Implementation and Detection of Modbus Cyberattacks

Panagiotis Radoglou-Grammatikis, Ilias Siniosoglou, Thanasis Liatifis, Anastasios Kourouniadis, Konstantinos Rompolos and Panagiotis Sarigiannidis

***Abstract*—Supervisory Control and Data Acquisition (SCADA) systems play a significant role in Critical Infrastructures (CIs) since they monitor and control the automation processes of the industrial equipment. However, SCADA relies on vulnerable communication protocols without any cybersecurity mechanism, thereby making it possible to endanger the overall operation of the CI. In this paper, we focus on the Modbus/TCP protocol, which is commonly utilised in many CIs and especially in the electrical grid. In particular, our contribution is twofold. First, we study and enhance the cyberattacks provided by the Smod pen-testing tool. Second, we introduce an anomaly-based Intrusion Detection System (IDS) capable of detecting Denial of Service (DoS) cyberattacks related to Modbus/TCP. The efficacy of the proposed IDS is demonstrated by utilising real data stemming from a hydropower plant. The accuracy and the F1 score of the proposed IDS reach 81% and 77% respectively.**

***Index Terms*—Intrusion Detection System, Modbus, Supervisory Control and Data Acquisition, Smart Grid, Smod**

I. Introduction

In the era of the Internet of Things (IoT), Information and Communication Technology (ICT) constitutes an integral part of the Critical Infrastructures (CIs). In particular, focusing on the energy domain, the conventional electrical grid is transformed into a new paradigm called Smart Grid (SG), by providing multiple benefits such as self-monitoring, two-way communication, self-healing, and distributed generation. However, SG raises critical cybersecurity hazards due to the vulnerabilities of ICT and mainly of the insecure communication protocols, such as Modbus, Profinet, Distributed Network Protocol (DNP3), and IEC 60870-5-104.

In this paper, we focus on the security of the Modbus/TCP protocol, which is commonly utilised by the Supervisory Control and Data Acquisition Systems (SCADA). Modbus/TCP does not include any authentication or access control mechanism, thus allowing potential cyberattackers to perform a plethora of cyberattacks such as Denial of Service (DoS), Man-in-the-Middle (MitM), and unauthorised access. In particular, the contribution of this paper is twofold; first, we investigate and enhance the various Modbus/TCP cyberattacks supported by Smod [1]. Smod is the most widely known pen-testing tool related to Modbus/TCP, aggregating a set of diagnostic and offensive features [1]. In this paper, we extended Smod with new five cyberattacks. Second, we provide an Intrusion Detection System (IDS) capable of detecting DoS attacks against Modbus/TCP.

The rest of this paper is organised as follows. Section II provides relevant works regarding the Modbus/TCP security. In Section III, we list the various cyberattacks supported by Smod and describe our extensions. Section IV analyses the architecture of our IDS, while Section V evaluates its efficacy. Finally, Section VI concludes this paper.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 787011 (SPEAR).

P. Radoglou Grammatikis, I. Siniosoglou, T. Liatifis, A. Kourouniadis, K. Rompolos and P. Sarigiannidis are with the Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani 50100, Greece - E-Mail: {pradoglou, psarigiannidis,isiniosoglou, aliatifis, ece01344, krobolos}@uowm.gr

II. Related Work

Many papers have examined the vulnerabilities of Modbus. More specifically, in [2], P. Huitsing et al. provided a detailed theoretical study about the various cyberattacks against the Modbus protocol. In particular, they classified the attacks into three categories, namely a) serial only attacks, b) Serial and TCP attacks, and c) TCP only attacks. In [3], B. Chen et al. executed and examined the impact of Modbus-related MiTM and TCP SYN flood attacks against a real testbed. Similarly, utilising a simulation environment, S. Li et al. in [4] performed four kinds of cyberattacks related to Modbus in order to collect appropriate traces that can be used for machine learning algorithms. Specifically, their dataset includes traces concerning 1) reconnaissance attacks, 2) response injection, 3) command injection and 4) DoS. Finally, A. Voyiatzis et al. [5], and S. Bhatia et al. [6] developed a Modbus/TCP fuzzer and a Modbus flooding attack tool, respectively.

