



# Suricata Language Server

Pass The Salt 2022

# About Me

## Stamus Networks

- Co Founder
- CTO

## Social Media

- @regiteric on Twitter (#jesors)
- <https://www.linkedin.com/in/ericleblond/>



## Open Source and Security

- Suricata
  - Developer
  - Board member of OISF
- Netfilter
  - “Emeritus” core team member



**Founded:** 2014 by Éric Leblond and Peter Manev

**Headquarters:** Indianapolis, USA and Paris, France

**Website:** [www.Stamus-Networks.com](http://www.Stamus-Networks.com)

### Open-Source Solutions:

- *SELKS* – open source Suricata turnkey distro
- *GopherCap* – open source PCAP playback tool
- *Stamus Networks App for Splunk* – integration of SSP and Suricata into Splunk
- *Suricata Language Server* – open source syntax checking and performance guidance for Suricata rules

### Commercial Solutions:

- *Stamus Security Platform* – commercial network threat detection, hunting and response solution

# Introduction to Suricata

Not just an IDS but a fully featured NSM and IDS engine

# Suricata: a threat detection engine

- Born: 2008
- Weight: 600000 lines of code
- Composition: C, Rust
- Eat: live packets and dead ones
- Produce: JSON files
  - Protocol logging
  - IDS Alerts
- Characteristics:
  - High speed
  - Open Source
  - Community driven
  - World famous





# Open Information Security Foundation

- Non-profit foundation organized to build a next generation IDS/IPS engine
  - Pay developers
  - Organize Suricon
  - Financed by consortium members
    - Big companies (Amazon, ...)
    - Startups (Stamus Networks, ...)
    - Governmental organizations (ANSSI, ...)
- Suricon:
  - Yearly user conference
  - Athens, Greece (Nov 9-11)
  - CFP is OPEN!

# Network Intrusion Detection System

- Signature based threat detection
- Language is an evolution of Snort one
  - Yes, sorry
  - With big feature additions
- Available Suricata ruleset
  - Proofpoint:
    - ETOpen
    - ETPro
  - MISP: support of export
  - ...

# Signature examples (½)

## Method Cobalt Strike Malleable C2 JQuery Custom Profile M2 / 1002033658

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Cobalt Strike Malleable C2 JQuery Custom Profile M2"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"/jquery-"; depth:8; content:".min.js"; endswith; http.header; content:"Referer|3a 20|http|3a|//code.jquery.com/|0d 0a|Accept"; fast_pattern; http.accept; content:"text/html,application/xhtml+xml,application/xml|3b|q=0.9,*/*|3b|q=0.8"; bsize:63; http.accept_enc; content:"gzip, deflate"; bsize:13; http.cookie; content:"__cfduid="; depth:9; isdataat:!172,relative; pcre:"/^[A-Za-z0-9_-]{171}$|Rs"; reference:md5,8c9903db02a29847d04d0fd81dd67046; classtype:command-and-control; sid:1002033658; gid:2; rev:3; metadata:affected_product Windows_XP_Vista_7_8_10_Server_32_64_Bit, attack_target Client_Endpoint, created_at 2020_09_22, deployment Perimeter, former_category MALWARE, malware_family Cobalt_Strike, signature_severity Major, updated_at 2022_03_24, mitre_tactic_id TA0011, mitre_tactic_name Command_And_Control, mitre_technique_id T1001, mitre_technique_name Data_Obfuscation;)
```



# Signature examples (2/2)

## Method Observed Malicious SSL Cert (Bazar Backdoor) / 1002032313

