History of Lua support in Suricata

You can’t fix things if they ain’t broken first

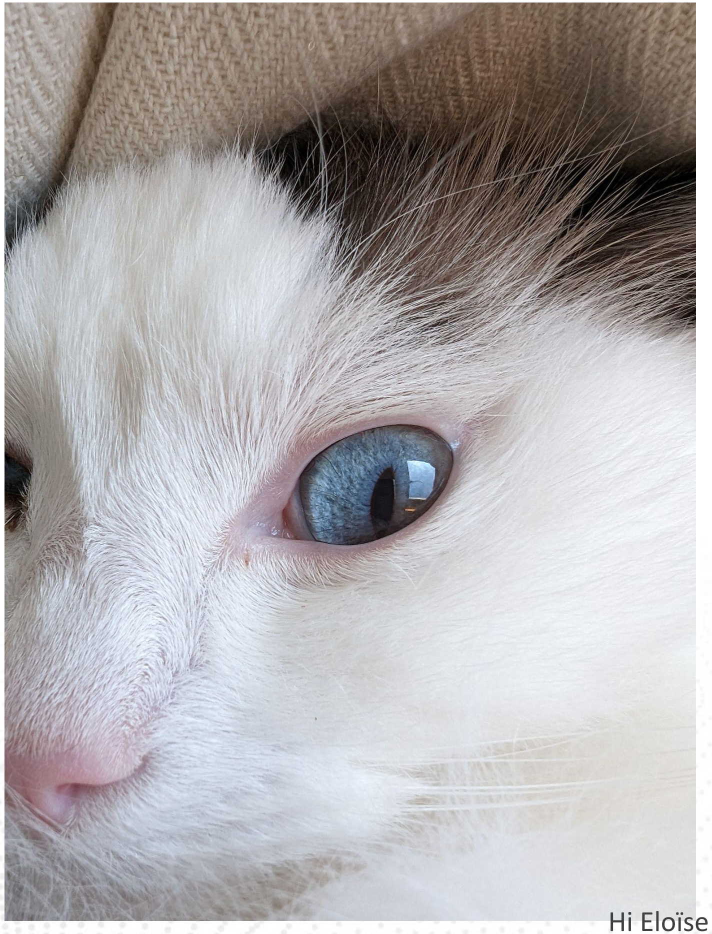
Who am I ?

Eric Leblond

- Co founder & CTO of Stamus Networks
- Member of OISF's board
- Contributor to Suricata since 2009
- Co-author of "The Security Analyst's Guide to Suricata"

Stamus Networks:

- Editor of a Suricata based NDR solution
- Contributor to Suricata



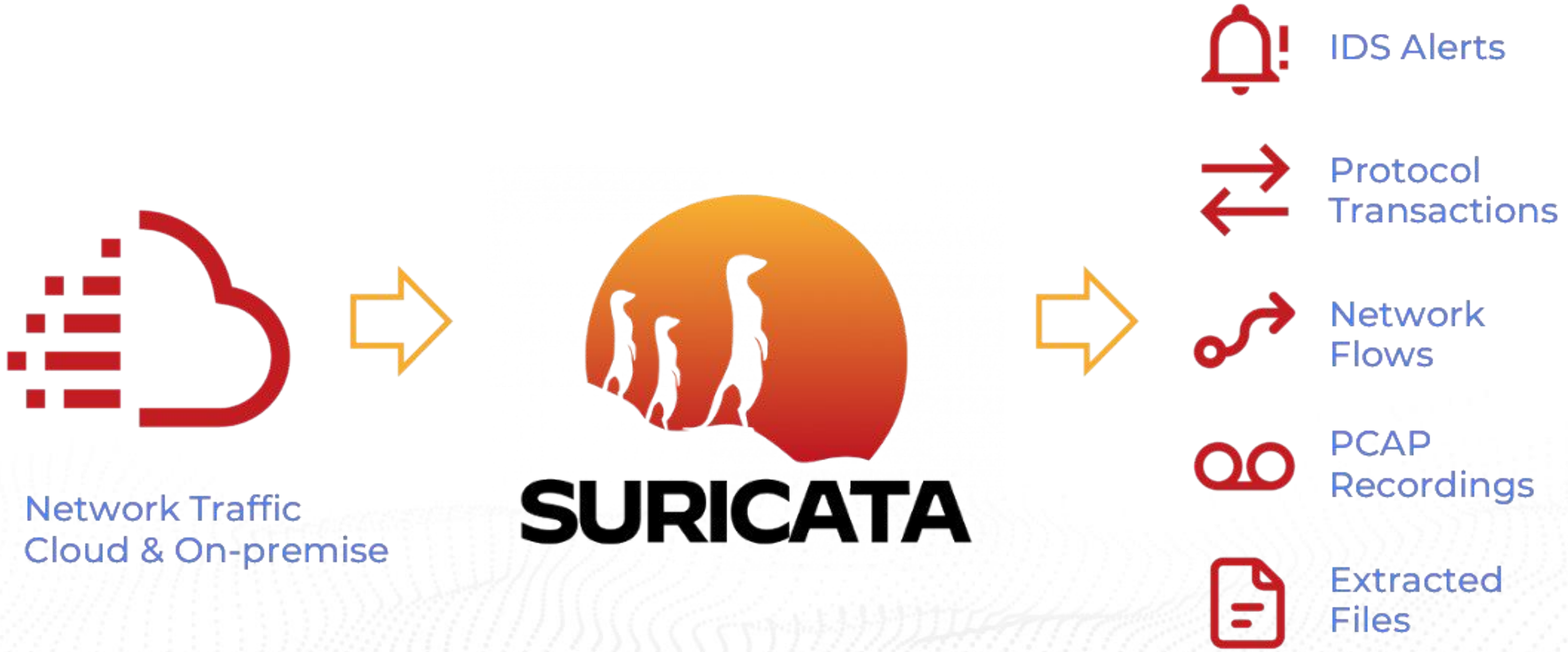
Hi Eloïse ;-)

