Converting an Electric Power Utility Network to Defend Against Crafted Inputs

Michael Millian, Prashant Anantharaman, Sergey Bratus, Sean Smith, Michael Locasto

Chapter 4

CONVERTING AN ELECTRIC POWER UTILITY NETWORK TO DEFEND AGAINST CRAFTED INPUTS

Michael Millian, Prashant Anantharaman, Sergey Bratus, Sean Smith and Michael Locasto

Abstract This chapter proposes a roadmap that employs secure parsers to eliminate the possibility of input-handling vulnerabilities in industrial control systems. Industrial control systems are responsible for maintaining the integrity of power grids. Complex communications networks constitute the backbones of these systems. Communications in industrial control networks must be processed correctly and they should not crash devices or enable attackers to access networked devices. Language-theoretic security is the practice of comprehensive input handling using secure parsers. This chapter demonstrates that the existing collection of secure parsers for industrial control protocols can cover the communications needs of industrial control networks. It discusses the merits of guarding industrial control networks using secure parsers, proposes a triage procedure for implementation and summarizes the security benefits and lessons learned.

Keywords: Industrial control networks, input handling, language-theoretic parsers

1. Introduction

Industrial control systems are increasingly connected to the Internet, either directly or via connections to Internet-connected devices. Industrial control protocols are used to interact with actuators and sensors that help operate important infrastructure assets such as the power grid. The risks posed by the cyber-physical nature of industrial control devices coupled with their network connectivity render the task of securing industrial control network communications a very high priority.

The principal goal of this research is to eliminate input-handling vulnerabilities in industrial control networks. Input-handling vulnerabilities are a class of

vulnerabilities with a long history and many modern examples [5–7, 19]. Previous work has shown that industrial control networks are not immune to these vulnerabilities – between 2013 and 2014 alone, more that 30 input-handling vulnerabilities were discovered in implementations of the DNP3 protocol in industrial control devices [2].

However, eradicating input-handling vulnerabilities presents some challenges. First, while a language-theoretic security approach has been applied to build secure parsers for industrial control protocols, the results have thus far been limited to academic research as opposed to production systems. Indeed, the adoption of these protocol implementations in real systems has been minimal. This research attempts to address the issue by clarifying the benefits and explaining how to use secure parsers.

In particular, a notional architecture of an industrial control network that employs secure parsers is presented. The notional architecture is a general network model that incorporates the components found in industrial control networks. The model loosely maps a real-world network without being tied to a single utility. It is shown that secure parsers cover the communications edges of this model. Indeed, all communications can be guarded using these parsers.

The second challenge is that industrial control networks employ a large variety of protocols. Securing a protocol implementation requires a careful examination of the protocol specification. Device manufacturers often subset or fork existing protocols, resulting in new protocols that must be analyzed thoroughly. Device manufacturers also implement proprietary protocols that significantly complicate protocol analysis. To address this challenge, best practices are proposed for creating new parsers and for subsetting or forking existing protocols.

The third challenge is updating industrial control devices. Because these devices perform vital operations, taking them offline or interrupting their ability to communicate are not viable options. Nevertheless, protocols that contain unsafe features must be made to meet language-theoretic security standards. This is accomplished by employing a triage procedure that enables industrial control devices to continue to operate during the transition.

The proposed approach focuses on subsetting existing industrial control protocols. A subset of a protocol is just the protocol with certain messages excluded. For example, an opcode is removed if the payload for the opcode is unsafe. No features are added to a protocol; rather, unsafe features are removed. As a result, all the industrial control devices that understand a protocol can understand the safe subset of the protocol.

2. Background and Prior Work

Input-handling vulnerabilities have plagued networked systems since their creation. Several well-known bugs – Heartbleed [6], Shellshock [18], Rosetta Flash [17] and Apple's goto bug [5] – involve input-handling vulnerabilities. Any program that accepts inputs must validate the inputs holistically to ensure that they comply with the protocol specifications. An input-handling vulnerability stems from a protocol violation. Typically this is due to a programmer

error, such as forgetting to check a condition. Sometimes, an input-handling vulnerability may arise not from a protocol violation *per se*, but from a deeper flaw in the protocol.

