Formal Security Assessment of Modbus Protocol

Roberto Nardone
DIETI,
Univ. di Napoli Federico II
Naples, Italy
roberto.nardone@unina.it

Ricardo J. Rodríguez
DIIS, Universidad de Zaragoza, Zaragoza, Spain
Dip. di Matematica e Fisica,
Seconda Università di Napoli, Caserta, Italy
rjrodriguez@ieee.org

Stefano Marrone
Dip. di Matematica e Fisica,
Seconda Università di Napoli, Caserta, Italy
stefano.marrone@unina2.it

***Abstract*—Critical infrastructures as water treatment, power distribution, or telecommunications, provide daily services essential to our lifestyle. Any service discontinuity can have a high impact into our society and even into our safety. Thus, security of these systems against intentional threats must be guaranteed. However, many of these systems are based on protocols initially designed to operate on closed, unroutable networks, making them an *easy* target for cybercriminals. In this regard, Modbus is a widely adopted protocol in control systems. Modbus protocol, however, lacks for security properties and is vulnerable to plenty of attacks (as spoofing, flooding, or replay, to name a few). In this paper, we propose a formal modeling of Modbus protocol using an extension of hierarchical state-machines that is automatically transformed to a Promela model. This model allows us to find counterexamples of security properties by model-checking. In particular, the original contribution of this paper is the formal demonstration of the existence of man-in-the-middle attacks in Modbus-based systems. Our approach also allows to formally evaluate security properties in future extensions of Modbus protocols.**

***Keywords*-SCADA control systems, Dynamic State Machines, Model checking, Cyber-Physical Security, Modbus**

I. Introduction

Modern society relies on large, complex heterogeneous systems that provide services as water treatment, power distribution, or logistics, to name a few. Those infrastructures upon which our economy, industry, and lifestyle depend are normally named *critical infrastructures* (CIs) [1]. They become critical since a loss of continuous operation of any of these services can impact our society and our safety.

These disruptions can have different origins, from man-made to unexpected acts of nature (e.g., earthquakes, powerful storms, or tsunamis). These acts of nature are intrinsically unexpected and unintended. On the contrary, man-made disruptions are unintended, when they are caused by lack of qualified and trained personnel, or intended, when they are intentional as acts of terrorism or sabotage.

CIs heavily rely on automated and distributed control systems that adopted information and communication technology solutions to support operation and monitoring of industrial and critical processes. In industry, these control systems are known as Supervisory Control and Data Acquisition (SCADA) systems. Many of these industrial systems, built using legacy devices and in some cases running legacy protocols, have evolved to operate in routable networks that directly expose a new attack surface for criminals [2].

As stated in [3], asset owners and industry partners from different critical sectors (such as chemical, health-care, transportation, or food and agriculture) reported thousands of cyber incidents during 2012. According to numerous industry reports, there is an increasing trend of threats to these critical infrastructures [4]. Apart from the well-known *Stuxnet* worm case in 2011 that was specially designed to exploit Siemens PLCs in SCADA networks affecting Iranian nuclear facilities [5], [6], other threats have become publicly known to target energy companies, research centers, or banking services, as *GhostNet*, *Flame*, *DarkSeoul*, or *DragonFly*, to name a few [7].

A well-established protocol in industrial control systems is Modbus [8]. Modbus protocol is at the top layer of the OSI model. Initially designed by Modicon in 1979 as a simple way for communicating control data between controllers and sensors using an RS232 port, it was widely adopted and deployed as a *de facto* standard in the industrial automation field. Modbus is a request/response (master/slave) protocol designed to operate over different links, such as serial buses, routable networks over TCP/IP, or intercommunicated buses throughout an RS-485 communication link: a unique address is assigned to each device intended to communicate using Modbus. Modbus commands are divided into read and write commands, device identification and diagnostic. A Modbus command message contains destination address, code function, and a sub-code function (it may be none). Normally, a device acting as master initiates a command containing the destination device, which replies after performing the function (or sub-function) requested by the master.