STAMVS NETWORKS

Suricata

- Born: 2008
- Weight: 600 000 lines of code
- Composition: C, Rust
- Eat: live packets and dead ones
- Produce: JSON files/output
  - Protocol transaction
  - IDS alerts
  - PCAP
- Characteristics:
  - High speed
  - Open Source
  - Community driven
  - World famous
- Software owned and managed by the Open Information Security Foundation

Suricata is far more than an IDS/IPS



Source: Stamus Networks

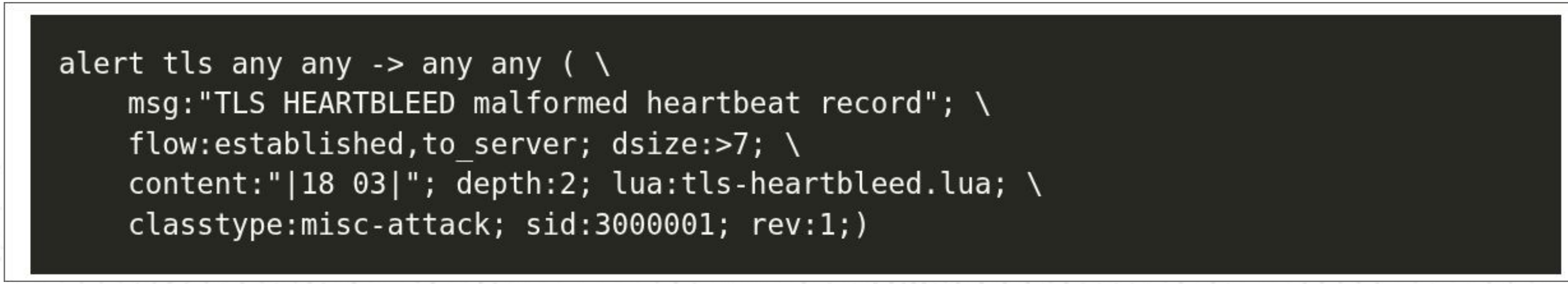
Introduction of Lua

Lua support in Suricata

- Introduced in Suricata 1.4
  - December 2014
- 2 main features:
  - Lua usage in signature
    - Dynamic detection
  - Lua output
    - Write data from the packet path to disk
    - Arbitrary logging

