Verification Tool for Securing RISC-V FPGA-Based Process Control System

Christine Johnson*, Ann Marie Reinhold*§, Clemente Izurieta*‡§, Bradley M. Whitaker† and Brock LaMeres†

*Software Engineering and Cybersecurity Laboratory, Montana State University, Bozeman, MT, USA
†Electrical & Computer Engineering Department, Montana State University, Bozeman, MT, USA
‡Idaho National Laboratory, Idaho Falls, ID, USA
§Pacific Northwest National Laboratory, Richland, WA, USA

***Abstract*—Industrial Control Systems (ICSs) are a crucial component of critical infrastructure and a popular target for cyberattacks. While most research focuses on the defense of large-scale networks of ICSs, it is critical to expand research on the small-scale networks of Process Control Systems (PCSs), which attackers may target to remain undetectable from the security of the larger network. One potential method of protection for PCS networks is the use of verification tools; however, existing research on verification tools focuses on the detection of attacks without mitigation. This research describes two instantiations of a verification tool for a PCS controlled by a RISC-V computer implemented on an FPGA. The first version of the verification tool provides passive detection, informing an engineering station of an attack, and the second version provides detection with mitigation against unauthorized command messages and malicious software downloads. An experimental testbed was developed to test both versions, consisting of: (1) a circuit to charge a battery; (2) an FPGA controller that controls charge/discharge based on user input; (3) an engineering station that provides control data, and updates firmware to the FPGA; and (4) a verification tool that verifies input forwarded by a passive serial tap, connected through the FPGA's hardware. The experimental data yielded promising results, with the tool successfully providing mitigation and detection against attacks on the serial communication channel between the engineering station and FPGA. This approach has applicability to common ICS computer devices, such as programmable logic controllers (PLCs).**

***Index Terms*—Industrial Control, Process Control, Field Programmable Gate Arrays, Reduced Instruction Set Computing, Computer Security, Hardware Security, Prevention and mitigation, Cyber-physical systems, Industrial Internet of Things.**

I. Introduction

Industrial Control Systems (ICSs) are an integral part of a nation's infrastructure and must operate continuously and without error. Many industrial systems were originally standalone operational technology systems, but the rise of modern-day information technology has led them to become increasingly interconnected with each other through local networks and the Internet. Although this interconnectedness allows for advances in ease of use and access to a system, it also increases the surface area of attacks. Security research around ICSs focuses on their larger network systems, with the goal of preventing attackers from gaining large-scale access to the control system. However, this may diminish well-rounded security for smaller portions of the system, as it is assumed that attacks will be stopped before they reach that level.

One of these important subcomponents of ICSs is Process Control Systems (PCSs). PCSs control singular processes within a larger ICS, such as adjusting the level of a water tank within a sewage plant. They are typically controlled with Programmable Logic Controllers (PLCs), which use the innately insecure Modbus communication protocol [3]. Many attacks take advantage of PCSs due to their lack of comprehensive implementation of security protocols. Notably, Internet-facing security on PLCs and other components is sometimes ignored because it interferes with system availability.

A pervasive issue of adding more security to control systems occurs when adding it has a negative impact on the availability of the systems. One technique to ensure the integrity of measurement data and control logic delivered to PLCs without impacting system availability is a verification tool. The passive nature of verification tools lets them monitor the control data provided to a PLC without preventing or delaying its arrival to the controller. Research on verification tools currently explores the ability to detect attacks, but research to prevent attacks has not received the same attention.

In our research, we create a verification tool that provides mitigation of these attacks in addition to detection without impacting the availability of the control system. We describe two instantiations of a verification tool for a PCS controlled by a RISC-V computer implemented on a Field Programmable Gate Array (FPGA). These instantiations separately provide *i)* passive detection, where the tool detects and informs the engineering station of an attack, and *ii)* detection with mitigation, where the attacks never reach the engineering station. The tool operates in the presence of unauthorized command message attacks where an attacker aims to modify the controller's logic in the engineering station. Though this tool is developed for a RISC-V FPGA, the approach has applicability to other common ICS computer devices such as programmable logic controllers (PLCs), which use ohter potentially vulnerable serial communication protocols.

