Readme

How to navigate the mess I threw in here

Prepared by Duncan Sparrell, sFractal

# OVERVIEW & PURPOSE

Intent of plugfest is to start the process of having reference SBOMs for the purpose of evaluating SBOM tools. sFractal is contributing just to move the ball forward due to his belief in the value of SBOM. All time spent on this project was pro bono and sFractal has no interest in selling any tools (although he does sell his time as a consultant when his spouse lets him take a break from retirement and he’s not too busy doing all his pro bono activities).

# OBJECTIVES

1. Show that SBOMs can be created using automated build tools.
2. Discover problems in formats, tools, process - AND SOLVE THEM!
3. Of secondary importance - make the community aware of QuadblockQuiz at RSAC in Supply Chain Sandbox at RSAC this year (and maybe rope volunteers to add more SBOM questions to the trivia/quiz part of the game)
4. Of secondary importance - make the community aware of Cybersecurity Automation Workshop coming in June and maybe rope volunteers in including their tools in that plugest
5. Of secondary importance - make the community aware of Elixir/Erlang/OTP and it’s value in programmer productivity, system resilience, and cybersecurity due to it’s “Let it fail” paradigm (ie it is designed to continue operating in the presence of hardware and software errors).
6. Of secondary importance - make the community aware of Neves - a resilient IoT operating system designed to take advantage of Elixir/Erlang/OTP’s resilience.

# Directory organization

* Blinky.sf.swift
* Blinky.env.hex
* QuadblockQuiz
* TwinklyMaha

Intent was to have several more blinky variants (eg adding upstream components, using it as a downstream component, doing SBOM of full firmware build, ..) but I ran out of time. All told I spent less than 2 hours on this - and most of that was on the readme’s. My intent was to have spent alot more.

# Blinky.sf.swift - single file - SwiftBOM generated

Blinky.sf.swift is based on Blinky provided by organizers (https://github.com/nerves-project/nerves\_examples/blob/main/blinky/lib/blinky.ex).

Sf = single file - meaning done in a vacuum of the environment it is being built for. Eg would work (sort of) on any system with Iex (interpretive elixir shell) installed.

swift = sbom created by manually entering data into SwiftBom tool (<https://sbom.democert.org/sbom/>)

Blinky.sf.swift directory contains:

* Readme.blinky.sf.swift
* CycloneDX-sfractal-blinky-0-1-1-1-4-2021-12-50.json
* CycloneDX-sfractal-blinky-0-1-1-1-4-2021-12-50.xml
* SPDX-sfractal-blinky-0-1-1-1-4-2021-12-50.spdx
* SWID-sfractal-blinky-0-1-1-1-4-2021-12-50.xml
* sfractal-blinky-0\_1\_1.png (as downloaded from the tool)
* ScreenShotGraph (screenshot with more info than download)

# Blinky.env.hex - elixir env - hex/sbom generated

Blinky.sf.swift is based on Blinky provided by organizers (https://github.com/nerves-project/nerves\_examples/blob/main/blinky/lib/blinky.ex).

Env means it was not just the single file, but an entire firmware build (sort of) but building the whole blinky application. But it did not get the upstream OS dependencies on dependencies. Intent was for that to be in next section I didn’t get to.

Hex means built using hex/sbom (<https://hexdocs.pm/sbom/readme.html>). Log files provided.

Files included:

* Readme.blinky.env.hex explains directory
* Bom.xml (that was what the tool named it)
* Log1.txt - log of creating raspberry pi firmware
* Log2.txt - log of creating sbom

# QuadQuiz.phx.hex -Outside Scope - elixir website - hex/sbom generated

*Outside scope - provided as extra*

See readme in the directory

# QuadQuiz.cloud.hex -Outside Scope - elixir website - hex/sbom generated

*Outside scope - provided as extra*

See readme in the directory

# Summary/Conclusions

*If only there was more time*

At least some material has been provided - hopefully enough ‘sweat’ to qualify.