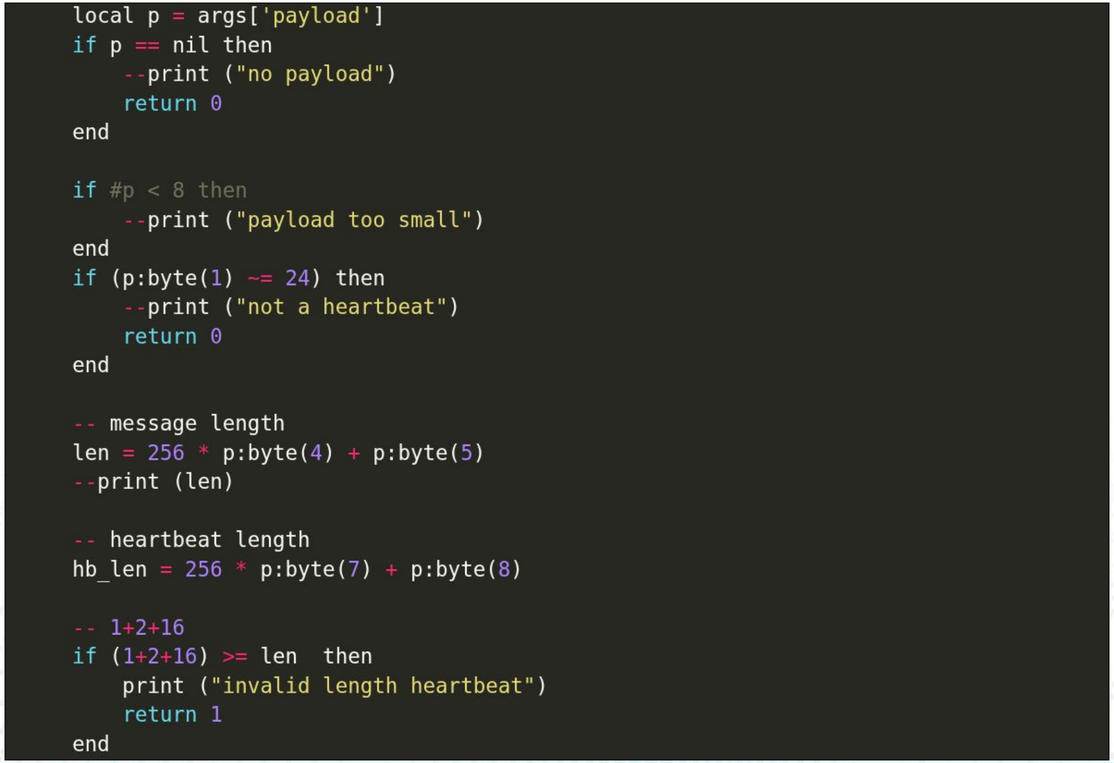
Lua signature for Heartbleed

The perfect use case for the feature:
https://inliniac.net/blog/2014/04/08/detecting-openssl-heartbleed-with-suricata/



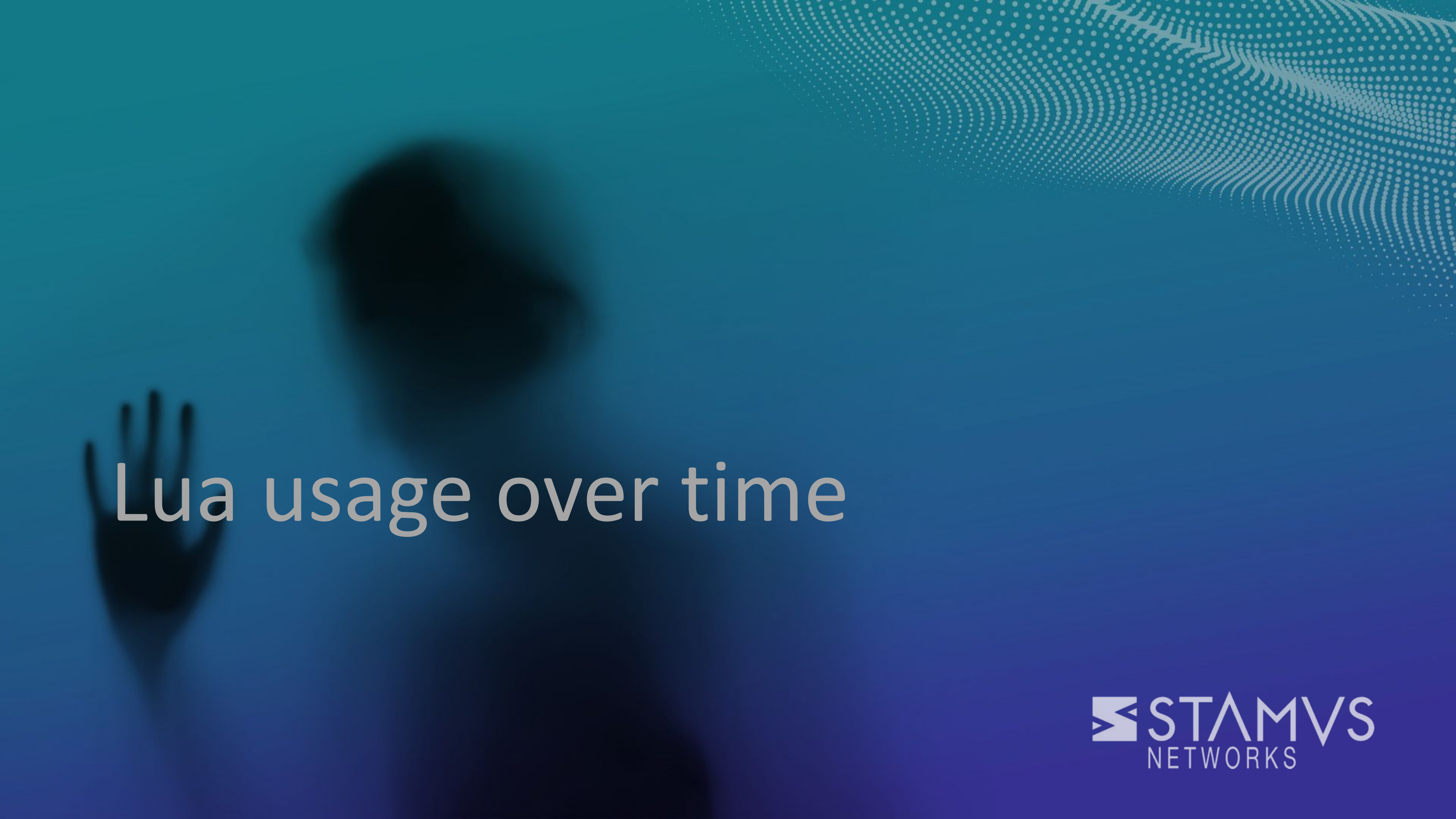
```
alert tls any any -> any any ( \
    msg:"TLS HEARTBLEED malformed heartbeat record"; \
    flow:established,to_server; dsize:>7; \
    content:"|18 03|"; depth:2; lua:tls-heartbleed.lua; \
    classtype:misc-attack; sid:3000001; rev:1;)
```