II. Background

An ICS comprises complex hardware and software components designed to automate processes within physical industrial systems. In cybersecurity, ICSs have received significant attention due to their role in national infrastructures and the

surge in harmful cyber-assaults targeting them. Stuxnet stands out as one of the most documented examples of such attacks [1]. In 2010, this attack targeted an Iranian nuclear facility, causing physical damage to the facility by taking advantage of a vulnerability within its Siemens PLC controlled PCSs to spread malicious control logic. PLCs are one of the common components used in the PCSs and are typically equipped with internet-facing components but not always secure-by-default. This allows attackers to have direct access to logic controls while avoiding the security of the larger ICS network.

A. Process Control Systems

A large majority of research for PCSs focuses on PLCs [2]. Many PCS components are proprietary, and therefore it can be hard to develop generalized defense techniques that adapt easily to all types. In particular, this prevents proper understanding of the operational logic and internal mechanisms of these embedded systems when developing defense techniques [14]. This research focuses on a control system where the controller is implemented with an FPGA. FPGAs are built around a matrix of configurable logic that allows them to be reconfigured and reprogrammed as needed and thus can be specialized to fit the specific needs of a control system. Although FPGAs are less common in ICS because PLCs tend to be cheaper and easier to use, FPGAs can be programmed to perform the same functionality within an ICS [4] [5].

B. Weaknesses

When discussing PCS weaknesses, the MITRE Common Weakness Enumeration (CWE) nomenclature is used [7]. In recent years, PCSs within ICSs have been implemented with greater amounts of interconnectivity, which improves their ease of use, but increases vulnerabilities due to the opening of internal attack vectors [9]. For example, to incorporate internet-facing PLCs into PCSs without negative effects on the control system's performance, many systems chose to ignore their security features [8]. This is CWE-655: Insufficient Psychological Acceptability, where a product's protection mechanism is too inconvenient to use, encouraging non-malicious users to bypass the mechanism. With this lack of properly implemented anomaly detection for PLCs to identify irregular control logic and measurements, methods of anomaly detection for PCSs have been a well-studied topic [6].

Another common weakness in PCSs is a lack of proper authentication and encryption that is innate to its communication protocols. Many systems use the Modbus communication protocol, which does not offer built-in encryption or strong authentication mechanisms [1]. This relates to weakness CWE-311: Missing Encryption of Sensitive Data, and allows attackers easy access to gather information on a system's measurements and commands for easier fabrication of attacks. PCSs also experience weaknesses due to human factors, such as malicious personnel and inexperienced users.

C. Attack Techniques

There are a plethora of attack techniques available against PCSs. Most techniques target the availability portion of the CIA triad, consisting of Confidentiality, Integrity, and Availability [2]. ICS availability is best described as *the probability that a system is up and running at some point in time?* Availability for ICSs is critical, as systems need to run uninterrupted to maintain proper functionality, and the entire ICS's availability can be affected by a single PCS. For example, at a Queensland sewer, false commands and data injected into a wastewater station caused one million liters of wastewater to be expelled into nearby waterways [11].

A frequent form of attacks on PCSs, commonly with internet-facing PLCs, is Adversary-in-the-Middle (AiTM), where attackers with privileged network access read or spoof network traffic [10]. A large portion of PCS attacks consist of false data injection and control logic injection AiTM attacks. In the MITRE ATT&CK ICS Matrix, these are the Spoof Reporting Message and Unauthorized Command Message techniques, in the Impair Process Control tactic, MITRE ID TA0106 [12]. With spoof reporting messaging, an attacker seeks to cause harm by altering reported measurement data used as feedback to control a system. With unauthorized command message attacks, an attacker sends malicious commands and logic to the controller. Both attacks can cause system assets to perform actions not aligned with its current state or beyond normal bounds [13]. Lastly, Denial of Service (DoS) attacks disrupt a system's ability to respond by sending a high volume of traffic that the system is incapable of handling [3].

