Integrity Checking of Railway Interlocking Firmware

Ronny Bäckman, Ian Oliver(✉), and Gabriela Limonta

Nokia Bell Labs, Espoo, Finland
{ronny.backman,ian.oliver,gabriela.limonta}@nokia-bell-labs.com

Abstract. While uses of trusted computing have concentrated on the boot process, system integrity and remote attestation of systems, little has been made on the higher use cases - particularly safety related domains - where integrity failures can have devastating consequences, eg: StuxNet and Triton. Understanding trusted systems and exploring their operation is complicated by the need for a core and hardware roots of trust, such as TPM module. This can be problematical, if not impossible to work with in some domains, such as Rail and Medicine, where such hardware is still unfamiliar. We construct a simulation environment to quickly prototype and explore trusted systems, as well as provide a safe means for exploring trust and integrity attacks in these vertical domains.

1 Introduction

The increasing use and implementation of digitalisation technologies and infrastructures enabled by the use of 5G communications, edge and far-edge computing into safety related and safety-critical verticals, such medical and rail, is inevitable. This brings an increasingly larger attack surface for a wide (and expanding) range of cybersecurity attacks. We can no longer rely upon network or device isolation as mechanisms to provide security - indeed the authors here argue that, except in some exceptional scenarios, there is no such thing as an isolated network.

The European Railway Agency (ERA) launched a study in 2018 amongst 10 countries to get an overview of the existing Command, Control and Signaling (CCS) systems. This was done to assist ERA with the deployment of European Rail Traffic Management System (ERTMS). ERTMS aims at replacing the different national train control and command systems in Europe [5,11]. The EN 50126 standard [3] specifies the CCS-systems safety and functional safety in railway applications. Key specifications are made for the development process [14]. While provision is made for maintenance, patching, update and system provisioning, implementation is left solely to the vendor or contractor that is in charge of maintenance [38]. These specifications furthermore do not touch upon the subject of platform trust, meaning integrity guarantees of hardware, firmware and software during the mentioned processes.

The Railway CCS-systems have been in the past vendor specific implementations for specific applications, components and interfaces to comply with

A. Casimiro et al. (Eds.): SAFECOMP 2020 Workshops, LNCS 12235, pp. 161–175, 2020.
https://doi.org/10.1007/978-3-030-55583-2_12

national specifications. This has made the systems hard to attack through from cyberspace. With new interoperability specifications in Europe and the demand to lower the cost of the old relay based systems, new standardized CCS-systems will emerge [32]. While vendors have already implemented remote maintenance on these systems, these are proprietary solutions that are protected by public key infrastructure if at all [10].

This paper is split into the following parts. We first describe the use of trusted computing technologies [28] to provide security and integrity guarantees and how this technology can be scaled up from its firmware roots into a larger scale set of trust services. We then describe how trusted computing can be utilised and investigated through the use of a simulation environment [29] and how this simulation environment can be utilised to accurately describe a railway signalling system. We then describe a simplified firmware or configuration attack upon a railway system. We then conclude with a discussion of the role of trusted computing and simulation in the railway environment and how it impacts how cybersecurity is viewed and how it can be utilised to develop sound cybersecurity procedures [15,16] with particular emphasis on the remote attestation and firmware tampering case.

2 Trusted Computing Concepts

A trusted system [1,33] can be defined as one that provides trusted execution environment (TEE) to the workload running on it *and* one where its integrity can be determined. A trusted execution environment is defined through the provisioning of one or more of the features listed below. We concentrate on the first point in this paper that a trusted system is one where the system and workload integrity can be checked and assured. Such facilities are usually integrated into processing environment and Trusted Platform Module (TPM) [6, 7].

- integrity measurement
- secure storage
- execution isolation
- authentication
- attestation
- physical location

The use cases for such mechanisms are based around integrity checking of the firmware, bootloader, operating system and application components as typically seen in systems using a TPM [2]. Other use cases relate to secure data storage, such as key management and disk encryption schemes. Further use cases such as DRM also exist though mainly in more specific areas such as found in embedded system software and mobile/telecommunications devices. In all of these cases there is an underlying reliance on a core root of trust which is provisioned typically through an initial set of measurement code and a measurement mechanism.

2.1 Platform Integrity and Boot

The x86 platform **measured boot** with legacy BIOS or UEFI is standardised by the TCG. The crucial point for establishing trust in a platform starts from the moment the platform is started. In the initial start up phase the first code to run on the platform is a process called the Core Root of Trust for Measurement (CRTM). The CRTM's purpose is to start a chain of trust, to accomplish this it needs to be able to control the environment of the platform in the initial phase.