```
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"Observed Malicious SSL Cert (Bazar Backdoor)"; flow:established,to_client; tls.cert_subject; content:"C=KZ, ST=Astana, L=Astana, O=NN Fern, OU=KZ System, CN=forenzik.kz"; bsize:66; fast_pattern; reference:url,twitter.com/z0ul_/status/1374121916143919106; reference:md5,4cf6fb8514073319e7759b4f66d13f08; classtype:domain-c2; sid:1002032313; gid:2; rev:1; metadata:attack_target Client_and_Server, created_at 2021_03_23, deployment Perimeter, former_category MALWARE, performance_impact Low, signature_severity Major, tag SSL_Malicious_Cert, updated_at 2021_03_23, mitre_tactic_id TA0042, mitre_tactic_name Resource_Development, mitre_technique_id T1587, mitre_technique_name Develop_Capabilities;)
```

# Writing Suricata Signatures

Bringing you own intelligence to threat intelligence

# Writing Suricata Signatures

Some dare to say out loud what people are thinking quietly



# Process of signatures writing

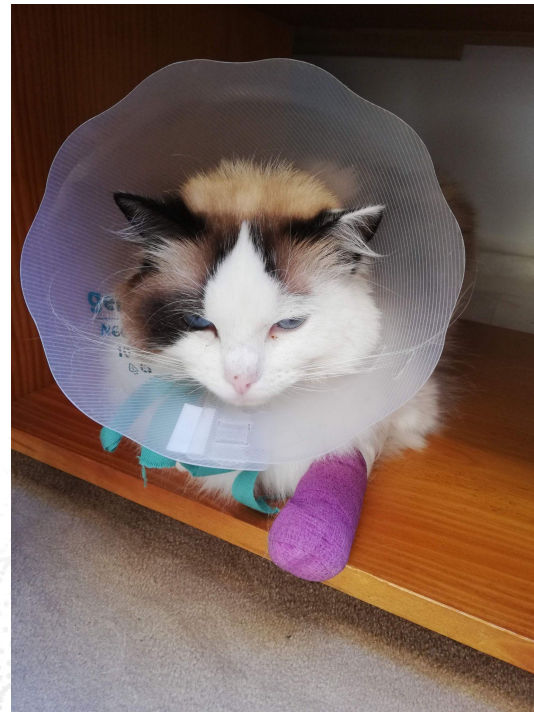
- Writing Suricata Signatures can be a nightmare:
  - Syntax inherited from Snort
  - With important evolution
- It is a repetitive process:
  - Write a signature
  - Check it is matching
  - Refine the signature

# Writing Suricata signatures the old way

- Use your editor
- Check the <https://suricata.readthedocs.io/en/suricata-6.0.4/> for the help
- Test the rules with Suricata
- Fix the rules
- Test the rules with Suricata
- Get a coffee
- Fix the rules
- Test the rules with Suricata
- Test rules performance
- Fix the rules



# Demo



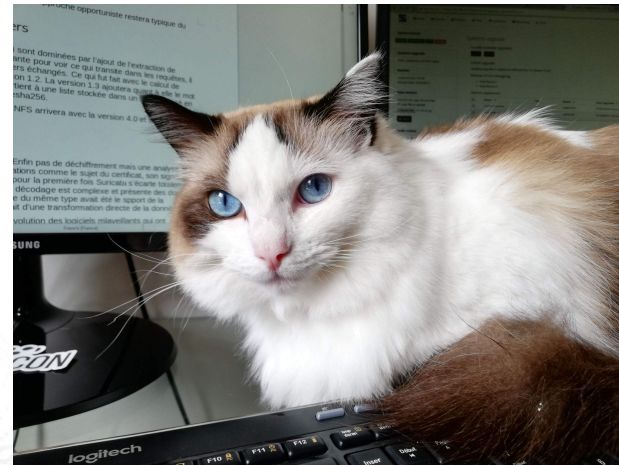
