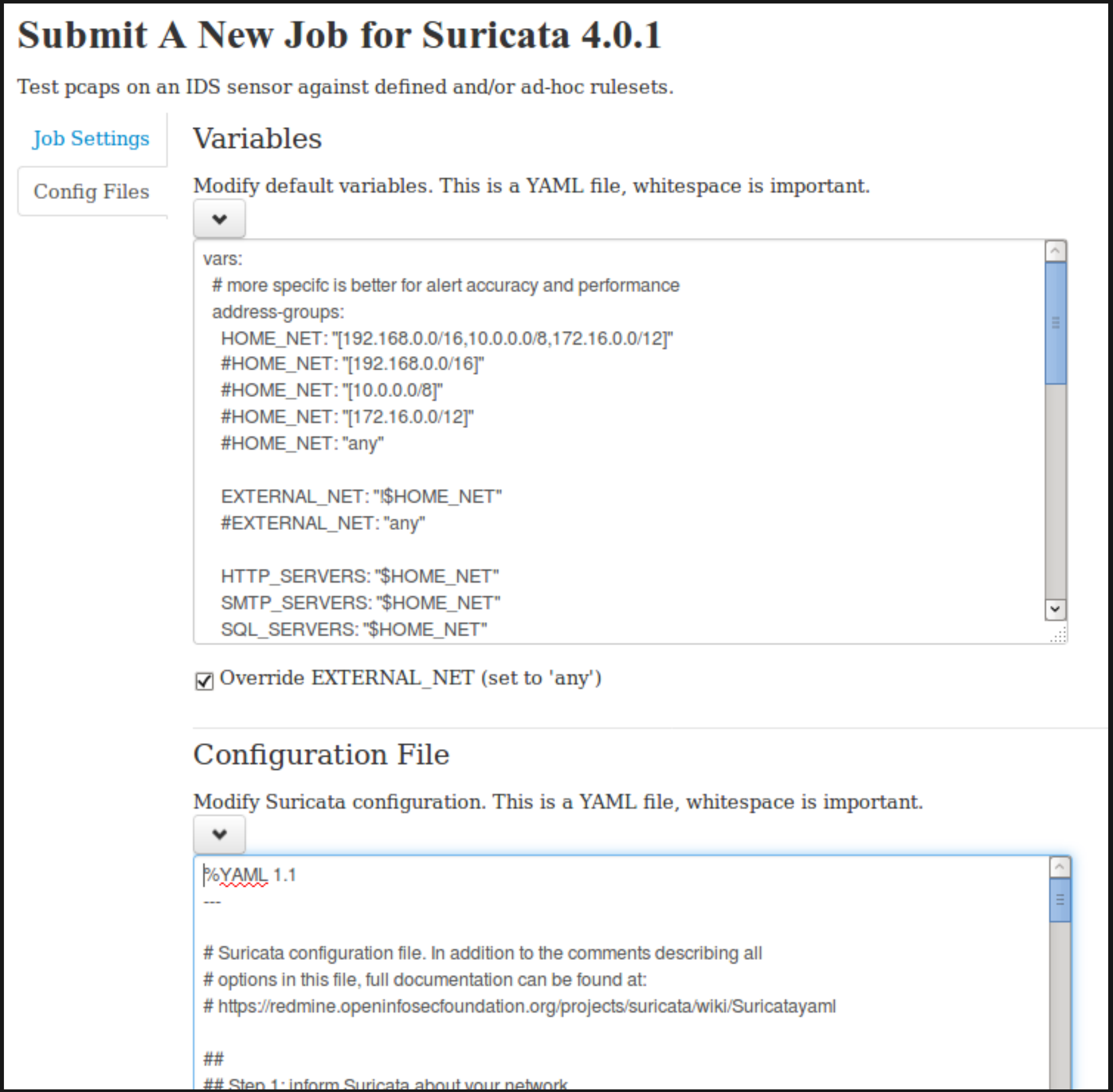
- Jobs cleared out of database periodically based on `redis_expire` (time) parameter in `dalton.conf`
- Jobs aren't purged from *disk* until Queue page loaded
  - Force cleanup by making a HTTP GET to `/dalton/controller_api/delete-old-job-files`
- Jobs can be accessed via `GET /dalton/job/<jobid>`
- Jobs files can be downloaded (link in Web UI on job page); includes pcap, rules, and settings for job.
  - `GET /dalton/sensor_api/get_job/<jobid>.zip`

Secureworks

Dalton Jobs

Config Files

- Edit settings on a per-job basis
- In the UI, variables are separated from the rest of the config
  - Override EXTERNAL_NET enabled by default



Submit A New Job for Suricata 4.0.1

Test pcaps on an IDS sensor against defined and/or ad-hoc rulesets.

Variables

Modify default variables. This is a YAML file, whitespace is important.

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

[✓] Override EXTERNAL_NET (set to 'any')