Modbus lacks for any security attribute (recall confidentiality, integrity, and availability): no authentication, no encryption, and no integrity at all are the main security concerns of this protocol. This lack of security awareness can be motivated since Modbus protocol was running in systems unreachable from external networks – surprisingly, insider attackers (e.g., disgruntled employees) were not either considered.

In this paper, we assess a security analysis of the Modbus protocol using formal models. In particular, we use a

high-level formalism based on hierarchical state machines, named Dynamic StaTe Machine (DSTM)[1], to model the Modbus protocol. Through an automatic transformation [9], a Promela model is obtained from the Modbus DSTM model to verify its security properties by model checking techniques. Our modeling allows to formally verify the weaknesses of Modbus protocol, as well as to provide a formal framework to evaluate solutions against security concerns. In particular, we demonstrate the existence of man-in-the-middle attacks in Modbus-based systems.

This paper is organized as follows. Section II reviews the related work. Section III introduces our approach to evaluate the Modbus protocol. The formal assessment of Modbus protocol is shown in Section IV. Section V concludes the paper and outlines future work.

II. Related Work

Modbus protocol and its security have been largely studied in the literature. The seminal work of Huitsing et al. [10] grouped the attacks on the Modbus serial and TCP protocols into three different categories: attacks to the Modbus protocol specification, attacks to vendor implementations of the Modbus protocol, and attacks to control system assets (i.e., information technology, networking, or telecommunications assets). Almost 60 attack instances were identified and classified according to this taxonomy, including spoofing, replay, and flooding attacks.

The risk of malicious traffic to Modbus/TCP protocol was demonstrated in [11]. By measuring Round-Trip Time and TCP Time-sequence Graph, they evaluated the impact of injection of malicious traffic. As expected, availability of the system was compromised. In [12], the authors confirmed that Modbus protocol was vulnerable to flooding attacks. They also proposed two algorithms to detect these attacks, based on anomaly and signature detection. Similarly, in [13] the Modbus/TCP communication channel between assets was modeled using deterministic finite automaton, later used to detect intrusions by network packet analysis.

Regarding the addition of security to Modbus protocol, Fovino et al. [14] proposed an extension of Modbus protocol to support authentication, non-repudiation, and replay protection. Although the security properties were fulfilled, their solution implicitly introduced overhead in terms of performance and packet size that may impact performance needed in real-time systems.

Security analysis of complex systems by means of formal methods and model checking in particular is not new in the scientific landscape. This paper could not, for sake of the space, cope with a complete survey of this topic: above all, we report the seminal work of Sheyner [15] with the definition of scenario and attack graph and their applications in network security [16]. The original contribution of this paper relies on the application of these techniques to the Modbus protocol in a technically sound and high usable model-driven methodology.

To the best of our knowledge, our work is the first one to formally verify the security properties of Modbus protocol, pinpointing its major flaws. Furthermore, we propose a modeling approach suitable for evaluating security solutions of the Modbus protocol and related applications.

[1]In this paper, we use DSTM interchangeably as singular and plural acronym.

III. General Approach

This section shows our approach for the formal assessment of the Modbus protocol. We exploit the well-known formal technique of model checking to generate counterexamples with respect to the security property to verify. In order to avoid the error-prone activity of modeling the Modbus protocol into the *low-level* syntax and formalism adopted by a model checker, a *high-level* state-based formalism (namely, DSTM) is used to describe the protocol as a set of state machines. Then, a model transformation is employed to derive automatically the notation (i.e., Promela) analyzable by the model checker (i.e., SPIN) from the source model. In the following, we describe the steps of our approach and give the basics of DSTM formalism, also discussing the motivations of its choice.

A. Overview

As shown in Fig. 1, the adopted approach starts from a high-level model of the protocol expressed in terms of state machines. Among the different existing state-based formalisms, we choose Dynamic StaTe Machine (DSTM), which is a recent extension of hierarchical state machines synthetically described in Section III-B. Through an automatic transformation [9] a Promela model is derived by the Modbus DSTM model. Promela is a textual verification modeling language introduced by Holzmann [17]. This formalism considers a model as a set of concurrent processes that exchange messages over a set of channels or through shared variables. The Promela model is later analyzed by the SPIN model checker [18].