# Suricata helpers

- Test the rules syntax:
  - `suricata -T -S my.rules`
- Analyse the rules:
  - `suricata --engine-analysis -S my.rules`
- Get keywords list that could be helpful:
  - `suricata --list-keywords | grep agent`
- Get documentation for a keyword:
  - `suricata --list-keywords=http.agent`

# Suricata engine analysis

- Suricata running mode doing analysis of ruleset
- 2 main output files:
  - Text file: rules\_analysis.txt
  - JSON format: rules.json
- Display:
  - Syntax errors
  - Warnings about potential performance issues
  - Optimization information
- Use it:
  - `suricata --engine-analysis -S tests/test.rules -l /tmp/`

# Suricata Language Server



# Excuse my French ? Language Server ?

- Language Server Protocol
  - Standardize communication between IDE/Text Editor and Language Server
  - Via JSON RPC
- Features
  - Auto complete
  - Go to definition
  - Find all references
  - Warnings
- More information:
  - <https://microsoft.github.io/language-server-protocol/>



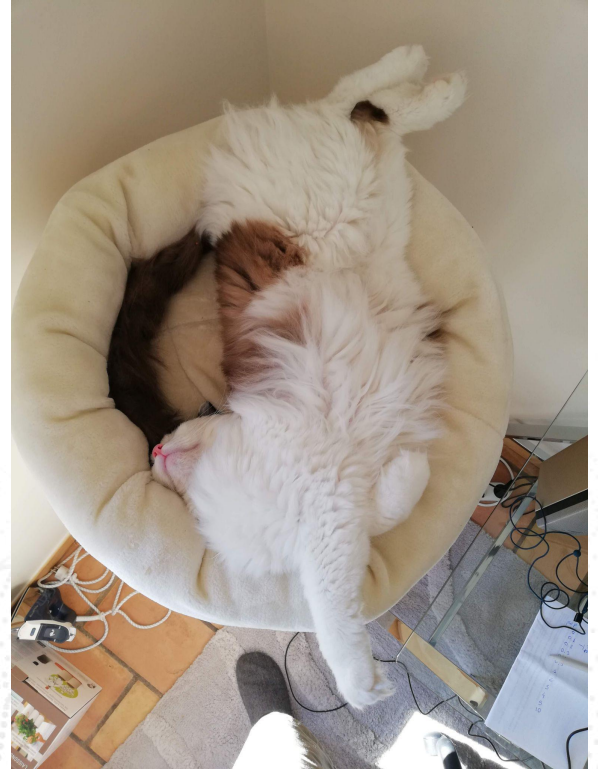
# Language Server support

- Most editors support LSP
  - VScode, Sublime Text, Atom
  - Emacs, Neovim, Kate
- A lot of language support
  - C, Rust, Python, Shell
  - Javascript, Java, C++
  - Latex
  - 176 language implementations are referenced

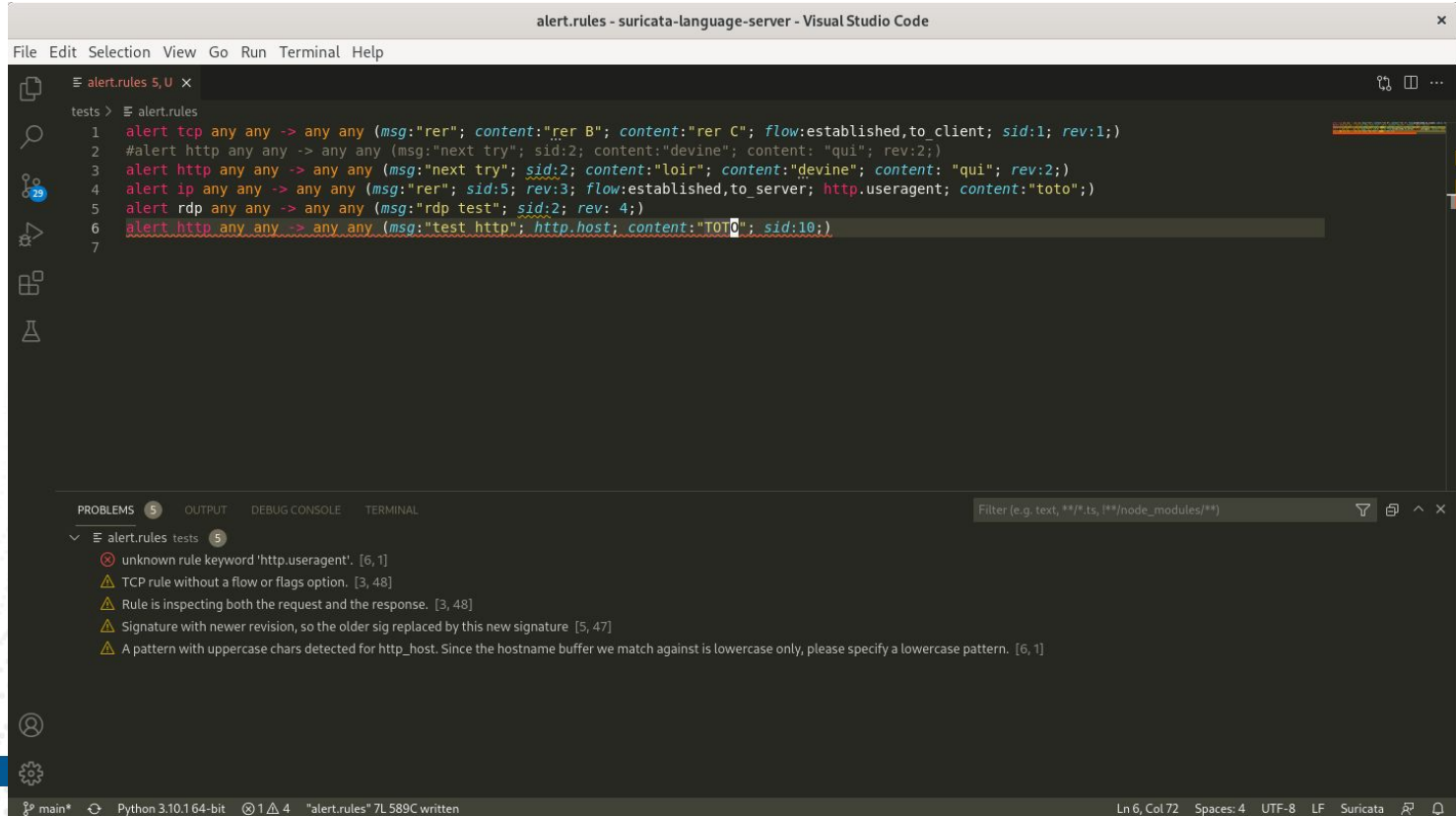
# Suricata Language Server

- A language server for Suricata signature
- Will get you
  - Syntax checking
  - Performance hints
  - Auto completion
- In your preferred editors
- Tested with:
  - Visual Studio Code
  - Neovim
  - Kate
  - Sublime Text 3

# Demo



# VS Code



# Neovim

