Firmware Management Best Practices Guide for Energy Infrastructure Embedded Control Devices

< Ken Masica, Systems Engineer, MSEE >

Written July 2018, Updated August 2020

Sponsored by the Department of Defense ESTCP Installation Energy and Water Program Area, Cybersecure Connectivity for Energy System Components and Military Installation Energy Infrastructure Topic #2, Project EW-201608

Lawrence Livermore National Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 31/08/2020	**2. REPORT TYPE** ESTCP Best Practices Guide	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE Firmware Management Best Practices Guide for Energy Infrastructure Embedded Control Devices	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Ken Masica, Daniel Quinlan	**5d. PROJECT NUMBER** EW-201608
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lawrence Livermore National Laboratory, 7000 East Ave L-495, Livermore, CA 94550	**8. PERFORMING ORGANIZATION REPORT NUMBER** EW-201608
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Environmental Security Technology Certification Program, 4800 Mark Center Drive, Suite 16F16, Alexandria, VA 22350-3605	**10. SPONSOR/MONITOR'S ACRONYM(S)** ESTCP
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) EW-201608

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper identifies a set of best practices regarding firmware management and security for embedded control devices that are critical components of an energy infrastructure system. Such systems are considered an aspect of Utility Monitoring and Control Systems (UMCS) in the DOD community, but the practices presented in this report are equally applicable to the civilian sector as well as the national critical infrastructure systems identified by the Department of Homeland Security that utilize embedded devices for control and monitoring purposes. The intended audience are vendors of embedded devices and firmware as well as the military bases that receive and apply updates of the firmware to their infrastructure. The importance of secure firmware management practices by both vendors and customers as well as the definition and operation of firmware within an embedded control device are provided initially for context and background. Secure development and distribution methods of vendor firmware is then addressed. The report next provides best practices for customer storage and organization of firmware in addition to the performance of security checks for verification purposes. Application of updates to embedded devices and retention practices of firmware are discussed as well. Finally, a section on how DOD detects and responds to malicious firmware is included as an example of how best practices can be integrated with critical infrastructure monitoring. References and an Appendix describing an example best practices firmware update process of for an electrical power distribution relay conclude the paper.

15. SUBJECT TERMS
Firmware Management, Energy Infrastructure, Embedded Control Devices, Cyber-Security Integrity, Electric Grid Facilities Management

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Daniel Quinlan
UNCLASS	UNCLASS	UNCLASS	UNCLASS	21	**19b. TELEPHONE NUMBER** *(Include area code)* 925-423-2668

Abstract

This paper identifies a set of best practices regarding firmware management and security for embedded control devices that are critical components of an energy infrastructure system. Such systems are considered an aspect of Utility Monitoring and Control Systems (UMCS) in the DOD community, but the practices presented in this report are equally applicable to the civilian sector as well as the national critical infrastructure systems identified by the Department of Homeland Security that utilize embedded devices for control and monitoring purposes. The intended audience are vendors of embedded devices and firmware as well as the military bases that receive and apply updates of the firmware to their infrastructure. The importance of secure firmware management practices by both vendors and customers as well as the definition and operation of firmware within an embedded control device are provided initially for context and background. Secure development and distribution methods of vendor firmware is then addressed. The report next provides best practices for customer storage and organization of firmware in addition to the performance of security checks for verification purposes. Application of updates to embedded devices and retention practices of firmware are discussed as well. Finally, a section on how DOD detects and responds to malicious firmware is included as an example of how best practices can be integrated with critical infrastructure monitoring. References and an Appendix describing an example best practices firmware update process of for an electrical power distribution relay conclude the paper.

Author

Ken Masica, MSEE, is an Electrical Engineer in the Systems Engineering Group at Lawrence Livermore National Laboratory (LLNL) specializing in the areas of secure network and communication system design, control system security analysis and hardening, defensible system architecture development, and vulnerability & risk assessment of U.S. critical infrastructure. He has thirty years of experience as a systems engineer, project manager, and principal investigator analyzing, implementing, and managing solutions for a wide range of systems and technologies including secure remote access and monitoring systems, smart metering infrastructures, building automation systems, SCADA control systems, counter terrorism response systems, secure wireless applications, high assurance communications, and national & international wide-area networks to support numerous enterprise, industrial, scientific, and defense related missions. He has led cyber security vulnerability assessments and system hardening of electrical power systems, nuclear reactors, dams and hydro-electric facilities, air traffic control systems, information systems, government facilities, and U.S. nuclear weapon material access control and protection systems. Ken also supported numerous programs that upgraded nuclear material security systems in the Russian Federation and served as a U.S.-Russian Nuclear Reduction Treaty Inspector. He actively publishes in trade magazines and journals in multiple industry sectors and presently serves on International Society of Automation (ISA) committees developing security standards for industrial control systems (ICS) and supervisory control systems (SCADA). Ken can be reached at Masica@LLNL.gov

Table of Contents

I. Introduction

This paper identifies a set of best practices regarding firmware management and security for Department of Defense (DOD) energy infrastructure systems and the embedded control devices that are critical components of those systems. The intended audience are the vendors that develop and distribute the firmware as well as the military bases that receive, verify, apply, and retain firmware updates. Elements of the best practice recommendations also include use of the firmware cyber assessment tools developed by Lawrence Livermore National Laboratory (LLNL) and funded by the DOD Environmental Security Technology Certification Program (ESTCP). Although this paper addresses the DOD environment, the recommended practices are also applicable to energy infrastructure systems operating in the civilian sector such as those at electric utilities, industrial plants, and commercial facilities.

Firmware management and security best practices should be included in the existing Cyber Security Management System (CSMS) for the base as well as incorporated into the control system specific Security Plans that should exist for the energy infrastructure systems. There are numerous systems that can comprise an energy infrastructure at a military base, and they are generally considered to be part of a more general-purpose *Utility Monitoring and Control System* (UMCS). Energy related components of a base UMCS, for example, can include Power Generation Systems (PGS), Power Distribution Systems (PDS), Building Automation Systems (BAS) and Advanced Metering Infrastructures (AMI). Some bases may also have traditional generator-based auxiliary power systems for emergency reserve during utility outages for critical loads while others may be rolling out micro-grids and implementing islanding practices for more self-sufficient operation during power outages and cyber-attacks. This report, therefore, will provide general guidance that is intended to be applicable to all types of energy infrastructure firmware while recognizing the wide variety of embedded devices that exist as well as the wide variety of security practices employed by vendors during firmware development and distribution to customers.

This paper begins by addressing the need for firmware security and management best practices to reduce risk associated with potential cyber-attack of embedded devices. A foundational section then provides a definition of firmware and how it operates within the context of an embedded control device. Following sections then address best practices for development and distribution of firmware by vendors. Vendors of commercial firmware vary widely in the number and type of security features they use during development as well as how they provide the firmware to their customers. Some vendors, for example, make their firmware updates available openly on the Internet for download by any user while others require registered customer product purchases or web accounts that are vetted for users to access their firmware. Therefore, a representative set of best practices by vendors will be included in this report and thus provide a list of security features to consider when choosing a vendor or discussing the topic of firmware security with current or future vendors. Additional sections of the paper then identify best practices for bases regarding the storage, verification, installation, and retention of firmware updates received by vendors. An overview of how DOD detects and responds to malicious firmware is included as an example of how best practices can be integrated with critical infrastructure monitoring. The final sections include a summary conclusion, technical references, and an appendix that provides a best practices example of how an electrical relay embedded device is updated with new firmware.