According to our approach, the Promela model of the Modbus protocol is enriched with temporal logic formulas derived from the set of requirements to assess. This enriched model is then analyzed by the SPIN model checker to generate a counterexample if the property to evaluate is not fulfilled. In this paper, we consider security properties which should ideally be true on the protocol: in this case, a counterexample represents the sequence of messages exploitable to compromise security.

The main reasons of choosing the DSTM formalism as the high-level representation of Modbus become clearer after the introduction of the formalism basics. These reasons are summarized as: (a) DSTM is an extension of state machines with a formal (textual and graphical) syntax and

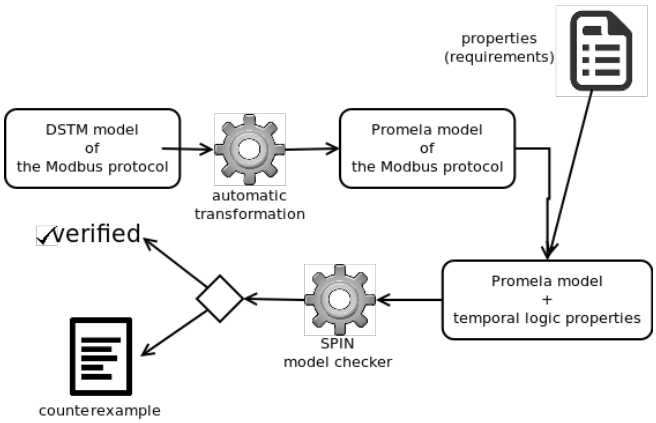

Figure 1. Approach overview.

semantics of both structural elements and annotations over transitions (i.e., triggers, conditions, and actions); (b) DSTM introduces *external channels* on which new messages are non-deterministically generated by the environment where the modeled system operates; and (c) an automatic transformation from DSTM to Promela exists that can be used to derive a complete and complex analyzable model of the Modbus protocol.

Specifically, the DSTM feature described in the item (b) is very useful in this case since it is not necessary to specify a priori the attack vector that is automatically found by the model checker.

B. The Dynamic StaTe Machines (DSTM) formalism

DSTM [19] are an extension of hierarchical state machines that add a novel semantics of fork and join syntactical constructs. This language allows for dynamic and recursive instantiation of machines, for preemptive termination, and for passing parameters at instantiation time. This new formalism explicitly offers to modelers the possibility of instantiating parametric machines by means of the *box* syntactical construct; it also removes the constraint, implicitly intended in many languages (e.g., UML), that the branches of a control flow exiting a fork must always be merged by a join element.

A DSTM model is a collection of parametric machines, channels (internal and external) and variables. Additionally, the definition of own datatypes is also allowed in DSTM, starting from *basic type*s (i.e., *Boolean*, *Integer* and *Enumerations*). Basic types can be composed to constitute *compound type*s and *multi-type*s. Compound types are structured types similar to records of basic types; multi-types, instead, are collections of basic and compound types. Channels can convey messages of any defined type (both a basic, compound, and multi-type). A DSTM channel can be either *internal* or *external*. Internal channels are entirely managed by the specified state machines. They have a buffer of a predefined length and transitions can fire according to the presence of messages over a channel, read its content, or remove a message from a channel. On the contrary, external channels do not have buffers: a message present over them is valid during a single step and can be only read without removal.

According to the DSTM semantics, at the starting of each step, a new message is present over external channels: if a message was generated during the previous step then this message is present in the current step; otherwise, a new message is non-deterministically generated over the possibilities given by the datatype.

A single machine is composed of vertices and transitions. Different kinds of vertices may be included in a machine. Nodes represent the possible control states. An initial node is also present in each machine, corresponding to the default entry. Moreover, a machine may contain additional entering nodes and different exiting nodes, representing different entering and exiting conditions. Boxes represent single or multiple machine instantiation (parallel procedure calls). Transitions are decorated with triggers, conditions, and actions, which are formally defined over a specific syntax. A transition entering a box models the instantiation of the machine(s) associated with the box and specifies the values of its parameters, while a transition leaving a box corresponds to a return from that machine(s).

