Defending Against Firmware Cyber Attacks on Safety-Critical Systems

Chris. W. Johnson DPhil, Mohammed Hashim Saleem,
Maria Evanglopoulou, Marco Cook, Rob Harkness
School of Computing Science, University of Glasgow, Glasgow, UK.

Tom Barker
EDF-Energy, Barnett Way, Barnwood, Gloucester, UK.

Keywords: Firmware, Hashing, Safety, Cyber Security.

Abstract

In the past, it was not possible to update the underlying software in many industrial control devices. Engineering teams had to 'rip and replace' obsolete components. However, the ability to make firmware updates has provided significant benefits to the companies who use Programmable Logic Controllers (PLCs), switches, gateways and bridges as well as an array of smart sensor/actuators. These updates include security patches when vulnerabilities are identified in existing devices; they can be distributed by physical media but are increasingly downloaded over Internet connections. These mechanisms pose a growing threat to the cyber security of safety-critical applications, which are illustrated by recent attacks on safety-related infrastructures across the Ukraine. Subsequent sections explain how malware can be distributed within firmware updates. Even when attackers cannot reverse engineer the code necessary to disguise their attack, they can undermine a device by forcing it into a constant upload cycle where the firmware installation never terminates. In this paper, we present means of mitigating the risks of firmware attack on safety-critical systems as part of wider initiatives to secure national critical infrastructures. Technical solutions, including firmware hashing, must be augmented by organizational measures to secure the supply chain within individual plants, across companies and throughout safety-related industries.

Introduction

Industrial Control System (ICS) play a crucial role in national infrastructures. In the past, these networks were isolated from the Internet and relied on specialist protocols, including Profibus and Modbus. However, they are increasingly accessible through MODBUS TCP/IP, PROFINET and Ethernet/IP gateways. This helps companies to monitor and control massively distributed production processes without duplicating network infrastructures (Ref. 1). Organizations create VPN links between their office based enterprise information systems and their operational networks using TCP/IP interfaces. This informs strategic decision-making and enables managers to continually monitor the productivity of underlying applications. Partly in consequence, IP-based protocols now account for 80% of all ICS installations (Ref. 2). This illustrates a dilemma that is at the heart of this paper:

- Previously the serial protocols used in ICS applications were inherently insecure; they were never intended to support encryption or strong authentication. However, they provided a degree of protection from mass-market malware because they were not widely understood;

- Today, TCP/IP variants of industrial protocols support encryption and authentication. However, more attackers understand their underlying implementation. ICS applications are vulnerable to denial of service/ransom attacks that were never intended to target safety-related processes.

A number of recent attacks have focused on the firmware that is used in ICS devices. Malicious agents can change the underlying code installed on PLCs, switches, smart sensor/actuators by first compromising the enterprise information systems and then using TCP/IP gateways to port their malware inside previously isolated industrial systems. PLCs are specialized microprocessor-based industrial devices that can be programmed in order to automate control of several machines and processes. Smart devices are lower level components that allow a degree of pre-programming/configuration in proprietary languages; including protection relays, temperature controllers and pressure transmitters.

This paper explains how firmware verification tools have been developed and deployed to protect UK critical infrastructures. This is complicated because different manufacturers use a host of different mechanisms to structure, encode and verify the updates that are distributed by physical media, including CD-ROMs, as well as different

Internet-based firmware servers. Later sections explain how technical innovations, including generalized firmware hashing tools, must be augmented by organizational measures to secure the supply chain within individual plants, across companies and throughout safety-related industries.

Firmware Attack Vectors for Safety-Critical Systems

The firmware that supports an embedded device in an industrial control system plays a similar role to that of an operating system in more conventional applications. It is stored in ROM, EEPROM or Flash memory. Without firmware, these devices are little more than 'bricks'. With firmware, they can be updated to improve reliability, to address bug reports and increasingly to provide security patches. Firmware updates are usually performed via interfaces provided by the device manufacturer, most often using Ethernet or RS-232 connections.

In safety-critical systems, these update mechanisms pose new challenges. Traditionally, the firmware of many ICS components was never routinely updated because of the additional costs associated with the verification and validation required to demonstrate the modifications did not compromise safety requirements. In such cases, critical processes remain vulnerable to previously published security exploits. Hence there is a growing conflict between security requirements to install firmware updates and the safety-related costs of insuring that any updates do not undermine integrity requirements in industrial processes.