On the other side, there are also several works focusing on detecting cyberattacks or anomalies against Modbus. In [7], T. Morris et al. provided a set of rules related to Modbus that can be used by known signature-based IDS like Snort and Suricata. Accordingly, in [8] N. Goldenberg and A. Wool presented a relevant anomaly-based IDS, which relies on a Deterministic Finite Automaton (DFA). In a similar manner, S. Anton et al. in [9] utilised and evaluated various machine learning classification techniques for detecting Modbus attacks. Finally, in [10], P. Wang et al. provided an IDS for Modbus based on honeypots' logs.

III. Modbus Cyberattacks and Smod Extension

Table I summarises a set of Modbus/TCP cyberattacks, which are supported by Smod. Next, the following subsections analyse our extensions by adding five new Modbus/TCP attack modules, namely a) *Modbus Teardrop Module*, *b) Flag Flood Module*, *c) Port Pool Exhaustion Module*, *d) Response Delay Module*, and *e) Baseline Response Replay Module*.

TABLE I: Smod Attacks

No	Cyberattack	Cyberattack Type	Description
1	modbus/dos/arp	DoS	DoS attack with Address Resolution Protocol (ARP) poisoning
2	modbus/dos/galilRIO	DoS	DoS attack against Galil RIO-47100 Programmable Logic Controller (PLC)
3	modbus/dos/writeAllRegister	DoS	DoS trying to write all registers
4	modbus/dos/writeSingleCoils	DoS	DoS trying to write all coils
5	modbus/function/fuzzing	Fuzzing	Fuzzing modbus functions
6	modbus/function/readCoils	Unauthorised Access/Fuzzing	Reads a specific of Coils
7	modbus/function/readCoilsException	Unauthorised Access/Fuzzing	Fuzzing read coils exception function
8	modbus/function/readDiscreteInput	Unauthorised Access/Fuzzing	Reads the status of specific discrete inputs
9	modbus/function/readDiscreteInputException	Unauthorised Access/Fuzzing	Fuzzing read discrete inputs exception function
10	modbus/function/readExceptionStatus	Unauthorised Access/Fuzzing	Fuzzing read exception status function
11	modbus/function/readHoldingRegister	Unauthorised Access/Fuzzing	Reads a specific amount of holding registers
12	modbus/function/readHoldingRegisterException	Unauthorised Access/Fuzzing	Fuzzing read holding registers exception function
13	modbus/function/readInputRegister	Unauthorised Access/Fuzzing	Reads a specific amount of input registers
14	modbus/function/readInputRegisterException	Unauthorised Access/Fuzzing	Fuzzing read input registers exception function
15	modbus/function/writeSingleCoils	Unauthorised Access/Fuzzing	Writes either 0 or 1 to a given coil
16	modbus/function/writeSingleRegister	Unauthorised Access/Fuzzing	Writes a specific value to a single register
17	modbus/scanner/arpWatcher	Reconnaissance Attack	ARP watcher
18	modbus/scanner/discover	Reconnaissance Attack	Identifies if the Modbus service is running in a field device
19	modbus/scanner/getfunc	Reconnaissance Attack	Enumares the function codes supported by a field device
20	modbus/scanner/uid	Reconnaissance Attack	Enumerates the function codes supported by a field device
21	modbus/sniff/arp	MiTM	ARP poisoning

A. Modbus Teardrop Module

Teardrop attack [2] is a DoS attack, which intends to crash the target by transmitting overlapping fragmented packets, thus severing the communication between the target and the other devices. In particular, the attacker sends fragmented Modbus/TCP packets with overlapping or scrambled fragment offsets in order to violate the fragmentation process. The Modbus payload can consist of either random or specific values. Algorithm 1 gives more details regarding the implementation of the attack.

