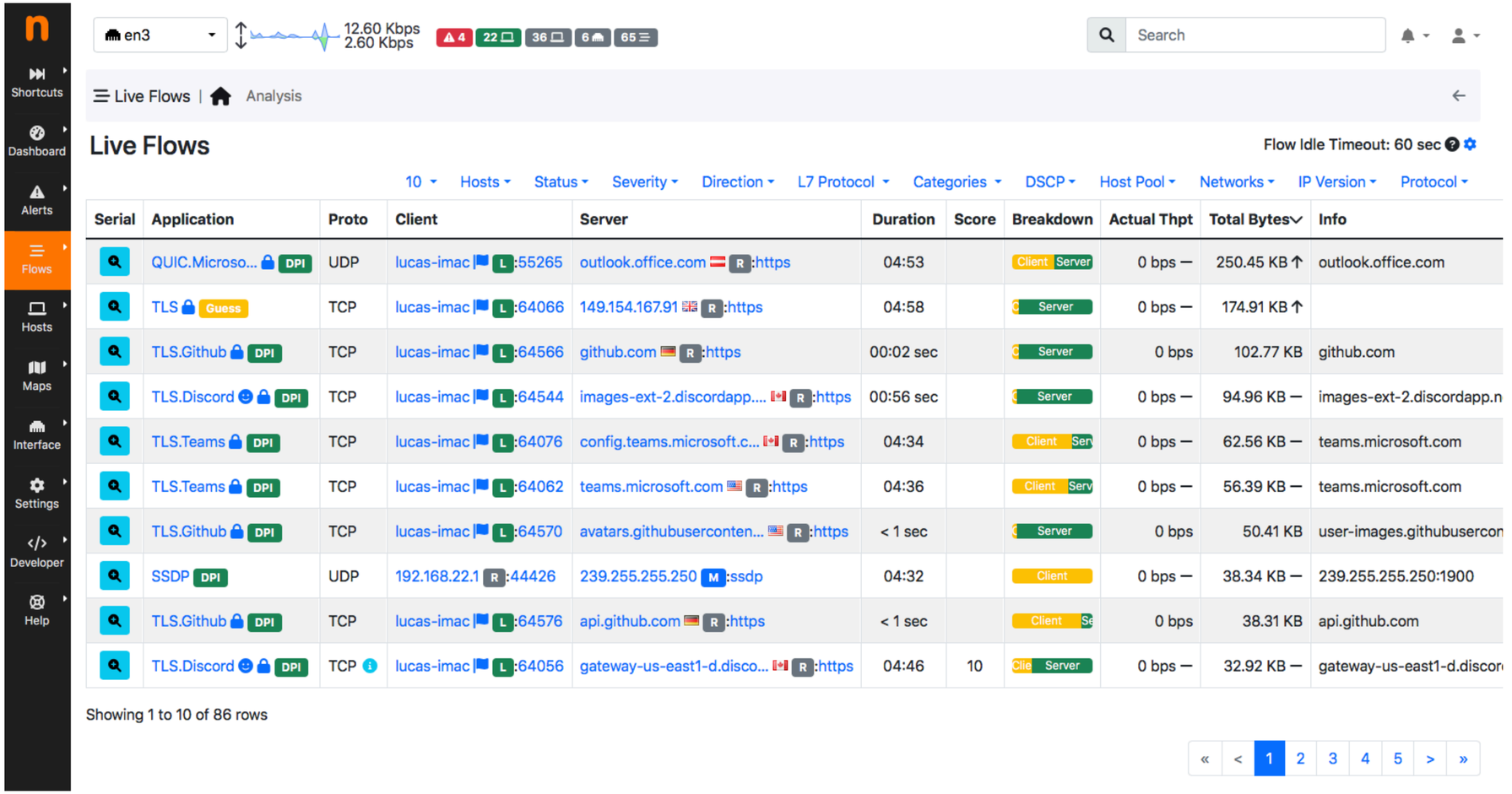
ntopng: an actionable event-driven network traffic analysis application

Luca Deri <deri@ntop.org>

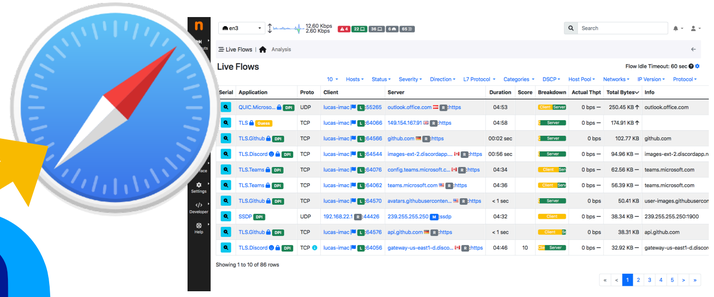
Alfredo Cardigliano <cardigliano@ntop.org>

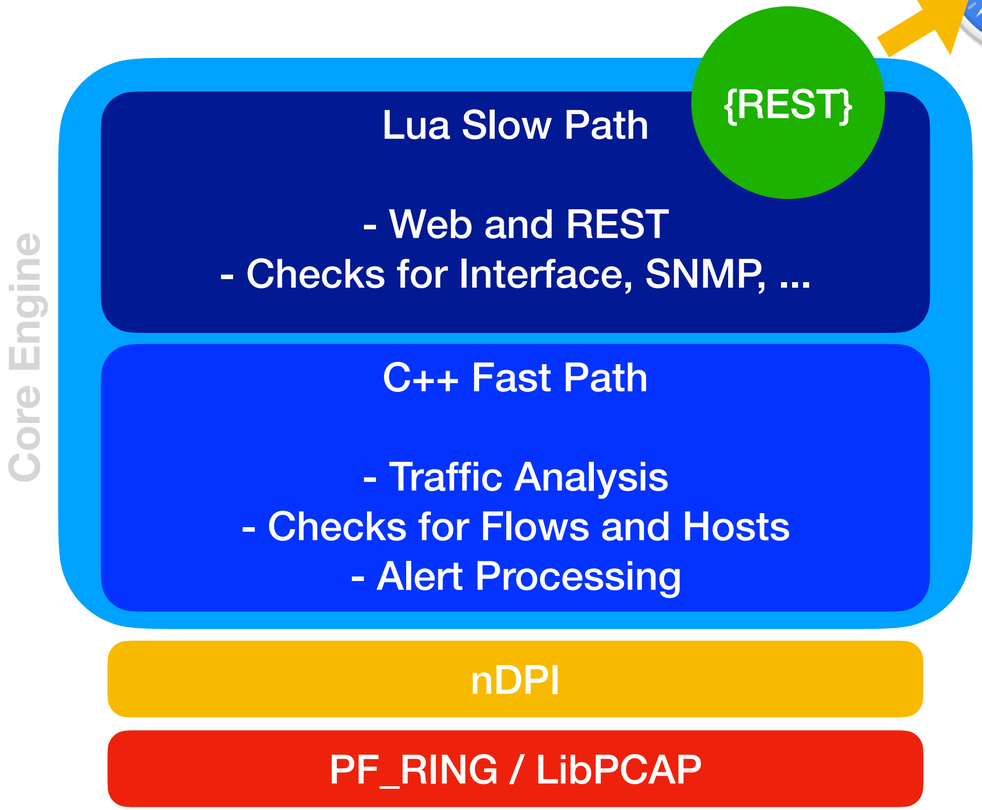
Welcome to ntopng

ntopng is a realtime network and cybersecurity traffic monitoring app.