Configuration File

Modify Suricata configuration. This is a YAML file, whitespace is important.

```
%YAML 1.1
---

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml

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

Sensor Config Files

- e.g. suricata.yaml or snort.conf
- Stored on disk
- Put in `engine-configs/<engine>/` directory on host
  - `engine-configs/suricata/`
  - `engine-configs/snort/`

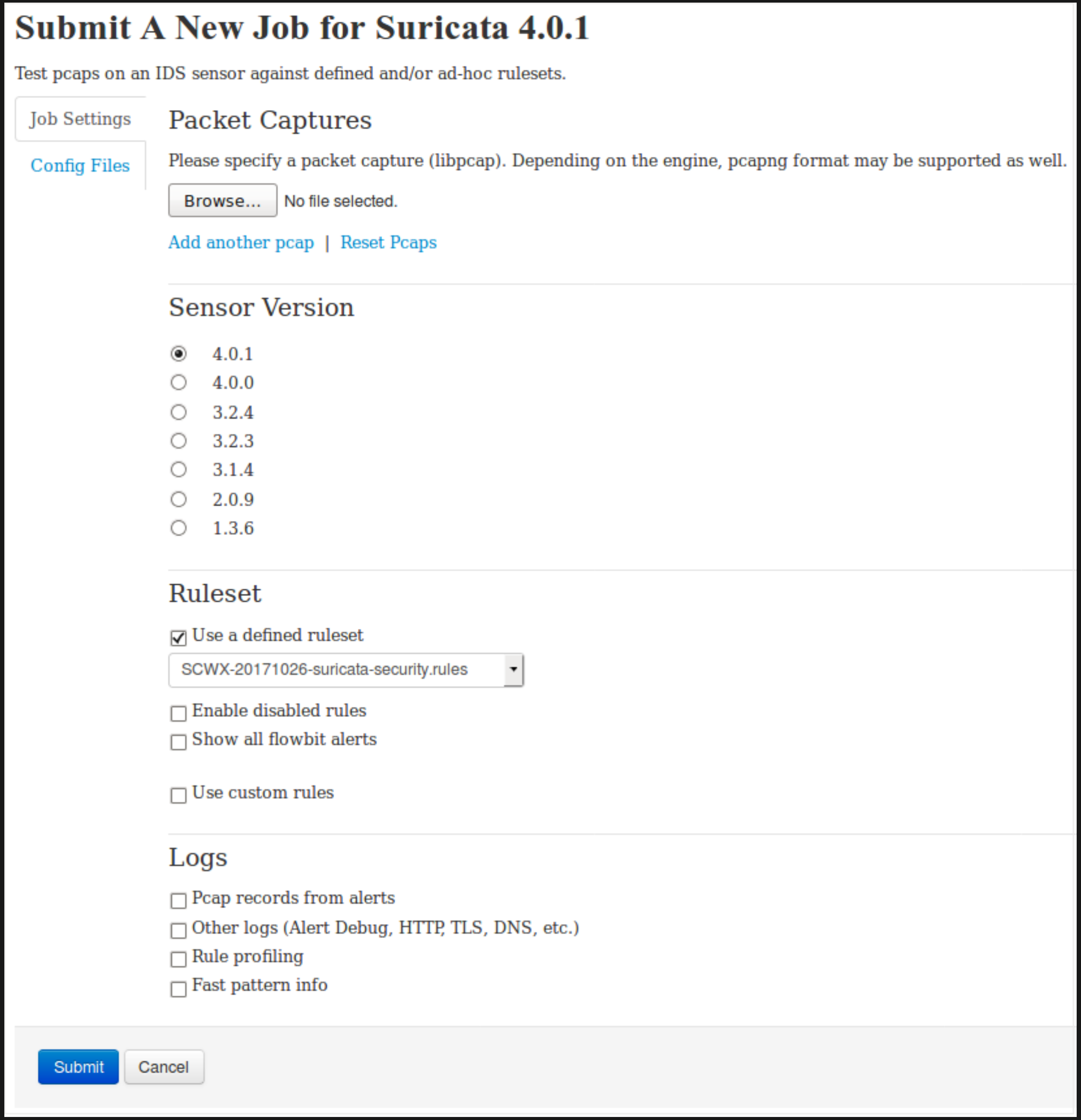
  `or`
- Put in `/opt/dalton/engine-configs/<engine>/` directory on container
- `engine-configs` directory shared with container so you can add on host or container
- Actually, `engine-configs` is a symlink to `app/static/engine-configs` so you can just put in there instead too

Sensor Config Files continued…

- Config filename should match up with SENSOR_TECHNOLOGY string submitted by Agent (controlled by `dalton-agent.conf`).
- Must start with ‘`suricata-`’ for Suricata agents
- Must start with ‘`snort-`’ for Snort agents
- And usually followed by “version” number(s)
- e.g. ‘`suricata-4.0.1.yaml`’
- But as long as the config filename “version” section matches up with Agent’s SENSOR_TECHNOLOGY value then it is fine.
- For example, if a sensor identifies as “suricata-5.9.1’ then the Controller will look for a file starting with ‘`suricata-5.9.1`’ in the `engine-configs/suricata/` directory
- If no exact match found, approximate match attempted based on version number(s)
- Default Suricata `.yaml` and Snort `.conf` files come with Dalton

Dalton Jobs

Job Settings



Submit A New Job for Suricata 4.0.1

Test pcaps on an IDS sensor against defined and/or ad-hoc rulesets.

Packet Captures

Please specify a packet capture (libpcap). Depending on the engine, pcapng format may be supported as well.

Sensor Version

- (●) 4.0.1
- ( ) 4.0.0
- ( ) 3.2.4
- ( ) 3.2.3
- ( ) 3.1.4
- ( ) 2.0.9
- ( ) 1.3.6

Ruleset

- [✓] Use a defined ruleset

SCWX-20171026-suricata-security.rules

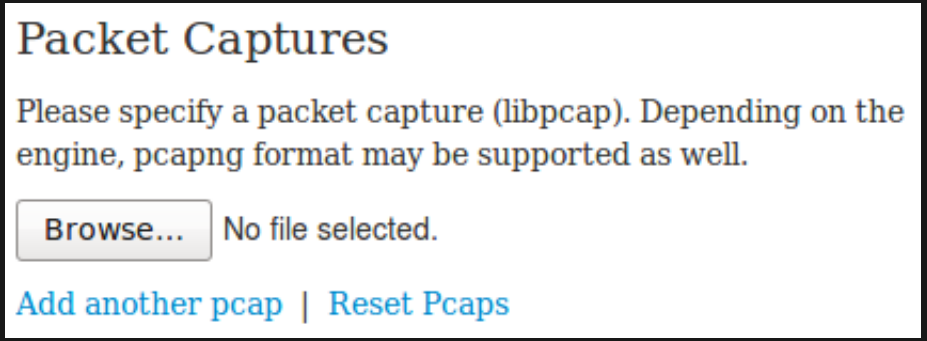
- [ ] Enable disabled rules