D. Defense Techniques

Defense techniques focus on detection and mitigation, also referred to as prevention, of intrusions, with a majority focusing on the former [2]. Detection techniques focus on identifying attacks using analysis and process techniques [21]. Among common intrusion detection methods are honeypots, verification tools, network traffic inspection, and dynamic watermarking. Honeypots are intended to lure attackers in and allow an attack to occur in a safe setting, so that information can be safely collected on the attack [15].

Mitigation techniques focus on reducing harm from attacks. One such technique is encryption for ICS networks. This addresses the lack of encryption within system network protocols, which often send plain text [2]. Verification tools work to confirm the veracity of firmware or software within the PCS, detecting malicious actions if any of these facets have been altered, typically grouped as techniques that validate program inputs [16], and techniques such as dynamic watermarking to determine the veracity of a signal within the PCS [17].

E. Serial Communication

Modbus protocol uses serial communication to send data over a line as a series of bits. Serial data standards such as Recommended Standard 232 (RS-232) standard, which is widely used in industrial monitoring and embedded systems such as PLCs, specify how serial data is transmitted [22]. A network Test Access Point (TAP) is a device that connects directly to cabling infrastructure to split or copy packets for use in analysis and security [18]. A serial tap is a network tap

specialized for serial communication. A passive tap is invisible to the network and may monitor traffic undetected.

F. RISC-V Architecture

We selected the RadPC edge computer developed by Resilient Computing as our controller [20]. It is implemented onto a Xilinx Nexys Artix-7 FPGA, has a RISC-V RV32I ISA process architecture, and has fault-mitigation procedures to help continue operations in the presence of space radiation. RISC-V [19] is a open standard instruction set architecture that focuses on reduced instruction set computer (RISC) concepts, which aim to simplify instruction sets while improving execution. RISC-V was chosen as it is usable for most practical computer use cases, and its reduced code size can reduce the binary size of code for use on small computers. Thus, it is a good consideration for use on deeply embedded systems like those we frequently encounter in ICS components.

III. System Design

Here, we define the PCS experiment and the verification tool. These were designed in the scope of a simple PCS which controls the charge of a 9 V Nickel-Metal Hydride (NiMH) rechargeable battery. The physical design of the charging circuit is shown in Figure 2. To allow the circuit to turn the battery charging on and off based on a digital signal from the FPGA, a 2N3904 bipolar junction transistor (BJT) is placed before the 12V wall source and used as a switch. Two versions of the verification tool were designed, one providing only detection and the other providing detection with mitigation.

A. Control System

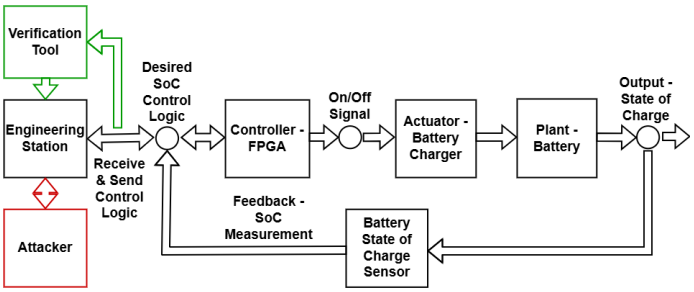

Fig. 1. Control Loop Block Diagram for Battery Charging Process Control System. Components within the main control loop are colored black, whereas external components that communicate with the control system are depicted in color. The legitimate external sources of communication in this diagram are represented with green arrows, whereas adversary sources are represented with red arrows.

The control system is shown as a negative feedback loop in Figure 1's control block diagram. We first describe the components existing within the control loop. The Engineering Station block allows human users to send and receive data from the FPGA through a USB to UART bridge. The FPGA Controller block component outputs an on/off signal for the battery charger based on feedback logic from the battery state of charge (SoC) sensor. The SoC Sensor block measures voltage across the battery to provide a current SoC. The Battery Charger block charges or discharges the Battery block based on feedback logic from the SoC Sensor block. Components outside the control loop includes an Attacker block, which can send and read data from the channel between the engineering station and the FPGA, and a Verification Tool block, which can read from the same channel and send data to the FPGA and engineering station. Figure 2 shows connections where PCS components are connected with wire, as well as the cable connections between the USB ports of the PCS components.

