Towards Securing Hard Real Time Networked Embedded Devices and Systems: a cBPF Implementation for an FPGA

Hans Dermot Doran, Sven Schneider, Prosper Leibundgut, Stephan Neuhaus
Institute of Embedded Systems
Zurich University of Applied Sciences
Winterthur, Switzerland
{doran, scdv, leiu, neut} @zhaw.ch

Stefan Eberli
Duagon AG
Dietikon, Switzerland
stefan.eberli@duagon.com

Abstract— In this body of work, we describe preliminary work in implementing a Berkely Packet Filter in its original conception, in an FPGA. The purpose is packet filtering and ingress traffic shaping in security-relevant applications in distributed embedded nodes. We specifically target PROFINET nodes in hard-real-time applications where network security is a known issue. We describe the motivation, implementation and verification including performance characteristics. We posit that such a filter can be used to not only for protection against simple denial-of-service attacks but also for ingress protocol management and be used in the implementation of system-wide security policies.

Keywords—Real Time Ethernet, Network Security, Packet Filtering, Berkeley Packet Filter, FPGA, PROFINET

I. Introduction

New security demands made on device manufacturers place challenges on the manufacturers' hardware. It is well understood that the cryptographic algorithms required for security are computationally expensive to a degree matching the computational expense of the run-time application, henceforth application. It is also well understood by industry that in the factory and process automation domains, device lifetimes of 20+ years are considered normal. The creativity of malicious intruders into systems and networks knows few bounds and the automation domain has, at least since Stuxnet [1], [2], understood that automation networks are very much at risk. What hasn't been given much consideration is the security-relevant maintenance effort required over the lifetime of the device. The operating systems of personal computers (PCs) are subjected to numerous updates over their lifetime. It is hard to imagine industry showing the same patience with factory downtime for security updates, including the inherent risk that the machine won't restart because of a faulty update on one node. We intuit that on-line bump-less updates will be preferred and the update should be verifiable, small and not affect the device booting process.

Network security in factory automation consists of (management) processes, protective algorithms and protective software. In-place processes help reduce the exposure to external threats. Protective algorithms help protect access to and traffic between, networked devices. Protective software includes that software than can detect, prevent and recover from a malicious attack. Our contention is that packet-filtering fits the above criteria for use in an industrial Ethernet security context. We demonstrate an implementation of the Berkeley Packet Filter. This paper will present our arguments and some experimental work.

The paper is structured accordingly – we complete this section with some further words on motivation and then relevant prior work. In Section II we shall describe the implementation activities so far and in Section III conclude with a summary of results and some thoughts on future work.

A. Motivation

In the context of some activities in the domain of Real Time Ethernet, specifically PROFINET security, the authors have been holding extensive discussions with industry representatives about the verification and validation (V&V) of HW/SW partitioned security implementations. Long-term maintainability concerns are a prominent feature of these discussions. To this end we are examining solutions that can be subjected to differential V&V [3].

The PROFINET Netload certification suite [4] consists largely of denial of service (DoS) tests. We previously conducted an unpublished security analysis where we noted a number of system-vulnerabilities over and above the relatively simple DoS tests. Whilst we made no comment on the protection against these vulnerabilities, we maintain that these are not solved by simple blacklisting and/or whitelisting. Other authors have noted similar complex vulnerabilities waiting to be discovered, for instance buffer saturation attacks [5].

There are a number of architectures available to implement protective structures, such as the `netfilter` [6] architecture known from Linux through which custom firewall functions can be hooked in (i.e adaptively instantiated). Embedded applications rarely need ad-hoc adaptability of the type offered by `netfilter`. Packet filtering infrastructure also offers the ability for traffic-shaping which we intuit to be especially useful in an embedded application, especially under real-time constraints.

The concept behind the Berkeley Packet Filter (BPF) is attractive as it is structured as a virtual machine and as such relatively easily implementable in HW. The original implementation of the BPF [7], henceforth the (classic) cBPF, has been superseded by the extended BPF (eBPF) [8] with a

We gratefully acknowledge the financial support of the Swiss Innovation Agency, Innosuisse, under project grant 42637.1 IP-ENG

virtual machine featuring a just in-time (JIT) compiler. Many of the attractive features of the eBPF come courtesy of the tight integration with the Linux kernel and hence use-cases that extend beyond packet manipulation. For our purposes – flexibility on a small scale – it appears that the implementation of the cBPF in hardware as a security orientated packet processor, warrants further investigation.