Many bugs have parsing errors at their root. Some work has been done to demonstrate this in specific domains (e.g., USB [12]), but no large-scale effort has been expended to label all parsing bugs as such. Another domain-specific work found more than 30 input-handling vulnerabilities in DNP3 protocol implementations [2]. In fact, only a few implementations were found to be free of vulnerabilities. They were immune because they employed very constrained subsets of DNP3 that significantly reduced their attack surfaces. This result supports the position that protocol subsetting can eliminate input-handling vulnerabilities.

The impacts of input-handling vulnerabilities range from device crashes to attackers gaining access to networks. Heartbleed enabled attackers to exfiltrate data; Apple's goto bug allowed man-in-the-middle attacks; Shellshock gave attackers direct access to systems. Given the ubiquity of input-handling vulnerabilities, it is imprudent to believe that industrial control networks, protocols and devices are immune to input-handling vulnerabilities. Device crashes may pose mild threats in information technology environments. Not so in industrial control networks where device crashes can disrupt critical infrastructure assets. Without question, it is imperative to ensure that industrial control networks are rendered immune to input-handling vulnerabilities.

2.1 Language-Theoretic Security

Language-theoretic security postulates that all inputs received by a program must be validated in their entirety by a parser developed from a formal grammar before any and all uses of the inputs by program internals. A program that receives an unanticipated input could be driven to a state that its developers did not anticipate. A language-theoretic-security-hardened parser ensures that input validation code is explicitly and clearly based on a formal grammar, that the validation code is logically separate from the code that processes the inputs, and that a program can never operate on inputs that have not been verified exhaustively. There is no room for inputs that are "almost correct" because these inputs cannot be meaningfully distinct from malicious crafted inputs.

In this work, a language denotes a set of allowed inputs. A protocol is specified using a grammar, a set of production rules that create the inputs that constitute the language. A parser is an implementation of the protocol in code.

A parser combinator is employed to construct a parser in a manner that clearly and explicitly represents the protocol. It is a toolkit or framework that produces code that visually resembles the formal grammar instead of multiple if-statements that check conditions. Parser combinators dramatically reduce the possibility of programmer errors (e.g., forgetting to check a condition).

In this research, the Hammer parser combinator tool [16] was used to implement parsers. Hammer was developed with a security focus, which is measured against the Chomsky hierarchy that classifies languages according to their com-

plexity [3]. The language classes range from regular expressions that are recognized/generated by finite state automata to recursively-enumerable languages that are recognized/generated by Turing machines. Note that `regex` tools in Perl, Python and JavaScript are actually more complex than regular expressions. Grammars that are deterministic-context-free or simpler are considered safe; this limit is discussed by Momot et al [14]. Parser combinator toolkits are useful for building parsers for binary protocols and for specifying byte-level constraints about languages. They also provide a way to represent top-down grammars. The Hammer tool parses inputs into abstract syntax trees.

2.2 Industrial Control Systems Security

Industrial control systems differ from traditional information technology systems and, consequently, require different security approaches. Industrial control networks interact with physical devices such as sensors and actuators using short messages with extremely low latency. In contrast, information technology networks transfer data using much larger packets with longer latency. Additionally, industrial control networks are typically deeper than information technology networks.

Much work has focused on ensuring the security of industrial control systems and networks. The prevailing security paradigm is defense-in-depth where security features and tools are added at each layer of the system or network to provide compound protection against external threats [8].

This research leveraging secure parsers complements the defense-in-depth model. Industrial control systems were originally designed for isolated, local use of analog equipment. Over the years, industrial control networks have been upgraded to support automation and remote access. New connections and capabilities pose new threats that industrial control systems were not designed to handle. The proposed approach is fundamentally about ensuring message security during the protocol design phase. It may require modifications of existing protocols if they do not meet the complexity-limitation requirements for security. Because protocol complexity is restricted rather than increased, the proposed approach dovetails with current defense-in-depth strategies. Existing security measures do not have to be replaced, they can work in concert with the proposed approach. Indeed, the approach can be used at every level of the defense-in-depth model to increase the security claims at a given level and between levels.

3. Notional Architecture

This section presents a notional architecture for a language-theoretic-security-compliant industrial control system at a utility. The notional architecture contains the general elements and components of a real-world network in an abstract representation that is not tied to a single utility.