II. Supply Chain Attacks and the Need for Firmware Best Practices

Adopting and incorporating best practices for management of embedded device firmware is important given the potential devastation that a firmware supply chain cyber-attack could entail. Because firmware is distributed as machine code, it is a form of a binary program that is in the format of a loadable and executable file on the microprocessor that is embedded in the energy infrastructure control device. Because it is assembled into the instruction set of the embedded microprocessor in the form of machine code, detecting a potentially devasting firmware update tainted with a targeted malware payload is difficult and inexact. Stuxnet is perhaps the most notorious and publicly known supply chain malware attack to date. According to open source studies, one of the Stuxnet malware payloads appears to have been an example of a man-in-the-middle form of attack uniquely applied to the firmware running on the Programmable Logic Controller (PLC) embedded control device for the Uranium enrichment cascade protection system at the target plant.

More generally, firmware security best practices are important for military base energy infrastructure systems for continuity of operation and assurance of mission execution. For electrical power distribution systems, applying best practices for managing firmware running on critical equipment such as protective relays are essential to include in a comprehensive base CSMS given their vital function of circuit breaker operation and the consequences of potential unauthorized/unintended open breaker activity and disruption of power availability. Likewise, Direct Digital Control (DDC) building automation systems now include application specific embedded microprocessor-based controllers which execute the logic that monitors and controls HVAC (Heating, Ventilation, Air Conditioning) equipment such as boilers, chillers, pumps, and air handling units to control temperature, humidity, and pressure within defined spaces inside both general and specialized facilities which may have critical requirements for essential personnel activity, material and weapons storage, and warfighting equipment/asset support, operations, & maintenance. For specialized energy infrastructure systems such as these, the embedded control devices that are elements of the power distribution and support facility operation provide critical system functionality and therefore the practice of firmware security management at military bases becomes an essential element of mission assurance.

III. Embedded Device Operation and Firmware Overview

This section of the paper will provide an overview of embedded device operation and describe the types and role of firmware found on such devices (for example a controller for a BAS found in a mechanical room, a protective relay found in an electrical substation, or a PLC found in the burner management system of a boiler in a central utility plant). It will therefore provide a conceptual foundation for the content presented in the remainder of this paper. For simplicity, the term *controller* will be used in this paper to represent the vast array of devices that have an embedded microprocessor that executes programmable logic for control and monitoring of energy and other critical infrastructure systems.

In general, a controller will consist of hardware, firmware, and software (i.e. control logic).

Controller *hardware* will consist of one or more circuit boards that contain the processor, memory, I/O channels, communication ports, and an assortment of other discrete and integrated circuit components that are generally in a sealed enclosure with a fixed configuration of components.

Controller *firmware* for embedded devices provides the low-level management of the device specific hardware. Because of the wide variety and complexity of controllers, it can either provide a complete and standardized Real-Time Operating System (RTOS) environment for processor and process management of more complex microprocessor-based devices that performing multiple functions and services or for less complex devices it can act as a more simplified but complete management environment that performs all control, monitoring and data manipulation functions written specifically for the device. Note that in this paper, we are considering controllers with that are powerful and complex enough to warrant an RTOS.

The two primary types of firmware found on controllers are:

1) *Boot Loader Firmware*: A small piece of machine code that executes on the embedded microprocessor after the POST (Power-On-Self-Test) of the hardware and loads the main image RTOS firmware.
2) *Main Image Firmware*: The larger machine code image that is loaded into the controller RAM memory by the boot loader are the device drivers and RTOS that perform the primary management functions of the controller including the low-level hardware input/output (I/O) processing as well as the higher-level system functions such as embedded processor management, task scheduling, file system management, etc.

Controller *software* are control programs (often referred to as control logic) that are scheduled and loaded by the RTOS firmware and executed on the embedded microprocessor hardware to perform a specific set of monitoring and control actions such as determining when a protective relay should trip a circuit breaker or when a building automation controller should modulate control valve positions or vary fan motor speed to ensure critical temperature, pressure, and ventilation space requirements. The RTOS firmware will schedule and the processor will execute the control logic that is developed in an external IDE (Integrated Development Environment) and downloaded to the controller hardware for storage and execution. Controller software is the equivalent of a high- level language in the computer world but in the control systems world the type of logic programming that is developed takes the form of interconnected micro-blocks, ladder diagrams, structured text, and instruction lists. (These are the four types defined by the IEC-61131 Open PLC controller standard.) Although once downloaded to the controller, the control logic is stored in non-volatile memory but is not generally considered firmware. During firmware updates, for example, control code will not change but the main and/or boot image will be upgraded.

A typical start-up and execution of a controller would therefore consist of the following steps:

1) The Controller hardware performs POST (Power-On-Self-Test) upon start-up.
2) The Boot Loader firmware is loaded onto the embedded processor for execution.
3) The Boot Loader loads the RTOS onto the embedded processor to manage the controller.
4) The RTOS schedules, loads, and executes the control logic on the embedded microprocessor to perform energy infrastructure monitoring and process control functions.

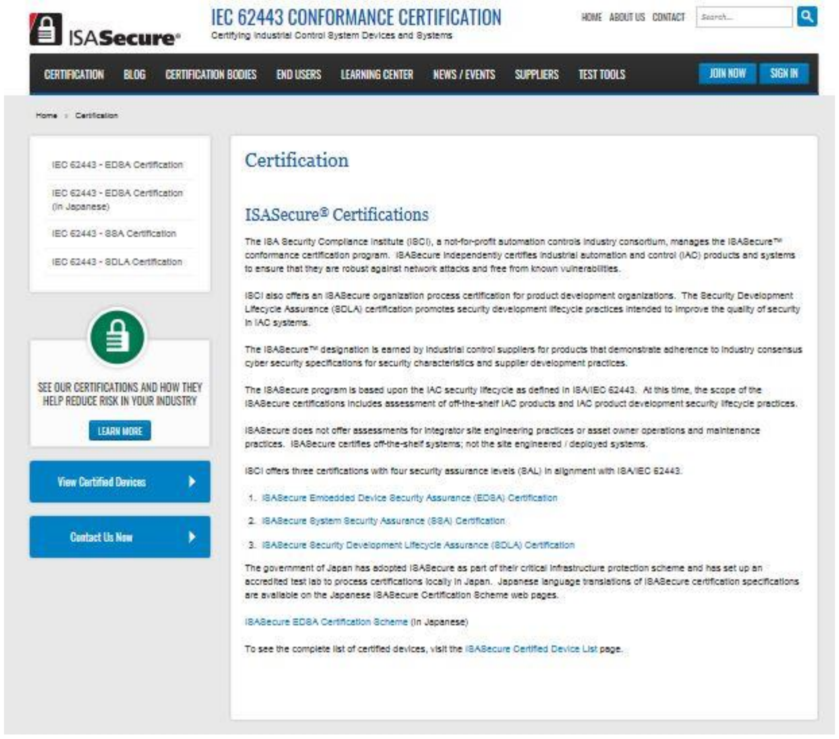
There are many traditional operating system functions that an RTOS will perform to manage the controller hardware, from processor and memory management to providing a network protocol stack and a file system structure. However, a key distinguishing feature of an RTOS is a micro-kernel structure that can prioritize, process, and react to external I/O and as it comes in, typically without buffer delay, and schedule real-time control software execution using a strict time-bound approach which has well-defined, fixed time constraints.

