Attacking SCADA systems: a practical perspective

Luís Rosa[1], Tiago Cruz[1], Paulo Simões[1], Edmundo Monteiro[1], Leonid Lev[2]
[1]CISUC-DEI, University of Coimbra, Portugal
[2]IEC – Israel Electric Corportation, Israel
{lmrosa, tjcruz, psimoes, edmundo}@dei.uc.pt, leonid.lev@iec.co.il

***Abstract*—As Supervisory Control and Data Acquisition (SCADA) and Industrial and Automation Control System (IACS) architectures became more open and interconnected, some of their remotely controlled processes also became more exposed to cyber threats. Aspects such as the use of mature technologies and legacy equipment or even the unforeseen consequences of bridging IACS with external networks have contributed to this situation. This situation prompted the involvement of governmental, industrial and research organizations, as well as standardization entities, in order to create and promote a series of recommendations and standards for IACS cyber-security.**

Despite those efforts, which are mostly focused on prevention and mitigation, existing literature still lacks attack descriptions that can be reused to reproduce and further research specific use cases and scenarios of security incidents, useful for improving and developing new security detection strategies. In this paper, we describe the implementation of a set of attacks targeting a SCADA hybrid testbed that reproduces an electrical grid for energy distribution (medium and high voltage). This environment makes use of real SCADA equipment to faithfully reproduce a real operational deployment, providing a better insight into less evident SCADA- and device- specificities.

Keywords—Industrial Control Systems, SCADA, Security

I. INTRODUCTION

Supervisory Control and Data Acquisition (SCADA) systems are used to manage and automate processes in critical infrastructures such as electricity grids or water distribution facilities. According to the ISA definition [1], SCADA-based Industrial and Automation Control Systems (IACS) are structured into five distinct levels: level 0, reserved for the sensors and actuators; level 1, that contains devices such as Programmable Logic Controllers (PLC's) and Remote Terminal Units (RTU's); level 2, composed of supervisory control equipment's such as the Human-Machine Interface (HMI); level 3: for the Manufacturing Execution Systems (MES), such as the systems hosting production planning software; and level 4 for the remaining business related systems.

The interconnection of level 0 and level 1 devices (e.g. PLC's and RTU's) and the interconnection of level 1 devices with level 2 devices (e.g. HMI's) are probably the most vulnerable points of IACS infrastructures. They were traditionally isolated and based on proprietary protocols and technologies without intrinsic security capabilities, relying on obscurity and air-gapping principles for such purpose.

Nevertheless, with the progressive adoption of Ethernet- and TCP/IP-based networks, standardized SCADA protocols and VPN-based remote access (to reduce maintenance costs), these networks are more connected than ever to the remaining infrastructure – the corporate network and even the Internet – either by sharing physical network and computing resources or via (not foolproof) interconnection firewalls, routers or gateways. This paradigm change drastically increases the risks, due to the increased system complexity, the introduction of new attack vectors and the amplified exposure of existing security vulnerabilities.

SCADA systems are intrinsically different from traditional ICT systems [2]. Automated real time physical processes do not need high throughput but demand continuous availability with guaranteed low delay and low jitter. More, their primary focus is on availability and service continuity – opposed to classic ICT systems, where information confidentiality and integrity come first [3]. SCADA systems also have much longer lifetime cycles, due to their high upgrade costs – easily reaching obsolescence by ICT standards. Even simple security patches take much longer to deploy, due to the need for previous testing and certification

Recognizing those specificities and risks, as well as the tremendous impact they can have on SCADA-based critical infrastructures such as energy grids, water distribution systems, transportation systems or factory plants, there is currently a strong investment on research towards enhancing the security of (both legacy and more recent) SCADA systems. There is an extensive literature researching various approaches for introducing IACS-specific intrusion detection mechanisms, as well as for improving the intrinsic security of SCADA systems.

However, due to logistic constraints and the difficulty of using real-world production systems for research purposes, not many works are based on wider testbed scenarios reproducing real infrastructures, instead using very simplified test benches or general-purpose datasets. Among these, the large majority is focused on the defensive perspective of the targeted infrastructure, instead of the attacker's point of view.

