Formal models of hardware peripherals

1. Introduction

1.1. Project identification and rationale

Security and privacy are important concerns. We study the mechanisms that prevent abuse of hardware peripherals. Our goal is to build and proof-check a model of secure hardware peripherals. Our networks, operating systems and applications must interact with secure hardware peripherals such as input/output devices (UART, USB, network controllers), interrupt controllers and coprocessors (GPU, FPGA).

There are formal models to test software and to guarantee data security. Hardware peripherals and their connections are usually not part of such tests. The scope of this project is to build and verify formal models of some common hardware peripherals such as USB, network controllers (NIC) and peripherals that have built-in processors (GPU, FPGA).

1.1.1. What's the problem?

Schneier's problem is that everybody can outsmart themselves and consequently you can also be intruded. ("Any security expert can invent a cipher that he himself can't break"). We study the hardware security applications of this problem.

The common data security patterns often use the placeholder actor names Alice, Bob, Carol, Eve, Mallory, Peggy, Victor, Trent etc for different roles and functions. Typically Alice and Bob are the ones sending data and the other guys are actors with different functions. Will they ever be 100 % safe?

1.2. Project collaborators

Niklas Rosencrantz (nik@kth.se)
Jonathan Yao Håkansson (jyh@kth.se)

1.3. Time plan

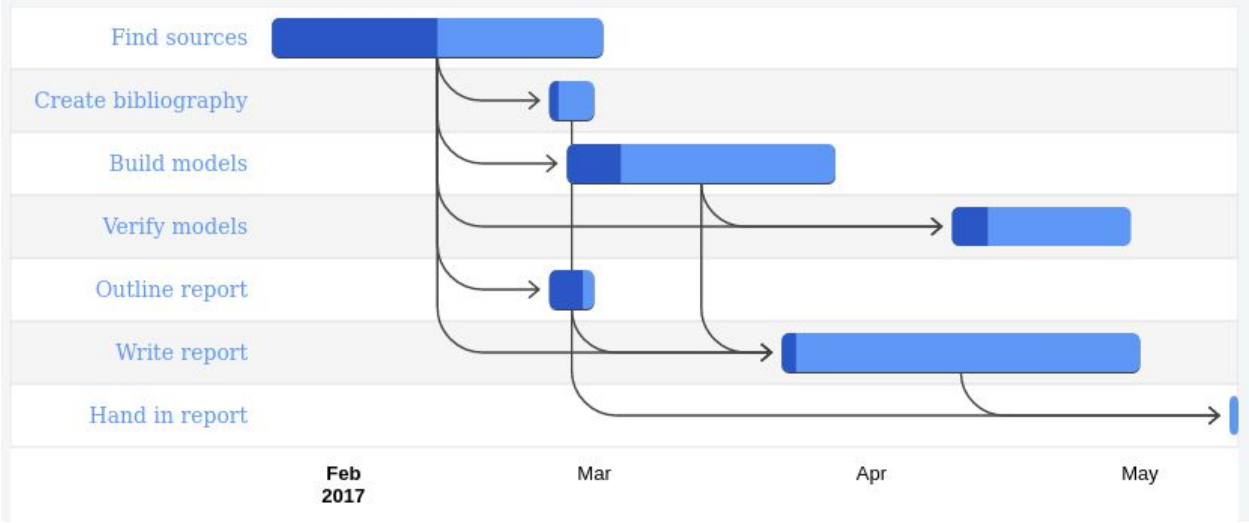

Figure 1.2 Gantt chart

1.4. What is formal verification?

Formal verification is a mapping done to ensure that the hardware schematic matches the HDL model. This means checking every possible input and checking every possible state of the model. It is tested that the outputs are the same for the same inputs and states.

In principle a formal verification is similar to a mathematical proof that covers all possible cases.

1.5. Benefits

The benefits is learning and getting results both from theoretical modelling and verifying and testing the models and verifying real hardware peripherals.

2. Background

3. Methodology

We build, examine and test hardware peripheral security models. We use tools that check Hoare contracts.

- We learn proof-checking
- We build a hardware peripheral security model and proof-check it. We test it with Quartus, Promela, LTL, Büchi automata, SPIN or similar tools
- We verify Hoare logic using Hoare contracts
- We learn the HOL proof-tool Isabelle and how it's better than the check testing framework that we already use
- We learn hardware abstraction layers (graph theory (python/oop?))
- We read the relevant docs about asserting hardware and firmware
- We look how to prove VHDL and/or Verilog
- We see what Quartus can, can't, won't or needs
- We analyse, investigate, build and check
- We look at brute forcing software and brute forcing hardware
- We discuss different models for example different architectures
- We look at a specific software problems such as Ken Thompson's gcc attack (1984). The Unix backdoor was compiled into next version of the compiler and not even visible in the source code.
- We examine hardware Trojans
- We mention common problems that appear and common patterns e.g. flattening
- We study the OSI layers specifically networks (ISO 8348, X.213), connections(ISO 8886, X.212) and the physical layer (ISO 10022, X.211)

3.1. USB

- QEMU info page
- QEMU git info docs
- USB drivers
- Communicating with hardware
- Device controller source e.g. OHCI, omap-specific
- OHCI controller specs

We use hol for proof-checking.

3.2. Useful utilities

- Wireshark. Sniffer and protocol analyzer tcpdump
- Command-line based sniffer netwox
- netcat (nc) - Lots of different tools, can be used for a simple client/server
- Nmap
- Iptables
- ufw ("uncomplicated firewall")

The verification could be performed using the HOL theorem prover (which? holzero?) and a model of the hardware (which?). We prove isolation of code and data of a hardware peripheral.

We demo on real hardware using a hypervisor that protects a peripheral.

4. Results

We document our results, both theoretically and empirically.

5. Conclusion

We expect to conclude that a computer user will not be 100 % safe in principle. We aim to conclude a result what the best and safest usage of hardware peripheral is.

6. Bibliography

[1] P. Mishra et al: Hardware IP Security and Trust, Springer 2017
[2] Ross Anderson, "Security Engineering", https://www.cl.cam.ac.uk/~rja14/book.html
[3] M. Hicks, M. Finnicum, S. King, M. Martin, and J. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in IEEE Symposium on Security and Privacy (SP) 2010, pp. 159–172
[4] Formality, User Guide, http://www.vlsiip.com/formality/ug.pdf, 2007
[5] 7] B. C¸ akir and S. Malik, "Hardware trojan detection for gate-level ics using signal correlation based clustering," in Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium, 2015, pp. 471–476.
[6] Wishbone bus [Online]. Available: http://opencores.org/opencores,wishbone
[7] Mads Dam, Trustworthy Security Using Formal Methods
http://www.ices.kth.se/upload/events/103/b38298d5102f4ca69fcb46d3effc45e1.pdf