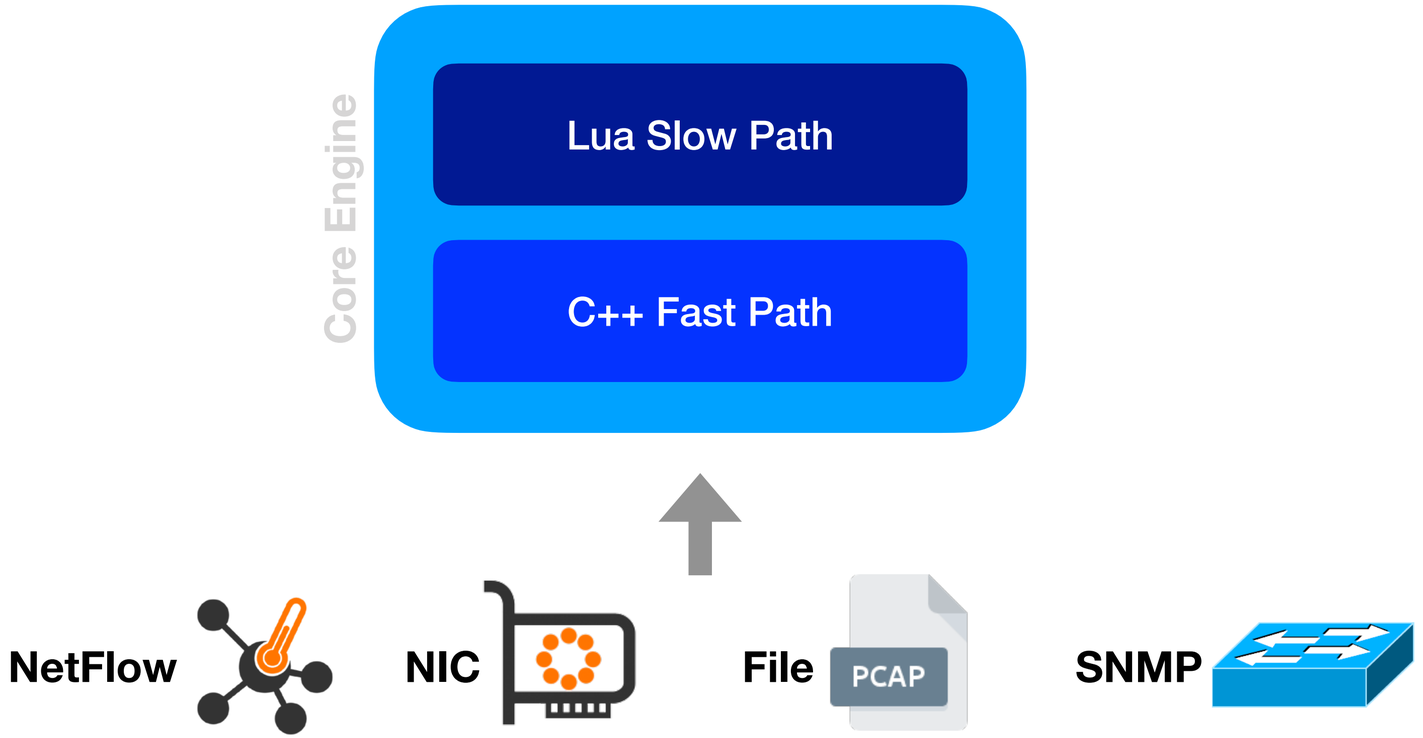


ntopng Architecture: Overview

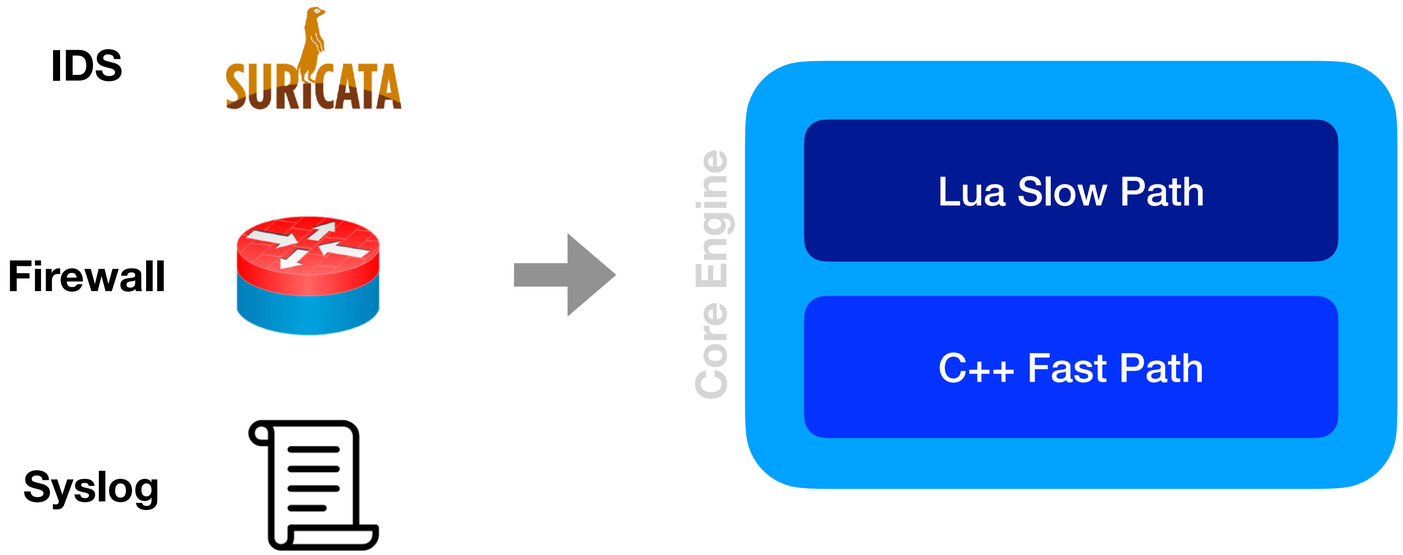




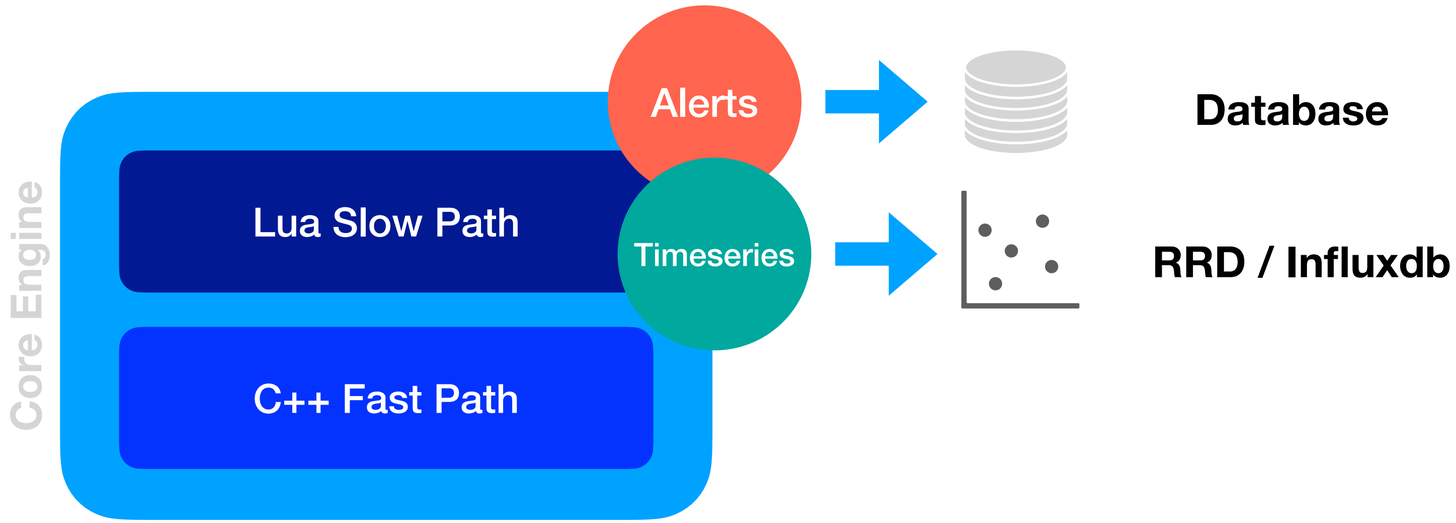
ntopng Architecture: Traffic Ingestion



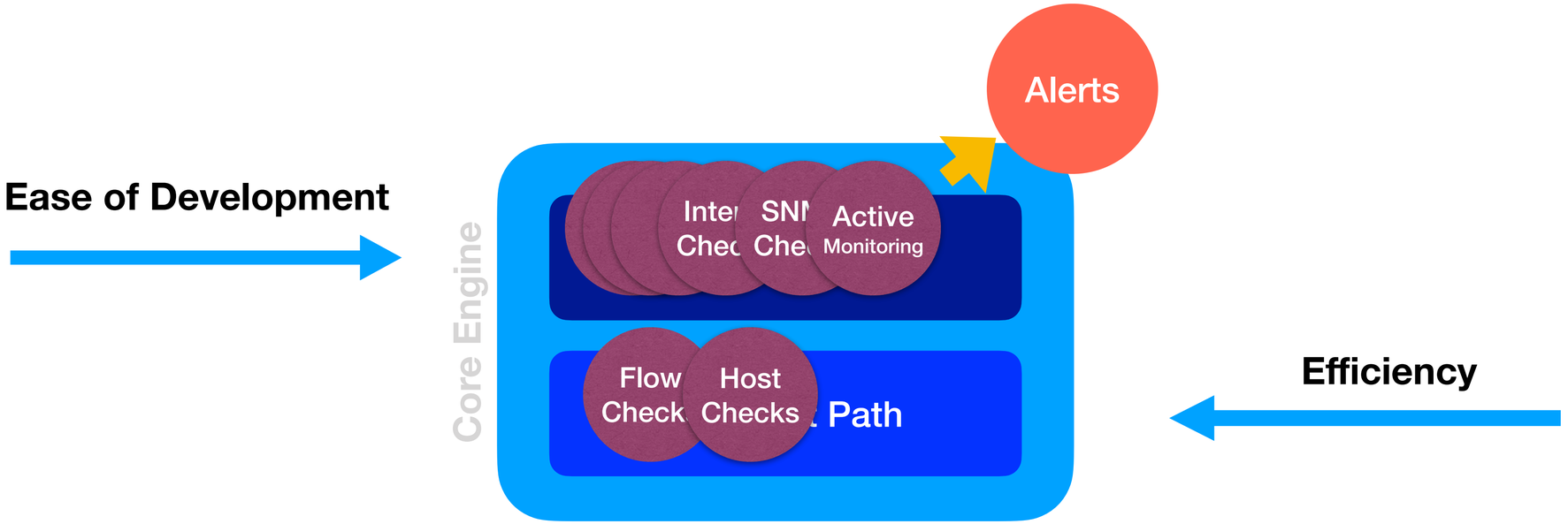
ntopng Architecture: Events Ingestion



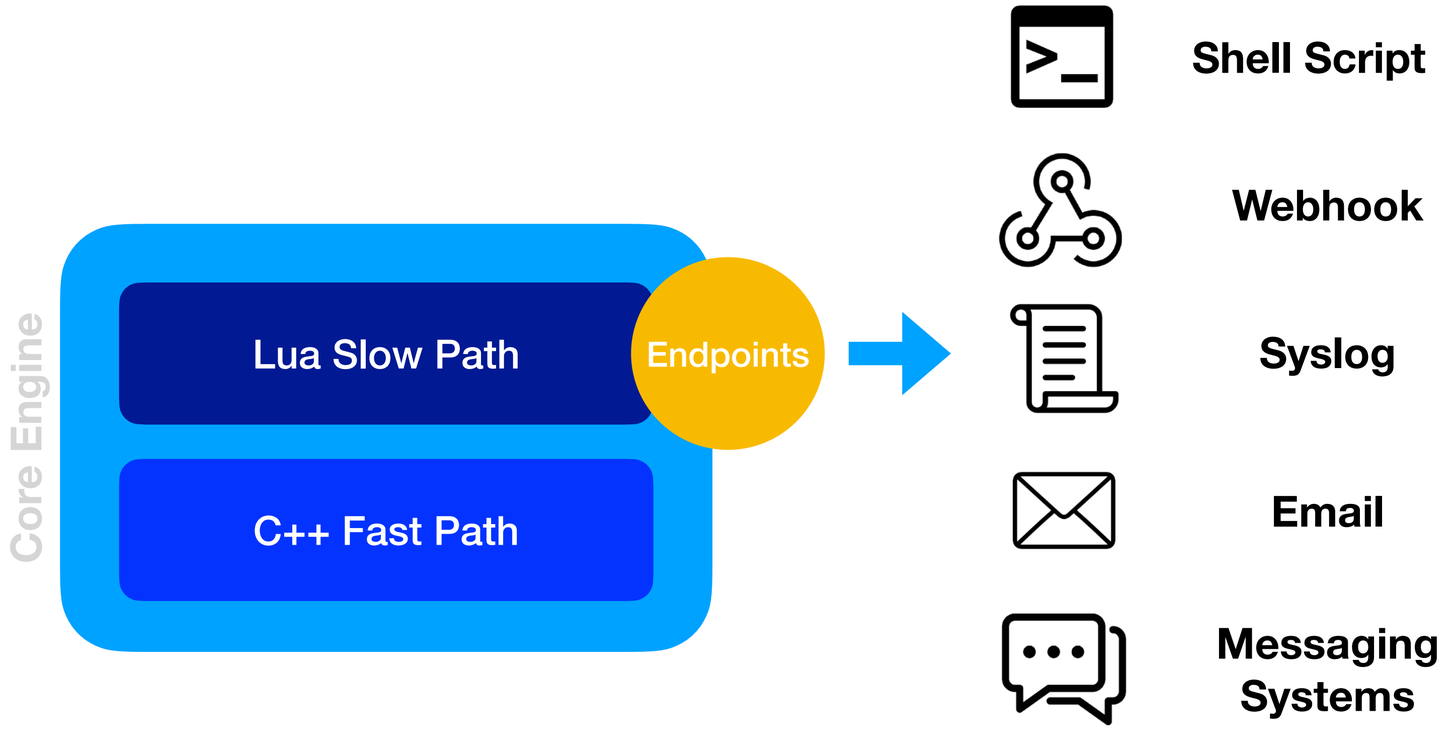
ntopng Architecture: Historical Data