First, the types of devices encountered in an electricity utility are specified, including the devices that are expected to communicate directly and the pro-

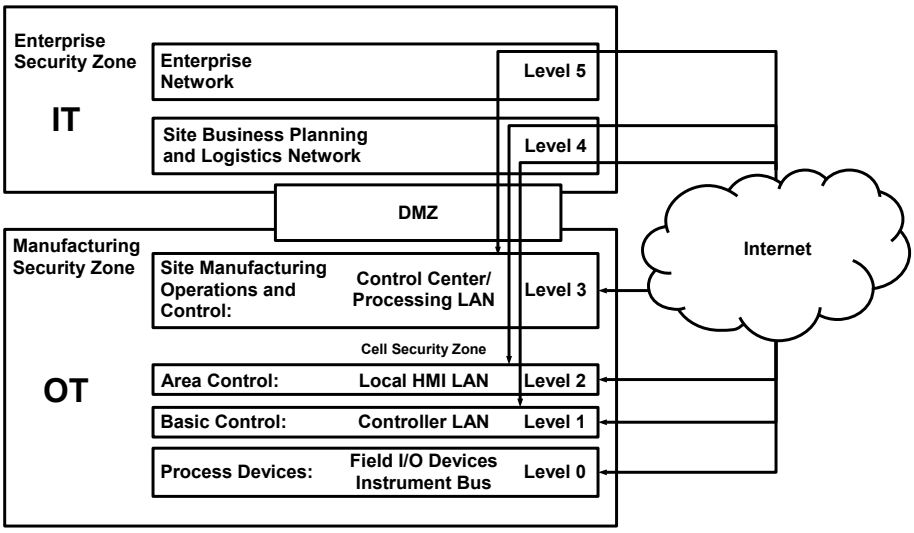

Figure 1. Purdue model architecture (adapted from [11]).

tocols they use for communications. Next, it is shown how secure parsers can provide coverage of the communication needs in the model such that all the communications can be guarded by the parsers.

Figure 1 shows the Purdue model architecture [11], which is annotated with the various paths that an attacker could use to access the industrial control network. The Purdue model has six levels: (i) enterprise network (level 5); (ii) business planning and logistics network (level 4); (iii) site manufacturing operations and control (level 3); (iv) area control (level 2); (v) basic control (level 1); and (vi) process devices (level 0). The levels are divided into several zones, where a zone corresponds to large-scale interconnectivity. Implementing clear boundaries between the zones is a best practice for enforcing multiple layers of defense.

This research focuses mainly on levels 2 through 0, which is called the cell security zone or the SCADA (supervisory control and data acquisition) zone. This zone comprises devices found in an electricity substation as well as devices that are directly involved in managing the substation. Level 2 is concerned with monitoring and controlling physical devices. The devices in this level include control center operation workstations, human-machine interfaces (HMIs), engineering workstations, security event collectors, operations alarm systems, communications front ends, data historians and network/application administrator workstations. Level 1 is concerned with sensing and manipulating physical devices. Devices in this level include dedicated operator workstations, programmable logic controllers (PLCs), control processors, programmable relays, remote terminal units (RTUs) and process-specific microcontrollers. Level 0 contains physical devices such as sensors, actuators, motors, process-specific automation machinery and field instrumentation devices [13].