- [ ] Show all flowbit alerts
- [ ] Use custom rules

Logs

- [ ] Pcap records from alerts
- [ ] Other logs (Alert Debug, HTTP, TLS, DNS, etc.)
- [ ] Rule profiling
- [ ] Fast pattern info

Dalton Jobs

Job Settings - pcaps



Packet Captures

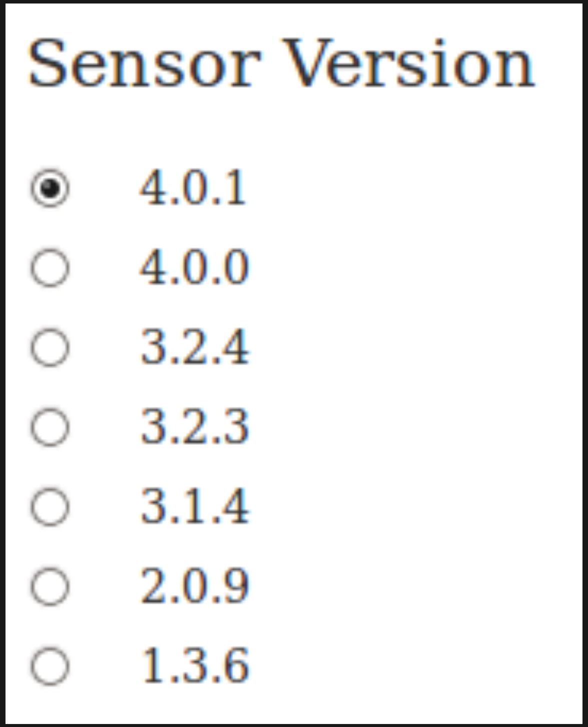
Please specify a packet capture (libpcap). Depending on the engine, pcapng format may be supported as well.

- Multiple pcaps can be sumitted per job
  - Even same one multiple times
  - Max configured by `max_pcap_files` in `dalton.conf`
- libpcap or pcapng, depending on what the sensor supports
- Can submit compressed / archive files!
  - Supported file extensions (and their inferred formats) are `.zip`, `.gz`, `.gzip`, `.bz2`, `.tar`, `.tgz`, and `.tar.gz`
  - If encrypted, password '`infected`' used on zip archives
  - Since zip and tar can contain multiple files, only files with `.pcap`, `.cap`, `.pcapng` extensions extracted
- Suricata jobs with multiple pcaps have pcaps merged on submission (mergecap)
  - Suricata doesn't (currently) support multiple pcaps in read pcap mode

Dalton Jobs

Job Settings – Sensor Version

- Jobs run on selected `SENSOR_TECHNOLOGY` (version)
- Determines queue for job
- Available versions based on current Agents

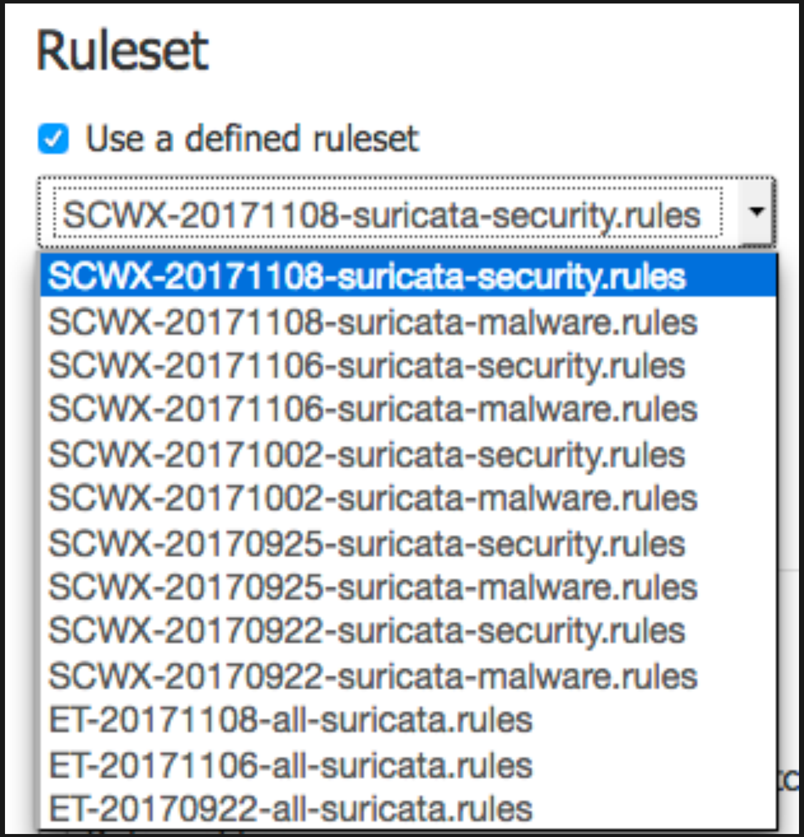


Sensor Version

[✓ 4.0.1]
4.0.0
3.2.4
3.2.3
3.1.4
2.0.9
1.3.6

Dalton Jobs

Job Settings - Pre-defined Rulesets



Ruleset

☑ Use a defined ruleset

SCWX-20171108-suricata-security.rules

- SCWX-20171108-suricata-security.rules
- SCWX-20171108-suricata-malware.rules
- SCWX-20171106-suricata-security.rules
- SCWX-20171106-suricata-malware.rules
- SCWX-20171002-suricata-security.rules
- SCWX-20171002-suricata-malware.rules
- SCWX-20170925-suricata-security.rules
- SCWX-20170925-suricata-malware.rules
- SCWX-20170922-suricata-security.rules
- SCWX-20170922-suricata-malware.rules
- ET-20171108-all-suricata.rules
- ET-20171106-all-suricata.rules
- ET-20170922-all-suricata.rules

- Stored on disk
- All rules in one file