The x86 platform boot and measurements are shown in Fig. 1. Measurements are taken during the phases of the boot, through to starting the required operating system. These measurements are written to the Platform Configuration Registers (PCRs) through a process of extension thus forming a simple Merkel Tree-like structure: $PCR_{new} := hash(PCR_{old}||new)$. The measurements are a mechanism to spot changes in the code and configuration of the firmware. Reference points need to be made of trusted states of these values by platform manufacturers, OEMs, vendors and customers.

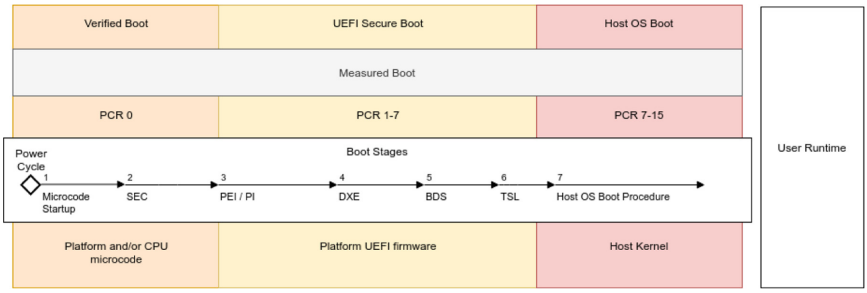

Fig. 1. Boot time measurements

2.2 Device Identity and Keys

The TPM provides key generation and storage features to provide unique keys which cannot be extracted. Incorporated in the TPM there are different seeds, which are multiple one-time programmable eFuses set during manufacturing time. These are used to create keys inside the TPM. Keys are protected in hierarchies as shown in Fig. 2 which can be locked during the manufacturing and supply-chain in a process called provisioning.

This combination of prior-provided keys can be utilised to form part of a device identity [4]. The TPM however provides two unique keys which themselves form the basis of the device's identity, these are known as the Endorsement (EK) and Attestation Keys (AK) and are used in a number of processes: the EK is effectively a root certificate and the AK - derived from this - is used in signing cryptographic measurements from the TPM.

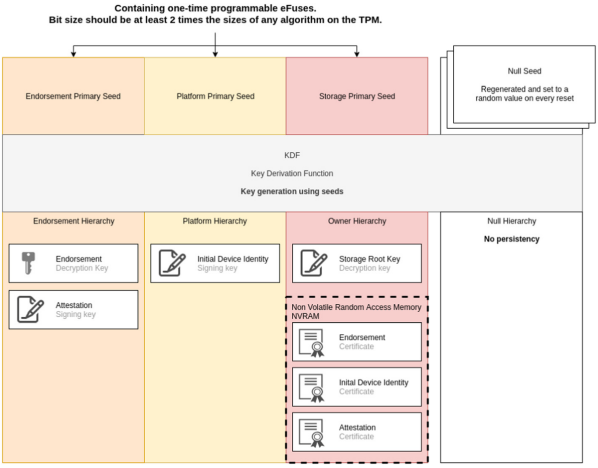

Fig. 2. Certificate and key storage for provisioning and attestation

The TPM uses an attestation structure - known as a quote - that contains a digest of one or more requested PCR measurements, certain device meta-data, the name of the signing key as well as limited nonce and user supplied data. This quote is also signed and verifiable against the TPM's specific attestation key further guaranteeing the provenance of the information.

2.3 Typical Integrity Attacks

Attacks against industrial control systems (ICS) can be described as cyber-physical attacks and they involve more layers than every day criminal attacks [9, 36]. The layers can be divided into an IT layer which is used to spread the malware, the control system layer which is used to manipulate process control and, finally, the physical layer where the actual damage is created.

Depending on the physical security of the system it can also be possible to directly target the Industrial Control System layer without touching the IT layer [37]. In railway systems the security of the control systems are not as high as in private production facilities. This makes the attack vector against the control system more compelling as has been seen with StuxNet and others [8,12,21,27].

One interesting approach to quantifying the amount of potential attacks against railway systems has been the HoneyTrain project [20][1] which has utilised a honey pot to simulate railway infrastructure [35]. While addressing a different area of security, specifically API attacks and denial of service, this approach has demonstrated clearly the amount of attacks (approximately 2.7 million individual attempts in one week in this study), the availability of attack vectors and the ease by which they might be utilised to deliver a much more destructive payload.

[1] https://news.sophos.com/de-de/2015/09/17/projekt-honeytrain-hackerwork/.

Combining attacks on the two lowest layers is also possible, especially if the levels are not properly secured. Tampering or substitution of sensors can provide useful outcomes when combined with tampered logic on the control level [39]. These attacks are many times possible due to lacking security protocols and procedures in these systems. Securing interconnections, cryptographic identities, configuration monitoring and signing are just a few things that always should be implemented on these systems but have not formally been used.

