Critical Reading of Gabriel Somlo's "Toward a Trustable, Self-Hosting Computer System"

Jon Trossbach, jt3691@nyu.edu, Nina Zeng, ninazeng@umich.edu

***Abstract*—The open source software development model has largely established its usefulness in allowing known vulnerabilities in software to be readily discovered and corrected while also allowing end users a better chance of attributing the functionality of their system to chosen compiled binaries. What Professor Somlo proposes is that we may be able to use FPGAs in order to guarantee end users be able to fully attribute *everything* their system does to a known set of HDL and software files. Though professor Somlo uses purely open source projects to demonstrate the possibility of this idea it is worth noting he does not believe this technique needs to be limited to open source implementations [1] but he does seem to acknowledge the role his technique could have in the realization of mass producible computing devices consistent with the essential freedoms set by the Free Software Foundation and the open source hardware definition set by OSHWA [2] [3].**

I. Summary

There are open source options for almost everything that gets compiled and ran on a computer. Using the Librem 15 – a computer the company Purism made which does it's best to respect the open computing ethos with a modern design – as a case study you can see from Figure 1 which is included as an appendix – the last unopened and unable-to-audit parts of a modern computer are the firmware and the RTL of the processor [4]. There is an argument to be made here that trustworthy hardware is different from that of trustworthy software though; and it says that in order for hardware to be protected from supply chain tampering it is best to obscure it before it reaches the fabrication facility and test it once it gets back to you in silicon form to ensure that it exhibits only the functionality you intended for it, nothing less and nothing more [5]. This trust model however has thus far been used to completely cut out most, if not all, end users from being verifiably able to trust the computing hardware they are buying; let alone the fact that such silicon verification is often hard to impossible and rarely ever attempted on commodity hardware.

In a novel a approach Professor Somlo's paper operates on an unorthodox hardware trust model, viz. that by virtue of randomize placement of security crucial hardware components within an FPGA we can work to create a trustworthy computing system where even an end user can verify all the attributes of its functionality. Somlo's hope and claim is that the randomizing of both potential hardware components and the placement of such components within an FPGA via it's bitstream with be enough to thwart a large class of hardware supply chain attacks or make it sufficiently difficult to do a successful supply chain attack and get any meaningful information out of it. In order to demonstrate his idea, Somlo built an entirely FLOSS RISC-V Linux system and tested it against well known processors from the 1990s. It performed only moderately worse than those processors for a number of benchmarks while running an out-of-order superscalar processor design with a clock rate of 65MHz.

II. Review

While many in the open source community have been discussing using FPGA's in the way Somlo has done, Somlo appears to have been the first to create a fully Linux capable computer with an entirely open tool chain. His work allows us to reconsider what is possible in terms of bringing end users into the trust models used to verify the devices on which we all rely. It is no stretch of the imagination that many users will be able to confidently rid their devices of undocumented blobs in the future and with more powerful FPGAs being primed for open tool chains, more of the computer will be verifiable like the Debian Project's reproducible system of compiled software binaries [6]. Adding reproducible bitstreams and microcode could easily be done in the ecosystem of the Free Software Conservancy, severely limiting the attack surface against the computing devices that are implemented using Somlo's approach. Similar to how reproducible binaries exist for the Debian operating system, reproducible synthesized gateware could be hosted similar to how operating systems images and related binaries are securely shared and hosted today. This is not a panacea however and there is still much to be worked out.

Ignoring side channel attacks that require physical access to the hardware, it is still possible – and Somlo didn't pay this much attention in his paper – that as a result of wide adoption of Somlo's approach, we may just see a new breed of hardware supply chain trojan that tries to embed itself in the FPGA's silicon and use some form of statistical technique to position and reconfigure itself in such a way that it intercepts sensitive information. As he stated in his paper it only takes about 20 LUTS to produce a severely harmful trojan, and aside from something like ptychographic x-ray imaging which today appears outside the reach of a mass production supply chain [7], there is plenty of room to try and fit one of these trojans into the silicon of an FPGA even though it might be hard to implement.

Taking this further, Somlo claims that such trojans would be detectable in either a nondestructive or destructive test of the FPGA's silicon and, economically, it's not an entirely unreasonable proposition to think that in the future an FPGA provider would likely be kept in check by a combination of market forces and independent researchers trying to credibly audit the reputation of an FPGA manufacturer. For this line of argument to hold water, the supply chain would need to introduce some ability to either reliably detect tampering or offer some level of non-repudiation about the hardware being unquestionably of their own manufacturing (which I will explore in the final Discussion).

No matter what, an FPGA will always perform worse than a hardcore and will not likely be usable for battery operated devices so from a typical end users perspective this may not be a viable solution given the wide scale adoption of everything from laptops to even lower powered devices like phones or watches. And even if everyone had a softcore FPGA computer at home, the mixing of trustable and untrustable personal devices could likely be enough to prevent people from habituating security practices to protect themselves as is well documented in the literature of more social science or human behavioral aspects of why information security is so difficult to maintain [8].