Some vendors now distribute SCADA firmware updates from their websites. This exacerbates security concerns. Potential attackers can download a sample of the firmware for embedded devices, enabling them to experiment using second-hand devices that can be bought via Internet based marketplaces. Penetration testing of embedded firmware can potentially discover backdoors. The distribution of firmware patches over the Internet now also means that legitimate users have to ensure that any firmware is not downloaded from a malicious site. Watering hole attacks rely on high-value targets being drawn to a compromised web site. Man-in-the-middle attacks insert illegitimate firmware on the route from a vendor's web site to the intended recipient; for example by compromising the addressing mechanisms used to identify the manufacturer's server.

The following list summarizes several different types of firmware attacks (Ref. 3); any single attack may borrow concepts form more than one category:

1. *System-Safety Patch Vulnerabilities.* As mentioned, software system safety requirements mean that many ICS components are unpatched. Hence previously published vulnerabilities can be used and re-used on safety-critical systems.

2. *Zero-day Exploits.* There is a growing market place in zero-day vulnerabilities – or attack methods that have not yet been patched. The closer that ICS components move to conventional architectures then the more likely they are to be affected by these attack methods. There is also a growing competence amongst state sponsored actors who are focusing on national critical infrastructures;

3. *Reverse Engineering.* These attacks build on aspects of the Stuxnet/Olympic Games attacks (Ref. 4). Penetration testing tools can be used to understand elements of the architectures used by manufacturers in their firmware updates. If digital certificates can be forged or undermined then it is possible to inject malicious code in an update that would otherwise leave the rest of the functionality unaffected until the attack is launched. Schuett et al. managed to modify functions identified during the disassembly stage of reverse engineering and injected them into the PLC firmware, causing denial of service to the operator under certain conditions (Ref. 5);

4. *Code mirroring.* Malicious code may also ensure that any attempt to query the firmware will make it seem that a valid installation is being used.

5. *Reload Death Spiral.* A simpler and more direct form of attack can be triggered in some devices; if a validation step fails late in the firmware installation then the device may halt while a new version of the firmware is downloaded. If that code is also corrupted then the device will repeat the download indefinitely.

6. *Firmware 'Bricking'*. A more general version of the reload-death spiral is an attack, which successfully installs malformed firmware in a manner that compromises the device. This is a simplified version of the reverse engineering attacks. It is not necessary to understand how to hide malicious code in a valid installation, only to get the device to load the compromised code that need not implement any valid computation (Ref. 4).

It is important to stress that tools are being developed to automate aspects of these attacks. Costin et al. (Ref. 6) describe a system that conducts large-scale static analysis looking for vulnerabilities and correlations across families of firmware. Zhu et al. (Ref. 7) propose algorithms to help determine the image base of firmware; for instance by identifying literal pools within a firmware file. This can then be used to determine a candidate base address for the firmware image. Their work focuses on devices that exploit ARM processors. However, many of the underlying concepts have more general application.

It is important to stress that, at present, it is not possible to directly validate the firmware that is running on many ICS components. Each manufacturer alters the binary representation of executable and resource files that are transferred between installation environments and the underlying ROM/EEPROM/Flash as the firmware update proceeds. The Glasgow group is actively engaged in developing forensic techniques that conduct this form of analysis on particular components. However, in general this requires a low-level understanding of specific devices including firmware architecture, memory encoding etc. Painstaking reverse engineering can gradually piece this information together, but it must be repeated for different manufacturers and devices. As a pragmatic, interim step, we want to update firmware and application software from a trusted source with a high-degree of confidence even if we cannot *prove* that the software running on a device has not been modified.

In order to defend safety-related ICS components against firmware attacks, it is first necessary to understand the underlying vulnerabilities. We, therefore, focused our initial work on two very different devices. The first is a widely used PLC; this was chosen because PLCs are the computational workhorses of safety-related processes. The second device was an IP-based security camera; this was chosen because of the obvious consequences from undermining the physical security of safety-related ICS applications[1]. We started to determine whether minor modifications to the firmware of an embedded device can by-pass firmware integrity checks.

The first step in attacking the PLC was to determine the version of firmware being run on the device, assuming that an attacker only had remote access to the ICS component. We were able to use public information sources to determine how to do this by inspecting the binary firmware image for the PLC. The same techniques also worked on the camera and PLCs from other manufacturers. Once the firmware version had been obtained for the devices, we downloaded valid firmware already running on both devices from the manufacturers' websites. We were then able to edit the binary files. Our analysis revealed a surprising range of responses on the devices when we made simple changes to the version identifiers. We were able to map out the range of legal and illegal version numbers for the camera. Moving outside the permitted range elicited an error message from the firmware update interface, numbers inside the approved range but still invalid for that version were not reported and the firmware was passed directly to the camera for installation. With the PLC, our edits caused the device-upload interface to hang. We were forced to conduct a manual reset resembling the firmware spiral mentioned as item 5 in the previous list of attack methods. The key point here is that even relatively simple changes provoke inconsistent responses on different ICS components, some of which could be exploited within a malicious attack unless defenders take measures to protect their safety-related systems.