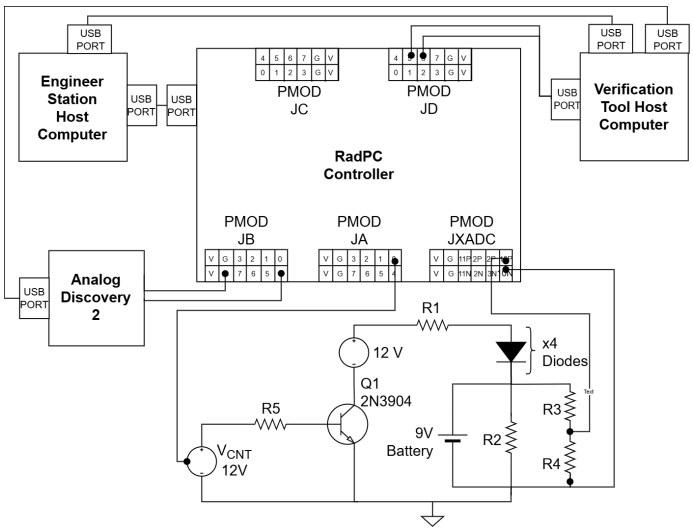

Fig. 2. Wired Connections of the process control system (PCS).

B. FPGA Logic

Basic logic code was provided for the RadPC board by the Resilient Computing lab and altered to fit the needs of the PCS controller. Alterations of UART traffic flow to RadPC varied for the different versions of the verification tool. For the detection version, all signals sent to the FPGA from the engineering station are received by the RX line of the primary UART (UART00) and saved to its RX register as normal. The signals are additionally saved to the secondary UART's (UART01's) TX register to be forwarded to the verification tool through its TX line. For the mitigation version of the tool, all signals sent to the FPGA from the engineering station are received by the RX line of UART00, but not saved to its RX register. The signals are instead saved to UART01's TX register to be forwarded to the verification tool through its TX line. Lastly, the RX register of UART00 saves data received by the RX line of UART01 from the verification tool. This flow of UART traffic simulates the passive serial tap used by the verification tool and lets UART01 to send control logic to be bootloaded onto the board, which only allows control logic saved to UART00's RX register to be bootloaded.

A secondary level of logic code was executed on the RadPC board. This code provides the operating control logic for the PCS based on feedback measurements. Every time heartbeat commands are received from the engineering station or verification tool, it samples a 12-bit resolution binary value

read in from RadPC's Analog-to-Digital-Converter, which reads across the battery's voltage, and the logic converts it to a decimal representation that includes a tens place digit, a ones place digit, and a tenths place digit. Using this decimal representation, it checks the current decimal voltage value against the desired maximum and minimum voltage value limits and turns the control signal off or on accordingly by altering the GPIO value. This results in a system that can alternate back and forth between the minimum and maximum charges to maintain a desired range of operation.

C. Engineering Station

The Engineering Station program allows the engineer to: (1) send in a value that will change the desired range of operation for the charging system or (2) send in a predetermined bitstream to bootload the logic code on the RadPC controller. The program allows the engineer to choose to set either the maximum or the minimum voltage, allowing for precision up to the first decimal point. The maximum voltage is allowed to be up to 9.5V and the minimum voltage must be greater than 1.0V. For any commands or firmware files being sent, the engineering station program tacks on the uncommon byte *'\x7f'*, a hex representation of Delete, to the data, just before the terminating *'\n'*. This allows the verification tool to identify the end of individual transmissions to RadPC. In cases where the program is not being actively used, a heartbeat command is sent out to keep RadPC's logic code from stalling, as RadPC does not currently support timeout logic for receiving data. Additionally, the engineering station is always listening for possible attack reporting from the verification tool. Multithreading was implemented to allow the station process data concurrently.