IV. Firmware Development Best Practices

Although the development of secure firmware and software for embedded control devices is beyond the scope of this paper, some basic best practice resources and basic principles will be identified so that customers of firmware updates have a conceptual understanding and awareness of the importance of secure code development and what potential questions to ask or what resources to consult to mitigate potential firmware coding flaws and bugs that can pose security threats. Additionally, customers can ask their supplier(s) what secure coding best practices, tools, and guidelines they employ. They can also inquire if their product is certified by an independent, third-party security testing organization.

Although more numerous in the IT world, organizations that provide operational technology (OT) secure coding guidance are more limited currently. The trend will hopefully change with the emergence of the Internet of Things (IOT), but presently the category of secure coding, development, and production of embedded control devices has not penetrated the large-scale production electrical power, building automation, or industrial control system industries. IOT systems in fact are presently notorious for their existing vulnerabilities and lack of the most basic security features. By contrast, it is imperative that production control system software for embedded devices undergo a rigorous validation and verification process to achieve functional, secure, reliable, and deterministic execution.

One example of a reference for embedded control system secure software and device development and certification is the *ISA Security Compliance Institute* organization (isasecure.org). They provide a lifecycle reference model as well as certify products that meet their criteria for secure control system device development based on the ISA99/IEC62443 set of industrial automation and control system standards. Below is a screen shot from the ISA Secure site explaining the certification services they provide:


ISASecure
IEC 62443 CONFORMANCE CERTIFICATION
Certifying Industrial Control System Devices and Systems
HOME ABOUT US CONTACT
Search...
CERTIFICATION BLOG CERTIFICATION BODIES END USERS LEARNING CENTER NEWS / EVENTS SUPPLIERS TEST TOOLS
JOIN NOW SIGN IN
Home » Certification
IEC 62443 - EDSA Certification
IEC 62443 - EDSA Certification (in Japanese)
IEC 62443 - SSA Certification
IEC 62443 - SDLA Certification
SEE OUR CERTIFICATIONS AND HOW THEY HELP REDUCE RISK IN YOUR INDUSTRY
LEARN MORE
View Certified Devices
Contact Us Now

Certification

ISASecure® Certifications

The ISA Security Compliance Institute (ISCI), a not-for-profit automation controls industry consortium, manages the ISASecure™ conformance certification program. ISASecure independently certifies industrial automation and control (IAC) products and systems to ensure that they are robust against network attacks and free from known vulnerabilities.

ISCI also offers an ISASecure organization process certification for product development organizations. The Security Development Lifecycle Assurance (SDLA) certification promotes security development lifecycle practices intended to improve the quality of security in IAC systems.

The ISASecure™ designation is earned by industrial control suppliers for products that demonstrate adherence to industry consensus cyber security specifications for security characteristics and supplier development practices.

The ISASecure program is based upon the IAC security lifecycle as defined in ISA/IEC 62443. At this time, the scope of the ISASecure certifications includes assessment of off-the-shelf IAC products and IAC product development security lifecycle practices.

ISASecure does not offer assessments for integrator site engineering practices or asset owner operations and maintenance practices. ISASecure certifies off-the-shelf systems; not the site engineered / deployed systems.

ISCI offers three certifications with four security assurance levels (SAL) in alignment with ISA/IEC 62443.

1. ISASecure Embedded Device Security Assurance (EDSA) Certification
2. ISASecure System Security Assurance (SSA) Certification
3. ISASecure Security Development Lifecycle Assurance (SDLA) Certification

The government of Japan has adopted ISASecure as part of their critical infrastructure protection scheme and has set up an accredited test lab to process certifications locally in Japan. Japanese language translations of ISASecure certification specifications are available on the Japanese ISASecure Certification Scheme web pages.

ISASecure EDSA Certification Scheme (in Japanese)

To see the complete list of certified devices, visit the ISASecure Certified Device List page.

Below are some best practices regarding control system software and device development for control system design that asset owners and customers should be aware of and can inquire with their supplier regarding adoption:

1) Identify third-party standards organizations that provide standards, best practice guidance, tools, and certification services for secure control system development.
2) Adopt a Security Lifecycle Assurance Approach.
3) Utilize open source RTOS, library, IDE, and tools for development.
4) Limit use of third-party binaries.
5) Keep complexity to a minimum.
6) Reduce remote/distributed code execution as much as possible to limit resulting system traffic flows that increase monitoring and security requirements and resources.

Vendors that follow the above general principles as well as the more specific guidance for secure control system software and device development and certification at sites such as the *ISA Security Compliance Institute* organization can provide a more comprehensive, secure, and least complexity approach to firmware development and implementation of embedded control system devices.

V. Firmware Distribution Best Practices

Vendors provide firmware updates to their customers using a wide variety of methods ranging from simply placing them on the Internet for download to shipping them on CD to registered product customers with current contact information based on confirmation requests from the customer. This section is not intended to cover all the possible ways in which controller firmware can be distributed but rather identify some current best practices for delivery and verification.

Below is an example secure firmware distribution and verification process that represents an ideal approach:

1) Availability to current, registered customers only that own the corresponding control devices for which the firmware update is available
2) Positive acknowledgement from current customer to verify contact information and interest in receiving the firmware update
3) Shipment via Out-of-Band (OOB) delivery on CD to the current customer contact information delivery address

4) Availability of a hash of the firmware should be provided to the customer so they can verify independently the integrity of the firmware update
5) Digitally signed firmware that is verified for integrity by the controller during the update process
6) Encrypted firmware that provides maximum security during the distribution, storage, and retention periods before and after the firmware is loaded on the controller

Note that if an online delivery model is used instead, the download process should still include all the other best practices outlined above in addition to the following security measures:

1) Customer registration code should be issued for firmware distribution website access
2) Vendor web site should be password protected using strong password policies (i.e. complexity and aging requirements enforced) to authenticate the customer
3) Connection to the web site should be encrypted (e.g. TLS 1.2 or greater)
4) Customer access based on being a current, validated product customer
5) Customer can only download firmware for products they have purchased
6) Provide a hash to verify the integrity of the downloaded firmware