- Put in `rulesets/<engine>/` directory on host
  - `rulesets/suricata/`
  - `rulesets/snort/`

  `or`
- Put in `/opt/dalton/rulesets/<engine>/` directory on container
- `rulesets` directory shared with container so you can add on host or container
- Must end in '`.rules`'

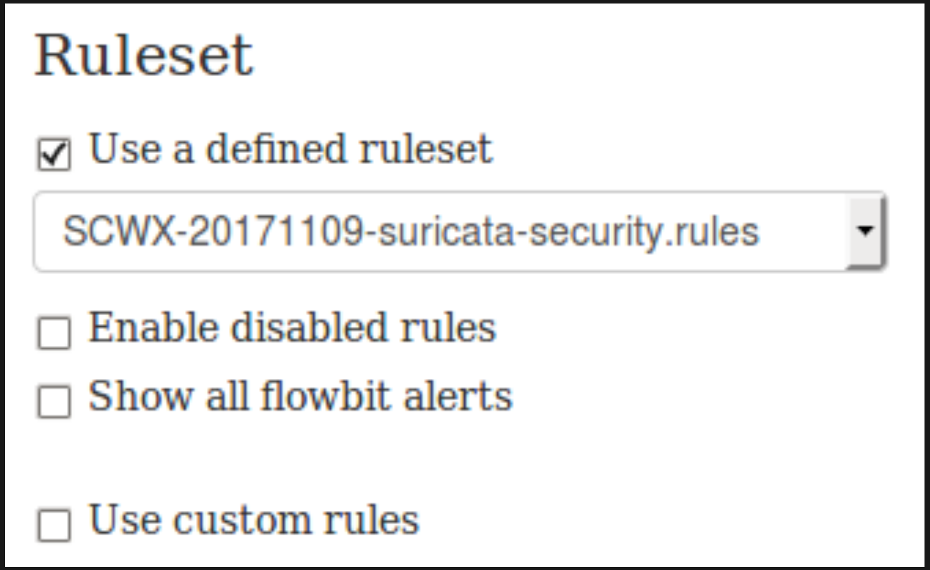
Dalton Jobs

Job Settings - Pre-defined Rulesets

- If no rulesets defined, the Dalton Controller will attempt to download the latest Suricata and Snort rulesets from Emerging Threats Open on launch
- Easy to automate getting new rulesets with tools like `rulecat` or `PulledPork`
- Dalton controller installs `rulecat` when built
- `python /usr/local/lib/python2.7/site-packages/idstools/scripts/rulecat.py --url https://ws.secureworks.com/ti/ruleset/<api_key>/Suricata_suricata-malware_latest.tgz --merged /opt/dalton/rulesets/suricata/SCWX-$(date +"%Y%m%d")-suricata-malware.rules`
- `python /usr/local/lib/python2.7/site-packages/idstools/scripts/rulecat.py --url https://rules.emergingthreats.net/open/suricata-1.3/emerging.rules.tar.gz --merged /opt/dalton/rulesets/suricata/ET-$(date +"%Y%m%d")-all-suricata.rules`

Dalton Jobs

Job Settings – Pre-defined Rulesets



Ruleset

[✓] Use a defined ruleset

SCWX-20171109-suricata-security.rules

[ ] Enable disabled rules

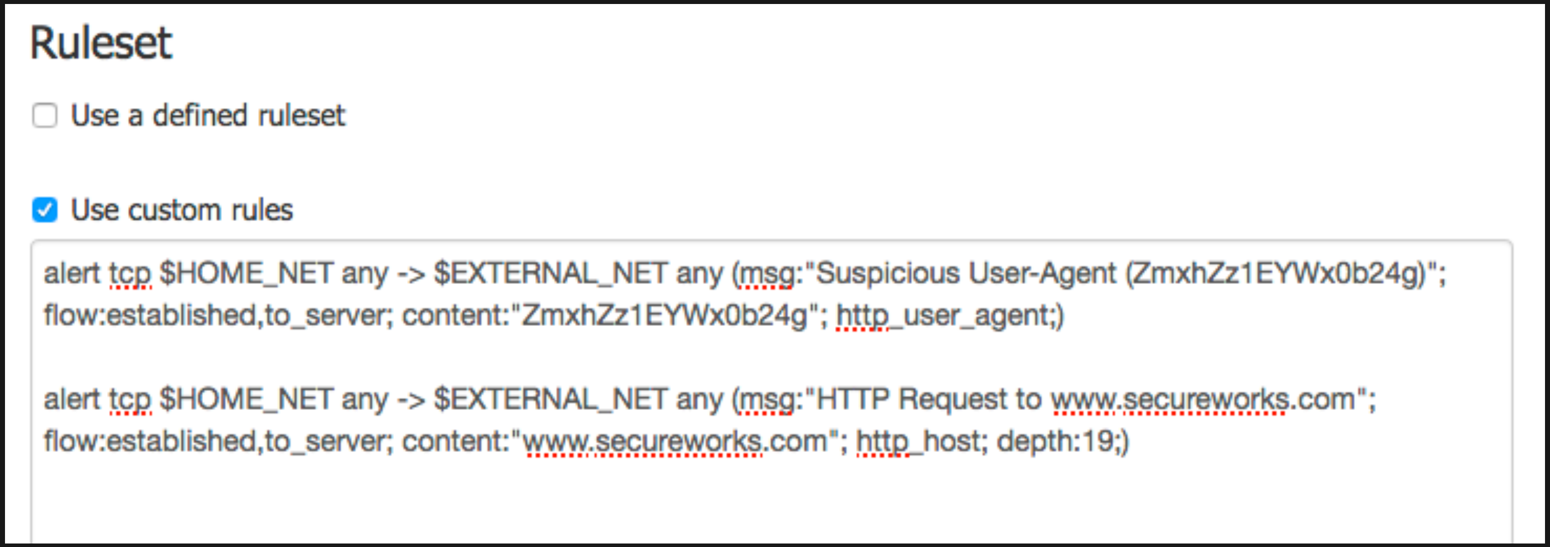
[ ] Show all flowbit alerts

[ ] Use custom rules

- Enable disabled rules
  - May cause issues with variables depending on ruleset and config
- Show all flowbit alerts
  - Removes '`flowbits:noalert;`' from rules

Dalton Jobs

Custom Rules



Ruleset

[ ] Use a defined ruleset

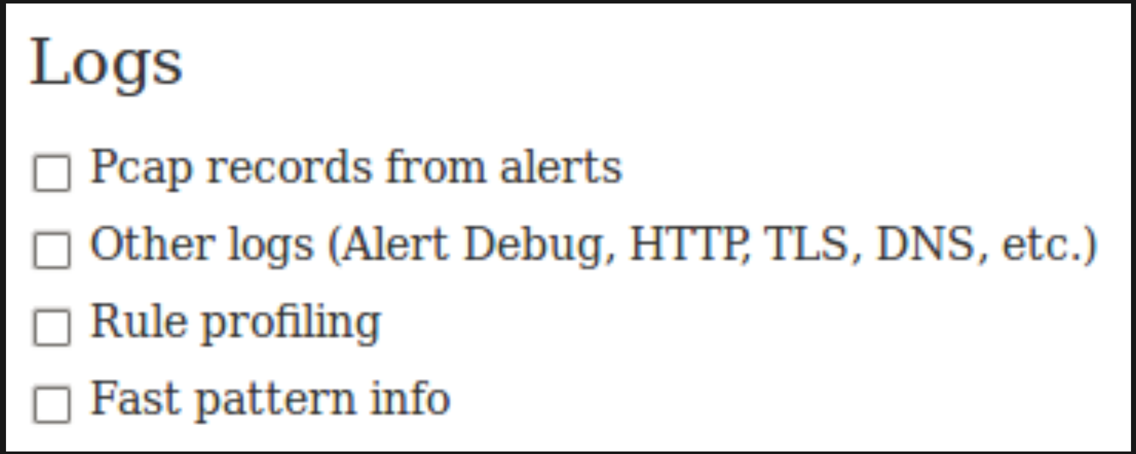
[✓] Use custom rules

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Suspicious User-Agent (ZmxhZz1EYWx0b24g)"; flow:established,to_server; content:"ZmxhZz1EYWx0b24g"; http_user_agent;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP Request to www.secureworks.com"; flow:established,to_server; content:"www.secureworks.com"; http_host; depth:19;)
```