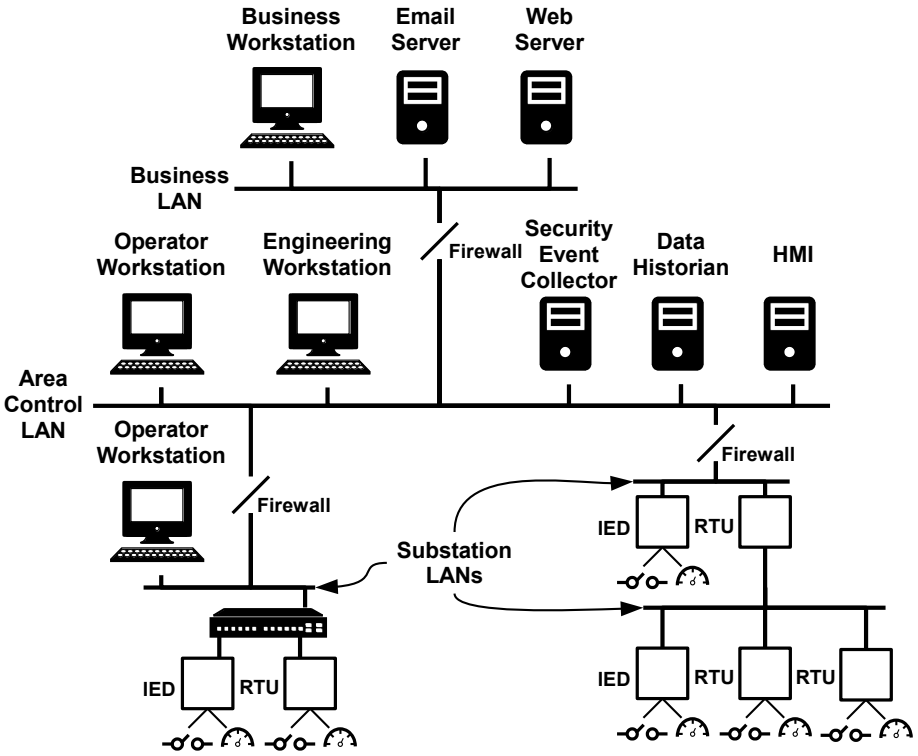

Figure 2. Notional architecture.

During the research, several real and development networks in the SCADA zone were examined. These networks are considered critical infrastructure assets, so detailed information about their network topologies cannot be published. In any case, large variances were observed in device types and layouts from substation to substation. Thus, it has been possible to develop a notional architecture that is not based on a single utility.

Figure 2 shows the notional architecture that is derived from previous models [10, 11, 20] as well as from real and development networks. The architecture is designed to be as generic as possible while still maintaining its utility.

The generic architecture enables the expression of coverage by focusing on a small set of protocols used at the edges (e.g., RTU-RTU, RTU-HMI and control-center-substation) without too much concern about the actual device models. While there are many protocols for a given edge (e.g., RTU-HMI), the notion of coverage means that at least one of the protocols is handled and, therefore, it is feasible to add the protection. Vendor-specific protocols exist, but many vendors provide devices that can handle multiple protocols (i.e., standard languages), so this concept of coverage is practical. Using popular protocols allows easier integration in existing ecosystems. The set of popular

protocols considered in this work was determined using informal as well as published surveys [10].

4. Analysis

This section discusses the coverage provided by secure parsers and their benefits and trade-offs.

4.1 Protocol Coverage

At this time, the authors of this chapter have implemented secure input handling for the DNP3, MMS (Manufacturing Message Specification), Modbus, IEC 61850-8-1 (GOOSE), IEEE C37.118 [1], SEL Fast Message, HTTP and Telnet protocols. This section discusses how this selection of protocols offers adequate coverage of industrial control network communications needs.

DNP3, MMS and Modbus are the *de facto* industry communications standards. These protocols allow for communications between the human-machine interface of a master station and remote terminal units, programmable logic controllers and intelligent electronic devices (IEDs). SEL Fast Message is a vendor-specific protocol for SEL devices that handles much of the same communications. GOOSE is used to broadcast or multicast event data fast and reliably in substations; GOOSE messages have a maximum latency of 4 ms. IEEE C37.118 is used to transmit phasor data over wide-area networks. HTTP and Telnet are used for communications between workstations and for configuring devices.

To reiterate, communications from level 2 downwards are covered by the popular DNP3, MMS and Modbus protocols as well as by the vendor-specific SEL Fast Message protocol. Level 1 substation/physical devices are covered by the GOOSE and IEEE C37.118 protocols. Finally, workstation-workstation communications are covered by HTTP and Telnet. By implementing parsers for these industrial control system protocols, a large degree of protection is provided for the majority of low-level (Purdue model) operational technology traffic in most industrial control networks. In particular, secure parsing is provided for the protocols that are responsible for manipulating physical devices, a task that has very high priority.

4.2 Benefits

The major benefit in using a parser combinator tool is the possibility of producing provably-correct code. A programmer implementing a parser should not have to worry about the correctness of a combinator just like a programmer typically does not worry about the correctness of a compiler.