B. Relevant Work

There is a substantial body of work on security for general purpose networked computer systems. Packet filters have long been used for firewall and traffic shaping purposes [9]. The BPF in both cBPF [7] and eBPF [8] variants have been well documented. An embedded (software) version of the eBPF was presented by [10].

Industrial network security has also been a subject of study. Modbus/TCP has undergone substantial scrutiny due to its importance in strategic infrastructure automation systems (electricity generation, chemicals etc.) making it representative for the application domain as a whole. Application-domain relevant works include [11] who develop a taxonomy of a substantial number of possible attacks, [12], [13] who provide a general overview of application-domain security issues, additionally referencing PROFINET and POWERLINK. A useful performance comparison of eleven TLS cyphers suites is provided by [14]. Of the hard real time protocols in existence, [15] take issue with the security of EtherCAT and for PROFINET we observe a substantial time-gap between the initial voicing of concerns [16] and results from the relevant specification committees [17].

Packet filtering in FPGAs has a long history especially in routing applications [9]. The usefulness of the eBPF coupled with the speed of FPGA processing has also not gone unnoticed and there are both research [18] and commercial [19] solutions available that convert eBPF code into hardware processing elements. To the best of our knowledge there is no hardware implementation of a cBPF.

One of the novelties of this body of work is therefore to propose a cBPF implementation in an FPGA.

II. Proof-of-Concept Implementation

A. Architecture

The packet filter will be integrated into a FPGA-IP based PROFINET communication controller [20]. The controller features a stringent data-flow design optimised for low-latency data transfer and both the packet filtering and the in-line encryption/decryption must be integrated into this architecture. The ingress system is shown in Figure 1. An ingress frame enters the delta unit which retains the frame until the filter bank instructs the delta unit to forward. Meantime the frame words are written to a configurable number of data buffer components, in this diagram three. The three buffers shown are for the communication classes isochronous, alarm and asynchronous. Once the frame has been identified and temporally validated (i.e no asynchronous frames in the PROFINET isochronous phase) the buffer components are notified for whom the frame is. This notification can be combined, usually at synthesize time, with assertion of an interrupt. This allows calling of a (prioritised) communication-class specific interrupt service routine which can then call the relevant driver/process.

We wish to maintain this low latency design in the light of new security requirements. In PROFINET there is an expectation of both authenticated real time (RT) and non-real time (NRT) packages using, for instance AES and SHA256. Both are block cyphers and a different body of work is investigating the integration of streaming cryptographic IP. In the context of the architecture the latency handling is an issue with the following considerations; there are NRT ingress frame that will not require authentication – for instance PROFINET's discovery and configuration protocol (DCP); If latency is present in either filtering and/or authentication decryption then these operations can be carried out in parallel – either unit, although in practice it will most likely be the filter unit, shall be able to drop a packet and so halt the work of the other. This results in the parallel architecture of Figure 2.

B. Implementation

The cBPF features 27 64-bit instructions, all of which are supported in this implementation. Inclusion of the multiply and division functional units are implemented as synthesise-time options. The current implementation supports 256 instructions instead of the maximum allowed 512 largely for convenience sake in instantiating necessary block-RAM. We anticipate that filters will not require longer code nor do we expect any issues extending the memory addressing range should this be necessary. The instruction RAM can be implemented as a dual ported RAM allowing the update of the instruction code. All instructions can be issued in one clock cycle and execution ranges from one to five clock cycles. In the case of the jump-on-true (jt) and jump-on-false (jf) instructions, a static branch prediction is used and the default action can be also be configured at synthseise-time. The interfaces of the cBPF component are noted in Figure 3.

An important latency minimizing feature is that the instruction execution can be halted if the data byte has not yet been received. In other words, the frame must not be completely received before filtering can start. The core cBPF synthesis to the following characteristics Table 1. It is clear that multiple such filters can easily be instantiated even in a small FPGA.