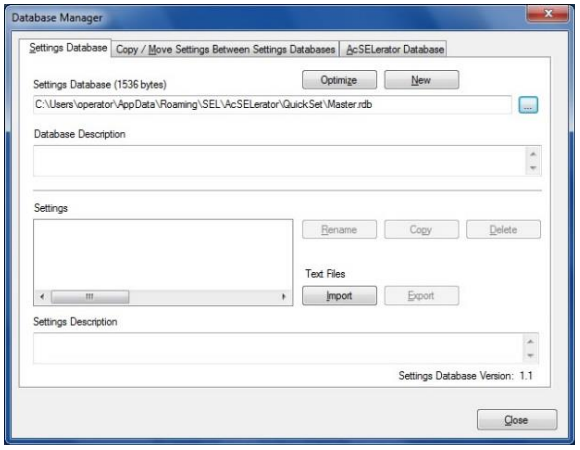
Although an adversary can always purchase the product and obtain the firmware, the infrastructure provider can attempt to vet the request before issuing the registration code and login credential (e.g. via sales representative or channel partner). Whether the firmware vendor uses the OOB delivery on CD method or an online delivery method, the firmware should ideally be digitally signed for integrity during the update process and protected via encryption during the storage and retention phases at the customer site. DOD customers, at a minimum, should only use firmware from vendors that require user authentication and provide a hash. (See the *References* section for links to sites that explain hashing as well as provide a free hash checking tool.) DHS is developing an Acquisition Guide with these requirements and NIST SP 800-161 Supply Chain also requires this.

VI. Firmware Storage and Organization Best Practices

After a firmware update is obtained, the customer should ideally employ a set of best practices to ensure the secure, synchronized storage of the updates. Doing so will allow the required version of updates to be available if a roll-back or system restoration is required and prevent multiple versions of the firmware from becoming distributed across multiple desktops of engineers and laptops of technicians in a decentralized and unsynchronized manner.

Ideally, all firmware updates should be stored in a centralized and secure location. This will provide a single repository to allow version control to be implemented and the firmware to be stored securely in single location. The single location can be a set of folders on a file server, a formal repository on network server, or a formal database server if the vendor or third-party tools available support these features.

For firmware updates of electrical relays, Schweitzer Engineering Laboratories (SEL) provides a software tool called AcSELerator that can be used to manage and apply firmware updates and centrally store the associated device configuration settings and parameters associated with the embedded device in a centralized location. When the firmware is stored in the same location as the settings, a single repository then exists for the production embedded device management including firmware and device parameters for the electrical relays. Shown below are two example screens for the AcSELerator Database Manager that indicates how a centralized Master file location can be specified. Although the example shows the Master repository location as a local filename on a Windows computer hard drive, the filename specification can use a server hostname in the file name instead to make the repository available over the network. It could also specify a database server name along with a database port number, in which case the central repository can be implemented as a *client-server database system* on the production network in order to make embedded device configuration files such as those for an electrical relay available in a database storage and distribution format. Likewise, the firmware stored on the same client-server system can be shared securely across the enterprise along with the settings for access by Engineers performing firmware updates on electrical relays to form a single location approach.


Database Manager

Settings Database (1536 bytes)
C:\Users\operator\AppData\Roaming\SEL\AcSELerator\QuickSet\Master.rdb

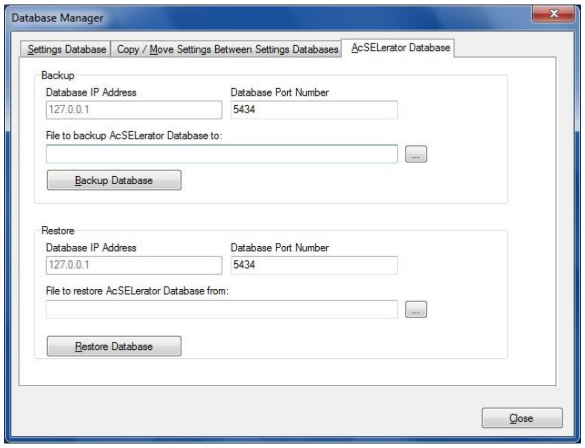
Database Description

Settings

Text Files

Settings Description

Settings Database Version: 1.1


Database Manager

Backup
Database IP Address: 127.0.0.1
Database Port Number: 5434
File to backup AcSELerator Database to:

Restore
Database IP Address: 127.0.0.1
Database Port Number: 5434
File to restore AcSELerator Database from:

The Engineers and/or Technicians that require access to the firmware for performing security testing or performing an update can then download a copy of the firmware from the server to work with and then delete the local, temporary copy when the testing or updating process is complete since the master copy resides on the server filesystem. There may also be a team of

Cyber Analysts that work with the Engineering Team to the perform cyber security checks on the target firmware that may also wish to access temporary copies of the firmware using such procedures. As will be described in the next section of this paper, such Cyber Analysts may also want to analyze the embedded device firmware updates using the suite of Binary Analysis Tools (BAT) being developed by LLNL as part of the DOD funded ESTCP Project.

An alternative, general approach is to use a general, third-party or public domain *Binary Repository Manager* tool that can be purchased or downloaded to implement a structured, centralized approach to firmware update organization and synchronization for all types of firmware. An example third-party binary repository is *Artifactory* which gives engineers and technicians the ability to manage a matrix of binaries such as firmware updates.

Lastly, note that if there are separate test and/or development networks they should have a separate firmware management database from the production network or alternatively have an internal security zone designated for the protection a single database using the ISA/IEC *Zone and Conduit* model for defining the minimum required traffic flows to support access to the dedicated firmware management database security zone from other security zones defined within the Operational Technology (OT) network infrastructure.