3 Simulation Environment

Tampering with railway signalling needs to be done in a controlled environment. We do not want real incidents to happen when testing. It is more secure, cost friendly and safer to simulate the system. When tampering with firmware on real devices there is a big risk that the system as a whole will not be recoverable. This section introduces a simulation framework that is utilised in the experimentation and attacking of the railway signalling system.

An overview of the framework can be seen in Fig. 3. Along with a management environment, each docker container corresponds to a real-World device with the addition of a set of tools to simulate the core root of trust, firmware and the measured boot process.

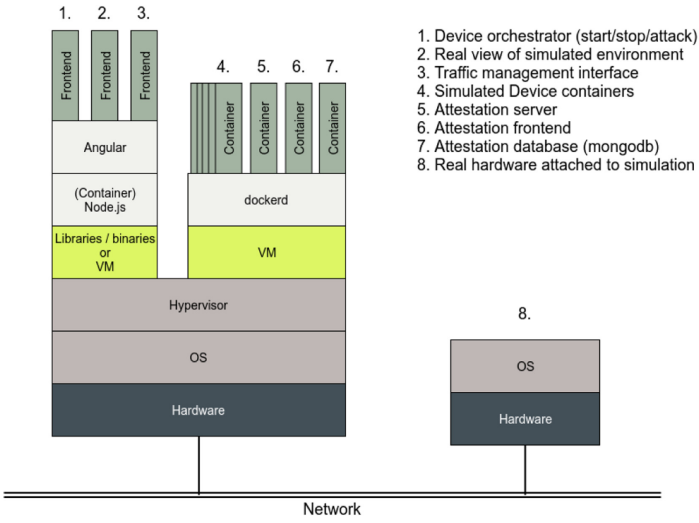

Fig. 3. Overview of simulation environment.

While providing a mechanism to develop simulated devices, it is also necessary for the simulation environment to provide two further services:

– Simulation Environment Management

– Attestation Services

The first refers to the functionality used to configure the devices and their communication (for rapid design, IP networking and MQTT message passing is used) and is utilised by the person in the role of setting up a simulation environment. The second contains the pre-built components specifically for the provisioning and remote attestation of the simulated (or real) trusted devices. These components are called by elements within the base containers automatically by the simulation environment. The remote attestation facilities would be presented to the users of the simulation as part of the cybersecurity forensics and failure detection processes.

4 Finnish Rail Traffic Management System

The Finnish rail traffic management system consists of four individual systems that work together: the interlocking, automatic train protection (JKV[2]), track vacancy monitor and the remote control system and is explained in [14].

The system uses track vacancy as the main safety criteria. A interlocking device takes the vacancy inputs and restricts traffic according to a predefined logic application. Trains are only allowed to move on tracks that are not occupied while speed is controlled by the interlocking system, partly through signalling and partly via track parameters.

There are two different methods for monitoring the track vacancy. The newer method is able to count train and carriage axles: if a carriage is broken loose from the train, the count between two sequential counters is not the same. The interlocking system will protect the track section between the counters and not let any trains pass. The older vacancy monitoring equipment depends on forming a track circuit; it can only report if the track section is occupied or not.

The interlocking system takes the inputs from the vacancy control system with the purpose to protect the railway environment by granting and prohibiting access for trains on track sections. Granting access is done by reserving track sections from the remote control system. A protective logic is implemented on the interlocking system that protects the rail environment from dangerous reservations. As an example it is not possible to reserve an already occupied track. It is not possible to reserve a soon to be occupied section, if stopping an incoming train is not possible.

Track switching is controlled by the interlocking system, so when a reservation is made and granted the track also switches to move the train to the designated endpoint. When the train moves on the reserved sections past vacancy control, track sections can be freed manually or even automatically. Track switches always try to protect the movement in a fail-safe manner, for example, if a stop signal is passed the train is routed to a free section.

The interlocking system is in charge of controlling all signals. The track side signals have three different modes *proceed*, *proceed 35* and *stop*[3]. These are shown

[2] [Automattinen] Junan Kulunvalvonta.

[3] The official Finnish terms are Aja, Aja35 and Seis.

by green, yellow/green and red lights on the physical track-side signal posts. An occupied track section is protected by a stop aspect, that section by a proceed 35 (warning) and that section by a proceed aspect. Other combinations are possible based on track speeds, train type (express vs freight), braking distances etc.

The Finnish automatic train protection system (JKV) is a second or additional signalling system that is installed on the train. It communicates with track side equipment, in this case a balise, which sends information to the train about the signalling and track information. The JKV system can react to lack of driver input and stop the train.