ntopng Architecture: Checks

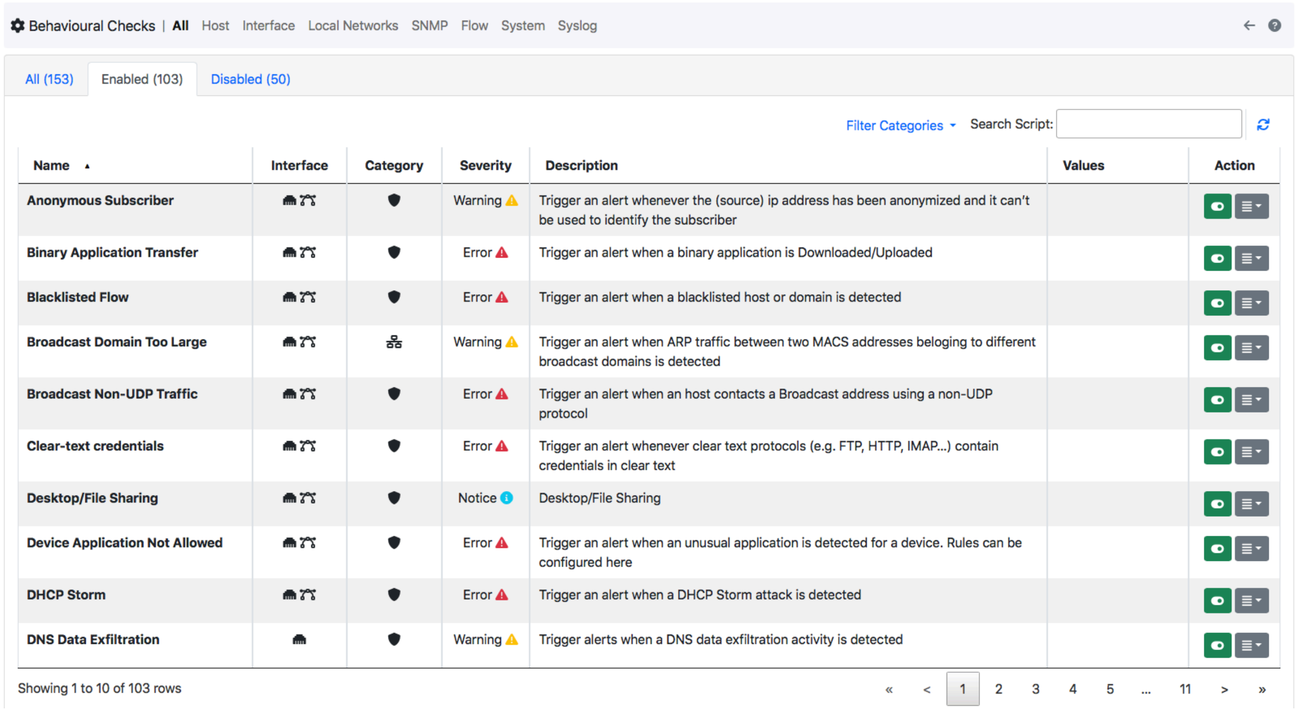


ntopng Architecture: Notifications



Problem Statement [1/2]

- How to extend ntopng's Behavioural Checks to meter something additional with respect to built-in checks?
- How can ntopng be used from languages such as Python to extract monitored data or traffic alerts?