```
1., alert.rules (6) •
H 1 alert tcp any any → any any (msg:"rer"; content:"rer_B"; content:"rer C"; flow:established,to_client; sid:1; rev:1;)
| 2 #alert http any any → any any (msg:"next try"; sid:2; content:"devine"; content: "qui"; rev:2;)
H 3 alert http any any → any any (msg:"next try"; sid:2; http.us|content:"loir"; content:"devine"; content: "qui"; rev:2;)
| Sticky Buffer http.user_agent Keyword er; http.user_agent; content:"TOT0");
W rev: 4;) ■ Signature with newer revision, so the older sig replace

sticky buffer to match specifically and only on the
HTTP User Agent buffer

[Documentation](https://suricata.readthedocs.io/en/
latest/rules/http-keywords.html#http-user-agent)
```



# Architecture and Availability



# Get information from the source

- Use Suricata helpers to get the information
  - Run suricata commands
  - Parse results
  - Return them as LSP message
- Advantage
  - We get information from **your** version of Suricata
- Disadvantage
  - You need to have a Suricata on your system

# Availability

- Available under GPLv3 license
- Source on Github:
  - <https://github.com/StamusNetworks/suricata-language-server>
- Seems to be pretty stable
- Tested on operating system:
  - Linux
  - Windows
- Tested with editors:
  - Neovim
  - VS code
  - Kate
  - Sublime Text 3

# Architecture

- Suricata Language Server is Python code
- Based on Fortran Language Server:
  - <https://github.com/hansec/fortran-language-server>
  - As it was:
    - Small code base
    - In Python
- Work by spawning Suricata command
  - Borrow Scirius code for that
  - <https://github.com/StamusNetworks/scirius>

# Making sense of warnings





# Directionality warning

- Bad rule

```
1.1 test.rules (1) x
W 4 alert tcp any any → any any (msg:"both ways"; content:"toto"; sid:4; rev:1;) ■ Rule inspect server and client

test.rules 4:65

▼ tests/test.rules 1
| △ Rule inspect server and client side, consider adding a flow keyword Suricata Engine Analysis [4, 64]
```

- Correct one

```
1.1 test.rules x
| 4 alert tcp any any → any any (msg:"both ways"; content:"toto"; flow:established,to_client; sid:4; rev:1;)
```

# Missing HTTP keywords

```
1., test.rules (1) x
1 #alert dns any any → any any (msg:"test dns no fp"; dns.query; content:"windows.com"; nocase; content:"grenui"; endswith; sid:1; rev:1;)
2 #alert dns any any → any any (msg:"test dns no fp"; dns.query; content:"tamere.com"; nocase; endswith; sid:2; rev:1;)
W 3 alert http any any → any any (msg:"let's match this"; content:"GET /toto"; flow:to_client,established; sid:3; rev:1;) ■ pattern looks
```

NORMAL main > 1 test.rules utf-8 < Δ < hog 33% 1:1

▼ tests/test.rules 1  
| Δ pattern looks like it inspects HTTP, use http.request\_line or http.method and http.uri instead for improved performance Suricata Engine

Trouble[-] 3:1

# Mixed content