D. Attacker

Attackers in ICS commonly seek to achieve the Impair Process Control tactic, with the end goal of affecting a system's availability. If the battery charger PCS was part of a larger ICS and the role of the battery was to provide power to an ICS component, attackers could prevent normal performance of the whole ICS by attacking the battery PCS. For example, an attacker may provide unauthorized command messages to the FPGA, such as improper maximum or minimum voltage limits, or their own harmful bitstream, which may instruct the charger signal to be off when it should be on or vice versa. These attacks could result in a dead battery or an overcharged battery, achieving the Impair Process Control tactic.

For ease of use in this system, the attacker is implemented as a separate functionality within the Python engineering station program, to allow access to the communication channel between the station and the RadPC controller. This allows the attacker to send in both malicious commands and malicious logic code. Some malicious commands include providing illogical voltage limits, which are too high, too low, or contrary to each other. The malicious logic code is sent as a compiled firmware file. For these attacks, it is assumed that the attacker may be aware of the proper forms for commands.

E. Verification Tool

The verification tool should run on a separate computer from the engineering station to remain isolated from the network. However, it must be connected to the station to send attack reports. It receives data from RadPC's UART00 port through hardware functionality. This is done by forwarding the data to RadPC's UART01 port, which is connected to the verification tool host by a FTDI Chip C232HD-DDHSP-0 USB to UART cable. The verification host is then able to receive the incoming serial transmission and transmit it to the verification tool. The serial port for communication was configured to the serial data requirements of the RadPC, specifying a baud rate of 115200, 8-bit data, no parity, and no flow control. Prior to being connected to the computer, an Analog Discovery 2 was used to confirm the UART signal being forwarded. When receiving data, the verification tool looks for the terminating bytes *'\x7f'* to separate individual transmissions.

All versions of the verification tool look for expected transmissions to be sent by the engineering station, and validate incoming transmissions against expected transmission formats. Any anomalous transmissions detected are logged and reported on its user interface. The tool focuses on verifying commands and logic code, rather than firmware. The verification tool ensures that voltage limit changes follow the correct format and comply with engineering station rules by tracking and validating the current voltage limits. Any commands or voltage values that don't agree with expected good values are saved to a log file *invalid_command_log.text*. The tool also verifies incoming logic code, by recording incoming bitstreams and performing checksums using SHA-256. The hash of the recorded bitstream is checked against the hash of a predetermined good bitstream, which was compiled from pre-approved logic code. If the incoming hash is not equal to the good hash, it is recorded to a log file *invalid_bin_log.text*. Multi-threading was used to allow the tool to receive and process data while sending back information at the same time.

Mitigation for DoS attacks was also considered as a potential problem with serial communication used by the verification tool. The bounds of the information being sent were considered, and it was determined that the longest a potential set of good data would be was the size of the trusted firmware file in bytes, plus its terminating byte. This size was set as the limit for the serial reading function, restricting it to read only up to that number of bytes before processing the data.

Two versions of the verification tool were implemented. The first version provided only passive detection, monitoring traffic sent to execute in the FPGA and reporting if an attack was detected. The second version provides both detection and prevention, requiring verification of traffic sent to the FPGA before it is executed, as well as reporting attacks back to the engineering station.

1) Version 1: Passive Detection: When unexpected transmissions are received, the verification tool logs the offending transmissions and alerts to possible attacks on it and the engineering station display. Passive detection reports potential

attacks but does not mitigate them, except for DoS prevention. No additional functionality is added beyond the base features. The FPGA also activates an LED to signal an attack, and the tool offers no prevention beyond DoS protection.

2) Version 2: Detection with Mitigation: This version of the verification tool adds prevention functionality. In practice, prevention is a combination of detection and response to prevent the effects of attacks. There are slight differences in the processing of communication for components in this version of the verification tool.

The FPGA cannot receive commands directly from the engineering station and instead receives them from the verification tool. The verification tool receives commands from the engineering station, which it verifies before forwarding them to be executed on the FPGA. To the engineering station however, commands appear to still be directly sent from itself to the FPGA's UART00. As the FPGA no longer looks to the engineering station for its commands, the engineering station program no longer sends a heartbeat signal. Instead, the verification tool provides a heartbeat signal in the absence of other commands to send to the FPGA.