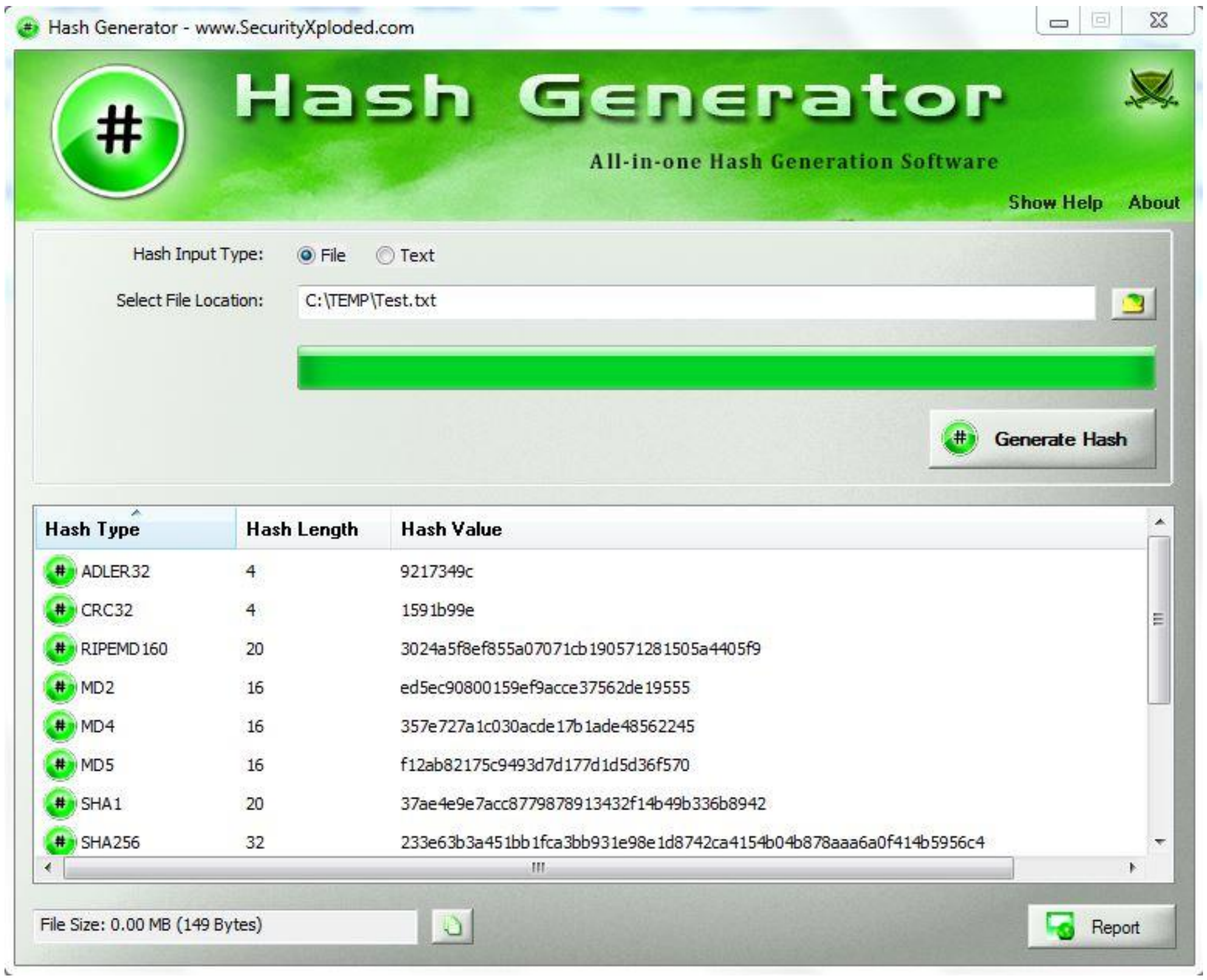
VII. Firmware Security Checking Best Practices

As noted in the previous section on the importance of firmware security, it is difficult to analyze firmware for security vulnerabilities because it is provided in the form of a *binary* file that is assembled into the machine code specific to the instruction set of the microprocessor of the control device. For example, a *PowerPC* microprocessor could be embedded in the most powerful controller in a BAS controller product line. The term “powerful” in this context is defined by support for a large number of digital & analog physical I/O channels, a large number of control programs that can execute simultaneously, support for all the industry standard BACnet fieldbus protocols, and a large capacity for equipment and process trend data sampling and storage. Other embedded processors such as the Intel Atom and ARM Cortex can serve as microprocessors for high-end controllers as well.

A basic security check that can be performed by end users applying a firmware update is to perform a file hash. Hashing is cryptographic method of performing file verification by transforming the file into a shorter fixed length string of characters. The vendor of the firmware update file must first perform and then make available the hash string of the file. Then the customer can also perform a hash of the file using freely available or built-in operating system hash checking tools to verify the vendor hash. The two independent file hashes should match for the security check to be successful. Although the hash string of a file does not uniquely identify it, it is a form of cryptographic integrity that makes it very unlikely that two different files will have the same hash value.

Hash checking is a fast and easy process provided that the vendor publishes their firmware file hash values. The Windows Operating System itself provides a built-in hashing tool accessible from the PowerShell for users comfortable working at the command line. Simply press the Windows Key or click on the Windows Start icon, then type PowerShell into the run command box. After the PowerShell screen opens, type "get-file hash FILEPATH" at the command prompt. By default, the SHA256 result will be shown, but other hashing algorithms can be specified.

An example of a free and easy to use graphical hash checking application is *Hash Generator*. It is a comprehensive tool that displays numerous hashing algorithm results and can check files as well as text. See the Reference section of this paper for the download location. Shown below is an example screenshot.


Hash Generator - www.SecurityXploded.com
Hash Generator
All-in-one Hash Generation Software
Show Help
About
Generate Hash
Report

Hash Input Type: [✓ File] Text

Select File Location: C:\TEMP\Test.txt

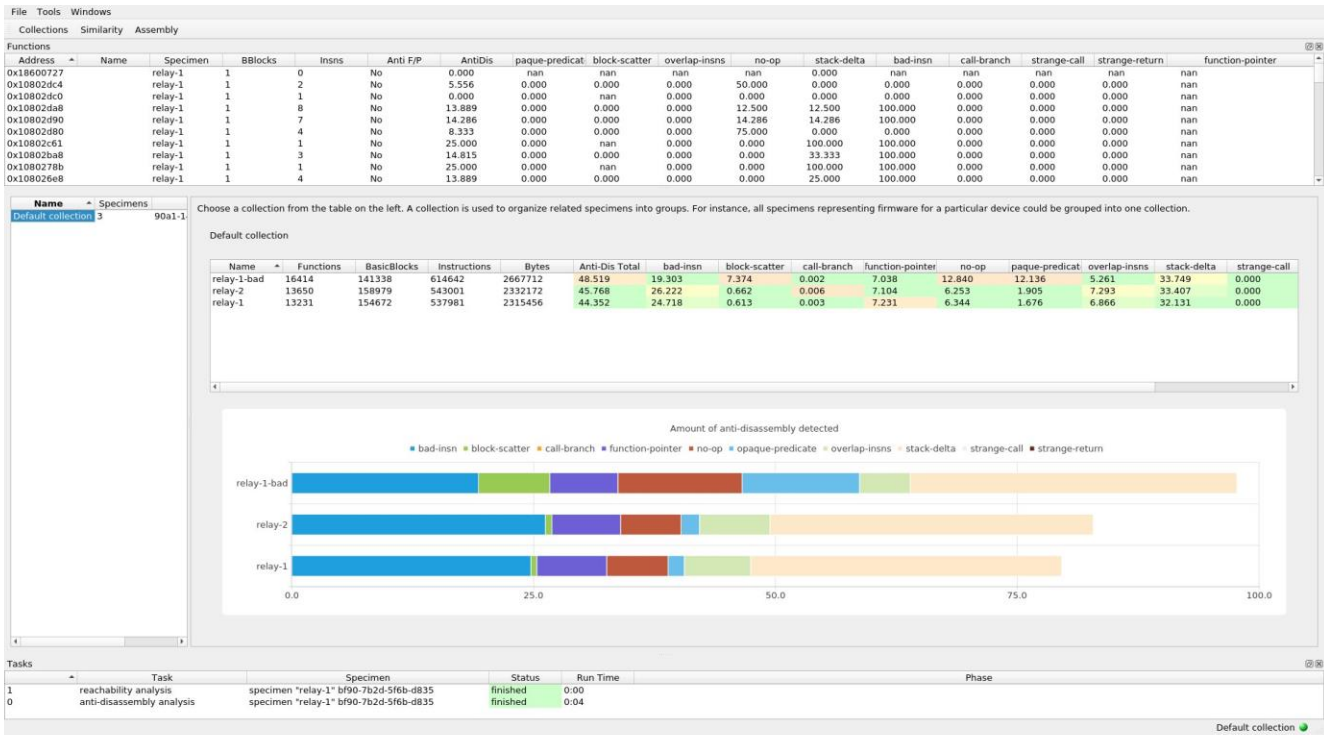
Hash Type	Hash Length	Hash Value
ADLER32	4	9217349c
CRC32	4	1591b99e
RIPEMD160	20	3024a5f8ef855a07071cb19057128150 5a4405f9
MD2	16	ed5ec90800159ef9acce37562de19555
MD4	16	357e727a1c030acde17b1ade48562245
MD5	16	f12ab82175c9493d7d177d1d5d36f570
SHA1	20	37ae4e9e7acc8779878913432f14b49b336b8942
SHA256	32	233e63b3a451bb1fca3bb931e98e1d8742ca4154b04b878aaa6a0f414b5956c4