The Glasgow Firmware Defender

A growing number of papers describe firmware attack vectors; fewer provide potential solutions. McMinn uses MD5 hashing and bit wise comparisons on the firmware for a PLC (Ref. 8). Hashing algorithms calculate characteristic values from a file. Changes to a file can be identified because the hash value will change if it is

[1] The identity of both devices is omitted given the sensitivity of the topic, further details are available from the first author of this paper.

recalculated. For example, if the number of characters in a file were used as the hash sum, any insertion or deletion would be identified because that value would change. However, if an attacker deleted a valid character and then inserted another one this might not be detected using this simple algorithm. This is an example of hash collision where the values of the hash function cannot detect the change. Unfortunately, MD5 suffers from a range of vulnerabilities including known problems with hash collision. More formally, a suitable hash function should ensure:

- *Pre-image Resistance* - Given an unknown message of arbitrary length x and its corresponding hash value y (H(x) = y), then it should be computationally in- feasible for any other message n for it's hash value m (H(n) = m) to match y i.e. $m \neq y$.

- *Second Pre-image Resistance* - Given a message x that produces a hash value y (H(x) = y) then it should be computationally infeasible for any different message n to produce the same hash value as x ($H(x) \neq H(n)$).

- *Collision Resistance* - Given x; y in M where M is the set of all possible messages, then $H(x) \neq H(y)$.

Others have focused on a range of alternate methods – for instance, modeling hysteresis effects on low-level transmissions from firmware servers. Xiao et al. (Ref. 9) provide a more general architecture intended to secure ROM, RAM, boot-loader and processes within ICS devices. This again illustrates the tensions that arise between safety and security when proposing detailed technical solutions to the threats against critical infrastructures. Xiao et al rely on encryption across the software stack. This would have enormous safety re-certification costs for legacy systems that support European and North American industry. It would incur enormous overheads in terms of the verification and validation processes that would be required to determine whether the upgrades had any impact on functional safety. As a specific example, many encryption techniques deliberately insert non-deterministic timing delays to disguise the algorithms they use. These delays undermine the timing analyses that are required to satisfy a host of safety related requirements.

In the long term it seems likely vendors and systems integrators will adopt techniques similar to those proposed by Xiao et al, as legacy systems are gradually replaced by more secure counterparts. In the meantime, there are significant threats to our existing infrastructures. We, therefore, began to develop tools that could help secure the firmware that is being deployed in European infrastructures. Our tool's sole purpose is to verify whether a given suspect file is genuine. It is possible to conduct byte-by-byte comparisons. The size of firmware images creates significant computational overheads for such an analysis, although hardware support can be provided. We, therefore, also built on the techniques pioneered by McMinn (Ref. 8). Recall that both the MD-5 and SHA-1 algorithms have been found to have high collision probabilities (Ref. 10). Attackers can exploit these collisions by manipulating a firmware image to produce the same MD-5 or SHA-1 hash value as the baseline. Rather than rely on a single algorithm with known vulnerabilities our toolset exploits several hash functions, eventually triggering byte-by-byte comparisons if doubts persist.

It is important to stress that many industrial processes rely on thousands of low-level devices, each running bespoke firmware. The techniques described above create significant potential overheads if the Firmware Defender raises a large number of false alarms. This might occur if, for example, the tool was incorrectly configured using incorrect hash values for baseline versions of legal ICS firmware. Conversely, an attacker could undermine our system if they could register an illegal hash value that characterized their compromised version of the firmware. For this reason, we had to develop extensive verification and access control measures to protect the integrity of the tool itself.

Before the defender can be used, it is first necessary to survey all the valid versions of manufacturer's software that might be deployed within a particular organization. Although most suppliers compute their own hash values, they use different techniques both to encode the files needed to install the firmware and also to calculate characteristic values. We, therefore, developed common algorithms that can be applied in a consistent way across the archives being transferred between many different suppliers and their safety-related customers. We can then re-compute the hash functions across different forms of firmware for many different devices. However, in order for this approach to be successful it is necessary to obtain a verified baseline copy. Safety-critical organizations must deploy physical and digital mechanisms to increase confidence that initial hash values are computed from reliable sources for the manufacturers' firmware. Again in the medium term this might be avoided if the ICS industry could agree on standard techniques for the generation of hash values across multiple suppliers.