IV. Results

Experimental data yielded promising results, with the verification tool successfully providing prevention and detection for the FPGA controller against unauthorized command message attacks on the serial communication channel between the engineering station and FPGA.

A. System Operation

A baseline of normal operation of the PCS was established by using the probes of a Digilent Analog Discovery 2, which reported voltage values back to a Python program. Maintenance of battery voltage by the PCS is confirmed with voltage charts. Figure 3 displays the battery voltage in blue and the charger control signal voltage in orange. The maximum and minimum voltage limits were changed every 2000 seconds, starting at voltage limits of 6.5 V to 8.0 V, then 6.8 V to 7.7 V, and ending at 7.0 V to 7.5 V. It was also confirmed that the FPGA could successfully bootload trusted firmware when requested by the engineer. Figure 4 shows normal operation of the engineering station, verification tool, and FPGA. The engineer sends in a low voltage limit of 1.3 V and a high voltage limit of 9.3 V, which are validated by the tool. It also shows successful updating of the voltage limits and charger control signal on the FPGA, which displays, in volts, the current voltage, the maximum voltage limit, and the minimum voltage limit, as well as the boolean charger control signal value, on its seven-segment display.

B. Detection

Three attack cases were sent to the FPGA: two out-of-bounds voltage limit commands, one for high and one for low, and a harmful firmware file. In all cases, the verification tool was able to passively detect and log, but not prevent the possible attacks, while displaying a detection message on its

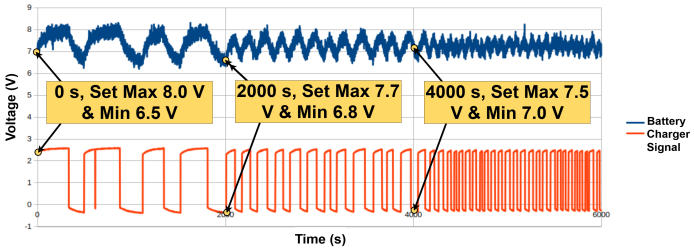

Fig. 3. Maintenance of three battery voltage ranges by the PCS

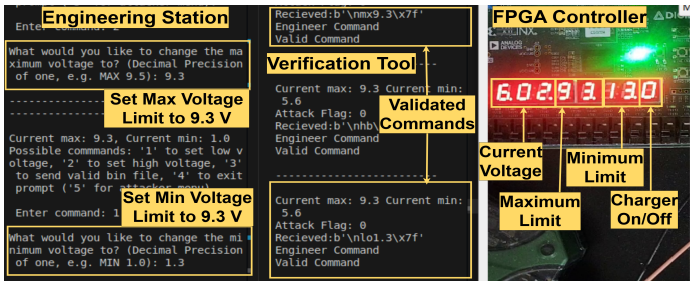

Fig. 4. Normal operation of the PCS experimental testbed.

user interface. Both versions were able to inform the FPGA and the engineering station of the attack. The reporting was done by sending a message to the engineering station and lighting an indicator LED on the FPGA.

Detection was achieved in all three attack cases. An example of the system's response to an out-of-bounds limit attacks is shown in Figure 5, which graphs the battery voltage and the charger control signal. In this case, the maximum voltage limit was 6.5V and the minimum voltage was 5.6V. The attacker sent in an unauthorized command message that set the maximum voltage limit to 9.9V. The system ran as normal for the first 60 seconds, until the attacker sent the harmful voltage limit. Because our system has physical limiters for the maximum voltage of the battery, this attack resulted in the battery never being able to hit that maximum limit and discharge again. In a system without a physical limiter, this may have resulted in an overcharged battery.