Parallel behavior can be modeled either by associating multiple machines with a single box, or by explicitly splitting and merging the control flow using the fork and join syntactical constructs. The transitions exiting a fork can enter both a box or a node: if no node is entered after a fork, a synchronous instantiation is performed (i.e., the machine is suspended waiting for the termination of the instantiated ones); otherwise an asynchronous instantiation is performed (i.e., the machine continues its execution). Join nodes allow for merging of multiple control flows from concurrently executing processes. There is no constraint such that a fork must be followed by a join. In general, a join either synchronizes the termination of the involved processes or forces their termination (if a preemptive transition enters the join node). Note that asynchronous forks, occurring within loops, allow for the dynamic instantiation of processes.

IV. Security Analysis of Modbus

In this section, we first define a reference scenario for the application of our approach to the security analysis of Modbus. Then, we explain all the steps from the Modbus modeling to discussion of obtained results.

A. Reference Scenario

Fig. 2 shows the scenario where we suppose to have a simple communication between Alice (a Modbus device acting as a master) and Bob (other Modbus device acting as a slave) over an open TCP/IP network. Bob monitors a physical binary process capturing the value of a variable

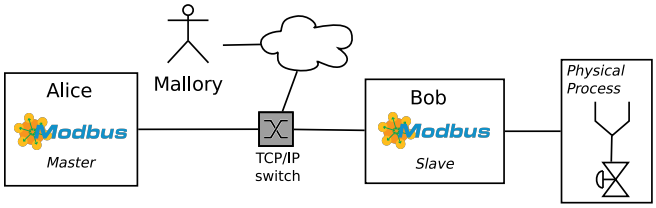

Figure 2. Reference scenario.

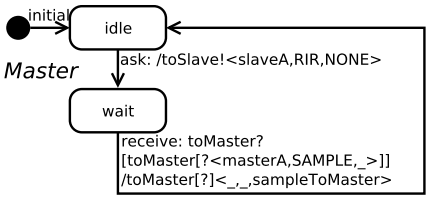

Figure 3. DSTM model of the Modbus master.

within the physical process upon Alice's request. Then, the value of this variable is sent over the network via Modbus protocol. This communication is threatened by Mallory (an attacker) who aims to perform a man-in-the-middle attack to take over this control system and send fake messages to Alice. More specifically, we want to prove whether is possible for Alice to receive messages that indicate a value of the physical process equal to one when the process is stuck at zero.

B. The Modbus DSTM Model

According to the described modeling and analysis methodology of Section III, a DSTM model of this scenario is built. This model comprises of three state machines: *Master*, *Slave*, and *Main*. The latter is the main machine of the DSTM (i.e., the first instantiated machine); it is trivial since it only instantiates the two other machines through a box. Before the description of the other two machines (representing the behaviors of Alice and Bob, respectively), it is necessary to describe the data portion of the Modbus model. In this model, the attacker Mallory is represented by the external environment, which sends valid messages towards both the Alice and Bob according to the semantics of the DSTM formalism.

Listing 1. Datatypes and variables.

```
//enumerations
Enum address {slaveA, masterA};
Enum fcode {RIR, DIA};
Enum subcode {NONE, RCM, FLOM};
Enum answer {EXCEPTION, SAMPLE};
//structures
Struct toMasterMsg {address, answer, Int};
Struct toSlaveMsg {address, fcode, subcode};
//channels
Chn external toMaster of toMasterMsg;
Chn external toSlave of toSlaveMsg;
//master's variables
Int sampleToMaster;
//slave's variables
address vaddress;
fcode vfcode;
subcode vsubcode;
Bool listenOnlyMode;
Int phenSample;
```

Datatypes and variables declared in the model are reported in Listing 1. Beyond the enumerations, we define the two structures *toMasterMsg* and *toSlaveMsg*, modeling the messages towards Alice and Bob, respectively. Two external channels are also defined: *toMaster* and *toSlave*, which are the input channel for the master and for the slave, respectively. According to the DSTM semantics, it indicates that when no messages are generated by Alice or by Bob, a message compliant with the message structure of the channel is non-deterministically generated by the external environment, i.e., Mallory. Furthermore, different variables are needed both by Alice's and Bob's behavioral state machines. The DSTM model of the Alice's behavior is shown in Fig. 3. It is a trivial cyclic state machine that in turn: (1) asks to the slave the value of its input register (transition *ask*) sending a Read Input Register (RIR) message, and (2) waits for a message containing a valid sample (transition *receive*).