```
1. test.rules (3) x
H 1 alert dns any any → any any (msg:"test dns no fp"; dns.query; content:"windows.com"; nocase; content:"grenui"; endswith; sid:1; rev:1;)
H 2 alert dns any any → any any (msg:"test dns no fp"; dns.query; content:"tamere.com"; nocase; endswith; sid:2; rev:1;)
H 3 alert http any any → any any (msg:"let's match this"; content:"/toto"; http.user_agent; content:"scirius"; flow:to_server,established; sid:3; rev:1;)
    ■ Fast Pattern "scirius" on http_user_agent

test.rules 3:138

▼ tests/test.rules 3
  △ Application layer "http2" combined with raw match, consider using a match on application buffer Suricata Engine Analysis [3, 137]
  Q Fast Pattern "windows.com" on dns_query Suricata Engine Analysis [1, 73]
  Q Fast Pattern "scirius" on http_user_agent Suricata Engine Analysis [3, 99]

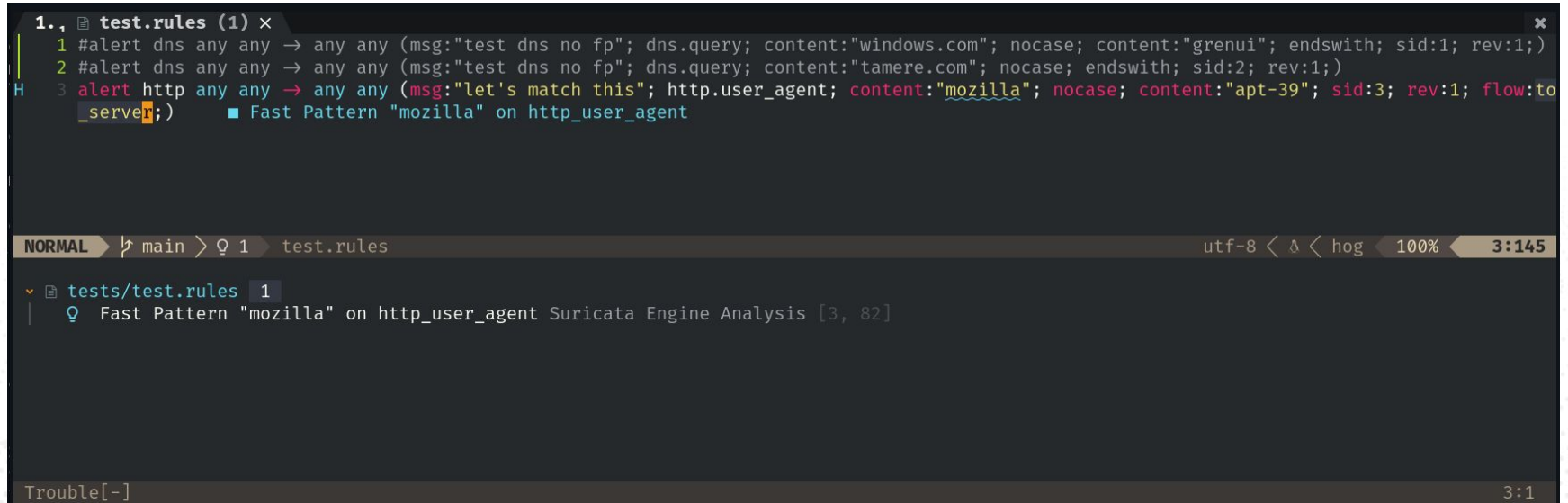
NORMAL main Trouble[-] 60% 3:1
```

# Fast pattern crash course

- Linear evaluation of ruleset is impossible
  - ETPro ruleset as more than 60000 rules
  - At 40Gbps this means 0.05 ns per rule
- Suricata needs to use heavy optimization
- Main one is multi pattern matching
  - Match multiple pattern in various buffer
  - Only evaluate signatures that contains the pattern
  - Ruleset MUST minimized the number of signature using same pattern
- How Suricata picks fast pattern ?
  - Longer pattern in content match
  - Follow *fast\_pattern* keyword instruction

# Fast pattern check

- Fast pattern is main optimization in detection engine
- Need to be done on a differentiator
- Suricata picks the longest pattern