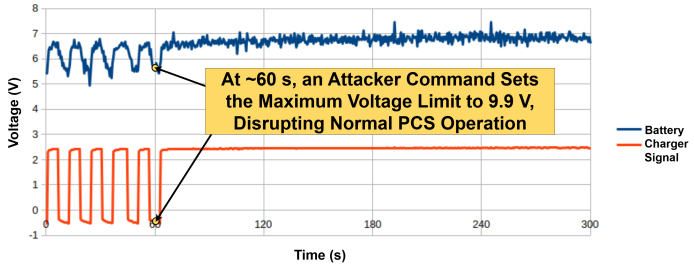

Fig. 5. Failure of battery maintenance after unauthorized command attack.

C. Prevention

Beyond detection, successful results were also achieved for prevention, where the attack needed to be addressed before it reached the FPGA. The second version of the verification tool

was able to prevent all three attacks from reaching the FPGA controller while allowing the system to perform as if the tool was not there. In every case, the tool successfully identified the data as part of an attack and blocked it from being forwarded. The battery's SoC maintenance was monitored and performed exactly as expected. Figure 6 displays the experimental PCS setup after the attacker sends malicious firmware, showing the verification tool's detection, the engineering tool's received attack report, and the FPGA running unaffected firmware. Figure 7 shows the setup after a malicious voltage limit has been sent by the attacker, requesting 9.9V as the maximum limit. It displays the verification tool's detection, the engineering tool's received attack report, and the FPGA's unchanged voltage limits.

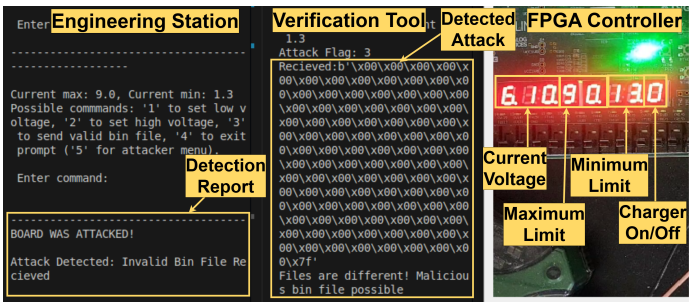

Fig. 6. Detection & mitigation of malicious firmware upload.

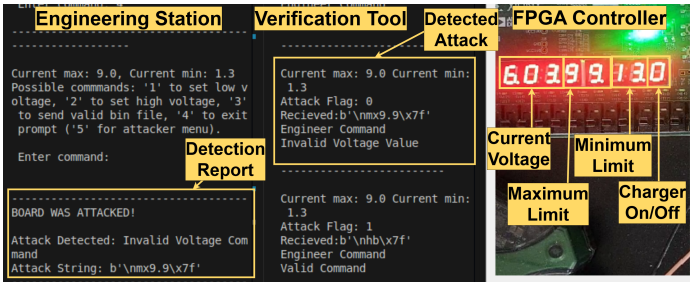

Fig. 7. Detection & mitigation of malicious voltage limit commands.

V. Conclusions

In conclusion, a verification tool was successfully developed which detects attacks using unauthorized command message techniques and provides mitigation against the attacks affecting a RISC-V FPGA controlled PCS. The key contribution is the development of mitigation functionality for the verification tool, which can be applied to other verification tools for PCS controllers that previously focused only on detection. Because the tool only requires the hardware inclusion of a passive serial tap to a system to monitor its traffic, it would be physically easy to incorporate this system into a PCS. However, rerouting communication to pass through the tool before it is forwarded to the controller for execution may prove more difficult.

Acknowledgment

This research was supported in part by NASA under award number 80NSSC23CA147, by Resilient Computing, LLC under subcontract number 4W9082, and by the Software Engineering and Cybersecurity Laboratory (SECL) at Montana State University. Opinions do not necessarily reflect those of NASA or Resilient Computing, LLC.