Algorithm 1 Teardrop Attack

procedure TEARDROP ▷ The execution of the attack
  while $k < numberOfPackets$ **do**
    $p \leftarrow CreateModbusPacket(fc, targetIP)$
    $fragments \leftarrow Fragment(p, fragmentSize)$
    for $fragment \in Fragments$ **do**
      $send(fragment)$
    $k \leftarrow k + 1$

Algorithm 2 Flag Flood Attack

procedure FLAGFLOOD
  while $x < numberOfPackets$ **do**
    $p \leftarrow CreatePacket(IP, TCP)$
    if $Flag$ == F **then**
      $c \leftarrow connectToTarget(targetIP, port)$
      $c.closeConnection()$
    else
      $c.send(p)$
      $attempts \leftarrow attempts + 1$
  $x \leftarrow x + 1$

B. Flag Flood Module

Flag Flood [2] belongs to the DoS category. In particular, it tries to flood the target by using a plethora of TCP packets with specific flags (ACK, FIN, SYN, and RST). Algorithm 2 provides the relevant implementation details.

C. Port Pool Exhaustion Module

The Port Pool Exhaustion [2] aims to deplete the available bandwidth of the target. When this attack module is activated, a plethora of concurrent threads is generated to connect to the target Modbus TCP port for a particular time duration, thus destructing the Modbus communication. Algorithm 3 gives the corresponding implementation details.

Algorithm 3 Port Pool Exhaustion Attack

procedure PORT POOL EXHAUSTION
  while $i < numberOfThreads$ **do**
    $c \leftarrow connectToTarget(targetIP)$
    $attempts \leftarrow 0$
    $startTime \leftarrow getTime()$
    if $c.Connected()$ && $attempts < 3$ **then**
      while $elapsedTime < attackTime$ **do**
        $c.send(KeepAlive)$
        $elapsedTime \leftarrow getTime() - startTime$
    else
      $c \leftarrow connectToTarget(targetIP)$
      $attempts \leftarrow attempts + 1$
  $i \leftarrow i + 1$

D. Response Delay Module

The response delay attack [2] belongs to the category of the replay attacks. It utilises first an ARP poisoning attack in

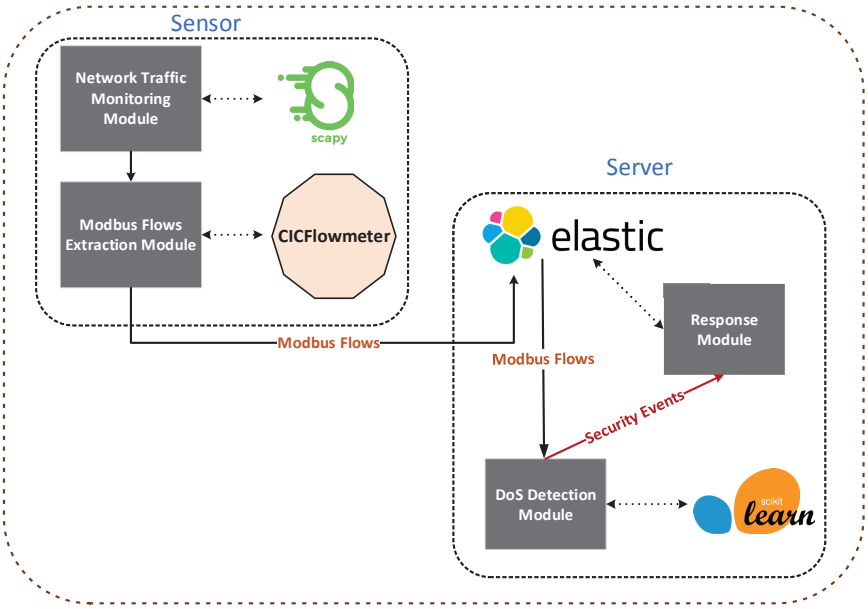

Fig. 1: Modbus/TCP IDS Architecture

Algorithm 4 Baseline Response Replay

procedure BRREPLAY
  while $x < DurationOfAttack$ **do**
    $sniff(filter, interface, prn = storePkt)$
    $send(packetList)$
    $x \leftarrow x + 1$
  while $attacking == True$ **do**
    $poisionTarget()$
  $restoreTarget()$