Safety is the sole purpose of the interlocking system, it takes the vacancy input, grants access to a remote control request if a safe passage can be routed, controls track side signals and data sent to the JKV system. Integrity of the interlocking system and the implemented logic is of high importance. When implementing an interlocking system it is tested, simulated and verified before production use. After it is put into use the interlocking integrity **is not actively monitored**.

5 Attacking the Trusted Railway Simulation

Attacking the firmware layer before the protective logic application can produce issues as described in [17,32]. Hardware initialization is done during the boot stage and this maps interfaces cards that produce outputs and take inputs from track side equipment. These inputs/outputs are in many cases analog and therefore can be directly swapped on the interlocking device. A trusted system would record the configuration measurements as cryptographic hashes of these components and write these to the TPM for subsequent remote attestation before admitting that device to the system.

For example a point device that controls the turnout of a train in a specific point could have 2 inputs and 2 outputs. The inputs tell the interlocking device where the pointing device turnout is. This is a simple analog circuit, if the circuit is closed the turnout is active. Through the outputs the turnout could be controlled. Note, this is intentionally (and maybe grossly) simplified - real systems rely upon multiple points within the interlocking to ensure functional integrity [23]; however the point here is to demonstrate a detectable misconfiguration in a digitalised system.

Swapping the input and output could - in the worst case - derail a train or cause a collision [34]. The interlocking device would believe that the turnout is connected to the correct track according to the application logic. However the input and output are swapped on firmware level, which of the application have no control of.

Firmware changes require a reboot of the device to take effect. Simulating device reboots with changes gives an opportunity to verify Traffic Management procedures so that right mitigation can be done and failed integrity can be corrected.

The mechanism for firmware updates typically requires a reboot of the system - this means that the effects of tampering can be hidden to take effect later and

not directly after a reboot changes in the firmware which are measured during the measured boot are detectable and actionable once they have been reported to the attestation mechanisms. Indeed some cases of malware [26] have included secondary attacks embedded within these [13].

5.1 Attack Anatomy

An example scenario is seen in Fig. 4 which shows the user-interface to the railway management system. In the simulator this interface is provided by its own container and the track and signalling devices as well as the interlock(s) similarly by their own containers.

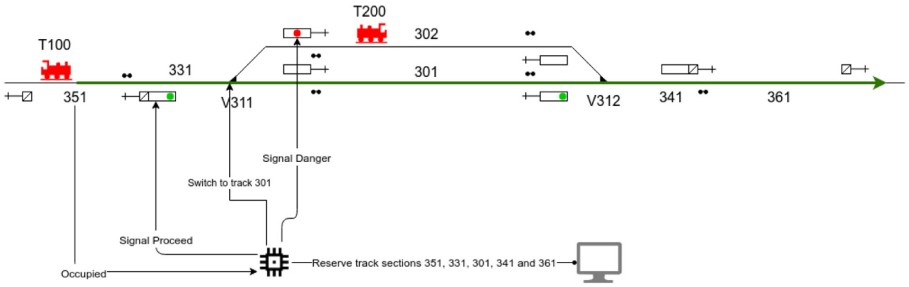

Fig. 4. Example of track section under attack

In this case we have a train (T100) approaching a switch (V311) set to enter track section (301). The signal protecting the switch (311) should display the appropriate aspect to allow the train to progress, typically a proceed or caution.

The attack upon this system - obviously relying upon digital components - is introduced by some vector as referenced earlier and need not be a direct attack against that device but also via some other attack medium. The attack here is designed to swap or misconfigure input and output signals. Figure 5 shows the input and output signals that will be mapped to each other in the attack. The delivery of the attack could further be made during normal maintenance operations [18]. The effect of swapping the input and output lines here would allow the train to cross the switch and enter the occupied section of track (302).

We introduce here three different attack triggers:

- Attacker triggers reboot after tampering.
- Passive reboot triggering waiting for maintenance reboot.
- Dynamic trigger activated a certain time after reboot.

As typical of firmware attacks, a reboot of the device is required to take effect. Either the reboot is directly triggered by the tampered firmware or we wait until a scheduled reboot is made.

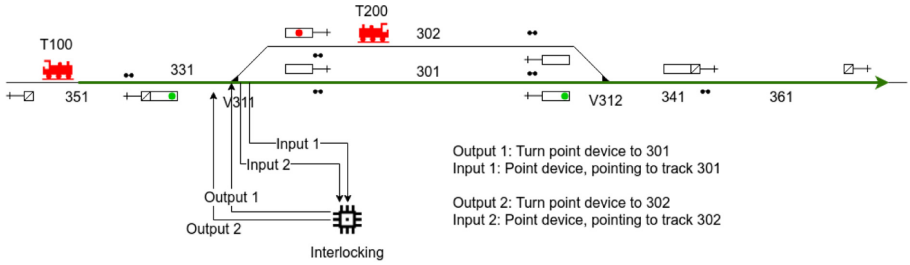

Fig. 5. Input-output signals that are flipped in the attack.