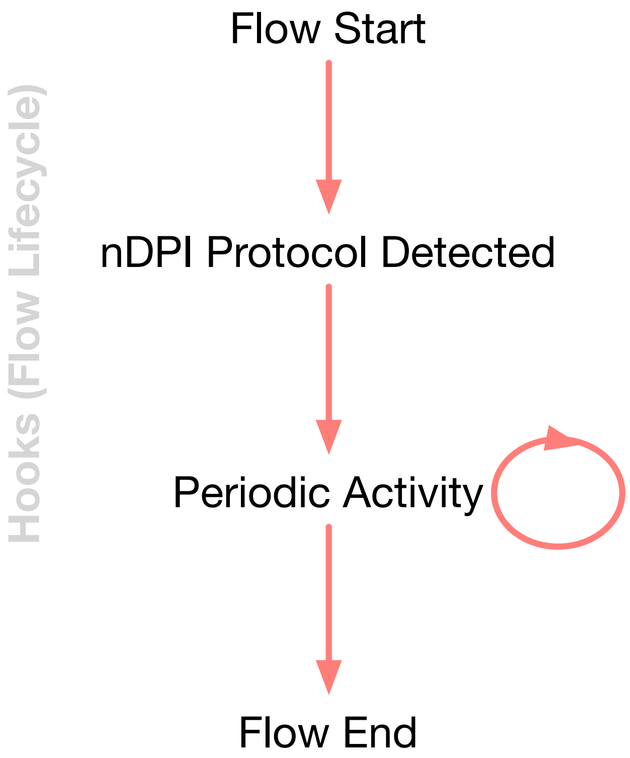


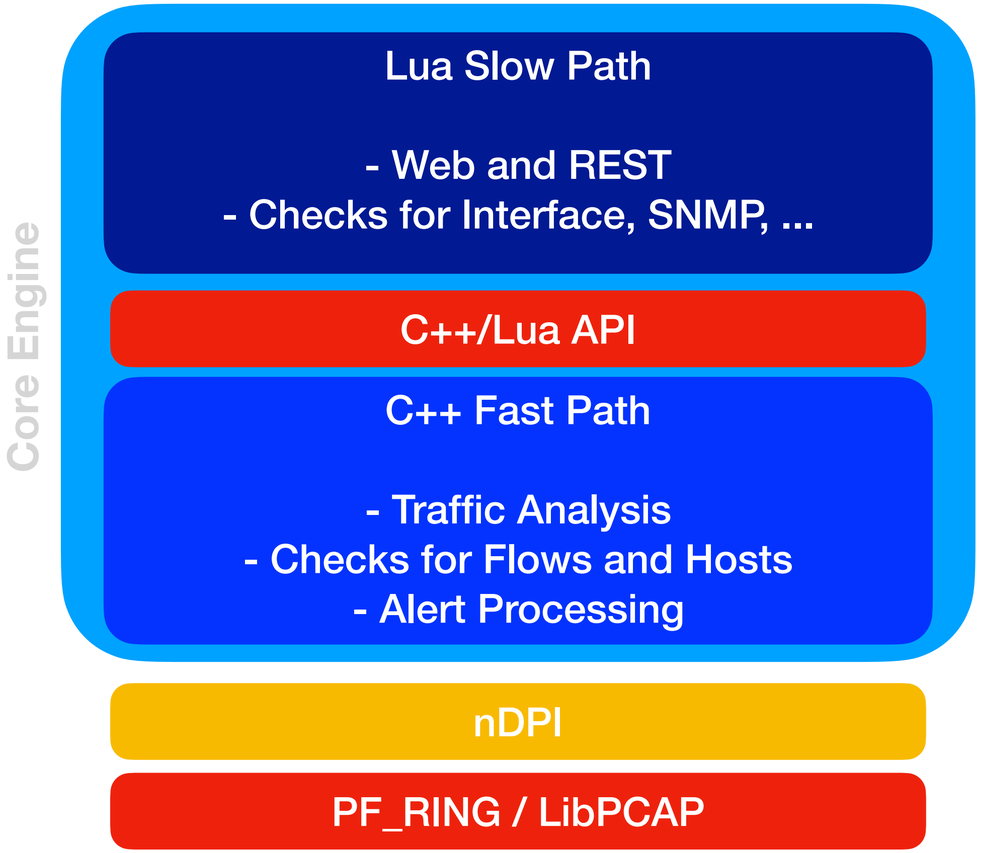
Name	Interface	Category	Severity	Description	Values	Action
Anonymous Subscriber			Warning	Trigger an alert whenever the (source) ip address has been anonymized and it can't be used to identify the subscriber		
Binary Application Transfer			Error	Trigger an alert when a binary application is Downloaded/Uploaded		
Blacklisted Flow			Error	Trigger an alert when a blacklisted host or domain is detected		
Broadcast Domain Too Large			Warning	Trigger an alert when ARP traffic between two MACS addresses beloging to different broadcast domains is detected		
Broadcast Non-UDP Traffic			Error	Trigger an alert when an host contacts a Broadcast address using a non-UDP protocol		
Clear-text credentials			Error	Trigger an alert whenever clear text protocols (e.g. FTP, HTTP, IMAP...) contain credentials in clear text		
Desktop/File Sharing			Notice	Desktop/File Sharing		
Device Application Not Allowed			Error	Trigger an alert when an unusual application is detected for a device. Rules can be configured here		
DHCP Storm			Error	Trigger an alert when a DHCP Storm attack is detected		
DNS Data Exfiltration			Warning	Trigger alerts when a DNS data exfiltration activity is detected		

Problem Statement [2/2]

- How can we give users the ability to write **efficient** Lua-based checks for flows and hosts without requiring C++ coding skills ?
- How can we allow people to create custom checks and alerts for **specific use cases** that they only have?
  - Trigger an alert if you see TLS traffic between host A and B.
  - Send me a Telegram notification if host A receives a TLS request with certificate issued by “O=GoDaddy.com"
  - Alert me of BitTorrent connections whose throughput exceeds 1 Gbit.
  - Report me about Zoom calls with bad quality.

Lua Scripting in ntopng [1/3]





Lua Scripting in ntopng [2/3]

C++

```
static luaL_Reg _ntop_host_reg[] = {
  { "ip",               ntop_host_get_ip              },
  { "mac",              ntop_host_get_mac             },
  { "name",             ntop_host_get_name            },
  { "vlan_id",          ntop_host_get_vlan_id         },

  { "is_unicast",       ntop_host_is_unicast          },
  { "is_multicast",     ntop_host_is_multicast        },
  { "is_broadcast",     ntop_host_is_broadcast        },
  { "is_blacklisted",   ntop_host_is_blacklisted      },

  { "bytes_sent",       ntop_host_get_bytes_sent      },
  { "bytes_rcvd",       ntop_host_get_bytes_rcvd      },
  { "bytes",            ntop_host_get_bytes_total     },
  { "l7",               ntop_host_get_l7_stats        },

  { "skipVisitedHost",  ntop_skip_visited_host        },
  { "triggerAlert",     ntop_trigger_host_alert       },
```

Lua Methods **Function Callback**

Lua

```
function dump_host()
   local rsp = {}

   rsp.ip                  = host.ip()
   rsp.mac                 = host.mac()
   rsp.name                = host.name()
   rsp.vlan_id             = host.vlan_id()
   rsp.is_unicast          = host.is_unicast()
   rsp.is_multicast        = host.is_multicast()
   rsp.is_broadcast        = host.is_broadcast()
   rsp.is_blacklisted      = host.is_blacklisted()
   rsp.bytes_sent          = host.bytes_sent()
   rsp.bytes_rcvd          = host.bytes_rcvd()
   rsp.bytes               = host.bytes()
   rsp.l7                  = host.l7()

   tprint(rsp)
end

dump_host()
```

Lua Scripting in ntopng [3/3]

```
void CustomFlowLuaScript::checkFlow(Flow *f, LuaEngine *lua) {
  lua->setFlow(f);
  lua->run_loaded_script(); /* Run script */

  if(f->isCustomFlowAlertTriggered()) {
    FlowAlertType alert_type = CustomFlowLuaScriptAlert::getClassType();
    u_int8_t c_score, s_score;
    risk_percentage cli_score_pctg = CLIENT_FAIR_RISK_PERCENTAGE;

    computeCliSrvScore(alert_type, cli_score_pctg, &c_score, &s_score);

    f->triggerAlertAsync(alert_type, c_score, s_score);
  }
}
```