While this is understandable – considering how difficult it is to build larger, more realistic testbeds and the fact that researchers aim is to improve the SCADA systems cyber-security awareness and capabilities – we believe it is also important to grasp the attacker's perspective, including the challenges he faces to implement a successful attack.

In this paper, we provide a practical description of somehow representative cyber-attacks (network based enumeration, communication hijacking and service disruption) targeting SCADA systems within a testbed that represents an

electricity grid (regional network of medium and high voltage distribution). This testbed consists of a hybrid environment that includes real networking and SCADA assets (e.g. PLCs, HMIs, process control servers) controlling an emulated power grid (so we can assess the possible impact of these attacks on the physical world). We explain those attacks and discuss some of the challenges faced by an attacker to implement them.

This work was performed in the scope of the CockpitCI [4] and ATENA [5] research projects, which aim at providing a holistic approach to security, safety and resilience of energy distribution grids, including the detection and prevention of cyber-attacks and the analysis of the mutual interdependency between their ICT assets (communications network, servers, SCADA control applications, PLC's and RTU's) and the energy side (e.g. transmission lines, substations, power transformers and generators, quality of energy service). Detection of cyber-attacks and situational awareness is a key part of these projects, and as such we built a specialized detection layer that has been extensively described and evaluated in previous works (e.g. [6-7]). This paper complements them by focusing not so much on the detection and mitigation solutions, but rather on the process of preparing and executing the attacks used for validation purposes. For sake of readability and representativeness, we decided to focus on simple, classic attacks, instead of more complex actions.

The rest of the paper is organized as follows. Next section, we discuss related work. Section III introduces the testbed environment we used. The implemented cyber-attacks are discussed in Section IV, and Section V concludes the paper.

II. Related Work

As already mentioned, existing research literature discusses different types of cyber-attacks against SCADA systems, such as Denial of Service (DoS) attacks [8-10], Man-in-the-Middle (MitM) attacks [11-12] or malware-based attacks [13]. Nevertheless, those discussions are usually focused on the defense mechanisms (and not on the attacks), based on small and/or simulated scenarios or lack detail on the practical implementation of the attack.

Post-incident research on real-world attacks are valuable sources. Rolf Langer's report on the well-known Stuxnet malware [14] targeting Iran Nuclear facilities is a good example of such sources. Other high-profile well covered include the Duqu malware [15] or the 2015 BlackEnergy attack allegedly responsible for power outages in the Ukrainian Power Grid [16]. These sources have the advantage of being based on real, successful attacks but are usually limited to the analysis of complex high-profile incidents often supported by nation-state resources – instead of simpler but representative attack profiles.

III. Target Environment

A. HEDVa Testbed

With the purpose of supporting the demonstration and validation of the CockpitCI framework, a testbed reproducing a regional-scale energy distribution network was built by Israel Electric Corporation (IEC). From ICT and SCADA perspectives this testbed is composed of real assets, including IT network, control and field level components, servers and services that typically integrate such a system. Within this scenario, an electrical distribution grid topology was entirely emulated using specialized software developed at IEC, given the practical impossibility of using a real, large-scale energy distribution infrastructure (composed of many substations and hundreds of kilometers of power lines).

This approach results in a hybrid testbed, where all ICT and SCADA components are real and "believe to be" monitoring and controlling a real energy grid. This is achieved by using an agent-based grid simulation model that uses real PLC equipment to emulate elements such as feeders or circuit breakers. The interface between the real and emulated domains of the grid scenario includes all the monitoring data and controls that would exist in a real operational environment.

Figure 1 provides an overview of this testbed (designated as HEDVa: Hybrid Environment for Design and Validation), of which only a subset will be relevant to the scope of this paper. By using such an environment, it became possible to research more complex interdependencies between different components (e.g. network, SCADA devices) and different domains (e.g. impact of ICT faults on the quality of energy on the different points of the grid). Furthermore, having a real deployment of ICT and SCADA systems allowed more realistic assessments and the collection of more extensive and realistic validation data.