The first trigger would be activated directly after tampering the device firmware. Due to fail safe procedures, which should be monitoring for certain kinds of misconfiguration, every device connected to the interlocking device would go into fail safe mode when active signals from the device are removed. This assumes however that the particular misconfiguration is detectable and that the incorrect configurations have been properly characterised.

Although it depends upon the implementation of the interlocking device reboot, we can assume that most of the implementations would show a notice of the reboot which should trigger reattestation. As reboots are indicative of a potential reconfiguration detecting and accurately reporting on reboots and being able to provide forensics of why a reboot took place is a critical part of firmware integrity security.

From the traffic controllers perspective an interlocking device reboot would be identified, if the reboot activity is monitored. Otherwise the reboot would just show up/seem as a moment of downtime for the device. In our scenario it is possible to discover a short disruption of the interlocking device service. This can be discovered by every controlled track side device from that interlocking device to go into fail safe for the rebooting time. Depending on traffic management procedures mitigation would be done to resume normal operations.

Attacks could wait for a maintenance reboot from after which the configuration change would be in place. More advanced attacks could also do a dynamic mapping and implement a trigger for the swap to occur during normal operating hours. This makes reboot monitoring not a full covered mitigation to discover firmware tampering.

5.2 Measurement

Detecting firmware attacks depends upon the element being trustable and measuring relevant parameters. In our railway elements as we have built in the simulation we measure the firmware, the configuration of the firmware and then finally the configuration and static software (or hardware) elements that make up the configurable parts of the element. In this case we take cryptographic measurements of the input and output port configurations and these are loaded into the PCR registers on board the TPM.

In all of the above cases the TPM provides information about device reboots through monotonically increasing counters that indicate the amount of power cycles. These counters are reported as metadata during the acquisition of a device quote of its measured configuration. Further forensics are provided by the use of a 'safe' counter which provides information whether the TPM embedded in that device was cleanly shut down or not.

5.3 Mitigation

Controller software is mostly firmware that is upgraded by flashing after the device is rebooted. So firstly we will see a reboot of a device when attacked. If the devices are constantly monitored, there will be a short downtime off a device if it is rebooted. The best practice would be to attest the device as soon as possible after the reboot.

Attestation of a device can be made locally or remotely. In local attestation the device itself is responsible for checking the measurements made against locally stored known good values. This was the default mechanism used in the earlier TPM 1.2 devices which did not take into consideration larger, distributed systems. This local attestation mechanism is now largely deprecated and not used in TPM 2.0.

In the newer TPM 2.0 standard we either utilise remote attestation system and rely upon the attestation integrating with other parts of the system to report trust failures, or, we can *seal* certain information in the TPM's non-volatile RAM that can only be obtained if a correct configuration is measured. In reality both mechanisms need to be used though with more of a focus on the remote attestation occurs for reliability and relatively ease of use and configuration.

We have constructed a remote attestation and integrated that with the simulation environment for railway signalling described here. This has been adapted from an earlier system targeted towards telecommunication systems [22,29–31]. Each device reports its immutable identity (via its EK and AK keys) and its configuration to the remote attestation service. Figure 6 shows the attestation server and the basic device identity information.

Upon reboot of a device, the remote attestation server would communicate with the device and report that the device has failed the correct system measurements as shown in Fig. 7. This is achieved using the quoting mechanism described earlier. Here we can also see that the device still has a valid signature and that the quote returned by the TPM is both syntactically and semantically valid.

The remeasurement of a device may be triggered in a number of ways. Firstly the reboot might have been triggered by some other system and the attestation server notified to take measurements. The device itself may also trigger attestation by reporting explicitly its reboot. Depending upon the nature of the environment, communication and real-time properties either or both systems might be utilised.