File Size: 0.00 MB (149 Bytes)

A more sophisticated and complementary set of security tools were developed during the DOD funded ESTCP Project. LLNL developed a suite of BAT software tools specifically for automated cyber-risk analysis of firmware binaries. These tools are freely available and can be used in an off-line manner to perform security checking of the firmware updates received from embedded control device suppliers. Below are capsule descriptions of the BAT tools.

1) *BAT1:* This tool is used to detect the presence of anti-disassembly technologies, often correlated to advanced forms of malware.
2) *BAT2:* This tool is used to analyze differences between two firmware releases to examine changes that could include malicious code.
3) *BAT3:* This tool analyses firmware for the presence of unused code, often a mechanism to obscure the behavior and complicate the analysis of both source code and binaries.
4) *BAT4:* This tool analyses firmware for "backdoor" exploits and the inputs required to exploit them.
5) *BAT5:* This tool evaluates specific paths to protected resources in the firmware, which aids reverse-engineering analysts find paths needed protection.

Shown below is an example screen from the BAT #1 tool.



Address	Name	Specimen	BBlocks	Insns	Anti F/P	AntiDis	paque-predicat	block-scatter	overlap-insns	no-op	stack-delta	bad-insn	call-branch	strange-call	strange-return	function-pointer
0x18600727		relay-1	1	0	No	0.000	nan	nan	nan	nan	0.000	nan	nan	nan	nan	nan
0x10802dc4		relay-1	1	2	No	5.556	0.000	0.000	0.000	50.000	0.000	0.000	0.000	0.000	0.000	nan
0x10802dc0		relay-1	1	1	No	0.000	0.000	nan	0.000	0.000	0.000	0.000	0.000	0.000	0.000	nan
0x10802da8		relay-1	1	8	No	13.889	0.000	0.000	0.000	12.500	12.500	100.000	0.000	0.000	0.000	nan
0x10802d90		relay-1	1	7	No	14.286	0.000	0.000	0.000	14.286	14.286	100.000	0.000	0.000	0.000	nan
0x10802d80		relay-1	1	4	No	8.333	0.000	0.000	0.000	75.000	0.000	0.000	0.000	0.000	0.000	nan
0x10802c61		relay-1	1	1	No	25.000	0.000	nan	0.000	0.000	100.000	100.000	0.000	0.000	0.000	nan
0x10802ba8		relay-1	1	3	No	14.815	0.000	0.000	0.000	0.000	33.333	100.000	0.000	0.000	0.000	nan
0x1080278b		relay-1	1	1	No	25.000	0.000	nan	0.000	0.000	100.000	100.000	0.000	0.000	0.000	nan
0x108026e8		relay-1	1	4	No	13.889	0.000	0.000	0.000	0.000	25.000	100.000	0.000	0.000	0.000	nan

Name	Specimens	
Default collection	3	90a1-1

Choose a collection from the table on the left. A collection is used to organize related specimens into groups. For instance, all specimens representing firmware for a particular device could be grouped into one collection.

Default collection

Name	Functions	BasicBlocks	Instructions	Bytes	Anti-Dis Total	bad-insn	block-scatter	call-branch	function-pointer	no-op	paque-predicat	overlap-insns	stack-delta	strange-call
relay-1-bad	16414	141338	614642	2667712	48.519	19.303	7.374	0.002	7.038	12.840	12.136	5.261	33.749	0.000
relay-2	13650	158979	543001	2332172	45.768	26.222	0.662	0.006	7.104	6.253	1.905	7.293	33.407	0.000
relay-1	13231	154672	537981	2315456	44.352	24.718	0.613	0.003	7.231	6.344	1.676	6.866	32.131	0.000

	Task	Specimen	Status	Run Time	Phase
1	reachability analysis	specimen "relay-1" bf90-7b2d-5f6b-d835	finished	0:00	
0	anti-disassembly analysis	specimen "relay-1" bf90-7b2d-5f6b-d835	finished	0:04	

For a more complete description of the development, evaluation, and usage of the BAT tools, see Section XII for references to the study completed by the Boeing Research & Technology (BR&T) group as well as the user guide created by members of the ESTCP Project Team.

VIII. Firmware Update Application Best Practices

The firmware update process will vary widely depending on the type of embedded controller, the vendor or reseller that supplies the controller, and the tools and corresponding procedures used to download the update into the controller. Therefore, this section will provide some general guidelines, with emphasis on integrating the best practices outlined previously regarding integrating security and reliability steps into the process such as verification of the firmware files, hashing of the file, security checking with the BAT tools, and using vendor supplied or third-party tools for a central repository for storing and organizing the firmware update files.

It is important to realize that the embedded device normally performs a restart operation to load and use the new firmware. Therefore, personnel must prepare for connected equipment to temporarily lose their I/O control and/or monitoring signal. Mitigations include putting equipment "in-hand" to temporarily use internal manual setpoint control such as a chiller in a building automation system or a PLC in *Run* vs *Remote* mode or switching to a redundant feeder and associated protection equipment when updating a relay if available.

In general, it is ideal to integrate embedded control equipment firmware updates with a formal *Preventative Maintenance* (PM) program for the control or electrical power system. A PM program entails regular, periodic, planned, and scheduled equipment maintenance procedures such as replacing bearings on motors or breaker maintenance in a substation. By integrating firmware updates and other control system related operational technology (OT) updates such as HMI software and operator workstation hardware, automation system software, automation and operating system patches, and vendor tools it ensures the supporting OT technologies that provide access to the control system and embedded control devices for development, configuration, updating, operation, and system monitoring are also kept current in addition to the control system equipment such as motors, chillers, boilers, air handling units, transformers, switches, circuit breakers, etc. being maintained during regularly scheduled and performed PM's .