Working with a number of industrial sponsors we found that end-users often operated many different firmware versions on a single family of devices deployed across a single production facility. Deployment of the toolhelped these companies to audit the firmware being used across a range of critical, legacy processes.

The safety-critical nature of this project meant that acceptance testing of the tool had to involve authorized participants from critical infrastructure companies. During a pre-deployment study, we determined whether plant engineers could use the tool to identify modified firmware during a simulated update on ICS components. One issue that arose during this initial evaluation was the additional time that might be required to train a sufficient number of staff to ensure that all updates were verified on a 24/7 basis. We initially focused on firmware – partly because this had been a target in the recent attacks on critical infrastructures in the Ukraine[2]. The engineers involved in our evaluation argued that the same level of protection should be extended to other levels of the software stack. In consequence, the application of our tools was extended in line with the GAMP 5 (Good Automated Manufacturing Practices) (Ref. 11) software categories illustrated in Table 1. For example, our tool was extended to perform periodic checks on the configuration data associated with the smart sensor/actuators, mentioned in the opening paragraphs.

Category	***Extending the Scope of Software Authentication***
1. Infrastructure software	Verify infrastructure software tools such as ladder logic interpreters and commercially available software such as anti-virus applications
2. Firmware	Verify firmware in devices that present some form of risk to the system.
3. Non-configured Products	Verify non-configurable soft- ware such as plugin software for certain applications that cannot be altered.
4. Configured Products	Verify configurable software such as software programs that are uploaded onto PLCs, compute hash values on configuration data.
5. Custom Applications	Verify bespoke software, which is developed in-house or from a third party. Examples of such software include macros, scripts, tools and applications used to auto- mate laborious procedures.

Table 1 – Using Good Automated Manufacturing Practices in the Validation of the Firmware Defender

Conclusions and Future Work

This paper describes work to defend European critical infrastructures against a growing range of cyber attacks. Many industrial control systems rely on devices that are hard to protect; they were never designed to be resilient against the threats that are emerging. New ranges of PLCs and of sensors provide support for on-board encryption and authentication. Unfortunately, it will be many years before these devices are used throughout the legacy systems that support national critical infrastructures [12]. Such improvements also depend upon the ability of engineers to identify ICS components that offer appropriate levels of security – not only for the threats that we see today but also for new forms of attack that will emerge over the lifetime of any devices that are procured for future systems. We are developing a systematic set of criteria that can be used to compare the levels of security that are provided by ICS components. The intention is that these questions will augment an existing set of characteristics that are considered by a subset of European infrastructure providers when they implement the IEC61508 standard. In other words, we are integrating questions about cyber security that are intended to guide the application of existing safety-related acquisition criteria.

[2] The recent attacks on Ukrainian infrastructures are presented in a companion paper for ISSC 2017 – C.W. Johnson, M. Evangelopoulou and T. Pavlova, Applying Lessons from Cyber Attacks on Ukrainian Infrastructures to Secure the Gateways onto the Industrial Internet of Things.

Previous sections have identified longer-term objectives for research in this area. Many of these relate to the tensions between safety and security. In particular, we have argued that there is an urgent need for techniques that can be used to speed up the validation and verification of firmware patches in safety-related applications. Without this many ICS components will retain known security vulnerabilities because we cannot demonstrate that these updates will preserve safety requirements. We have also identified specific concerns over conventional security techniques, including the insertion of non-deterministic delays to frustrate timing attacks on cryptographic algorithms. Further work should consider means of bounding these delays across complex systems so that the cumulative effect of these mechanisms does not undermine system safety.

In the meantime, we have the pragmatic aim of defending existing legacy infrastructures. In order to better understand the threat, we have described the effects of uploading compromised malware onto a PLC and also onto a security camera used to preserve the physical security of ICS applications. Later sections have described the design and high-level implementation of a tool that can detect firmware modifications. Ideally, this should be capable of reverse engineering the code running on a suspect device. However, this raises many of the intellectual property barriers and manufacturer differences that complicate ICS forensics. In contrast, our tool integrates a broad range of hashing tools that can be used across the software supply chain to authenticate any code that is transferred onto particular components. The tool has been developed in close cooperation with field engineers. We claim that although firmware verification is far from the perfect defence, it helps to increase confidence in the integrity of safety-critical infrastructures at a time when a growing array of state-sponsored cyber-threats pose an existential threat to the systems we all depend upon.

References

1. C.W. Johnson. Securing the Participation of Safety-Critical SCADA Systems in the Industrial Internet of Things. In C. Sandon, R. Piggin, M. St. John Green, Paul Casely and Chris Johnson (eds.), Proceedings of the 11th International Conference on System Safety and Cyber Security, The IET, Savoy Place, London, 11-13th October 2016.