Fig. 4 depicts the slave state machine. This state machine is typical of a Modbus slave and can be found in its specification [8]. Starting from the top, transition *T1* sets the variable *listenOnlyMode* to false. Then *T2* waits for a message over the channel *toSlave*; when a new message is received, its fields are stored in the variables *vaddress*, *vfcode*, and *vsubcode*. In the following three steps, before entering the node *executeMbFunction*, the three fields of the received message are verified: *T3* ensures that the received function code is valid, *T4* verifies the validity of the address field, while *T5* and *T6* verify the coherence between the received function code and subcode (if a RIR message is received then the subcode must be equal to NONE, otherwise a valid subcode is required). Transitions *T7*, *T8*, and *T9* perform the execution of the received function: when a device diagnostic (DIA) message is received the variable *listenOnlyMode* is set consequently; otherwise, a sample of the phenomenon (equal to 0) is prepared. Transitions *T10* and *T11* turn back to the first node, possibly sending a message with the sample to the master if a RIR message has been received and the variable *listenOnlyMode* is equal to false. Transitions from *T12* to *T15* model exceptions and send the related exception message to the master.

C. Security Analysis

We define now a proper logical formula indicating that the security property is always fulfilled to get a complete counterexample describing the attack scenario whether the property is violated. The property, as a CTL formula, is shown in Equation 1 and uses `sampleToMaster` (`SM`), a variable defined in the DSTM model. It can be described as

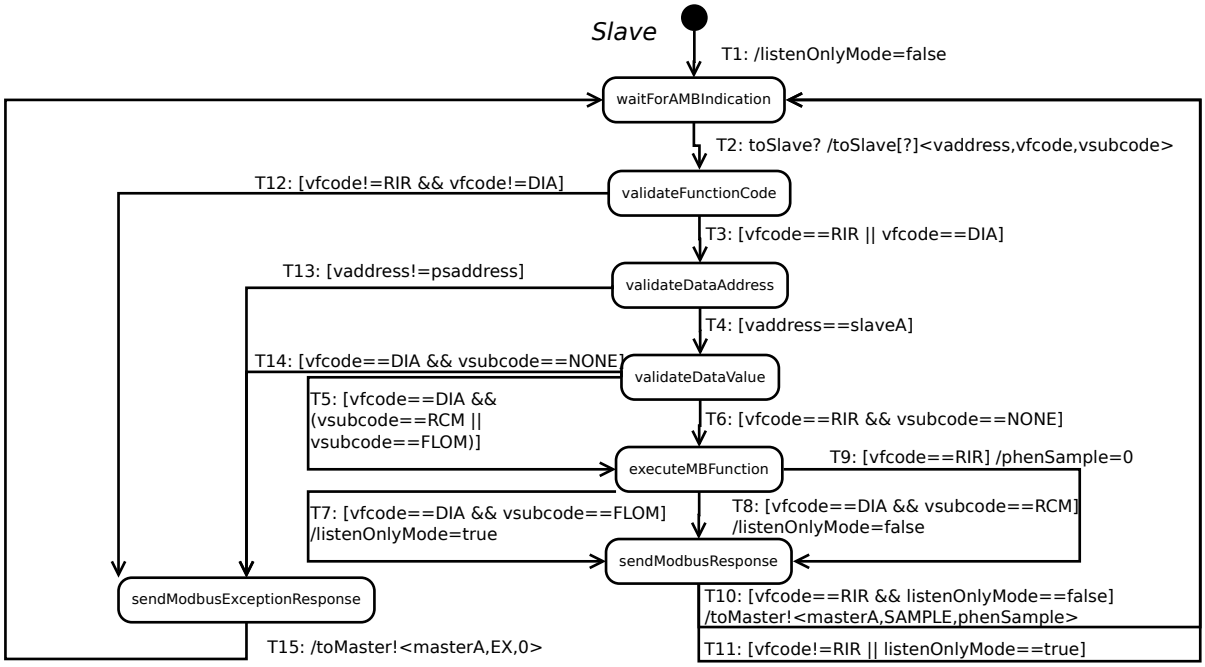

Figure 4. DSTM model of the Modbus slave.