Call LuaVM

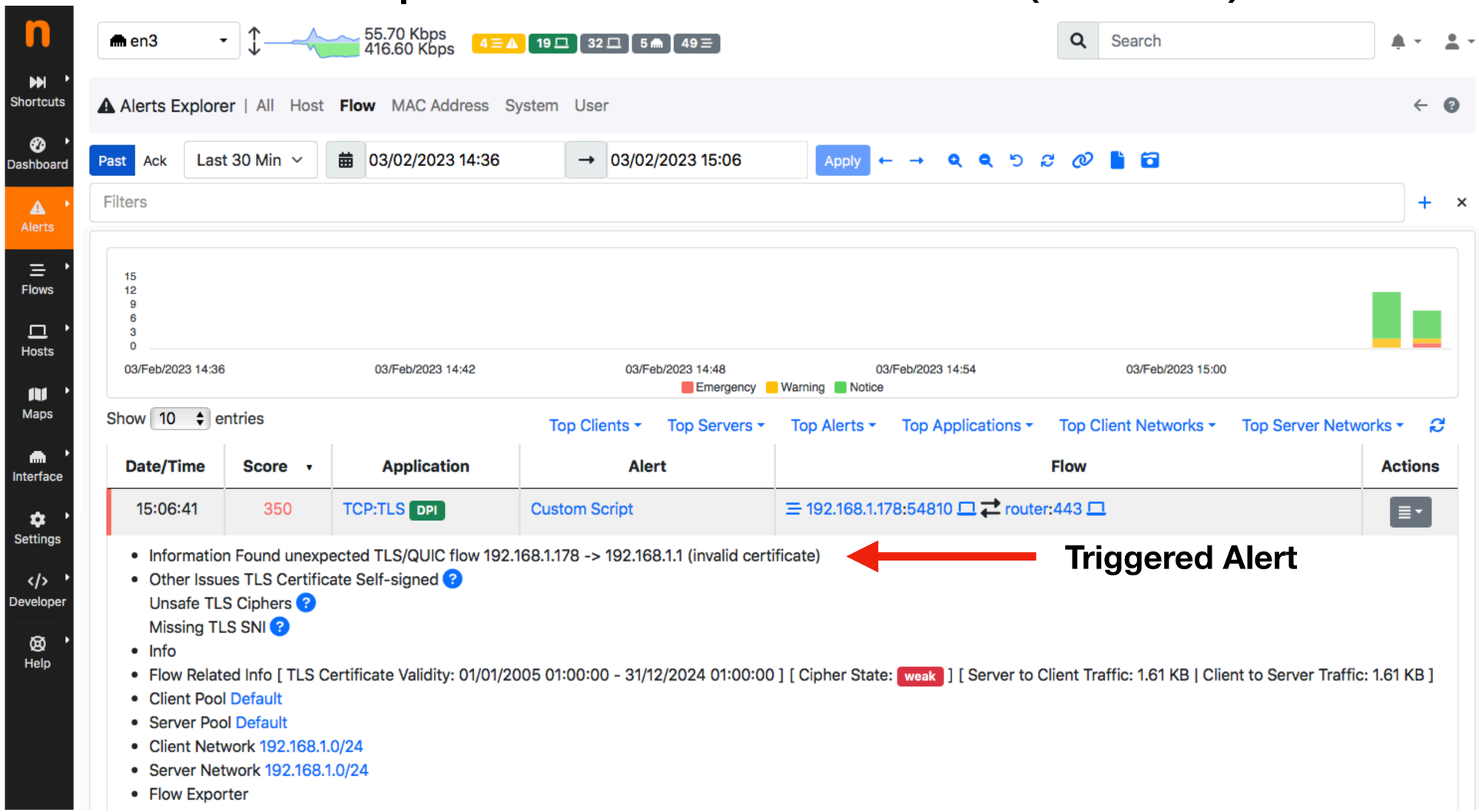
Simple Flow Check (User Script)

Current Flow

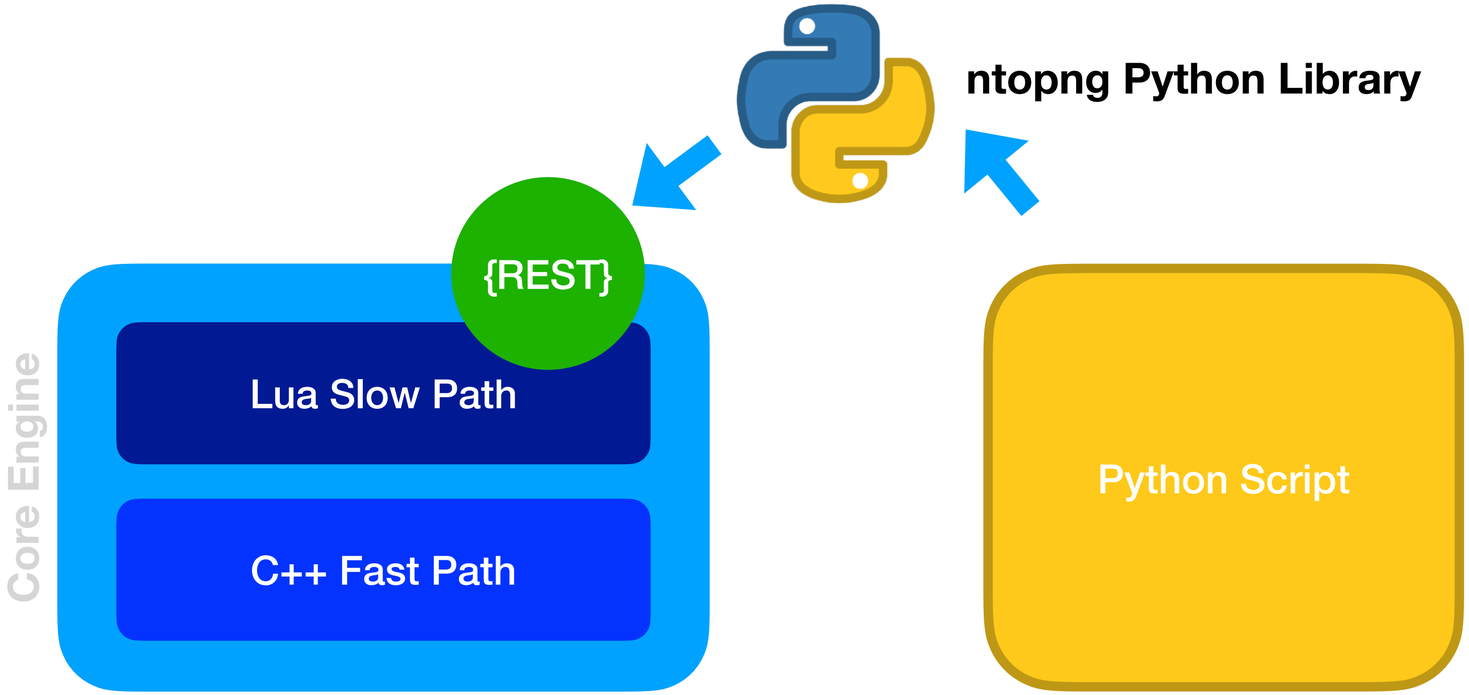
```
local t = flow.tls_quic()

if((flow.cli() == "192.168.1.178") and (flow.srv() == "192.168.1.1") and (t ~= nil)) then

   if(t["protos.tls.issuerDN"] == "CN=AGCOMBO, O=Technicolor, OU=1827SAZCH") then
      local score   = 111
      local message = "Found unexpected TLS/QUIC flow 192.168.1.178 -> 192.168.1.1 (invalid certificate)"

      flow.triggerAlert(score, message)

      dump_flow()
   end
end

return(0)
```

Check Overhead (Intel i3, 2010)
- Lua: ~30 usec
- C++: ~1 usec

Simple Flow Check (Alert)



Python 3 API




```
$ pip3 install ntopng
```

Script Example: Live Data

```
#!/usr/bin/env python3

from ntopng.ntopng import Ntopng

auth_token = "532b8dbe1092e591435c7a13f2"

ntopng_handle = Ntopng(auth_token=token)

interface = ntopng_handle.get_interface(iface_id)

hosts = interface.get_active_hosts()

for host in hosts:
        print(host['ip'] + " => ", end='')
        print(host)
```