References

[1] Biçoku, K. Security vulnerabilities in industrial PLC software: A taxonomy and a systematic mapping study. (2021)

[2] Asghar, M., Hu, Q. & Zeadally, S. Cybersecurity in ICSs: Issues, technologies, and challenges. *Comp. Networks*. **165** pp. 106946 (2019)

[3] Gao, W. & Morris, T. On cyber attacks and signature based intrusion detection for MODBUS based industrial control systems. (2014)

[4] Monmasson, E. & Cirstea, M. FPGA design methodology for industrial control systems—A review. *IEEE Transactions On Industrial Electronics*. **54**, 1824-1842 (2007)

[5] Monmasson, E., Idkhajine, L., Cirstea, M., Bahri, I., Tisan, A. & Naouar, M. FPGAs in industrial control applications. *IEEE Transactions On Industrial Informatics*. **7**, 224-243 (2011)

[6] Cárdenas, A., Amin, S., Lin, Z., Huang, Y., Huang, C. & Sastry, S. Attacks against process control systems: Risk assessment, detection, and response. *Proceedings Of The 6th ACM Symposium On Information, Computer And Communications Security*. pp. 355-366 (2011), https://doi.org/10.1145/1966913.1966959

[7] The Mitre Corporation. Common weakness enumeration. (https://cwe.mitre.org/)

[8] Klick, J., Lau, S., Marzin, D., Malchow, J. & Roth, V. Internet-facing PLCs - A new back orifice. *Blackhat USA*. pp. 22-26 (2015)

[9] Byres, E. & Lowe, J. The myths and facts behind cyber security risks for industrial control systems. *Proceedings Of The VDE Kongress*. **116** pp. 213-218 (2004)

[10] Yoo, H. & Ahmed, I. Control logic injection attacks on industrial control systems. *IFIP International Information Security Conference*. (2019), https://api.semanticscholar.org/CorpusID:189926624

[11] Reaves, B. & Morris, T. Discovery. Infiltration, and denial of service in a process control system wireless network. (2009)

[12] The Mitre Corporation. MITRE ATT&CK ICS matrix. (https://attack.mitre.org/matrices/ics/)

[13] Alsabbagh, W. & Langendörfer, P. A flashback on control logic injection attacks against PLCs. *Automation*. **3**, 596-621 (2022)

[14] Wang, Z., Zhang, Y., Chen, Y., Liu, H., Wang, B. & Wang, C. A survey on programmable logic controller vulnerabilities, attacks, detections, and forensics. *Processes*. **11** (2023), https://www.mdpi.com/2227-9717/11/3/918

[15] Mesbah, M., Elsayed, M., Jurcut, A. & Azer, M. Analysis of ICS and SCADA systems attacks using honeypots. *Future Internet*. **15** (2023), https://www.mdpi.com/1999-5903/15/7/241

[16] McMinn, L. & Butts, J. A firmware verification tool for programmable logic controllers. *Critical Infrastructure Protection VI: 6th IFIP WG 11.10 International Conference, ICCIP 2012, Washington, DC, USA, March 19-21, 2012, Revised Selected Papers 6*. pp. 59-69 (2012)

[17] Kim, J., Ko, W. & Kumar, P. Cyber-security with dynamic watermarking for process control systems. *2019 AIChE Annual Meeting*. (2019), https://www.researchgate.net/profile/Jaewon-Kim-5/publication/364657063_Cyber-security_with_dynamic_watermarking_for_process_control_systems/links/6356146f8d4484154a2b4c46/Cyber-security-with-dynamic-watermarking-for-process-control-systems.pdf

[18] Gigamon understanding network TAPS. (https://www.gigamon.com/resources/resource-library/white-paper/understanding-network-taps-first-step-to-visibility.html,2023)

[19] Sharma, S. RISC-V architecture: A comprehensive guide to the open-source ISA. (https://www.wevolver.com/article/risc-v-architecture-a-comprehensive-guide-to-the-open-source-isa,2023)

[20] Resilient computing. Products. (https://resilient-computing.com/products/)

[21] Izurieta, C. & Prouty, M. "Leveraging SecDevOps to Tackle the Technical Debt Associated with Cybersecurity Attack Tactics," *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*, Montreal, QC, Canada, 2019, pp. 33-37, doi: 10.1109/TechDebt.2019.00012.

[22] Dawoud, D. & Dawoud, P. Serial communication protocols and standards. (River Publishers,2022)