And finally this may be minor given how much we are already re-imagining the hardware we are using in Somlo's approach but Somlo explicitly ignored how trojanized firmware or persistent memory devices like SSDs or HDDs might be protected against or incorporated into a system level verification technique like he was eluding to in the conclusion of his paper. Unless we mearly incorporate the embedded processors typically responsible for such devices in our FPGA-softcore configuration, it is not clear how persistent storage is to gain trust within a future computing system.

III. Discussion

As eluded to above, assuming that a manufacturer gave you an FPGA and that it was not tampered with en route to you is too strong of a presumption to ensure your hardware is secure. What could be done to ensure that the hardware from the manufacturer is the same hardware you received? I propose that we could ship the FPGA while connected to a battery within tamper resistant packaging and a maximal-density bitstream configuration loaded into it such that an output which takes advantage of the maximal-density of the FPGA correlates to a value placed on a publicly verifiable ledger on a blockchain network hiding the proof-value of the verifiable FPGA configuration using fully homomorphic encryption similar to the zk-SNARKs protocol (a well-tested NTT blockchain scheme). This could be done by loading onto the FPGA a series of hash functions followed by a variable size XOR or series of XOR's to fill the rest of the LUTs. The output of this function can feed into itself during the course of shipment. Upon receiving the package, if the values do not align you would have an indication that your device has been tampered with en route to you. One may imagine a scheme as follows: the initial seed and the hash function configuration are uploaded to the blockchain. The consumer would receive the end timestamp and the output. They could then download the hash function configuration and the seed and run it to verify that at the particular timestamp at which they stopped the FPGA computation, that indeed they were suppose to receive that particular output. It would be difficult, if not impossible, to meaningfully recreate the output from the bitstream configuration if this configuration were able to be both reliable end exploiting the maximum information density of the FPGA given an initial secret seed value. On top of this there could be a series of tests the end user could do to verify the functionality of the FPGA upon receiving it, thus allowing the end user to verify that they have an FPGA with the manufacture specifications; nothing less and nothing more.

The bitstream placement is also a well known place that attackers have targeted [9] but the small embedded processor responsible for the bitstream could be made with transistors that are verifiable with a commodity microscope or even with the naked eye [10]. And though this may seem inconvenient, it is nonetheless achievable and the ability to verify will often serve more as a good enough deterrence rather than as a true security practice in most use cases.

In a different discussion vein, wide scale adoption of Somlo's approach still has performance challenges and – though security may be affected – it is worth finding out the performance to security trade offs of adding small ASICs into the FPGA that affect the critical paths of a processor which would allow faster clock rates of a given FPGA-softcore implementation. For example, the write back system is often the most lengthy critical path in the system and could have its number of required LUTS reduced with a reconfigurable ASIC or other logic capable of accelerating computation within the FPGA.

Finally, getting an estimate for the maximum clock rate of these FPGA-softcores could go a long way in allowing us to find funding and sell the promise of Somlo's approach. For example, today in 2021 the maximum advertised clock rate for an FPGA appears to be 2GHz for a 7nm CMOS implementation‘ [11] but the critical path of all the LUTS strung together might require a clock rate that is much slower than 2GHz. Clock rate estimates for a 3nm FPGA-softcore implementation should also be done for the purpose of marketing these ideas in order to fund research into these areas.

IV. Conclusion

There are limited resources with which the open source community can experiment, is Somlo's approach the right direction for the community to invest it's resources today? I think so. Whether or not Somlo's approach remains popular for 5 years or 1000 years, the open source community is going to be able to experiment and democratize processor designs in a way that will enable everyone to discover the most secure processor designs and maybe even be able to enable a system level architecture that looks impenetrable when compared to what we have today through the rapid democratized experimentation that only an open community can facilitate.

References

[1] G. L. Somlo, “Toward a trustable, self-hosting computer system,” in *2020 IEEE Security and Privacy Workshops (SPW)*, 2020, pp. 136–143. [Online]. Available: https://youtu.be/5IhujGl_-K0?t=955

[2] F. S. Foundation, “What is free software.” [Online]. Available: https://www.gnu.org/philosophy/free-sw.html

[3] O. S. H. Association, “Definition.” [Online]. Available: https://www.oshwa.org/definition/

[4] A. B. Haung, “Impedance matching expectations between risc-v and the open hardware community,” 2017. [Online]. Available: https://riscv.org/wp-content/uploads/2017/05/Wed1100-impedancematch-huang.pdf

[5] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, “Trustworthy hardware: Identifying and classifying hardware trojans,” *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[6] S. F. Conservancy, “Reproducible builds.” [Online]. Available: https://reproducible-builds.org/

[7] M. H. et al., “High-resolution non-destructive three-dimensional imaging of integrated circuits,” *Nature*, vol. 543, no. 1476-4687, pp. 401–406, 2017.

