Your new Threat Hunting tool for Linux

GitHub / Twitter: 0xrawsec

Repository: https://github.com/0xrawsec/kunai

Who is in front of you ?

RawSec

First Name: Quentin **Last Name**: JEROME **Age**: last_year + 1

Job: Freelance Security Consultant/Researcher working in Luxembourg

- **Background**: doing Incident Response, digital forensics, threat hunting on endpoints, detection engineering …
- **Now:** Open-Source developer mainly in Rust, Go, C, Python. At the origin of several projects: Gene, WHIDS, golang-evtx, golang-misp, golang-etw …

I tend to speak only at local conferences: MISP/CTI Summit, Hack.lu … maybe I should try to export myself out of Luxembourg !

RawSec

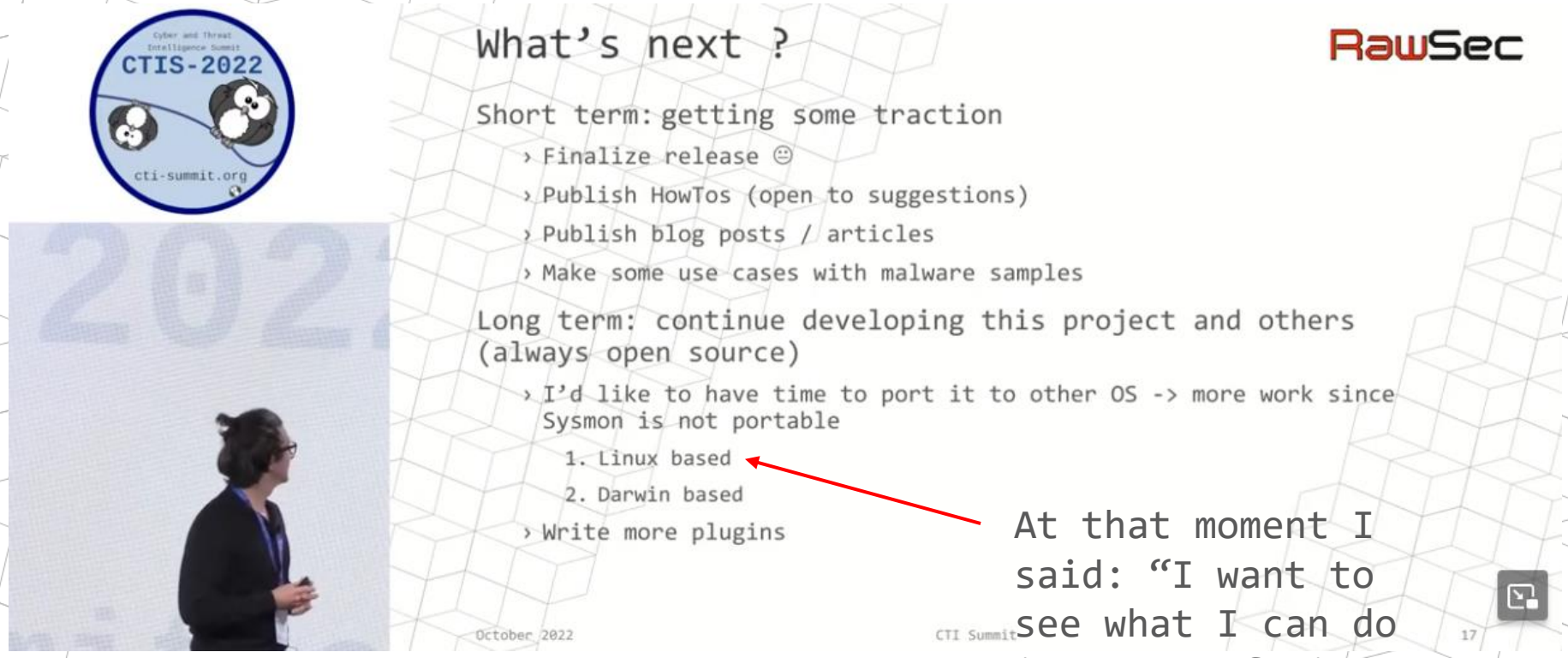
The Context

The story of creating a new monitoring tool for Linux

Why this project ?

RawSec

Because I told you so last year !



At that moment I said: "I want to see what I can do in terms of Linux monitoring"

More seriously !

RawSec

Sysmon is a really nice tool and I waited a long time after the Linux version … but I was also a bit disappointed when it went out !