"it is always true that, once SM is equal to 1, it still remains equals to 1".

$$AG((SM == 1) \Longrightarrow AG(SM == 1)) \qquad (1)$$

This CTL expression is translated into a Promela never claim that, attached to the Promela model obtained by the DSTM mode, is analyzed by SPIN. A counterexample is obtained whose excerpt of, focusing on the messages exchanged between parties, is reported in Listing 2.

Listing 2. Man-in-the-middle attack steps

```
M->A: <masterA,SAMPLE,0>
M->B: <slaveA,RIR,NONE>
M->A: <masterA,SAMPLE,0>
...
M->B: <slaveA,RIR,NONE>
M->A: <masterA,SAMPLE,0>
A->B
B->A
M->B: <slaveA,DIA,FLOM>
M->A: <masterA,SAMPLE,0>
A->B
M->A: <masterA,SAMPLE,0>
M->B: <slaveA,RIR,NONE>
M->A: <masterA,SAMPLE,0>
A->B
M->A: <masterA,SAMPLE,0>
M->B: <slaveA,RIR,NONE>
...
A->B
M->A: <masterA,SAMPLE,0>
M->B: <slaveA,RIR,NONE>
M->A: <masterA,SAMPLE,0>
A->B
M->A: <masterA,SAMPLE,0>
M->B: <slaveA,DIA,FLOM>
M->A: <masterA,SAMPLE,1>
A->B
M->A: <masterA,SAMPLE,0>
M->B: <slaveA,RIR,NONE>
M->A: <masterA,SAMPLE,1>
A->B
M->A: <masterA,SAMPLE,0>
M->B: <slaveA,RIR,NONE>
M->A: <masterA,SAMPLE,1>
...
```

The counterexample clearly states that Mallory (M) can exploit the lack of security mechanisms by sending to Bob (B) a diagnostic message and forcing the listen only mode status and then, acting as a slave, by responding to the data request from Alice (A) setting the value of the physical process to one, regardless its real value.

D. Discussion

Let us justify the plausibility of this scenario. Modbus specification addresses diagnostic functions only in serial line and ModbusPlus communication [8]. However, some solutions allow these functionalities also in scenarios which exploit the tunneling of serial communication over TCP/IP and TCP/IP Modbus gateways [20].

Regarding performance, as the DSTM model proposed in this paper just focused on a subset of the Modbus functionalities, the amount of time SPIN needs to compute the reported counterexample is very negligible (much less than one second on a medium/high performance laptop). Moreover, SPIN does not generate the shortest possible counterexample: optimizing strategies are necessary to automatically generate compact solutions.

Although up to date there are no test made on a larger model in the context of Modbus security, first and promising

results about the scalability of a similar approach using DSTM language and automatic test generation were reported on the railway domain [9], [19].

V. Conclusions and Future Work

Critical infrastructures are complex, heterogeneous systems that rely on automated and distributed control systems built using legacy devices and running legacy protocols. A wide adopted protocol in industrial control systems is Modbus. However, Modbus lacks for any security concern.

In this paper, we analyzed security of Modbus using formal models. In particular, we used a high-level formalism based on hierarchical state-machines (namely, Dynamic StaTe Machine) to obtain a model suitable for model-checking. Our approach allowed to find security flaws in Modbus protocol on complex control scenarios by instantiating and parameterizing the proposed state-machine models. Here, we proved the existence of man-in-the-middle attacks in Modbus-based systems.

As long-term goal, we aim at completing the modeling of Modbus protocol including all functionalities and to create a library of security properties to capture the traditional cyberattacks in SCADA systems.