ALMs	ALUTs	DSP Blocks
912	1162	2

Table 1: Intel FPGA Resources Required for cBPF Implementation

C. Verification and Validation

The core IP was verified in an initial phase using the standard `tcpdump` filter set [21] in simulation. A proof was given by instantiation in a FPGA. The viability of a cBPF as part of a security concept is a multi-faceted argument and will be discussed in the next section. Our (in simulation) verification setup can send a sequence of frames to the device under test which can check the sequence and set status bits if a correct sequence is detected. This setup is duplicated in essence in a real-world differential-test setup.

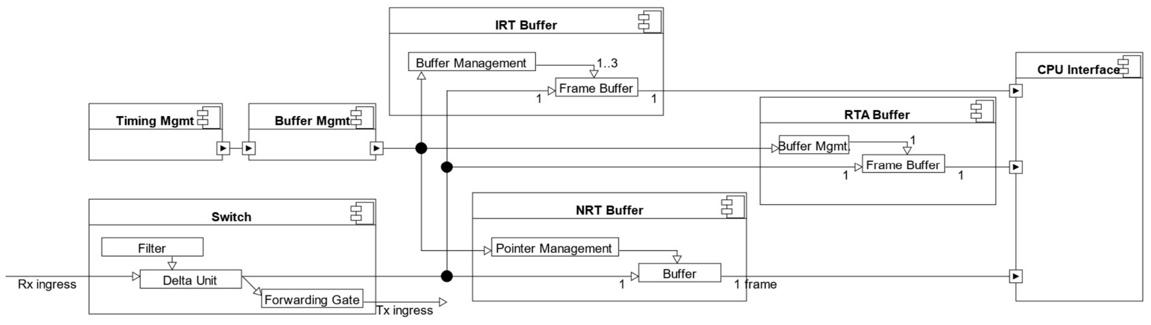

Figure 1: Frame Management RTE Switch. Rx path left, Tx path right. The buffers can be determined at synthesis-time and are generally divided into traffic types (Isochronous, Alarm, Asynchronous according to handling requirements). Isochronous real-time buffers are often instantiated as double or triple single-frame buffers, whereas asynchronous traffic can be placed in a larger FIFO buffer.

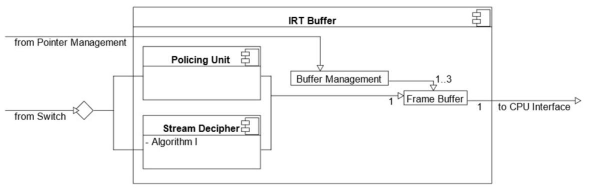

Figure 2: Proposed Expansion of a Buffer Subsystem (example IRT) with cBPF Filter and Streaming Decryption

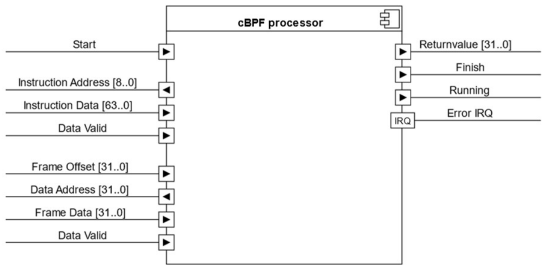

Figure 3: Interface Description cBPF IP

III. Discussion and Further Work

To a certain extent the availability of a cBPF asks more questions than it answers, many of which open a discussion on the sophistication of a security concept on different levels. We shall approach those questions that we consider offer innovations in future technology and understanding of security systems. The first question we shall approach is to what degree a cBPF is useful in a stand-alone security context and how this can be expanded to a security-aware protocol management system. The second question we approach is that of how to integrate a device fitted with this technology in a security-aware networked embedded system. For both, innovations out of the scope of this paper are necessary.

As previously noted, the cBPF is a relatively simple filter system whose ecosystem consists of applications such as `tcpdump`, an application not considered relevant to embedded security. The distribution of frame buffer memory is different in embedded and server systems so the concept of traffic shaping in terms of security requires some rethought when applied to embedded systems. In this body of work, we wish to situate the filter in a system-wide context. At a minimum the filter should be cycle-aware and Rx buffer-size aware. A concept of state, which the cBPF does not possess, is necessary.