Connect to ntopng

Get a live interface instance

Print all active hosts

Script Example: Historical Data

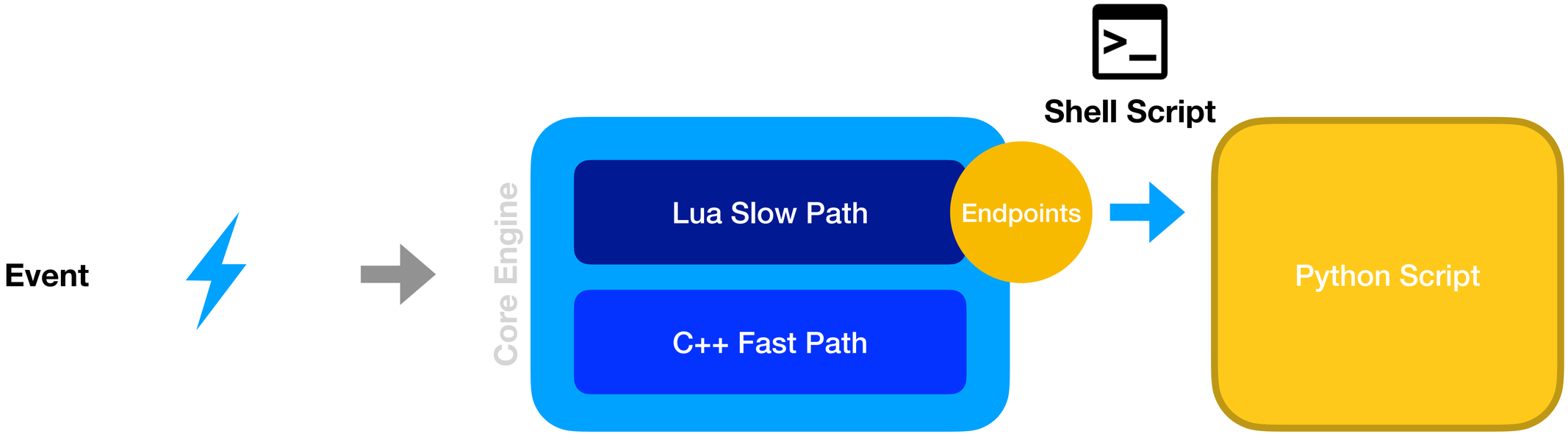
```
#!/usr/bin/env python3

from ntopng.ntopng import Ntopng

auth_token = "532b8dbe1092e591435c7a13f2"

ntopng_handle = Ntopng(auth_token=token)

interface = ntopng_handle.get_interface(iface_id)
historical = interface.get_historical()

epoch_end    = int(time.time())
epoch_begin  = epoch_end - 24*60*60
all_stats = historical.get_alerts_stats(epoch_begin, epoch_end)

for stats in all_stats:
    print(stats['label'])
    print(stats['value'])
```