Proofs of correctness of the combinators in Hammer remain to be done. However, as far as this work is concerned, only two possibilities exist – either every combinator is correct or there are bugs in one or more combinators. If a bug is found in a combinator, it can be corrected without having to rewrite

the parsers built using the combinator (although they would have to be recompiled using the updated combinator library). This is because each combinator performs a function that is fully understood under formal computational theory, so the function signature of each combinator is set, only the internals may change. After this proof work is complete, every secure parser built with these combinators immediately derives the full benefits of provable security.

The other benefit of the parser combinator is that it reduces the effort undertaken by the programmer who works on a parser. A key observation is that attention should be paid to match the complexity of the parser to the complexity of the protocol and this attention must be baked in during development and implementation. Traditional parser programming involves a number of if-statements that check conditions. It is easy to miss a condition – as in Heartbleed and Apple's goto bug. However, even when a fix is provided, it is still difficult to compare the new parser against the protocol and demonstrate that they match completely [14].

Using a parser combinator simplifies the comparison task, and thus decreases the likelihood of errors, and simplifies the implementation of fixes should errors occur. A parser combinator tool produces code that visually matches the structure of the grammar, rendering the verification of equality trivial. Furthermore, a tool like Hammer does not have combinators that would allow the programming of complex constructions such as Turing machines. If a programmer cannot implement a protocol feature using a parser combinator, then it is an indication that, perhaps, the feature is unsafe and that a subset of the protocol without the feature should be used. Ideally, this practice of subsetting protocols leads to protocols being designed without unsafe features.

The end result of using a parser combinator is a parser that only accepts messages in the protocol specification. The task of implementing protocols safely can thus be broken down to designing protocols and designing parser combinator tools.

Previous work with DNP3 has demonstrated the practicality of the approach for industrial control system protocols [2]. Implementing the DNP3 parser revealed that the specification mentions that the transport layer payload contains at least one byte, but that a zero-length application layer message would cause unhandled exceptions in certain implementations. Each protocol that was implemented contained such features, which were usually handled by if-checks in the parser. The language-theoretic security approach to parser construction considers such packet structure features when writing the parser, significantly decreasing the likelihood that a check is omitted.

4.3 Trade-Offs

The major trade-off that comes with a language-theoretic-security-based parser is the need to subset a protocol when inherently unsafe features are discovered. The cost associated with this modification is the possibility that network devices regularly or occasionally transmit messages using the unsafe features. Experience has shown that such messages are a small, if any, fraction

of actual traffic. However, there are situations where the trade-offs could be greater depending on the use cases.

Maintaining unsafe protocol features is dangerous. Unsafe features most often relate to message format as opposed to message content, especially in the case of industrial control networks. Of course, it may be necessary to use certain kinds of messages and there are always development costs involved in making changes. However, the real costs arise from the risks of an attacker crashing devices, exfiltrating data or seizing control of devices.

5. Triage Procedure

This section discusses the roadmap for incorporating language-theoretic-security-hardened parsers in industrial control networks so that electric utilities may realize the security benefits. The roadmap involves a three-step plan for engaging with utilities and vendors. The first step is to develop the secure parsers and incorporate them on a per-device basis in a laboratory setting. The second step is to create a virtual substation in the laboratory. The third step is to work with utilities and vendors to replace parser implementations in device firmware via their product refresh cycles.

5.1 Protocols and Devices

The first step is to write and test parsers for industrial control protocols. At this time, parsers have been implemented for eight protocols: DNP3, MMS, Modbus, IEC 61850-8-1 (GOOSE), IEEE C37.118, SEL Fast Message, HTTP and Telnet. Accomplishing this task in full requires the complete list of protocols used by utilities.

For each protocol of interest, the protocol specification is obtained and a secure parser is written and tested. At first, parser testing is performed using a bump-in-the-wire implementation. A key requirement is to ensure that the messages passed by each parser allow normal device operations.

However, some inherent difficulties exist. Obtaining documentation for industrial control protocols can be difficult. Many protocol specifications have to be purchased – their costs range from a few hundred dollars to several thousand dollars. A protocol specification may not cover the complete protocol; some protocols import other protocols to leverage existing work and offset the design burden (e.g., data encoding formats and protocol data units). The specifications of these embedded protocols might also have to be purchased.