- Can be used with or without pre-defined ruleset
- Literally (becomes) a rules file
  - One rule per line
  - Lines starting with '#' are ignored (and blank lines)
- (next to) No error checking
- If no '`sid`' provided, one is added
- I use this every time I craft a rule

Dalton Jobs

Logs


Logs

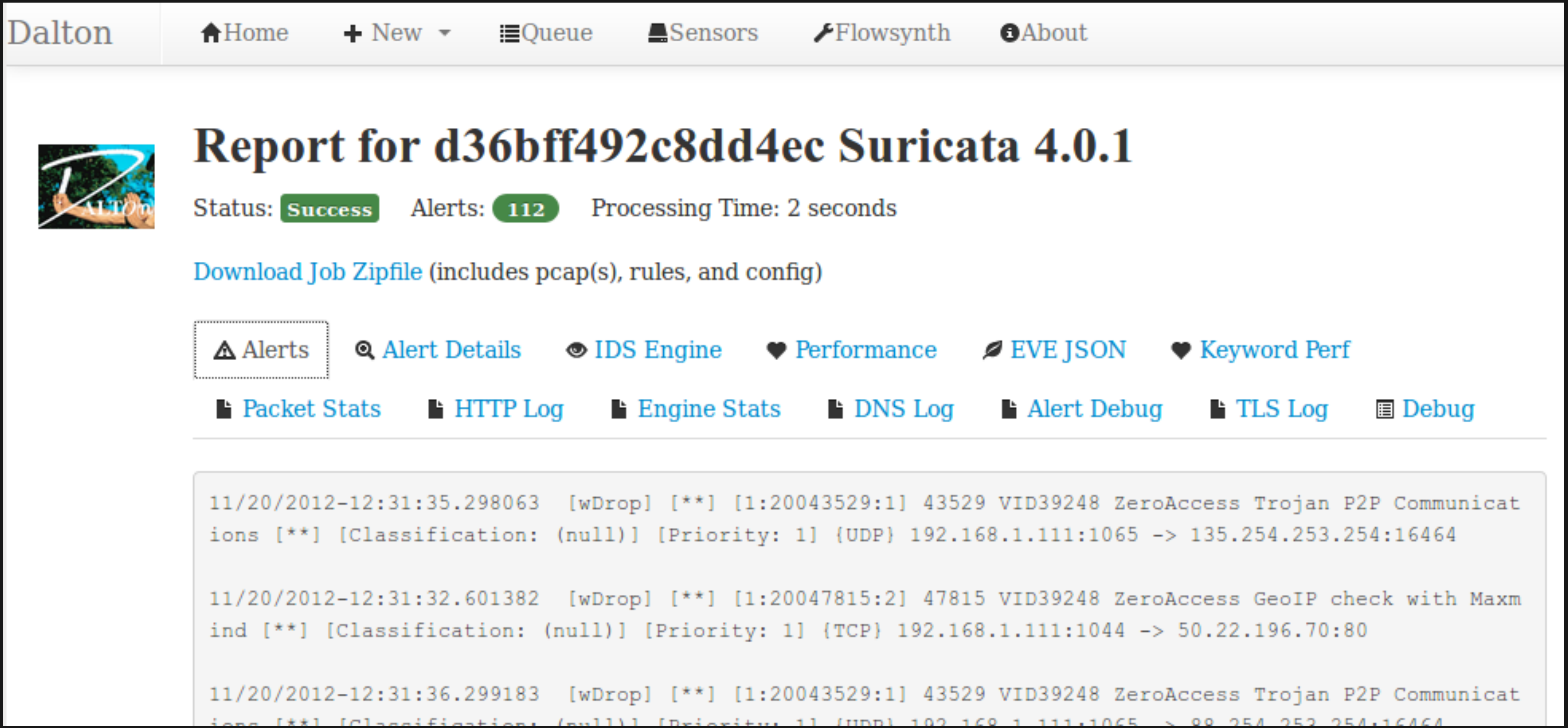
- ☐ Pcap records from alerts
- ☐ Other logs (Alert Debug, HTTP, TLS, DNS, etc.)
- ☐ Rule profiling
- ☐ Fast pattern info