Some best practices to keep in mind during the firmware update process:

1) Perform the firmware update in a test/development environment first if possible before applying the update on the production control or power system. (See the *References* section for a link to the ESTCP Test and Development Environment.)

2) Use a *role-based* approach to planning and executing the firmware update process. (For example, *Protection Engineer, Power System Technician, and Cyber Analyst*.)
3) Develop a *Firmware Update Implementation and Test Plan* that includes all steps of the update from security checking to application to post installation testing and documentation of the results, version, and date applied.
4) Perform a backup of the current configuration if a roll-back or recovery is required.
5) Plan the update during a preventative maintenance period if possible when equipment is expected to be temporarily out of service. Doing so will address the reboot issue and ensure that firmware is kept up to date along with other aspects of equipment maintenance. This will avoid having the controller not being updated at all or being so outdated that multiple intermediate releases must be applied before the most current update can be performed.
6) Integrate security pre-checks such as hashing of the firmware file and use of the BAT tools. If the hash does not match or the BAT tools produce a flag, have a *Cyber Analyst* investigate the firmware before proceeding with the update.
7) Work with a temporary copy of the firmware file checked out of the firmware binary file repository or the vendor supplied firmware management tool. The permanent copy should reside in the repository or centralized vendor system management tool. This avoids the poor practice of having multiple copies of firmware, configuration, and logic programs files spread out across multiple desktops of *Protection Engineers* and laptops of *Power System Technicians* within the organization in a desynchronized and disorganized manner.
8) Create a post-update backup of the system configuration to save for future system recovery if needed.

IX. Firmware Retention Best Practices

As noted in the section on firmware management tools, it is ideal to retain previous versions of firmware updates applied to all devices currently in service or serving as spares. Having the firmware (along with the control programs and logic, configuration settings, etc.) will allow an embedded control device to be restored if necessary to a previous build. Also, retaining a version history will permit regressive security testing and checking if desired by the organization utility group or a more specialized cyber security group of the firmware updates planned or previously implemented using the tools described in this paper or other available binary analysis tools. A vendor supplied device management tool or a third-party binary repository manager tool can be used to retain all the controller firmware in a single location that can maintain the desired version history and be secured to prevent unauthorized access.

X. DOD Firmware Monitoring

An initiative by the United State Cyber Command (USCYBERCOM) referred to as the Advanced Cyber Industrial Control System (ACI) Tactics, Techniques, and Procedures (TPP) has the goal to better defend DOD critical infrastructure and embedded systems and serves as an example for how firmware best practices can be integrated with a monitoring and management model. For the full report issued by DOD see the references section.

The ACI TTP uses a detection, mitigation, and recovery model to defend critical infrastructure. Regarding firmware best practices, the detection phase uses a set of integrity checks when an anomalous event occurs that include:

1) Identifying valid firmware versions
2) Verifying firmware against a baseline
3) Detecting unexpected firmware changes
4) Calculating the hash value of the firmware
5) Roll-back to a previous version if incident recovery is required

A full workflow process is then defined that use the firmware integrity checks in addition to a complete set of integrity checks for the entire critical infrastructure system and used by ACI operators to detect deviation from baseline operation while performing Routine Monitoring. These checks provide actions which assists the operator in determining whether a cyber event regarding firmware is in progress. A segmentation strategy is used in Mitigation to isolate a compromised embedded system that has failed an integrity check regarding firmware. After the system is stabilized, the operator proceeds to Recovery. For compromised embedded device firmware, recovery can include re-imaging the compromised firmware from a known and trusted source, verifying the latest updates are applied, and saving the new hash value for the firmware.

The DOD model represents how best practices can be integrated with a tactical management model for critical infrastructure protection regarding firmware. Having a set of robust procedures for monitoring an embedded system and then mitigating and recovering from compromised firmware using a set of best practices is an ideal approach.

XI. Conclusion

This paper has focused on the concept of firmware that runs on embedded microprocessor controllers such as those used to monitor and control energy infrastructure systems on military bases such as electrical power distribution and building automation systems. The applicability of the best practices and tools used to implement them are more broadly applicable, however, to numerous other embedded control device applications and the wide variety of critical infrastructure systems they monitor and control from dam spillway gates to passive train control braking to nuclear material protection systems. The need to address firmware security from a supply chain management threat perspective as well as the role that firmware has in the operation of an embedded control device was provided as a foundation for the subsequent content of the paper. Sections on best practices for the customers and end users of controller firmware updates in terms of supplier development practices, distribution, storage and organization, security checking, updating, and retention have been described with the intention of providing awareness and general guidance regarding the subject of firmware security. The purpose of this paper was to make customers aware that firmware updates are a potential supply chain management threat issue, understand the critical role of firmware in the operation of embedded controllers used in energy infrastructure monitoring and control, and know the set of best practices and tools available that can help mitigate potential firmware security threats. The DOD ACI TTP model was also introduced to demonstrate how best practices can be integrated with a tactical model for detecting and responding to compromised firmware on embedded systems.

Following this concluding section are a list of references cited in this paper and provided for further background and reading. An example firmware update process for an electrical relay type of embedded device is also provided as an appendix to pull together the concepts presented and demonstrate an integrated security approach using some of the best practices and tools identified in this paper.

XII. References

1) Stuxnet: "To Kill a Centrifuge"; Langner; March 2017; *https://www.langner.com/wp-content/uploads/2017/03/to-kill-a-centrifuge.pdf*

2) Hash Checking Explained: *https://www.howtogeek.com/67241/htg-explains-what-are-md5-sha-1-hashes-and-how-do-i-check-them/*

3) Free Hash Checking Tool: https://securityxploded.com/getsoftware_direct.php?id=2211

4) Schweitzer Compass Tool: *https://selinc.com/products/compass/*

5) Schweitzer SEL acSELrator Tool: *https://selinc.com/products/5030/*

6) Binary Repository Tools: *https://en.wikipedia.org/wiki/Binary_repository_manager*

7) Artifactory: *https://jfrog.com/artifactory/*

8) Binary Analysis Tools Evaluation: "*ESTCP Operational Usage Report*"; Joshua Cazalas & Adam Brown; The Boeing Company.

9) Binary Analysis Tools User Guide: "*ROSE Binary Analysis Tools User Guide"*, E. Banks, K. Masica, R. Matzke, D. Quinlan, & W. Hutton

10) The ESTCP Test and Development Environment: https://serdp-estcp.org/Tools-and-Training/Installation-Energy-and-Water/Cybersecurity/Test-and-Development-Environment

11) The DOD ACI TTP: www.acq.osd.mil/eie/Downloads/IE/ACI%20TTP%20for%20DoD%20ICS.pdf

XIII. Appendix: Example Best Practice Firmware Update for an Electrical Power Relay

Below is an example process for performing a firmware update on an electrical power distribution relay device that incorporates the best practices outlined in this paper in the workflow process. A step-by-step description is provided first followed by a flowchart representation.

