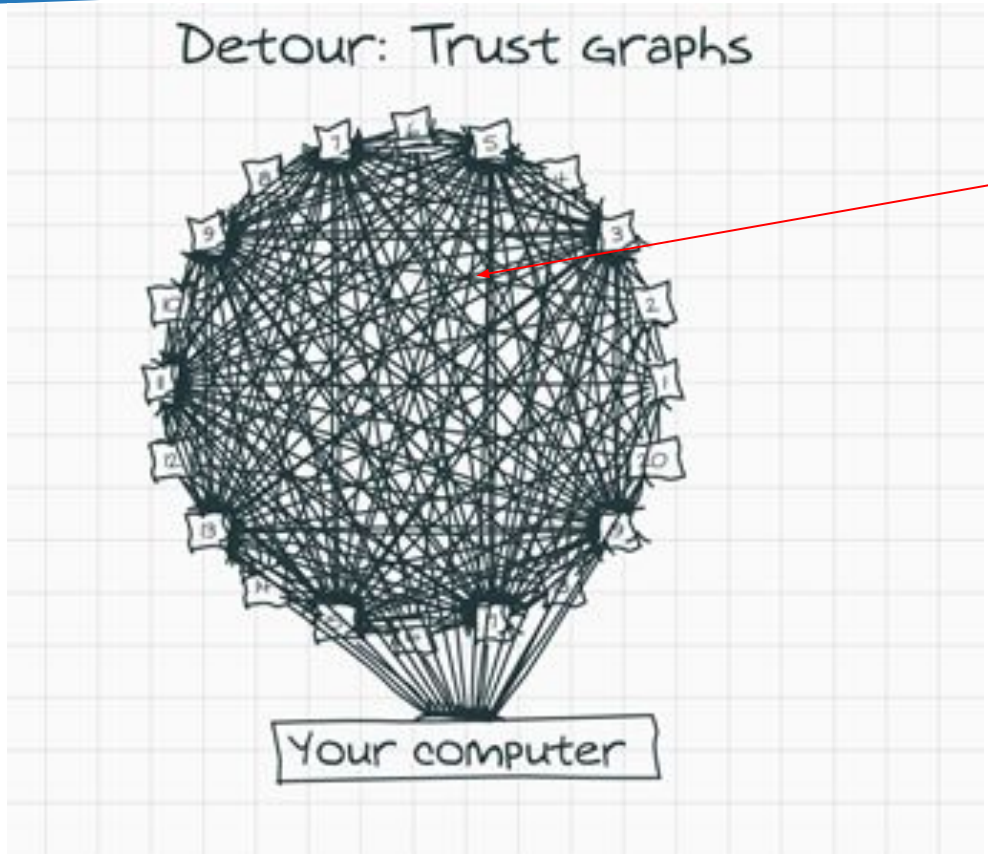
Reflections on Supply Chain security
CERT Vendor Conference, April 2023

Iván Arce, Chief Research Officer @ Quarkslab

FOUR VULNERABILITIES I LEARNED FROM

- The first vulnerability I've ever reported : April 22nd, 1997.
  »BIND vulnerabilities and solutions https://seclists.org/bugtraq/1997/Apr/86
- The last vulnerability I've reported : February 23nd, 2023.
  »TCG TPM2.0 implementations vulnerable to memory corruption https://www.kb.cert.org/vuls/id/782720
- Over 26 years I've coordinated reporting and disclosure of 100s of vulns.
  I learned something new from almost each one of them.
- Whenever a difficult/complex issue appeared I've asked CERT/CC for help. They obliged 🙏
- To talk about Supply Chain security I picked 4 *relatively* recent vulnerability reports.
  Each provided me some insight.
- 4 data points over 7 years ➜ Please allow me to generalize.

..BUT FIRST : SUPPLY CHAIN SECURITY TRUST GRAPH



YOU
ARE
HERE

Trust graph != dependency graph
The trust graph is not a tree
but if it was..
what would be the root?

Image courtesy of:
Thomas Dullien (Halvar Flake), *Re-architecting a defendable Internet*, O'Reilly Security Conference, Amsterdam 2017
https://drive.google.com/file/d/0B5hBKwgSgYFacC1jejJYSE1LTlk/view?resourcekey=0-JaTOSUC_e5A7yzCkPeEGHQ

CVE-2016-5080

Title: Heap memory corruption in ASN.1 parsing code generated by Objective Systems Inc. ASN1C compiler for C/C++

Summary:

Memory corruption. ASN.1 Compiler generated code that induced a heap overflow in the resulting program. Vendors use/inlined generated code in the firmware/software.
ITU endorsed ASN.1 compiler >> Self-developed ASN.1 parser. *Eppur si muove*

Where is your SBOM God now ?
Dependency graph != trust graph

CVE-2017-7932

Title:Vulnerabilities in High Assurance Boot of NXP i.MX microprocessors

Summary:

Memory corruption. Stack overflow in ASN.1 parser in Boot ROM, when parsing X.509 certificate specifying key to verify firmware signature. Entire i.MX family affected, not fixable (vendor issued new silicon). MCUs embedded in larger systems.

Follow the KISS principle
Make things updatable, but plan for unfixable vulns too

CVE-2022-20233

Title: Attacking Titan M with Only One Byte

Summary:

Memory corruption. An out of bounds write in Google's Titan-M chip while parsing Keyparameters in an ImportKey request. The vulnerability allows to write a single byte with the fixed value of "0x1" to the chip's memory (with some contrains).

Everything is exploitable unless sufficiently studied to say otherwise
Logging, debugging, monitoring FTW

CVE-2022-20233

Title: Vulnerabilities in the TPM 2.0 reference implementation code

Summary:

Memory corruption. OOB write to, and OOB read of, TPM memory due to bad parsing of encrypted parameters in a TPM2.0 API request. The vulnerability allows to write 2 bytes past the buffer to receive commands. Impact highly dependant on each vendor's implementation. DoS, memory leak, code exec, sandbox escape.Hardware (firmware), software, cloud providers affected. Open source reference implementations.

Reference implementations are the supply chain of supply chains
Opaqueness makes risk assessment more difficult
Your EOL is not my EOL

WHAT DID I LEARN ?

OBSERVATIONS ON SUPPLY CHAIN SECURITY

Memory Corruption. Didn’t we solve this already ?
Move to memory safe languages, adopt mitigations.

Know your trust graph. Upstream, downstream, sidestream.

SBOM is not the answer, SBOM is the question. The answer is yes.

SBOM is not sufficient.

Keep it simple, make it updateable.

More transparency, better tooling. Binary analysis is a must.

Bugs can’t be hidden. Vulnerability researchers are your friends.

Reference implementation must be EXTRA VETTED. Standards bodies beware.

THANK YOU

Quarkslab