- Pcap records from alerts
  - Really just unified2 data so includes ExtraData records
- Other logs (Suricata only)
  - Engine Stats*
  - Packet Stats*
  - Alert Debug
  - HTTP Log
  - DNS Log
  - TLS Log
  - EVE Log

  *returned even if “Other logs” not checked
- Rule profiling
  - From the engine's rule performance profiling output
- Fast pattern info (Suricata only)
  - Requires two Suricata runs

Dalton Jobs

Job Results



Report for d36bff492c8dd4ec Suricata 4.0.1

Status: Success Alerts: 112 Processing Time: 2 seconds

Download Job Zipfile (includes pcap(s), rules, and config)

```
11/20/2012-12:31:35.298063  [wDrop] [**] [1:20043529:1] 43529 VID39248 ZeroAccess Trojan P2P Communicat
ions [**] [Classification: (null)] [Priority: 1] {UDP} 192.168.1.111:1065 -> 135.254.253.254:16464

11/20/2012-12:31:32.601382  [wDrop] [**] [1:20047815:2] 47815 VID39248 ZeroAccess GeoIP check with Maxm
ind [**] [Classification: (null)] [Priority: 1] {TCP} 192.168.1.111:1044 -> 50.22.196.70:80

11/20/2012-12:31:36.299183  [wDrop] [**] [1:20043529:1] 43529 VID39248 ZeroAccess Trojan P2P Communicat
ions [**] [Classification: (null)] [Priority: 1] {UDP} 192.168.1.111:1065 -> 88.254.253.254:16464
```

Secureworks

Dalton API

Job Results

- RESTful API to retrieve data about submitted jobs
- Job results available in the Web UI are exposed via API
- Response is JSON
  - values / data not necessarily JSON
  - EVE log is, if that is enabled in config
- `GET /dalton/controller_api/v2/<jobid>/<key>`
- `<key> : [alert|alert_detailed|all|debug|error|ids|other_logs|perf|start_time|statcode|status|submission_time|tech|time|user]`

Teapot Jobs

short and stout

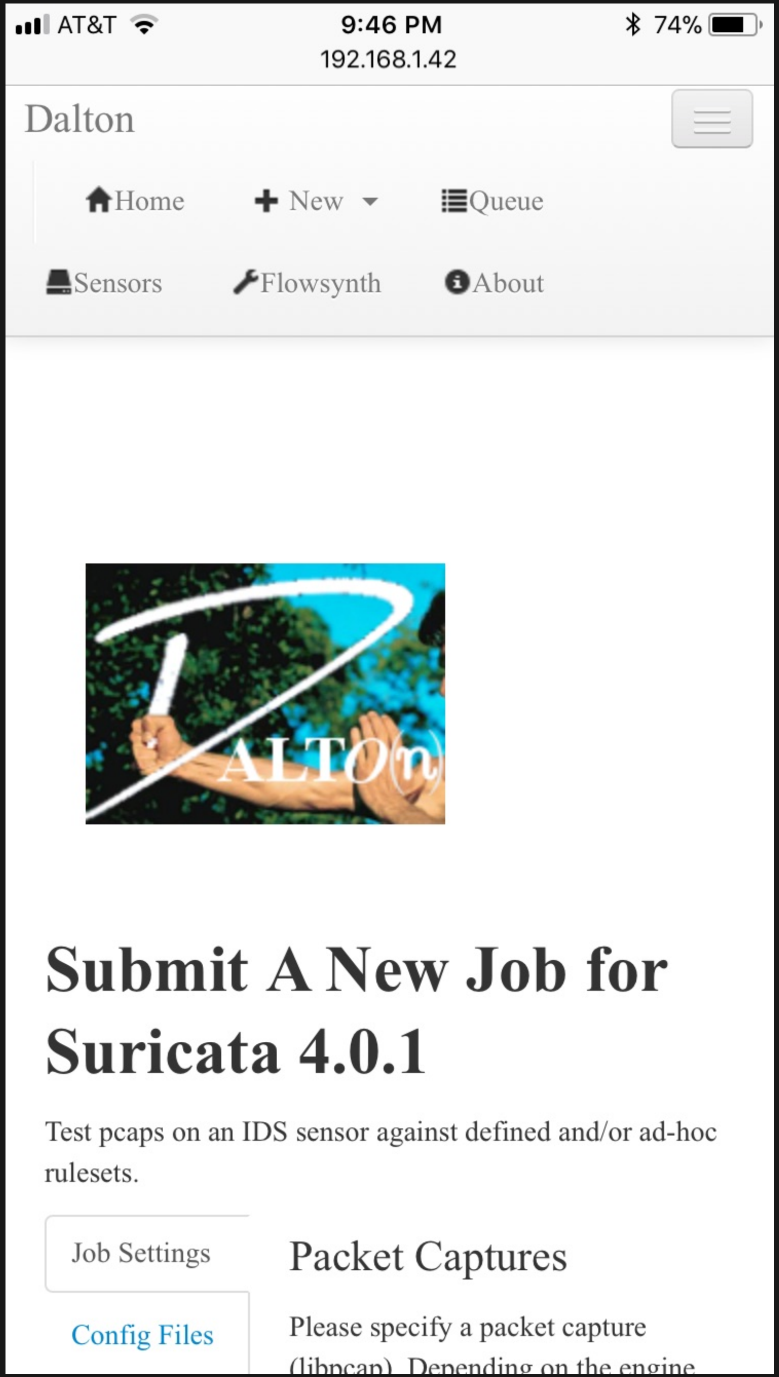
- Short lived
  - job expire timeouts are configured with the `teapot_redis_expire` option in `dalton.conf`
- Like any other job except:
  - Submitted using the `teapotJob` POST parameter (with any value)
  - Have a `job_id` that starts with '`teapot_`‘
  - Submission of a teapot job results in the `job_id` being returned instead of a redirect page
- Designed for voluminous and/or programmatic submissions
  - Although Dalton’s programmatic job submission capabilities are currently less than ideal