Another challenge is that there is neither uniformity nor good practice when it comes to describing a protocol. Some specifications are all prose and the developer must create the protocol grammars. Even worse are situations where the specifications include state machines or grammars, but their functionalities do not match the prose [2]. This causes divergent implementations depending on how closely the developer reads the documentation. Until protocol specifications improve, close readings of the available specification are essential.

When a protocol has unsafe features, the correct subset of the protocol has to be determined before a parser can be developed. An example of an unsafe feature is nested length fields. Inclusion of nested length fields requires inner length agreement (e.g., the inner length should not exceed the outer length). This constraint cannot be described purely in terms of packet structure using a context-free language because it requires complete parsing of the outer and inner fields to determine agreement. If adherence to the protocol is not maintained by the packet structure of the packet, but left to after-the-fact checks, it is common for one or more checks to be forgotten [5, 6].

After the parsers are written and tested as bump-in-the-wire implementations to ensure that devices can operate as required, the native parsers must be replaced with security parsers on a per-device basis. This action is required because industrial control protocols have maximum latency requirements and parsing every message twice can be expensive. Incorporating a secure parser as the native parser provides security benefits beyond traditional intrusion detection. Intrusion detection systems have difficulty providing insights into encrypted messages, but every message must be decrypted and parsed. Thus, incorporating secure parsers as the only native parsers in a device adds precise security properties.

5.2 Virtual Substation

After implementing the full range of parsers for industrial control protocols and incorporating them in devices, the next step is to create and operate a virtual substation with hardened devices in a laboratory environment. Before deploying the parsers in real critical infrastructure assets, it is necessary to guarantee that the individual devices and the consequences on a network with these devices operating under normal and stress conditions are well understood.

The virtual substation would be a fully-functioning substation that runs in parallel with real-world networks but does not affect the operation of the networks. It could accept real-time data or replayed captures and would operate real or simulated devices. Developers would conduct analyses to ensure correct operations of the virtual substation with no risk to the larger network.

This step can motivate hardened devices via the list of vulnerabilities that the parsers would prevent. It would also demonstrate to utilities and vendors that hardened devices are viable in operational environments.

5.3 Deployment

The final step involves field deployments of the hardened devices. This step must address all the real-world constraints that were not considered in the previous two steps. In particular, industrial control networks are slow to incorporate changes and the changes made may be expected to last for decades. Nevertheless, existing refresh cycles can be leveraged to push language-theoretic-security-based parsers to devices in the form of firmware updates.

5.4 Current Status

The project is currently in the first step in the roadmap. Eight protocol parsers have been developed and tested as bump-in-the-wire implementations in confidential field trials [9]. Parsers for a proprietary JSON-based protocol have also been incorporated in General Electric devices [15].

The parsers will be made available as open source or under similar licenses. Instead of each developer having to implement a parser to read input in a specific format, the project goal is to create a standard library for each parser. It would be very useful if the crypto-idiom "don't roll your own crypto" could be extended to parsers – "don't roll your own parsers." The number of vulnerabilities that have arisen from poor parser code supports this point of view.

Code for the DNP3 and C37.118 parsers is available on GitHub [4]. The remaining parsers will be added to the master repository in the near future.

6. Conclusions

This chapter has presented the design and implementation of an industrial control network that exclusively employs language-theoretic-security-compliant parser implementations. The collection of secure parsers for industrial control protocols cover the communications needs of industrial control networks while eliminating input-handling vulnerabilities that could be exploited by denial-of-service and remote code execution attacks. The roadmap described in this chapter describes how electric utilities could deploy the security-hardened parsers in their industrial control networks via standard product refresh cycles, reaping the associated security benefits in a cost-effective manner.

Any opinions, findings, conclusions or recommendations expressed in this chapter are those of the authors and do not necessarily reflect the views of the U.S. Air Force, DARPA, United States Government or any agency thereof.

Acknowledgement

This research was supported by the U.S. Air Force and DARPA under Contract No. FA8750-16-C-0179 and by the U.S. Department of Homeland Security under Award No. DE-OE0000780.