In Fig. 7 we see that the device has failed its expected measurements test which is indicative of some change to the configuration. As we have mentioned

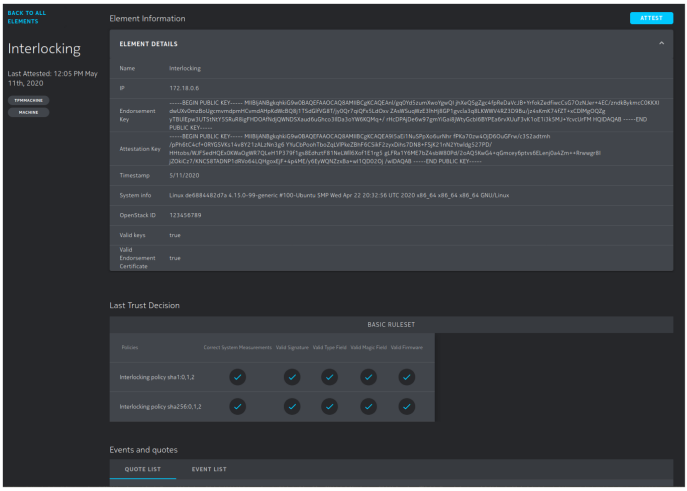

Fig. 6. Device identity

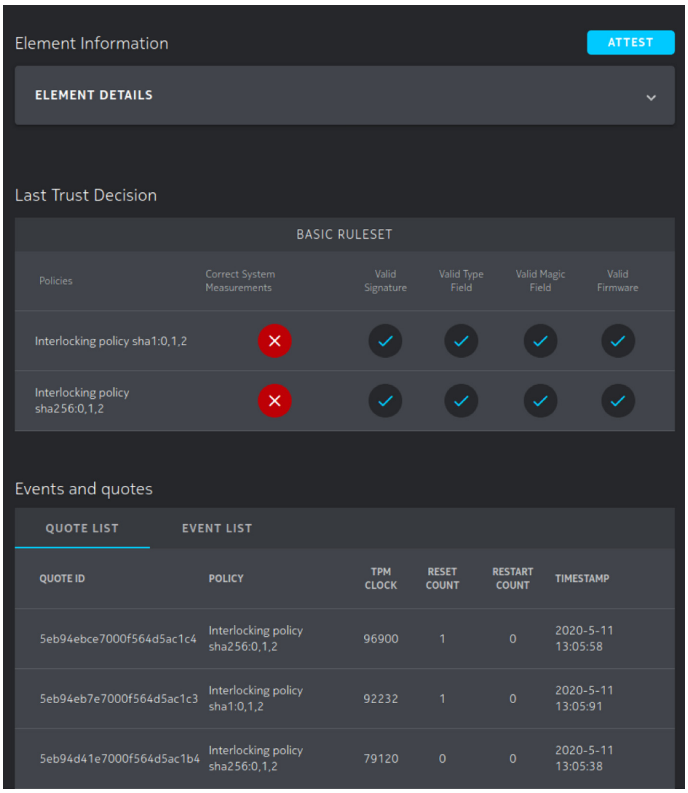

Fig. 7. Device showing configuration trust failure/error

other tests has passed successfully so we can be sure that the correct device is present.

Once we have identified a change in some measurements we proceed with the initial forensics where we examine the particular PCRs reported by the TPM as shown in Fig. 8. In this case we see that there has been a change in PCR1 which depending upon the semantics of the PCRs, at least on an x86 system, would indicate a change in the actual firmware itself.

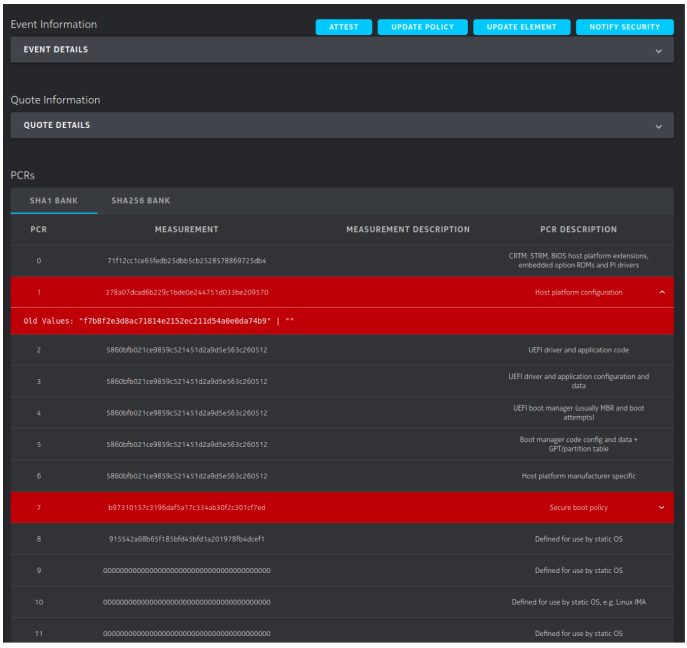

Fig. 8. TPM forensics using PCR listing

Further forensics can now start, for example examining the TPM boot logs exposed by the UEFI firmware on x86 machines. Similar mechanisms would have to be developed for other architectures, such as ARM, where this has not been standardised.