[8] R. Anderson, “Why information security is hard - an economic perspective,” in *Seventeenth Annual Computer Security Applications Conference*, 2001, pp. 358–365.

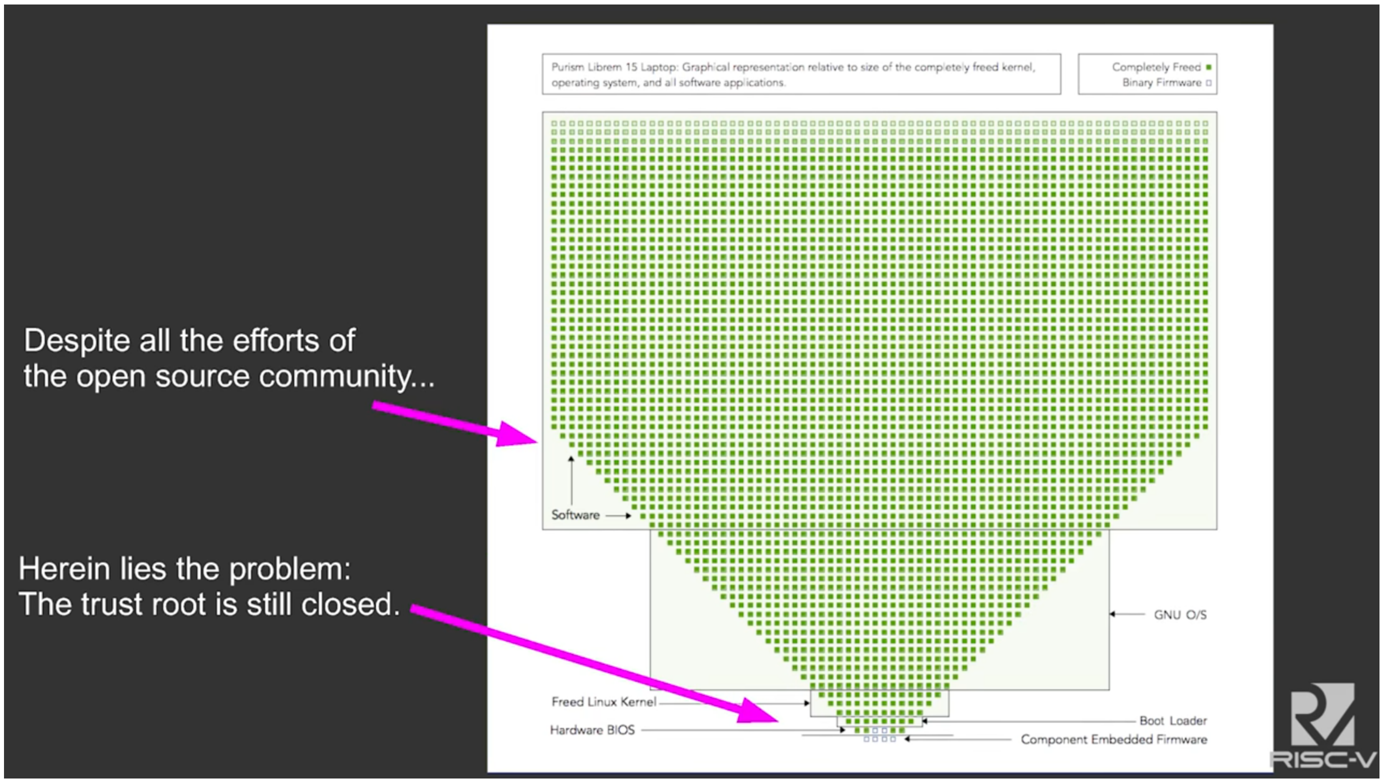
[9] P. Swierczynski, M. Fyrbiak, P. Koppe, and C. Paar, “Fpga trojans through detecting and weakening of cryptographic primitives,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1236–1249, 2015.

[10] E. Schlaepfer, “The monster 6502.” [Online]. Available: https://monster6502.com/

[11] A. S. Corporation, “Speedster7t fpga datasheet (ds015),” p. 23. [Online]. Available: https://www.achronix.com/sites/default/files/docs/Speedster7t_FPGA_Datasheet_DS015_0.pdf

Appendix

YouTube Presentation: https://youtu.be/_AvEd_DKU3E


Despite all the efforts of the open source community...

Herein lies the problem: The trust root is still closed.

Purism Librem 15 Laptop: Graphical representation relative to size of the completely freed kernel, operating system, and all software applications.

Fig. 1. The Librem 15's Hardware and Firmware are the only closed aspects in its design [4]