References

[1] P. Anantharaman, K. Palani, R. Brantley, G. Brown, S. Bratus and S. Smith, PhasorSec: Protocol security filters for wide-area measurement systems, *Proceedings of the IEEE International Conference on Communications, Control and Computing Technologies for Smart Grids*, 2018.

[2] S. Bratus, A. Crain, S. Hallberg, D. Hirsch, M. Patterson, M. Koo and S. Smith, Implementing a vertically-hardened DNP3 control stack for power applications, *Proceedings of the Second Annual Industrial Control System Security Workshop*, pp. 45–53, 2016.

[3] N. Chomsky, Three models for the description of language, *IRE Transactions on Information Theory*, vol. 2(3), pp. 113–124, 1956.

[4] Dartmouth's PKI/Trust Lab, C37.118PMU and dnp3, GitHub (`github.com/Dartmouth-Trustlab`), 2018.

[5] P. Ducklin, Anatomy of a "goto fail" – Apple's SSL bug explained, plus an unofficial patch for OS X! *Naked Security* (`nakedsecurity.sophos.com/2014/02/24/anatomy-of-a-goto-fail-apples-ssl-bug-explained-plus-an-unofficial-patch`), February 24, 2014.

[6] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson and M. Bailey, The matter of Heartbleed, *Proceedings of the Internet Measurement Conference*, pp. 475–488, 2014.

[7] J. Freeman, Exploit (& fix) Android "master key," *The Realm of the Avatar Blog* (`www.saurik.com/id/17`), 2013.

[8] B. Galloway and G. Hancke, Introduction to industrial control networks, *IEEE Communications Surveys and Tutorials*, vol. 15(2), pp. 860–880, 2013.

[9] L. Hay Newman, The Hail Mary plan to restart a hacked US electric grid, *Wired*, November 14, 2018.

[10] C. Hurd and M. McCarty, A Survey of Security Tools for the Industrial Control System Environment, INL/EXT-17-42229, Revision 1, Idaho National Laboratory, Idaho Falls, Idaho, 2017.

[11] Industrial Control Systems Cyber Emergency Response Team (ICS-CERT), Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies, Idaho Falls, Idaho, 2016.

[12] P. Johnson, S. Bratus and S. Smith, Protecting against malicious bits on the wire: Automatically generating a USB protocol parser for a production kernel, *Proceedings of the Thirty-Third Annual Computer Security Applications Conference*, pp. 528–541, 2017.

[13] R. Lee, Detecting the Siemens S7 worm and similar capabilities, *SANS Industrial Control Systems Security Blog* (`blogs.sans.org/industrial-control-systems/2016/05`), May 8, 2016.

[14] F. Momot, S. Bratus, S. Hallberg and M. Patterson, The seven turrets of Babel: A taxonomy of LangSec errors and how to expunge them, *Proceedings of the IEEE Cybersecurity Development Conference*, pp. 45–52, 2016.

[15] Office of Cybersecurity, Energy Security and Emergency Response, From Innovation to Practice: Re-Designing Energy Delivery Systems to Survive Cyber Attacks, U.S. Department of Energy, Washington, DC (`www.energy.gov/sites/prod/files/2018/09/f55/CEDS%20From%20Innovation%20to%20Practice%20FINAL_0.pdf`), July 2018.

[16] M. Patterson, Parser combinations for binary formats, in C; Yes, in C; What? Don't look at me like that, GitHub (`github.com/UpstandingHackers/hammer`), 2017.

[17] M. Spagnuolo, Abusing JSONP with Rosetta Flash, *Michele Spagnuolo Blog* (`miki.it/blog/2014/7/8/abusing-jsonp-with-rosetta-flash`), July 8, 2014.

[18] Symantec Security Response, ShellShock: All you need to know about the Bash Bug vulnerability, *Symantec Security Response Blog* (`www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability`), September 25, 2014.

[19] K. Torpey, The DAO disaster illustrates differing philosophies in Bitcoin and Ethereum, *CoinGecko Buzz* (`www.coingecko.com/buzz/dao-disaster-differing-philosophies-bitcoin-ethereum`), July 4, 2016.

[20] C. Veitch, J. Henry, B. Richardson and D. Hart, Microgrid Cyber Security Reference Architecture, Version 1.0, Sandia Report SAND2013-5472, Sandia National Laboratories, Albuquerque, New Mexico, 2013.