6 Conclusion

The work here has presented an attack on the integrity of a railway interlocking environment using the firmware and configuration of those systems in such a way - through simulation - that it causes changes in the measurable - in a trusted sense - aspects of the system. This allows us to safely understand the role of trusted computing in the firmware attack scenarios and develop techniques and plans to deal with such incidents.

This has been developed as the case study for a larger simulation environment for trusted systems. This provides us therefore with an educational environment where - in this case - railway signallers and control role operators can explore their reactions and develop mechanisms for the successful diagnosis and recovery.

Future work includes developing the scenarios shown in conjunction with trusted hardware and remote attestation specifically designed for safety-critical systems. It must be pointed out that dealing with trust failures is relatively novel and an unexplored area. Beyond preliminary investigations into root cause analysis there is currently no mechanism for trust failure forensics, analysis and recovery - especially in the case of safety critical systems [24].

A number of aspects have come out of this work so far which require more discussion. Firstly the response to a failure in a safety-critical system can not be a reboot/reinstall or taking that element out of usage [19]; instead a more sophisticated mechanism of managed degradation of functionality put in place. In the signalling case this might require manual intervention and imposition of procedures such as 'drive on sight.'

The real-time characteristics and data transmission restrictions also come into play. The size of a TPM quote is around 1000 bytes, but the time to generate, sign and check a quote might take a number of seconds [25]. So while the reporting of attestation information is well within specifications such as the EuroBalise data transmission specification, the real-time properties present significant problems to the obtaining of timely information. If a quote take 3 s to generate and check, then a high speed train at 300 kmh will travel approximately 250 m during that time. Even at slower speeds this might prove to be a significant obstacle especially when mitigation mechanisms need to put into place.

Finally handling of trust failures needs to be properly developed. While integration of cybersecurity procedures into normal railway signalling procedures is starting to happen, trust failure forensics and management is undeveloped. We have examined how root cause analysis can be integrated and automated with the remote attestation at least in the telecommunications server environment. Extending this into rail and other safety-critical domain use cases is ongoing.

Acknowledgements. This work has been partially funded by EU ECSEL Project SECREDAS (Grant Number: 783119).

References

1. Trusted computing platform alliance main specification. Trusted Computing Group (2002)
2. Trusted platform module library, part 1: architecture. Trusted Computing Group (2016). Version Number: Level 00 Revision 01.38
3. EN 50126–1. European Committee for Electronic Standardization (2017)
4. TCG TPM v2.0 Provisioning Guidance. Trusted Computing Group (2017). Version Number: 1.0
5. Hybrid ERTMS/ETCS Level 3. EEIG ERTMS Users Group (2018). Version Number: 1C