1) Obtain the new firmware update file from the vendor for the target relay.
2) Perform a hash security check on the firmware file to verify integrity.
3) Perform BAT tool checks on the firmware file for cyber security assurance.
4) Place the firmware update file in a central server location, binary repository, or vendor tool.
5) Develop a *Firmware Update Implementation and Test Plan* for the target relay.
6) Ensure that the *Firmware Update Implementation and Test Plan* has been reviewed and approved, all required permissions to proceed have been obtained, and that safety procedures such as Lockout-Tagout (LOTO) have been authorized and implemented.
7) Copy the current/active firmware version file (if available) and new firmware update file from the central storage location or binary repository to a Laptop and use as local temporary copies.
8) Backup the current target relay settings.
9) Take the active feeder and target relay out of service via scheduled outage or switching the active feeder to a redundant feeder if available.
10) Confirm target relay is out of service and the redundant feeder (if available) is operating properly with no alarms present.
11) Confirm no other process or physically connected devices are active on the target relay.
12) If necessary, try downloading currently active firmware from the target relay to the connected laptop if it is not already available per Step #4 (for potential rollback).

13) Complete firmware upgrade process using the vendor firmware upgrade software or procedures.

14) After the firmware upgrade has completed, perform a settings comparison to the earlier saved settings if supported by the vendor configuration management software.

    a) If settings are a 100% match to earlier settings, proceed to the next step.

    b) If settings are not a 100% match, rollback to load settings saved earlier during backup.

    c) If settings comparison is not possible, load settings saved earlier during backup.

15) Confirm all data connections to/from the relay are active and working as expected.

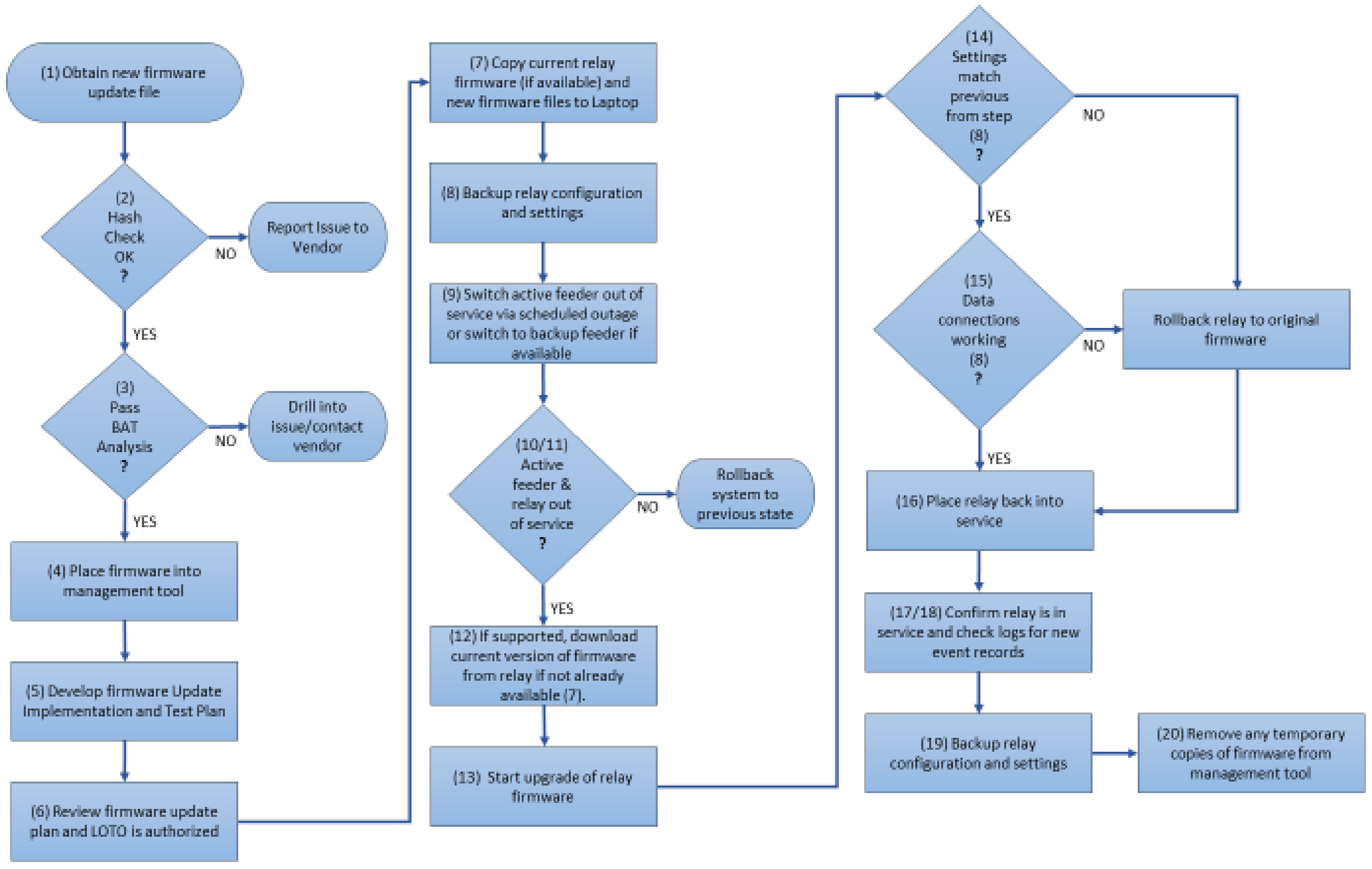
16) Place target relay back into service by closing open switchgear to restore power or switching the backup feeder (if available) to the previously active feeder.

17) Confirm the target relay is in service and confirm the backup feeder (if available) is out of service.

18) Check active feeder relay logs for new records and unusual events.

19) Perform a backup of the current relay configuration, settings, and logic programming.

20) Remove the temporary copies of the firmware from the Laptop. (Permanent copies retained in the central server location, binary repository, or vendor tool if applicable.)


(1) Obtain new firmware update file
(2) Hash Check OK ?
NO
Report Issue to Vendor
YES
(3) Pass BAT Analysis ?
NO
Drill into issue/contact vendor
YES
(4) Place firmware into management tool
(5) Develop firmware Update Implementation and Test Plan
(6) Review firmware update plan and LOTO is authorized
(7) Copy current relay firmware (if available) and new firmware files to Laptop
(8) Backup relay configuration and settings
(9) Switch active feeder out of service via scheduled outage or switch to backup feeder if available
(10/11) Active feeder & relay out of service ?
NO
Rollback system to previous state
YES
(12) If supported, download current version of firmware from relay if not already available (7).
(13) Start upgrade of relay firmware
(14) Settings match previous from step (8) ?
NO
YES
(15) Data connections working (8) ?
NO
Rollback relay to original firmware
YES
(16) Place relay back into service
(17/18) Confirm relay is in service and check logs for new event records
(19) Backup relay configuration and settings
(20) Remove any temporary copies of firmware from management tool