Connect to ntopng

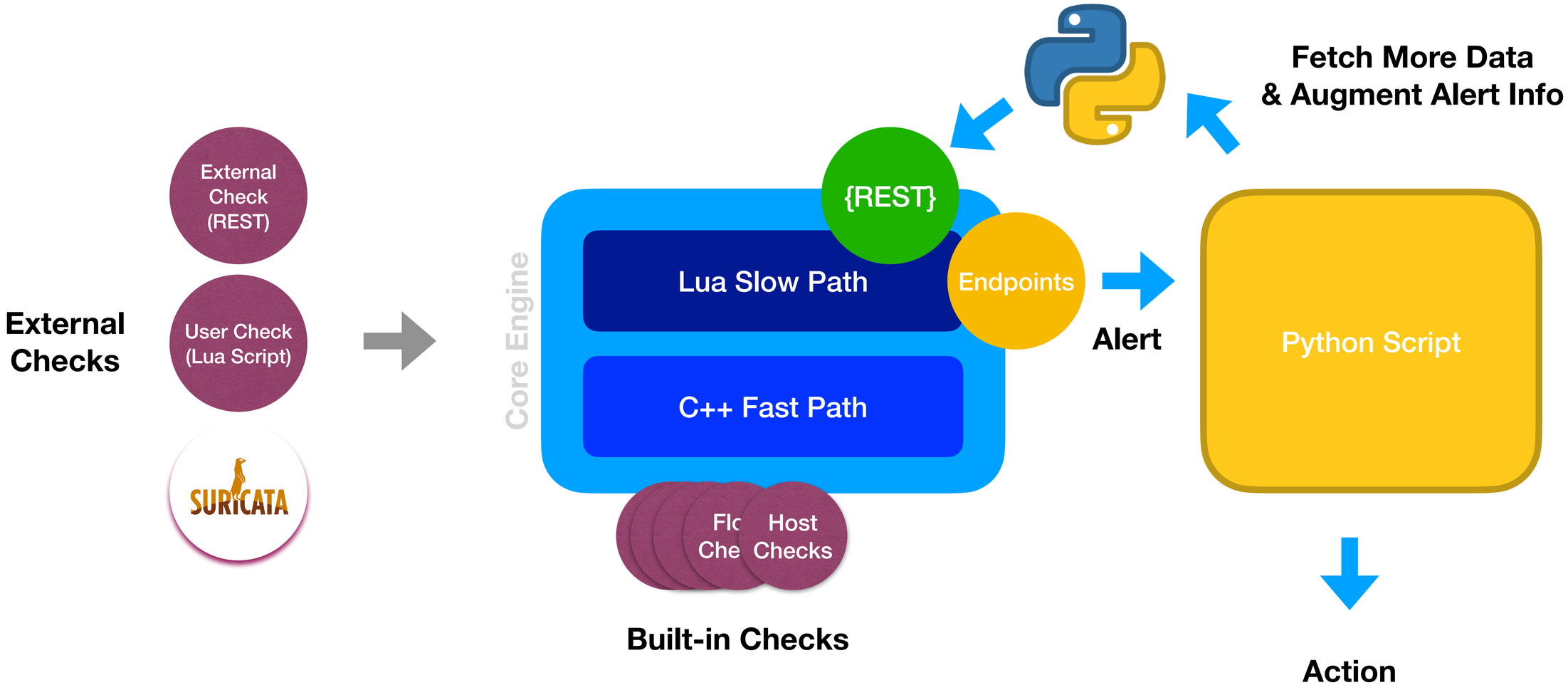
Get an historical interface instance

Get the alerts summary for a time interval

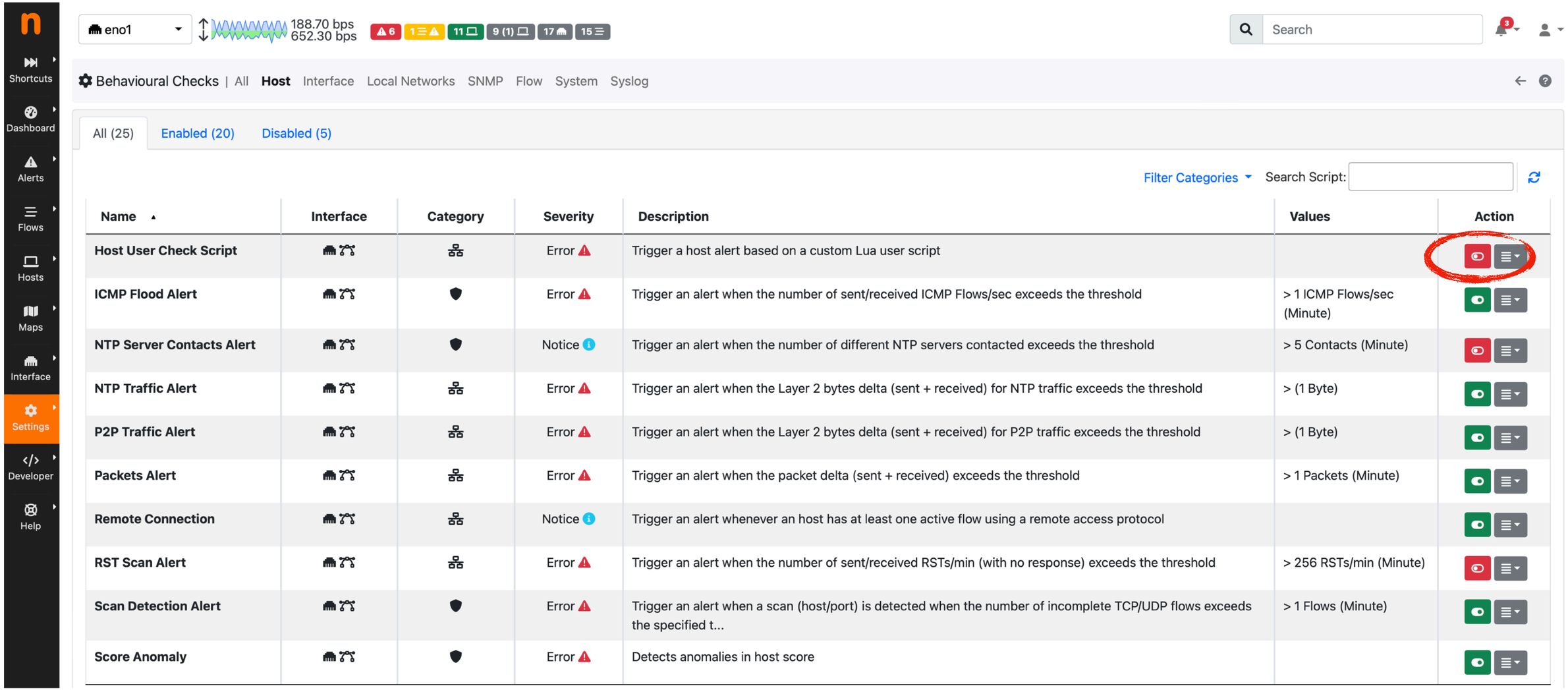
Python Script as Alert Handler



Let's Put Pieces Together



Step 1: Enable The Check

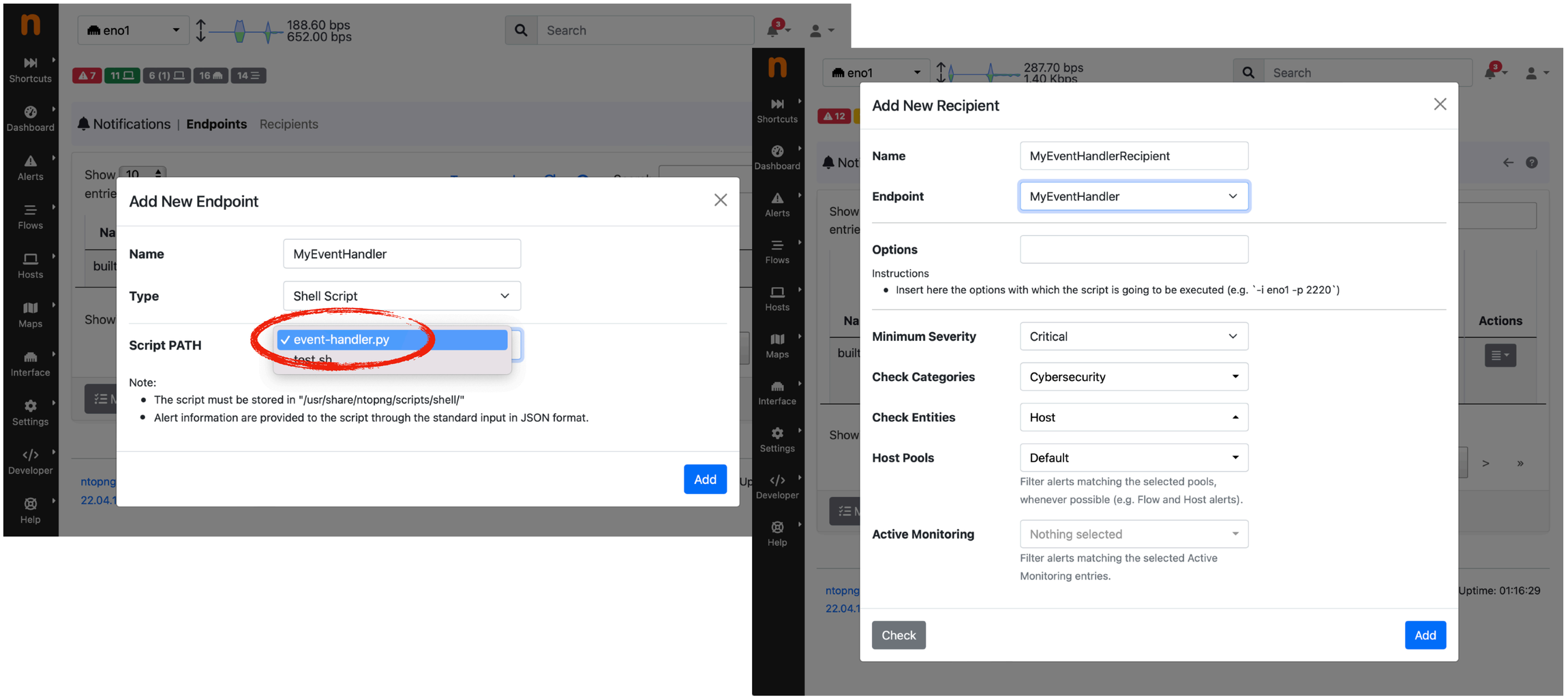


Name	Interface	Category	Severity	Description	Values	Action
Host User Check Script			Error	Trigger a host alert based on a custom Lua user script		
ICMP Flood Alert			Error	Trigger an alert when the number of sent/received ICMP Flows/sec exceeds the threshold	> 1 ICMP Flows/sec (Minute)	
NTP Server Contacts Alert			Notice	Trigger an alert when the number of different NTP servers contacted exceeds the threshold	> 5 Contacts (Minute)	
NTP Traffic Alert			Error	Trigger an alert when the Layer 2 bytes delta (sent + received) for NTP traffic exceeds the threshold	> (1 Byte)	
P2P Traffic Alert			Error	Trigger an alert when the Layer 2 bytes delta (sent + received) for P2P traffic exceeds the threshold	> (1 Byte)	
Packets Alert			Error	Trigger an alert when the packet delta (sent + received) exceeds the threshold	> 1 Packets (Minute)	
Remote Connection			Notice	Trigger an alert whenever an host has at least one active flow using a remote access protocol		
RST Scan Alert			Error	Trigger an alert when the number of sent/received RSTs/min (with no response) exceeds the threshold	> 256 RSTs/min (Minute)	
Scan Detection Alert			Error	Trigger an alert when a scan (host/port) is detected when the number of incomplete TCP/UDP flows exceeds the specified t...	> 1 Flows (Minute)	
Score Anomaly			Error	Detects anomalies in host score		

Step 2: Write The Alert Handler

```
$ cat /usr/share/ntopng/scripts/shell/event-handler.py

#!/usr/bin/env python3

import sys
import json

input = sys.stdin.readlines()
alert = json.loads(input[0])

if alert["alert_id"] == host_user_check_script:

    user_info = json.loads(alert["json"])

    log(alert["ip"])
    log(user_info["message"])
```

Step 3: Set The Handler As Recipient



Step 4: Extend The Alert Handler

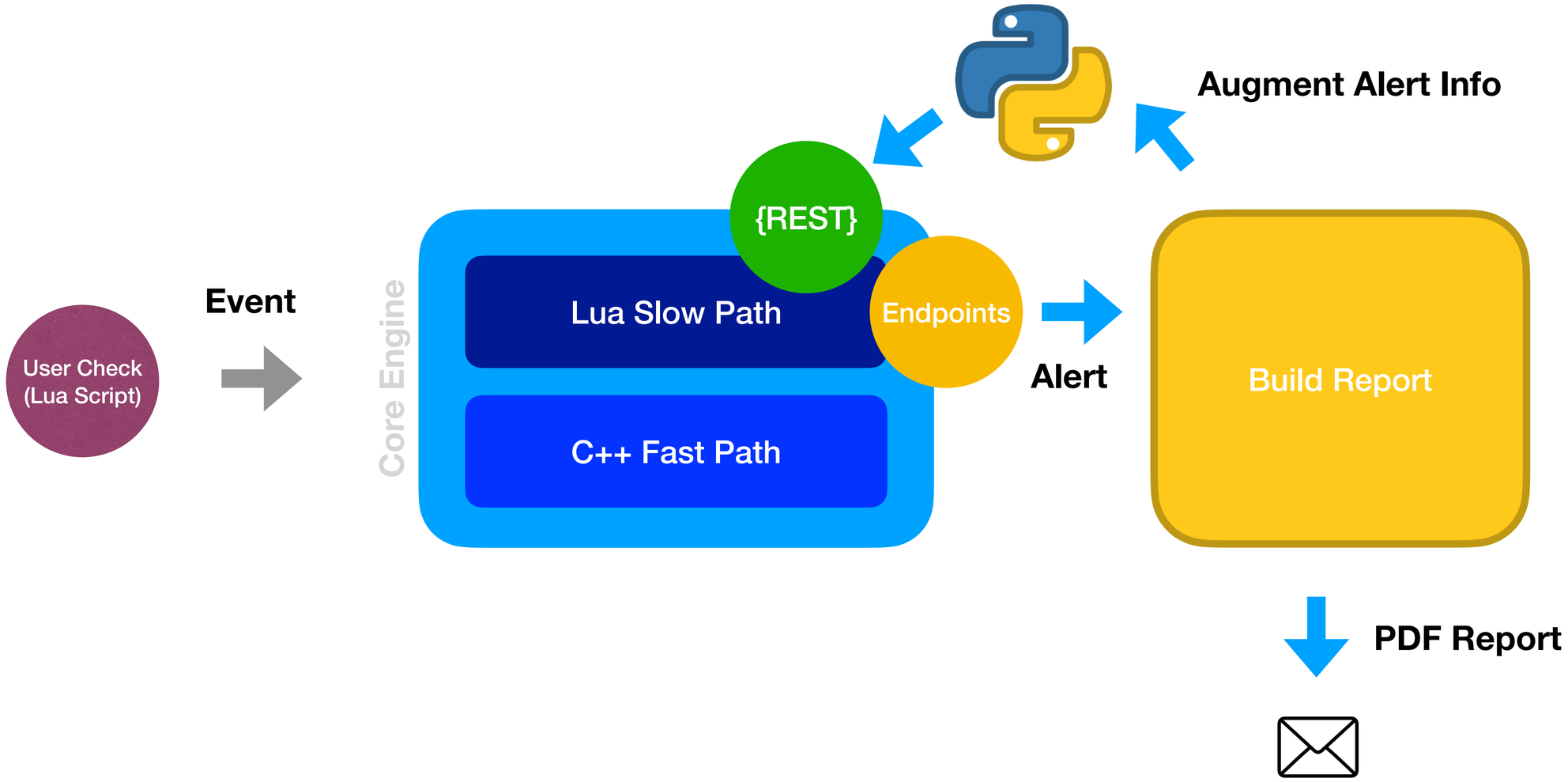
```
from pathlib import Path
from redmail import EmailSender
from ntopng.ntopng import Ntopng
from ntopng.report import Report

[...]

auth_token = "532b8dbe1092e591435c7a13d561db73"
report_path = "/tmp/report.pdf"

ntopng_handle = Ntopng(auth_token=token)

generator = Report(ntopng_handle, alert["ifid"], alert["ip"])
generator.build(report_path)

email = EmailSender(host=e_host, port=e_port, username=e_username, password=e_password)

email.send(sender = e_sender, receivers = [e_recipient], subject = e_subject,
attachments = { "report.pdf": Path(report_path) } )
```