order to insert the attacker between the two communication points. Thus, the attacker is able to obtain the exchanged request packets and send the appropriate responses with a specific delay time. In other words, the response packets are transmitted only by the attacker with a specific delay. Such delays can have disastrous consequences in an industrial environment. The implementation of the attack was based on iptables and the netfilter queue. Algorithm 5 summarises the attack's steps. In particular, it consists of three threads. The first one sends periodically fake ARP packets. The second thread waits for a specific amount of time and then forwards all the packets in the queue. Finally, the main thread receives all the packets and appends them in the queue.

E. Baseline Response Replay Module

The baseline response replay attack [2] aims at confusing or even interrupting the communication between two endpoints, by capturing and replaying the information sent between two devices. Specifically, the attacker executes an ARP poisoning attack in order to receive the exchanged traffic. Next, some of the packets are replayed to the destination. Algorithm 4 provides the implementation steps.

Algorithm 5 Response Delay

procedure RECEIVEPACKETS(packet)
  if packet destined for HMI_IP **then**
    $packets\_queue.append(packet)$
  else
    $packet.accept()$
procedure FORWARDPACKETS(delay)
  while True **do**
    $sleep(delay)$
    for $p \in packets\_queue$ **do**
      $p.accept()$
procedure EXPLOIT(HMI_IP, RTU_IP, queue_id, delay)
  $thread_1 \leftarrow ARPPoison(HMI\_IP, RTU\_IP)$
  $thread_2 \leftarrow ForwardPackets(delay)$
  $thread_1.start()$
  $thread_2.start()$
  $nfqueue \leftarrow NetfilterQueue(queue\_id)$
  $nfqueue.run()$

IV. MODBUS/TCP IDS

Fig. 1 presents the architecture of the proposed Modbus/TCP IDS, which consists of two main components called *a) Sensors* and *b) Server*. *Sensors* are distributed throughout the network, while they are responsible for capturing and parsing the Modbus/TCP network traffic of each subnet. In particular, a *Sensor* consists of two modules, namely *a) Network Traffic Monitoring Module* and *b) Modbus Flows Extraction Module*. On the other hand, *Server* receives from the various *Sensors* the Modbus flows and identifies which of them are related to a DoS attack. *Server* includes two modules: *a) DoS Detection Module* and *b) Response Module*.

The following subsections analyse each module.

A. Network Traffic Monitoring Module

The *Network Capturing Module* is responsible for capturing periodically the network traffic. To this end, the Scapy library was utilised. In particular, the various Packet Capture (PCAP) files are generated based on two criteria: a) when their size is equal to a specific threshold or b) when a specific amount of time is equal to a second threshold. These thresholds are defined based on each use case.

B. Modbus Flows Extraction Module

The *Modbus Flows Extraction Module* receives the PCAP files generated by *Network Traffic Capturing Module* and is responsible for extracting the corresponding Modbus/TCP flows using the CICFlowMeter software [11]. These flows are stored in an Elasticsearch database of *Server*. CICFlowMeter generates 83 features for each Modbus flow that are used in order to detect the DoS attacks.

C. DoS Detection Module

The *DoS Detection Module* receives from the Elasticsearch database the Modbus/TCP flows and undertakes to detect potential DoS attacks by using classification machine learning models. The efficacy of these models is presented in Section V. The produced security events are stored in a different index of the Elasticsearch database of *Server*.

D. Response Module

The *Response Module* informs the user about the security events based on a web-based user interface. To this end, Kibana of Elastic Stack was used.