6. TCG PC Client Platform Firmware Profile. Trusted Computing Group (2019). Version Number: Level 00 Revision 1.04
7. TCG PC Client Platform Firmware Profile Specification. Trusted Computing Group, June 2019. Version Number: 1.04
8. Assante, M.J., Conway, T., Lee, R.M.: German steel mill cyber attack. Technical report, SANS Industrial Control Systems (2014)
9. Basnight, Z., Butts, J., Lopez Jr., J., Dube, T.: Firmware modification attacks on programmable logic controllers. Int. J. Crit. Infrastruct. Prot. **6**(2), 76–84 (2013)
10. Bastow, M.D.: Cyber security of the railway signalling & control system (2014)
11. Buurmans, K., Koopmans, M., Rijlaarsdam, R., Es, A.V., Vliet, M.V.: Feasibility study reference system ERTMS: final report, digitalisation of CCS (control command and signalling) and migration to ERTMS. Techreport, European Railway Agency (2018)
12. Falliere, N., Murchu, L.O., Chien, E.: W32.Stuxnet dossier. Technical report (2011). Volume: Version 1.4
13. Gotora, T.T., Zvarevashe, K., Nandan, P.: A survey on the security fight against ransomware and Trojans in Android. Int. J. Innov. Res. Comput. Commun. Eng. **2**(5), 4115–4123 (2014)
14. Kantamaa, V.M., Sorsimo, T.: Rautatieturvalaitteet. Otavan Kirjapaino Oy (2018)
15. Karjalainen, M., Kokkonen, T., Puuska, S.: Pedagogical aspects of cyber security exercises. In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 103–108. IEEE (2019)
16. Kokkonen, T., Hautamäki, J., Siltanen, J., Hämäläinen, T.: Model for sharing the information of cyber security situation awareness between organizations. In: 2016 23rd International Conference on Telecommunications (ICT), pp. 1–5. IEEE (2016)
17. Konstantinou, C., Maniatakos, M.: Impact of firmware modification attacks on power systems field devices. In: 2015 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 283–288. IEEE (2015)
18. Kour, R., Aljumaili, M., Karim, R., Tretten, P.: eMaintenance in railways: issues and challenges in cybersecurity. Proc. Inst. Mech. Eng. Part F J. Rail Rapid Transit **233**(10), 1012–1022 (2019)
19. Kour, R., Thaduri, A., Karim, R.: Railway defender kill chain to predict and detect cyber-attacks. J. Cyber Secur. Mobil. **9**(1), 47–90 (2020)
20. Kühner, H., Seider, D.: Security engineering für den schienenverkehr. In: Eisenbahn Ingenieur Kompendium, pp. 245–264 (2018)
21. Langner, R.: To kill a centrifuge. Technical report, The Langner Group (2013)
22. Hippelainen, L., Oliver, I., Lal, S.: Towards dependably detecting geolocation of cloud servers. In: 2nd International Workshop on Security of Internet of Everything, SECIOE 2017, Helsinki, Finland. IEEE, August 2017
23. Lim, H.W., Temple, W.G., Tran, B.A.N., Chen, B., Kalbarczyk, Z., Zhou, J.: Data integrity threats and countermeasures in railway spot transmission systems. ACM Trans. Cyber-Phys. Syst. **4**(1), 1–26 (2019)
24. Limonta, G., Oliver, I.: Analyzing trust failures in safety critical systems. In: Proceedings of the 29th European Safety and Reliability Conference (ESREL) (2019)
25. Limonta Marquez, G.: Using remote attestation of trust for computer forensics. Master's thesis, 10 December 2018
26. Mago, M., Madyira, F.F.: Ransomware software: case of wannacry. Eng. Sci. **3**(1), 258–261 (2018)
27. Matrosov, A., Rodionov, E., Bratus, S.: Rootkits and Bootkits. No Strach Press Inc., San Francisco (2019)

28. Oliver, I., et al.: Experiences in trusted cloud computing. In: Yan, Z., Molva, R., Mazurczyk, W., Kantola, R. (eds.) NSS 2017. LNCS, vol. 10394, pp. 19–30. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64701-2_2
29. Oliver, I., et al.: A testbed for trusted telecommunications systems in a safety critical environment. In: Gallina, B., Skavhaug, A., Schoitsch, E., Bitsch, F. (eds.) SAFECOMP 2018. LNCS, vol. 11094, pp. 87–98. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99229-7_9
30. Oliver, I., Lal, S., Ravidas, S., Taleb, T.: Assuring virtual network function image integrity and host sealing in telco cloud. In: IEEE ICC 2017, Paris, France, May 2017
31. Oliver, I., Ravidas, S., Hippeläinen, L., Lal, S.: Incorporating trust in NFVI: addressing the challenges. In: Proceedings of 20th Innovations in Clouds, Internet and Networks Conference, ICIN 2017, Paris, France (2017)
32. Pasquale, T., Rosaria, E., Pietro, M., Antonio, O., Ferroviario, A.S.: Hazard analysis of complex distributed railway systems. In: 2003 Proceedings of the 22nd International Symposium on Reliable Distributed Systems, pp. 283–292. IEEE (2003)
33. Proudler, G., Plaquin, D., Chen, L., Balacheff, B., Pearson, S.: Trusted Computing Platforms: TCPA Technology in Context. Prentice Hall, Upper Saddle River (2002)
34. Anthony Hidden, Q.C.: Investiation into the Clapham junction railway accident. UK Department of Transport, November 1989
35. Schindler, S., Schnor, B.: Honeypot architectures for IPv6 networks. Ph.D. thesis, Universität Potsdam, Mathematisch-Naturwissenschaftliche Fakultät (2016)
36. Schuett, C., Butts, J., Dunlap, S.: An evaluation of modification attacks on programmable logic controllers. Int. J. Crit. Infrastruct. Prot. **7**(1), 61–68 (2014)
37. Shila, D.M., Geng, P., Lovett, T.: I can detect you: using intrusion checkers to resist malicious firmware attacks. In: 2016 IEEE Symposium on Technologies for Homeland Security (HST), pp. 1–6. IEEE (2016)
38. Stumpp, K.: Draft of the security-by-design and of railway cyber security management system standards. Technical report, European Union Funding for Research and Innovation (2019)
39. Thaduri, A., Aljumaili, M., Kour, R., Karim, R.: Cybersecurity for eMaintenance in railway infrastructure: risks and consequences. Int. J. Syst. Assur. Eng. Manag. **10**(2), 149–159 (2019). https://doi.org/10.1007/s13198-019-00778-w