2. L. Zhang. An Implementation of SCADA Network Security Testbed. arXiv preprint arXiv:1701.05323, 2017.

3. G. Loukas. Cyber-Physical Attacks: A Growing Invisible Threat. Chapter 5, pp165-166. Butterworth Heinemann/Elsevier, Waltham, MA, USA, 2015.

4. Z. H. Basnight. Firmware Counterfeiting And Modification Attacks on Programmable Logic Controllers. Master's thesis, Graduate School of Engineering and Management Air Force Institute of Technology Air University, Wright Patterson Air Force Base, Ohio, USA, 2013. Available on: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA583401, last accessed May 2017..

5. C. Schuett, J. Butts, and S. Dunlap. An Evaluation of Modification Attacks on Programmable Logic Controllers. International Journal of Critical Infrastructure Protection, 7(1):61-68, 2014.

6. A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, and S. Antipolis. A Large-Scale Analysis Of The Security Of Embedded Firmwares. In Proceedings of the 23rd USENIX Security Symposium. August 20–22, 2014, San Diego, CA, USA/USENIX Security, pp 95-110, 2014.

7. R. Zhu, Y.-A. Tan, Q. Zhang, Y. Li, and J. Zheng. Determining Image Base Of Firmware For ARM Devices by Matching Literal Pools. Digital Investigation, 16:19-28, 2016. https://doi.org/10.1016/j.diin.2016.01.002

8. L. R. McMinn. External Verification of SCADA System Embedded Controller Firmware. Master's thesis, Graduate School of Engineering and Management Air Force Institute of Technology Air University, Wright Patterson Air Force Base, Ohio, USA, 2012. Available on: https://www.hsdl.org/?view&did=756306, last accessed May 2017.

9. M. Xiao, Y.-Q. Li, S.-h. Chen, and J.-S. Su. Security Enhancement on Firmware for the Internet of Things. DEStech Transactions on Computer Science and Engineering, (WCNE), 2016. Available on: 10.12783/dtcse/wcne2016/5146

10. P. Gauravaram and L. R. Knudsen. Cryptographic Hash Functions, Springer Verlag, Heidelberg, Germany, pp 59-79, 2010.

11. J. DeSpautz, K.S. Kovacs, G. Werling, GAMP Standards For Validation Of Automated Systems. Pharmaceutical Processing, 11 March 2008. Available on: http://www.pharmpro.com/article/2008/03/gamp-standards-validation-automated-systems, last accessed May 2017.

12. C. W. Johnson, R. Harkness, and M. Evangelopoulou. Forensic Attacks Analysis and the Cyber Security of Safety-Critical Industrial Control Systems. In Proceedings of the 34th International System Safety Conference, Orlando, USA 8-12 August 2016, International System Safety Society, Unionville, Virginia, USA, 2016.

Biography

Chris Johnson, DPhil, School of Computing Science, University of Glasgow, Glasgow, Scotland, G12 8RZ, Scotland, U.K., telephone – +44 (141) 3306053, facsimile – +44 (141) 3304913, email – Johnson@dcs.gla.ac.uk.

Chris Johnson is Professor and Head of Computing Science at the University of Glasgow in Scotland. He leads a research group devoted to improving the cyber-security of safety-critical systems. He has developed forensic guidance on behalf of the UK civil nuclear industry and helped develop European policy for the cyber-security of aviation – including ground based and airborne systems.

Maria Evangelopoulou, School of Computing Science, University of Glasgow, Glasgow, G12 8RZ, Scotland, U.K., telephone – +44 (141) 330-6053, facsimile – +44 (141) 330-4913, e-mail – Maria.Evangelopoulou@glasgow.ac.uk.

Maria Evangelopoulou is a Research Assistant working on a joint FAA/US Navy project in Glasgow University, looking at safety and security analysis of network data. She attained her MSc in Intelligence and Security Informatics from the University of Abertay and a BSc Technology Management from University of Macedonia in Greece. Maria's current research is concerned with the investigation of Cyber Situation Awareness Methods and Techniques in Cloud Networks and other kind of systems.

Mohammed Hashim Saleem, School of Computing Science, University of Glasgow, Glasgow, G12 8RZ, Scotland, U.K., telephone – +44 (141) 330-6053, facsimile – +44 (141) 330-4913, e-mail – mhsaleem100@gmail.com

Hashim Saleem attained an MSci in Computing Science from the University of Glasgow; driving the implementation of the security toolkit for firmware in embedded devices used within ICS.