V. Evaluation Results

Table II compares the efficacy of the proposed algorithms used in order to train the various intrusion detection models. For this evaluation, a) Accuracy, b) F1 score, c) True Positive Rate (TPR), and d) Precision were used. These metrics are defined and described thoroughly in [12]. Regarding the dataset used for the training and testing process, we combined real Modbus/TCP data from a power plant in Greece as well as DoS data of [13]. The overal dataset was divided into two subsets: a) training dataset (70%) and b) testing dataset (30%). The scikit-learn Python library was used for the training and testing process. According to the evaluation results, Adaboost and Random Forest classifiers give the most efficient results in terms of Accuracy and F1.

TABLE II: DoS Detection Evaluation Results.

Model	Accuracy	Precision	TPR	F1
SVM-Linear	0.645	0.879	0.336	0.487
Random Forest	**0.811**	**0.964**	**0.647**	**0.774**
Naive Bayes	0.650	0.989	0.304	0.465
KNN	0.667	0.996	0.336	0.503
MLP	0.8017	0.942	0.642	0.764
AdaBoost	**0.812**	**0.964**	**0.647**	**0.775**

VI. Conclusions

This paper is focused on the security of the Modbus/TCP protocol. In particular, first, we investigated and enhanced the Smod pen-testing tool, by introducing new five attack modules. Subsequently, we provided an anomaly-based IDS capable of discriminating DoS attacks related to Modbus/TCP. The evaluation analysis demonstrates the efficiency of the proposed IDS since Accuracy and F1 score reach 81% and 77% respectively.

VII. Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 787011 (SPEAR).

References

[1] J. Luswata, P. Zavarsky, B. Swar, and D. Zvabva, "Analysis of scada security using penetration testing: A case study on modbus tcp protocol," in *2018 29th Biennial Symposium on Communications (BSC)*, June 2018, pp. 1–5.

[2] 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, 12 2008.

[3] B. Chen, N. Pattanaik, A. Goulart, K. L. Butler-Purry, and D. Kundur, "Implementing attacks for modbus/tcp protocol in a real-time cyber physical system test bed," in *2015 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*. IEEE, 2015, pp. 1–6.

[4] S.-C. Li, Y. Huang, B.-C. Tai, and C.-T. Lin, "Using data mining methods to detect simulated intrusions on a modbus network," in *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)*. IEEE, 2017, pp. 143–148.

[5] A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, "A modbus/tcp fuzzer for testing internetworked industrial systems," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2015, pp. 1–6.

[6] S. Bhatia, N. Kush, C. Djamaludin, A. Akande, and E. Foo, "Practical modbus flooding attack and detection," in *Proceedings of the Twelfth Australasian Information Security Conference (AISC 2014)[Conferences in Research and Practice in Information Technology, Volume 149]*. Australian Computer Society, Inc., 2014, pp. 57–65.

[7] T. H. Morris, B. A. Jones, R. B. Vaughn, and Y. S. Dandass, "Deterministic intrusion detection rules for modbus protocols," in *2013 46th Hawaii International Conference on System Sciences*. IEEE, 2013, pp. 1773–1781.

[8] 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.

[9] S. D. Anton, S. Kanoor, D. Fraunholz, and H. D. Schotten, "Evaluation of machine learning-based anomaly detection algorithms on an industrial modbus/tcp data set," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–9.

[10] P.-H. Wang, I.-E. Liao, K.-F. Kao, and J.-Y. Huang, "An intrusion detection method based on log sequence clustering of honeypot for modbus tcp protocol," in *2018 IEEE International Conference on Applied System Invention (ICASI)*. IEEE, 2018, pp. 255–258.

[11] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *ICISSP*, 2017, pp. 253–262.

[12] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems," *IEEE Access*, vol. 7, pp. 46 595–46 620, 2019.

[13] P. Simoes, "Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process," in *Critical Information Infrastructures Security: 13th International Conference, CRITIS 2018, Kaunas, Lithuania, September 24-26, 2018, Revised Selected Papers*, vol. 11260. Springer, 2019, p. 230.