```
1., test.rules (1) x
1 #alert dns any any → any any (msg:"test dns no fp"; dns.query; content:"windows.com"; nocase; content:"grenui"; endswith; sid:1; rev:1;)
2 #alert dns any any → any any (msg:"test dns no fp"; dns.query; content:"tamere.com"; nocase; endswith; sid:2; rev:1;)
H 3 alert http any any → any any (msg:"let's match this"; http.user_agent; content:"mozilla"; nocase; content:"apt-39"; sid:3; rev:1; flow:to
   _server;) ■ Fast Pattern "mozilla" on http_user_agent

NORMAL main > Q 1 test.rules utf-8 < Δ < hog 100% 3:145

▼ tests/test.rules 1
| Q Fast Pattern "mozilla" on http_user_agent Suricata Engine Analysis [3, 82]

Trouble[-] 3:1
```



# Fast pattern And future





# Rules.json on ETPRO

- Analyzed in a Jupyter notebook
  - Parse the JSON
  - Build data structure
- Some patterns are hugely used
  - 3400+ signatures seen on a single content
  - Potential performance impact

# Demo

	pattern	degree
57	22 method 22 3A	3432
5	&amp;cvv=	204
45	C=US, 20 O=Let 27 s 20 Encrypt, 20 CN=Let 27 s 20 Encrypt 20 Authority 20 X3	147
44	&amp;ssn=	72
24	2F default.asp?	65
15	5C 00 p 00 i 00 p 00 e 00 5C 00 00 00	64
14	5C pipe 5C 00	64
21	2F search.asp?	60
58	00 01 86 A0	59
35	2F vehiclelistings.asp?	54
10	PE 00 00	51
1	.exe	51
20	2F search.php?	49
31	2F detail.asp?	49
9	Mozilla 2F 3.0 20 (compatible 3B 20 Indy 20 Library)	48

# SLS future

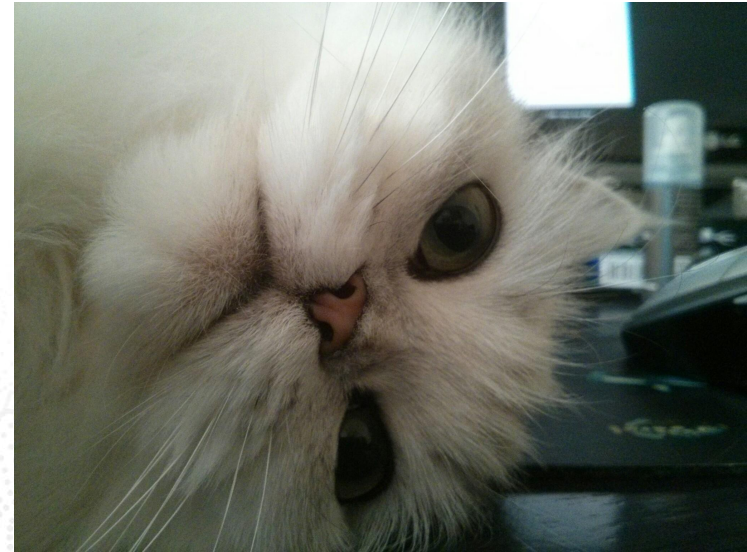
- Known limits
  - Only run complex syntax check on file with less than 1000 lines
- Evolution
  - Add hints about fast pattern usage
    - Find a way to use rules.json on global ruleset
  - Performance analysis
    - Find a way to use profiling
  - Better completion

# Conclusion



# Conclusion

- Suricata Language Server is:
  - Easy to use
  - Gives a lot of information
- Thanks to Suricata developers for the helpers
- Feedback and contributions are welcome



# Questions ?

- Suricata Language Server:
  - <https://github.com/StamusNetworks/suricata-language-server>
  - VS Code plugin: Suricata Intellisense
- Contact me:
  - Twitter: @regiteric
  - Mail: [el@stamus-networks.com](mailto:el@stamus-networks.com)
- Suricon:
  - Athens, Greece, Nov 9, 10, 11, 2022
  - <https://suricon.net/>