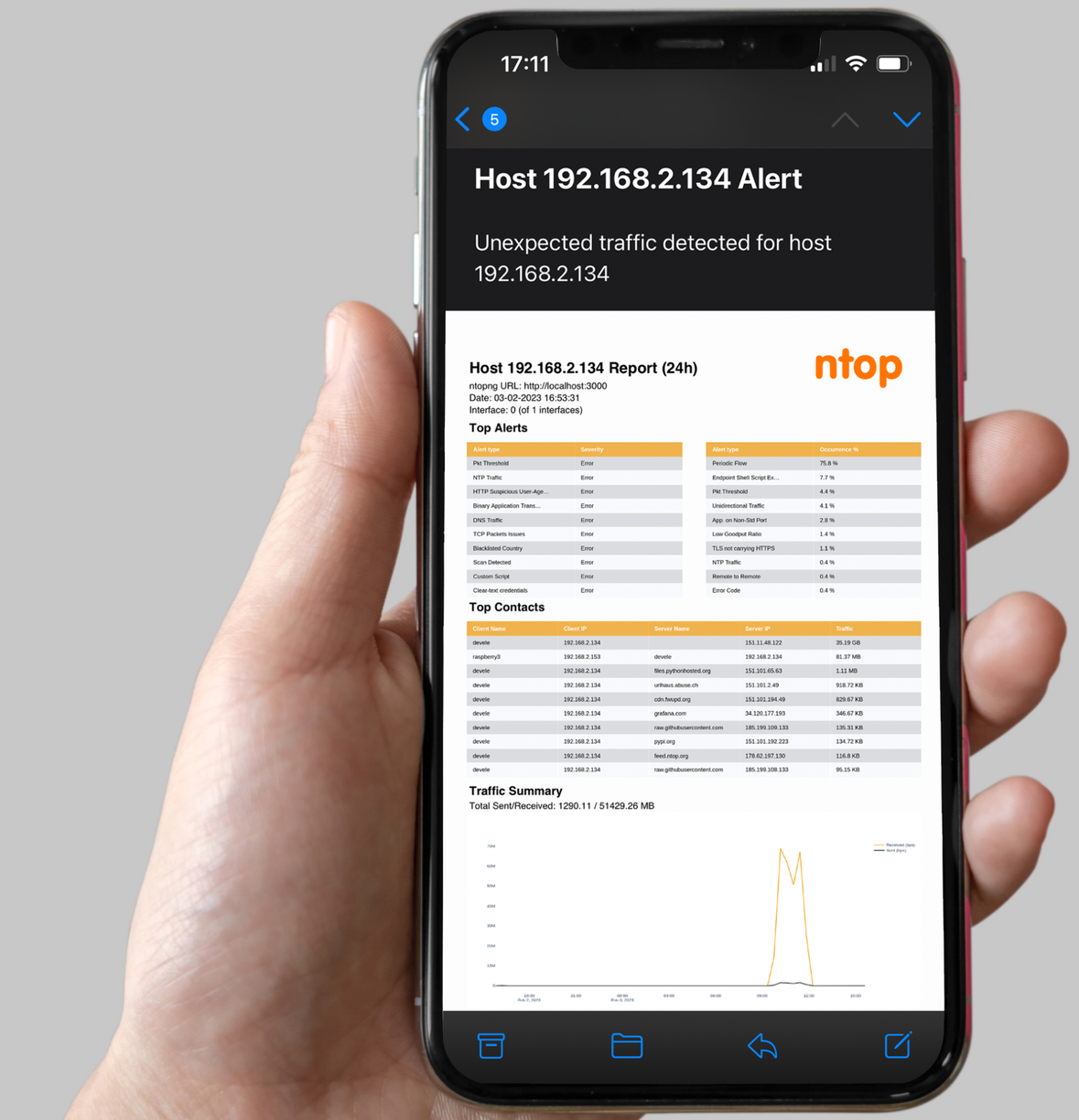
Connect to ntopng to fetch more data

Build a PDF report

Send the report by email

Example: Alert Report




17:11
5
Host 192.168.2.134 Alert
Unexpected traffic detected for host 192.168.2.134
Host 192.168.2.134 Report (24h)
ntop
ntopng URL: http://localhost:3000
Date: 03-02-2023 16:53:31
Interface: 0 (of 1 interfaces)
Top Alerts
Alert type | Severity
Pkt Threshold | Error
NTP Traffic | Error
HTTP Suspicious User-Age... | Error
Binary Application Trans... | Error
DNS Traffic | Error
TCP Packets Issues | Error
Blacklisted Country | Error
Scan Detected | Error
Custom Script | Error
Clear-text credentials | Error
Alert type | Occurrence %
Periodic Flow | 75.8 %
Endpoint Shell Script Ex... | 7.7 %
Pkt Threshold | 4.4 %
Unidirectional Traffic | 4.1 %
App. on Non-Std Port | 2.8 %
Low Goodput Ratio | 1.4 %
TLS not carrying HTTPS | 1.1 %
NTP Traffic | 0.4 %
Remote to Remote | 0.4 %
Error Code | 0.4 %
Top Contacts
Client Name | Client IP | Server Name | Server IP | Traffic
develo | 192.168.2.134 | | 151.11.48.122 | 35.19 GB
raspberry3 | 192.168.2.153 | develo | 192.168.2.134 | 81.37 MB
develo | 192.168.2.134 | files.pythonhosted.org | 151.101.65.63 | 1.11 MB
develo | 192.168.2.134 | urlhaus.abuse.ch | 151.101.2.49 | 918.72 KB
develo | 192.168.2.134 | cdn.fwupd.org | 151.101.194.49 | 829.67 KB
develo | 192.168.2.134 | grafana.com | 34.120.177.193 | 346.67 KB
develo | 192.168.2.134 | raw.githubusercontent.com | 185.199.109.133 | 135.31 KB
develo | 192.168.2.134 | pypi.org | 151.101.192.223 | 134.72 KB
develo | 192.168.2.134 | feed.ntop.org | 178.62.197.130 | 116.8 KB
develo | 192.168.2.134 | raw.githubusercontent.com | 185.199.108.133 | 95.15 KB
Traffic Summary
Total Sent/Received: 1290.11 / 51429.26 MB

Final Remarks

- ntopng is now a versatile flow collector and traffic analysis application.
- It combines traditional traffic visibility with cybersecurity analysis.
- Lua-based engine scripting enables the creation of behavioural checks that can be implement in seconds by non-skilled developers.
- They are fast enough to be used in Gbit networks, but converting them to C++ is required for scaling up.
- Python bindings enable programmers to use ntopng as both a live and historical data source, and also extend ntopng in areas such as PDF reporting not currently available as native feature.
- All code is available at https://github.com/ntop/ntopng