The Bad

- No SSL/TLS
  - Can add fairly easily to nginx container
- No authentication or authorization
  - Certainly possible (it has been done before, just not in this open source release)
  - Can DoS controller
- Redis key value limited to 512MB

The Good

- Fast!
- Configurable
  - Open Source
  - Use your own sensors / configs
- Don’t have to submit to 3rd party / Internet
- Can be used locally / offline
  - If built beforehand
- Flowsynth WebUI built-in



Submit A New Job for Suricata 4.0.1

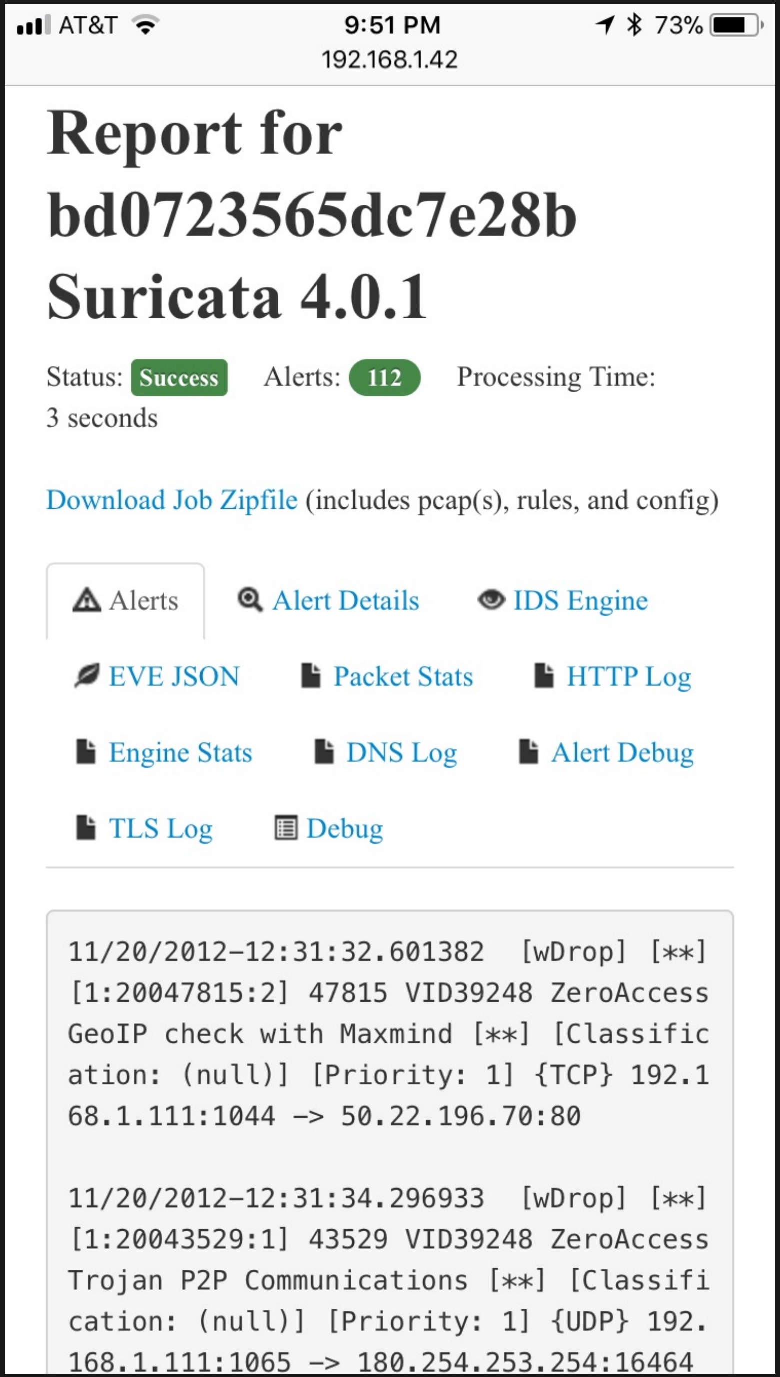
Test pcaps on an IDS sensor against defined and/or ad-hoc rulesets.

Packet Captures

Please specify a packet capture (libpcap). Depending on the engine

Works on mobile!

(Thanks bootstrap!)