- My personal opinions:
  - XML config … really !
  - Events in XML … 2nd really !
  - Developed in C++ (or how to make nobody contribute to your project)
  - Same events on Windows and on Linux ! How is that even possible ?
    - Windows and Linux are two different OS
    - Some events or parts of events don’t make sense at all on Linux
  - Almost no activity on the repo between 2022 and 2023 (main dev of first release left MS just after initial release)

Maybe what I accepted on Windows I don’t on Linux ?

(Do you think I should see someone ?)

End of 2022: YOLO !

RawSec

Instead of waiting, why not implementing my own concept of what “Sysmon for Linux” should be?

Objectives:

- Taking all the good ideas in Sysmon
- Provide relevant events for threat hunting and detection
  - Not too raw
  - Not too refined
- Something simple
- Documented
- Open-Source: people can understand and modify

Some questions though !

- How to do it ?
  - Kernel module
  - eBPF
- In which language ?
  - C, C++, Go, Rust …

Answering the questions

RawSec

Which technology ?

- After some research writing a kernel module does not seem to be a good idea ! Since a little while a lot of efforts are made in Linux kernel to develop eBPF
- eBPF : Bytecode executed in a VM on Kernel side. Exactly meant for that purpose

Which language to write userland and kernel code ?

- C: unleash the full power of eBPF but it will be hard to make the userland part (security, libraries …)
- C++: probably the same as C but worst to write
- Rust: nice libraries allowing to write eBPF code directly in Rust are being developed
- Go: does not compiles to eBPF bytecode ! eBPF programs needs to be written in another language. Only the loader can be in Go. I know Go but I don’t want to juggle between two languages.

Decision is taken: let’s write that stuff in Rust !

Other issues arise …

RawSec

- I don’t know how to write Rust !
- eBPF … what’s that ?
- Linux kernel … don’t know more

Sounds like we have a roadmap

1. Learn Rust
2. Learn about eBPF
3. Kernel … let’s learn on the field !

RawSec

Interlude

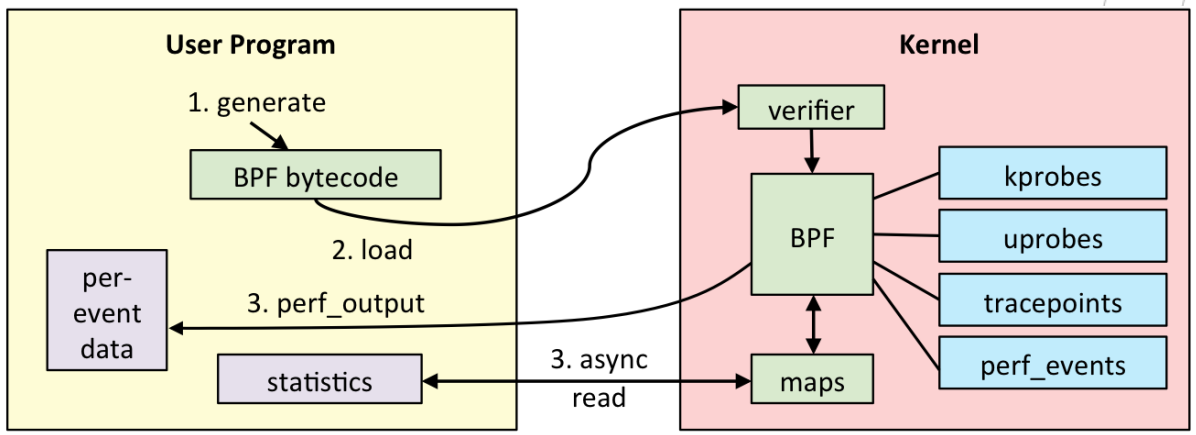
I can now explain how eBPF works

RawSec

C, C++, Rust compile down to eBPF bytecode

eBPF bytecode:

1. Is verified (side effects checking)
   - Number of instructions is limited
   - Checks kernel read/write
   - Checks null pointers
2. Can be attached at different places:
   - Syscalls (tracepoints)
   - Kernel functions (kprobes)
   - Sockets
   - Network Interfaces
   - …
3. Executed in Kernel inside a VM (in theory we cannot escape)
4. Communicates with userland via shared memory using different structures



Source: https://www.brendangregg.com/eBPF/linux_ebpf_internals.png

eBPF: for the non technical people

RawSec

RawSec

Seven Months Have Passed

First Kunai Release in June

Finally something to throw

RawSec

Taking all the good of Sysmon:

- Task/process UUID
  - Used to track activity across events
- File hashes:
  - File executed
  - Mmapped shared objects

And even more:

- Events in JSON
- Tasks' ancestors (until init)
- Script execution and their interpreter
- BPF/eBPF programs being loaded
- Data sent over the network and estimate data entropy