Heartbleed: the lua script



```lua
local p = args['payload']
if p == nil then
    --print ("no payload")
    return 0
end

if #p < 8 then
    --print ("payload too small")
end
if (p:byte(1) ~= 24) then
    --print ("not a heartbeat")
    return 0
end

-- message length
len = 256 * p:byte(4) + p:byte(5)
--print (len)

-- heartbeat length
hb_len = 256 * p:byte(7) + p:byte(8)

-- 1+2+16
if (1+2+16) >= len  then
    print ("invalid length heartbeat")
    return 1
end
```

Lua usage over time



Only used by Chris Wakelin

- Chris Wakelin is a threat researcher at Proofpoint/EmergingThreats
- Lua activity:
  - Github repository: https://github.com/EmergingThreats/et-luajit-scripts
  - Suricon talks: https://suricon.net/wp-content/uploads/2019/01/SuriCon2018_Wakelin.pdf
  - …

Reasons

- Distribution issue
  - Lua is coming from system
  - Rules writers don’t know:
    - Which lua version is available if available
    - Which lua modules are available
- Lua output
  - Distribution
  - Nobody found a use case
    - But they exists

The boiling frog effect

Paradigm change slowly

Rethinking signatures security

- 2014 and before:
  - Signatures are coming from trusted partners
  - No real supply chain attack risk
    - Was it even a word at the time ?
- 2023:
  - Ruleset is built using Github repositories
    - It is done via https://rulezet.org/ in 2025
  - We can not really trust Suricata signatures anymore
- Impact:
  - CVE on dataset: https://nvd.nist.gov/vuln/detail/CVE-2023-35852
  - CVE on lua: https://nvd.nist.gov/vuln/detail/CVE-2023-35853

CVE on Lua

In Suricata before 6.0.13, an adversary who controls an external source of Suricata rules may be able to execute Lua code. This is addressed in 6.0.13 by disabling Lua unless allow-rules is true in the security lua configuration section.

Let’s iterate

Lua - 2nd floor

- Open distribution implies
  - Security
  - Reproducibility
- Strategy:
  - Sandboxed Lua
    - Limit interaction risk
  - Vendored Lua
    - Embedded into Suricata binary
- Details:
  - CPU credit limitation

Conclusion

Let’s wrap it up.

Conclusion

Global Take Away

- Feature is as good as it is for the ecosystems
- Documentation and evangelisation are keys
- Question status quo: risk model may have evolved

Suricata and Lua

- Will reboot be a success ?
- Known problems have been fixed
- Will it be enough ?

SEE YOU IN 10 YEARS