Report for bd0723565dc7e28b Suricata 4.0.1

Status: Success Alerts: 112 Processing Time: 3 seconds

Download Job Zipfile (includes pcap(s), rules, and config)

```
11/20/2012-12:31:32.601382  [wDrop] [**]
[1:20047815:2] 47815 VID39248 ZeroAccess
GeoIP check with Maxmind [**] [Classific
ation: (null)] [Priority: 1] {TCP} 192.1
68.1.111:1044 -> 50.22.196.70:80

11/20/2012-12:31:34.296933  [wDrop] [**]
[1:20043529:1] 43529 VID39248 ZeroAccess
Trojan P2P Communications [**] [Classifi
cation: (null)] [Priority: 1] {UDP} 192.
168.1.111:1065 -> 180.254.253.254:16464
```

TODO / Improvements / Ideas

Dalton

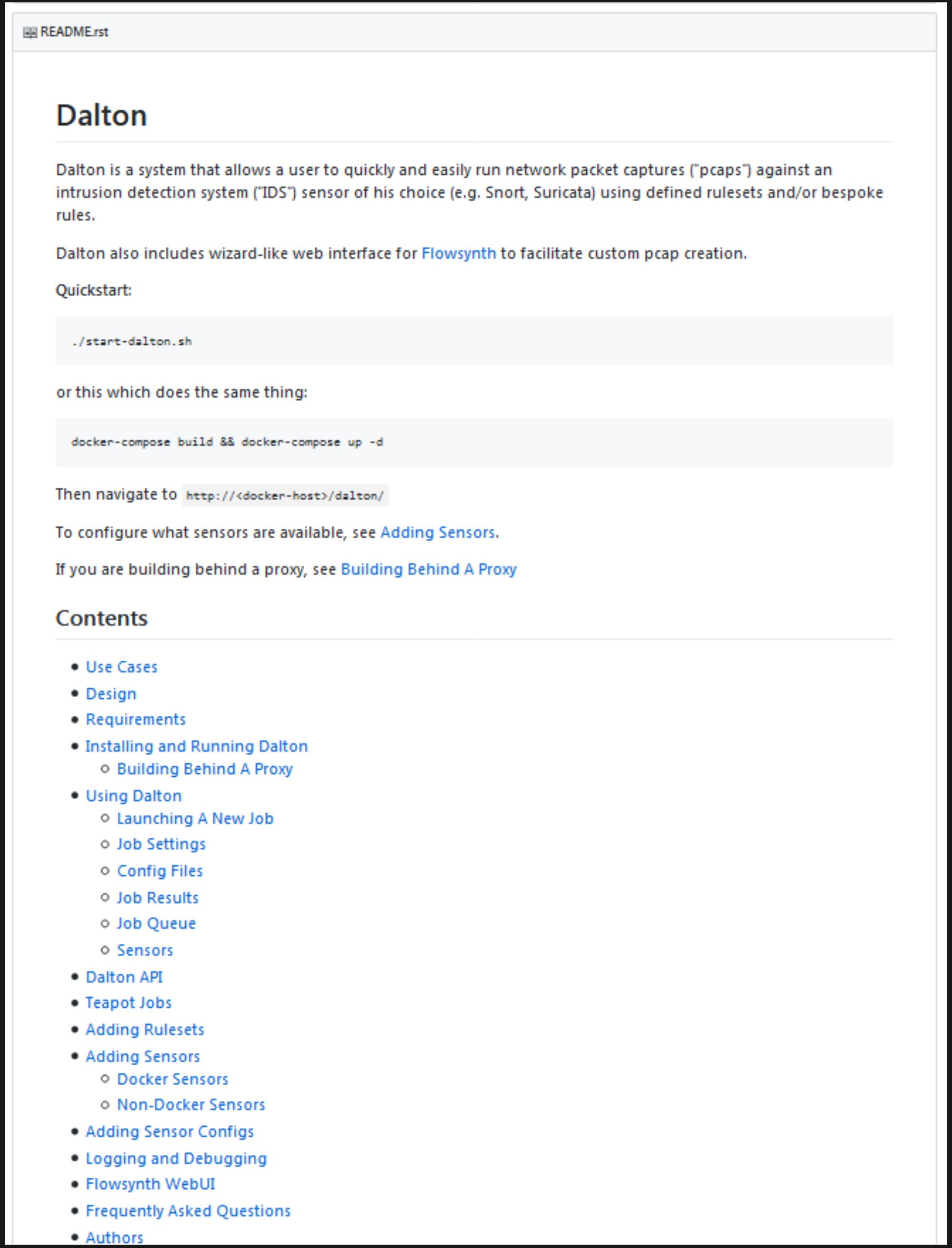
- Job submission API
- Ruleset management from Web UI
  - add / delete / schedule
- Authentication & Authorization
  - Users and Agents
- API route for just EVE JSON log
- Update Bootstrap and/or jquery?
- Templates for Flowsynth WebUI
  - SMTP, FTP, etc.
- Better Docker scripts / Makefile
- Build in corporate proxied environments (fight the proxy)
- Build / publish Docker images
- More eloquent job queue than agents polling every second
- Custom rules - syntax checker
- Automated rule problem finder
- Pcap to Flowsynth language converter (Flowsynth idea)

More Information

- **Read the README**

  (seriously!)

- **And the source code**



README.rst

Dalton

Dalton is a system that allows a user to quickly and easily run network packet captures ("pcaps") against an intrusion detection system ("IDS") sensor of his choice (e.g. Snort, Suricata) using defined rulesets and/or bespoke rules.

Dalton also includes wizard-like web interface for Flowsynth to facilitate custom pcap creation.

Quickstart:

```
./start-dalton.sh
```

or this which does the same thing:

```
docker-compose build && docker-compose up -d
```

Then navigate to `http://<docker-host>/dalton/`

To configure what sensors are available, see Adding Sensors.

If you are building behind a proxy, see Building Behind A Proxy

Contents

Questions?

Dalton:

- https://github.com/secureworks/dalton

Flowsynth:

- https://github.com/secureworks/flowsynth