A. Adding State

Applying filtering as traffic shaping in an isochronous context requires, at the very least, an understanding of cycles. Such identification in a PROFINET IRT switch will be achievable through status bits generated by the internal switch logic. An internal communication-controller state status word such as buffer-fill status can be made available. Application SW may wish to impart protocol-state information. The system controller may also wish to impart security state information to a security process in the node. Enabling these features imply an extension to the cBPF as it does not support state, once a filter instruction has run its course the machine is reset. The store instructions for instance, will only store either the accumulator or the index register into scratchpad memory. It follows there are at least two methods of noting state. A memory added onto the beginning and/or end of a (in this case notional) frame whose contents have been written by other circuit elements (for instance internal buffer state and/or cycle phase) and/or extending the instruction set. Current work is adding a dual-ported memory to the notional frame which can be written by internal circuits and software and read by the cBPF.

Both these options look initially attractive, but such manipulations will immediately remove the ability to directly transfer filter code from a PC/server system and differentially test the HW implementation. In other words, the benefit of "write-once run everywhere" and differential V&V is, at least partially, lost. We leave discussions on this point to future work.

B. Security-Aware Protocol Management

In a stand-alone context the discovery and configuration protocol (DCP) of PROFINET is an example of a somewhat problematic protocol. It is used at device boot to handle the fundamentals of device reachability namely IP address, hostname and the like. It is an unsecured protocol whose most egregious feature is a "Reset" broadcast message that can be, as the name suggests, used to reset the base connectivity parameters of the entire network. It may have uses in the boot phase of a network but during normal operation there is no call for it other than to act as an open invitation for abuse via a spoofing attack. In this context it also acts as a simple example for demonstrating a more sophisticated approach to security in multi-protocol environments.

The PROFINET literature is very clear on the use of DCP during the boot phase but omits to make any mention of its use in recovery mechanisms, for instance when the system controller has deemed the device to require a reboot. As an example of protocol management might function we propose a simple DCP protocol management system comprising of two phases, initalisation and initalised. In the default initalisation phase the cBPF accepts all DCP packets, including Reset. Once the device has been initalised, device software sets a bit in the status memory attached to the frame buffer of the cBPF and from that point on, all incoming DCP frames are dropped. We are currently verifying this functionality in simulation. This novelty represents self-management of protocol behaviour by a device to increase security.

C. System-Wide Security Management

We can extend the device-sided protocol management to system-wide management by integrating a feedback loop. Cutting off DCP communication after device initalisation limits the system controller's ability to change networking parameters, for whatever reason. To enable this, we leave to future work a so-called security profile. Most communication protocols operate an object model where device and application parameters (instantiated as objects) are gathered into a holding structure (instantiated as a profile) and accessed, by both application and remote node, at specified memory locations on the device. These parameters are device parameters such as device version, manufacturer string and, in the case of application parameters, motor torque, current and the like. Objects represent an abstraction of an inter-process communication channel. Virtual profiles - without direct connection to external I/O - can usually be instantiated. By instantiating a security profile, security parameters of the device can be situatively changed by the system controller to optimise the security integrity of the entire system. The authors are unaware of whether such a proposal has been previously made. This notional security profile could also function as the mechanism by which cBPF code is uploaded/updated to the device. Since the objects in the profiles are accessed over secure mechanisms as defined by the specific protocol, the security integrity of this profile would possess the integrity of the rest of the implementation.

D. Conclusion

In the light of this discussion we would summarize the attractivity of the cBPF as a device-sided building block supporting device security as well as potentially facilitating integration of the device in system wide security policies, both of which – as automation security is derived from IT security which assumes open networks, not closed ones like automation networks - we consider novelties.

REFERENCES

[1] T. M. Chen and S. Abu-Nimeh, 'Lessons from Stuxnet', *Computer*, vol. 44, no. 4, pp. 91–93, Apr. 2011, doi: 10.1109/MC.2011.115.

[2] R. Langner, 'Stuxnet: Dissecting a Cyberwarfare Weapon', *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, May 2011, doi: 10.1109/MSP.2011.67.

[3] A. Walz and A. Sikora, 'Exploiting Dissent: Towards Fuzzing-Based Differential Black-Box Testing of TLS Implementations', *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 278–291, Mar. 2020, doi: 10.1109/TDSC.2017.2763947.