Find all details: https://why.kunai.rocks

What makes Kunai special

RawSec

Implementation wise:

- Events arrive sorted in chronological order
  - Not always the case with other tools (at least from the exp I have with Sysmon on Windows)
- Built-in “on host” correlation
  - Always more performant, accurate and easier to do as early as possible
- Activity deep down to containers

Documentation/support wise:

- It is actually documented and documentation will be maintained
- There is a chat to share experience, ideas and frustrations
- I am there to listen to users and develop new features/events
- I would really like to build an open community of users around this project so that we can all improve Linux threat detection

On host correlation

RawSec


```
{
  "data": {
    "command_line": "curl --dns-servers 8.8.8.8 http://why.kunai.rocks",
    "exe": "/usr/bin/curl",
    "query": "why.kunai.rocks",
    "proto": "udp",
    "response": "0xrawsec.github.io",
    "dns_server": {
      "ip": "192.168.1.1",
      "port": 53,
      "public": false
    }
  },
  "info": {
    "host": {
      "hostname": "bionic-container",
      "container": "docker"
    },
    "event": {
      "source": "kunai",
      "id": 61,
      "name": "dns_query",
      "uuid": "24981eed-63a6-9c38-b048-2c3ef6995aa8",
      "batch": 275
    },
    "task": {
      "name": "curl",
```


DNS query comes first, right ?


```
  "data": {
    "command_line": "curl --dns-servers 8.8.8.8 http://why.kunai.rocks",
    "exe": "/usr/bin/curl",
    "dst": {
      "hostname": "0xrawsec.github.io",
      "ip": "185.199.110.153",
      "port": 80,
      "public": true,
      "is_v6": false
    },
    "connected": true
  },
  "info": {
    "host": {
      "hostname": "bionic-container",
      "container": "docker"
    },
    "event": {
      "source": "kunai",
      "id": 60,
      "name": "connect",
      "uuid": "2ba54779-6eca-4d05-4335-177c59e8f90e",
      "batch": 275
    },
    "task": {
      "name": "curl",
```


connect() syscall connects to an IP not a domain !

We get info from the DNS query and push the information back to the connect event

Container monitoring

RawSec

You: Not a big deal …
containers are just tasks running in namespaces!

Me: True, though when it comes down to hash files some tricks need to be found.

You: Why ?

Me: Accessing files inside a container (from outside) requires a bit of extra work.

The file you see above does not exist outside of the container (where kunai is running)

A rocky road toward a first release

RawSec

- Before feeling comfortable writing Rust -> 1 month
- Writing eBPF programs in Rust
  - Initially used RedBPF (https://github.com/foniod/redbpf) framework
  - When I developed 80% of the project I moved everything to Aya (https://github.com/aya-rs) -> another 1.5 month
- I wanted to use features (i.e. BPF CO-RE) not officially supported by Aya
  - Contributed to bpf-linker (eBPF object file linker)
  - A lot of time spent on reading code, testing, bug tracking/reporting in the following projects:
    - bpf-linker
    - LLVM (the compiler infrastructure)
    - Aya
- A lot of time spent trying to make the eBPF verifier happy (this guy is pretty picky …)

Conclusion: developing eBPF is not easy !

The future of this project

RawSec

Still a lot to do:

- Migrate to latest version of Aya / Rust
- Support more containers type
- Enhance CI/CD with event testing
- Improving user experience
  - Advanced configuration / event filtering
- Maintenance
  - Compatibility with new kernels
- R&D
  - Developing new probes (waiting for your ideas)
  - Running Kunai on phones -> building phoneypots ?

Advertisement Time

RawSec

For Aya (Rust library for eBPF):

- The community is great and always there to help
- Very rigorous development
- They are always happy to receive contributions
- If you are considering developing eBPF based project, you should seriously take a look at it!

For Kunai: I know (sometimes) corporations/institutions don’t like when things are free

- Support contract
- Trainings
- Feature development
- Develop all your eBPF dreams

RawSec

Thank you all !

Special thanks to @alessandrod and @vadorovsky and everyone else making Aya such a great project

Thanks to @adulau, @gallypette and CIRCL staff to always support my craziest ideas

Contact via Twitter/Mastodon/Github **@0xrawsec**

Feel free to open issues, ask questions, give feedbacks/suggestions …

References:

Kunai: https://github.com/0xrawsec/kunai

Aya: https://github.com/aya-rs

eBPF: https://docs.kernel.org/bpf/index.html

Questions / Comments

If the presentation was not technical enough, let’s chat together afterwards!