Acknowledgments

The research of Roberto Nardone and Stefano Marrone was supported in part by research project CRYSTAL (Critical System Engineering Acceleration), funded from the ARTEMIS Joint Undertaking under grant agreement no. 332830. The research of Ricardo J. Rodríguez was supported in part by the EU H2020 Research and Innovation Program under grant agreement no. 644869 (DICE), in part by the Spanish MINECO project CyCriSec (TIN2014-58457-R), and in part by Universidad de Zaragoza, Fundación Bancaria Ibercaja, and Fundación CAI, "Programa Europa XXI de Estancias de Investigacion" ref. n. IT 8/16.

References

[1] Department of Homeland Security, *National Security Strategy*. The White House, May 2010, available at http://www.whitehouse.gov/sites/default/files/rss_viewer/national_security_strategy.pdf. Last check: 28/10/2016.

[2] ENISA, "Can we learn from SCADA security incidents?" Tech. Rep., 2013.

[3] R. Kozik and M. Choras, "Current Cyber Security Threats and Challenges in Critical Infrastructures Protection," in *Proceedings of the 2nd International Conference on Informatics and Applications (ICIA)*, Sep. 2013, pp. 93–97.

[4] R. Walters, "Cyber Attacks on U.S. Companies in 2014," *The Heritage Foundation – National Security and Defense*, no. 4289, pp. 1–5, October 2014, issue Brief.

[5] J. P. Farwell and R. Rohozinski, "Stuxnet and the Future of Cyber War," *Survival*, vol. 53, no. 1, pp. 23–40, 2011.

[6] A. Sood and R. Enbody, "Targeted Cyberattacks: A Superset of Advanced Persistent Threats," *IEEE Security & Privacy*, vol. 11, no. 1, pp. 54–61, Jan 2013.

[7] K. Rauscher, "Writing the Rules of Cyberwar," *IEEE Spectrum*, vol. 50, no. 12, pp. 30–32, 2013.

[8] "MODBUS Application Protocol Specification v1.1b3," MODICON, Inc., Industrial Automation Systems, Tech. Rep., 2012.

[9] R. Nardone, U. Gentile, M. Benerecetti, A. Peron, V. Vittorini, S. Marrone, and N. Mazzocca, "Modeling railway control systems in Promela," *Communications in Computer and Information Science*, vol. 596, pp. 121–136, 2016.

[10] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the Modbus protocols," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37–44, 2008.

[11] T. H. Kobayashi, A. B. Batista, J. P. S. Medeiros, J. M. F. Filho, A. M. Brito, and P. S. M. Pires, "Analysis of Malicious Traffic in Modbus/TCP Communications," in *Proceedings of the 3rd International Workshop on Critical Information Infrastructure Security (CRITIS)*. Springer, 2009, pp. 200–210.

[12] S. Bhatia, N. Kush, C. Djamaludin, J. Akande, and E. Foo, "Practical Modbus Flooding Attack and Detection," in *Proceedings of the Twelfth Australasian Information Security Conference (AISC)*, ser. AISC '14, vol. 149. Australian Computer Society, Inc., 2014, pp. 57–65.

[13] N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63–75, 2013.

[14] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "Design and Implementation of a Secure Modbus Protocol," in *Proc. of the 3rd Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection*. Springer, 2009, pp. 83–96.

[15] O. M. Sheyner, "Scenario graphs and attack graphs," Ph.D. dissertation, Pittsburgh, PA, USA, 2004, aAI3126929.

[16] J. M. Wing, "Scenario graphs applied to network security," 2007.

[17] E. Mikk, Y. Lakhnech, M. Siegel, and G. J. Holzmann, "Implementing statecharts in PROMELA/SPIN," in *Proceedings of the 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques*, 1998, pp. 90–101.

[18] G. J. Holzmann, *The SPIN model checker: Primer and reference manual*. Addison-Wesley Reading, 2004, vol. 1003.

[19] R. Nardone, U. Gentile, A. Peron, M. Benerecetti, V. Vittorini, S. Marrone, R. De Guglielmo, N. Mazzocca, and L. Velardi, "Dynamic state machines for formalizing railway control system specifications," *Communications in Computer and Information Science*, vol. 476, pp. 93–109, 2015.

[20] E. D. Knapp and J. T. Langill, *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 2014.