[4] 'PROFINET Netload Robustness for Security Guideline (former Security Level 1 Netload)'. https://www.profibus.com/download/profinet-netload-robustness-for-security-guideline-former-security-level-1-netload (accessed Jun. 13, 2023).

[5] P. Ferrari, E. Sisinni, A. Saifullah, R. C. S. Machado, A. O. De Sá, and M. Felser, 'Work-in-Progress: Compromising Security of Real-time Ethernet Devices by means of Selective Queue Saturation Attack', in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, Apr. 2020, pp. 1–4. doi: 10.1109/WFCS47810.2020.9114505.

[6] 'netfilter/iptables project homepage - The netfilter.org project'. https://www.netfilter.org/ (accessed Jun. 13, 2023).

[7] S. McCanne and V. Jacobson, 'The BSD packet filter: a new architecture for user-level packet capture', in *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*, in USENIX'93. USA: USENIX Association, Jan. 1993, p. 2.

[8] B. Gregg, *BPF Performance Tools (Book)*. Addison Wesley, 2019. Accessed: Jun. 13, 2023. [Online]. Available: https://www.brendangregg.com/bpf-performance-tools-book.html

[9] D. Antoš, V. Řehák, and P. Holub, 'Packet Filtering for FPGA-Based Routing Accelerator', in *Prague*, Prague: CESNET, z. s. p. o., 2006. Accessed: Jun. 13, 2023. [Online]. Available: https://is.muni.cz/publication/630886/cs/Packet-Filtering-for-FPGA-Based-Routing-Accelerator/Antos-Rehak-Holub

[10] K. Zandberg and E. Baccelli, 'Minimal Virtual Machines on IoT Microcontrollers: The Case of Berkeley Packet Filters with rBPF', in *2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*, Dec. 2020, pp. 1–6. doi: 10.23919/PEMWN50727.2020.9293081.

[11] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, 'Attack taxonomies for the Modbus protocols', *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37–44, Dec. 2008, doi: 10.1016/j.ijcip.2008.08.003.

[12] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, 'A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics', *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1942–1976, thirdquarter 2020, doi: 10.1109/COMST.2020.2987688.

[13] A. Volkova, M. Niedermeier, R. Basmadjian, and H. de Meer, 'Security Challenges in Control Network Protocols: A Survey', *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 619–639, 2019, doi: 10.1109/COMST.2018.2872114.

[14] M. K. Ferst, H. F. M. de Figueiredo, G. Denardin, and J. Lopes, 'Implementation of Secure Communication With Modbus and Transport Layer Security protocols', in *2018 13th IEEE International Conference on Industry Applications (INDUSCON)*, Nov. 2018, pp. 155–162. doi: 10.1109/INDUSCON.2018.8627306.

[15] A. Granat, H. Höfken, and M. Schuba, 'Intrusion Detection of the ICS Protocol EtherCAT', *dtcse*, no. cnsce, May 2017, doi: 10.12783/dtcse/cnsce2017/8885.

[16] J. Akerberg and M. Bjorkman, 'Exploring Security in PROFINET IO', in *2009 33rd Annual IEEE International Computer Software and Applications Conference*, Jul. 2009, pp. 406–412. doi: 10.1109/COMPSAC.2009.61.

[17] K.-H. Niemann, 'IT security extensions for PROFINET', in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Jul. 2019, pp. 407–412. doi: 10.1109/INDIN41052.2019.8972209.

[18] J. C. Vega, M. A. Merlini, and P. Chow, 'FFShark: A 100G FPGA Implementation of BPF Filtering for Wireshark', in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, May 2020, pp. 47–55. doi: 10.1109/FCCM48280.2020.00016.

[19] 'Agilio eBPF Software - Netronome'. https://www.netronome.com/products/agilio-software/agilio-ebpf-software/ (accessed Jun. 13, 2023).

[20] D. Gunzinger, C. Kuenzle, A. Schwarz, H. D. Doran, and K. Weber, 'Optimising PROFINET IRT for fast cycle times: A proof of concept', in *2010 IEEE International Workshop on Factory Communication Systems Proceedings*, May 2010, pp. 35–42. doi: 10.1109/WFCS.2010.5548637.

[21] 'tcpdump(1) man page | TCPDUMP & LIBPCAP'. https://www.tcpdump.org/manpages/tcpdump.1.html (accessed Jun. 13, 2023).