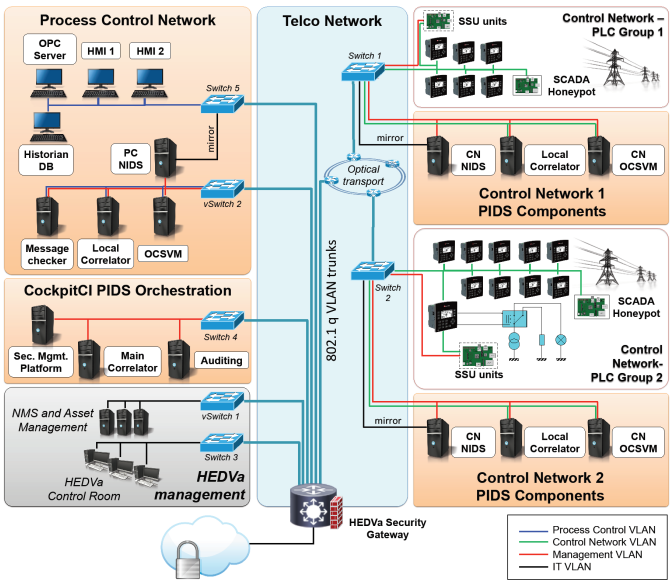

Figure 1: Overview of the HEDVa Testbed [6]

B. The Modbus Protocol

Among the wide range of different SCADA protocols available, the HEDVa Testbed uses Modbus over TCP/IP [17-18]. Modbus is a protocol used to query field data using a polling client/server approach. Communication is based on query/response transactions identified by a transaction ID field and distinguished by a function code field. According to the Modbus data model, different types of tables are mapped into the PLC memory (such as discrete inputs, coils or holding registers). These values are queried via their respective function code and memory address (see Figure 2).

There is no built-in mechanism (or fields) for authentication, authorization or encryption. Hence, without proper security enforcement in the remaining network stack, it

becomes possible to dissect the Modbus messages payload (i.e. critical information from a physical process).

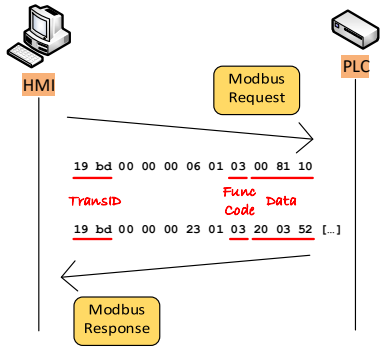

Figure 2: Example of the interaction between two Modbus devices

Forging communication or field data is also possible by simply crafting a valid value for the transaction ID field (see Figure 3), as this value is frequently predictable (due to lack of randomness in poor Modbus implementations) or even blindly discarded by some Modbus implementations. Moreover, Modbus/TCP runs on the top of non-encrypted TCP sessions.

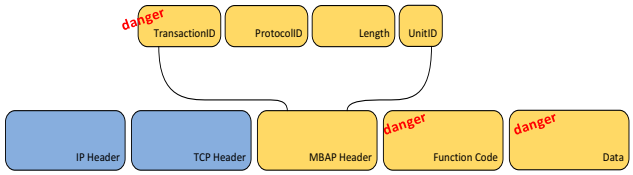

Figure 3: Modbus Frame and header format

Even considering the real-time nature of the underlying processes, the polling based mechanism provided by the Modbus protocol is not effectively real-time. The intervals between each request directly impact the delay time between a change in the physical process and the time the change is observed by the HMI operator. This results in a small but viable time window for hijacking communications before the Operator and/or the HMI application notice any changes.

Despite all these security vulnerabilities of Modbus apparently making the attacker's work too easy, Modbus holds a significant market share (over 20%, considering all its variations [19]) and many of the other protocols are not much different. This means the testbed represents of a large subset of the systems currently in operation.

Several open source components can be used to build Modbus hacking tools, such as the Nmap's modbus-discover script [20] or Modscan [21] that allows to map and enumerate PLCs using Modbus over TCP within a network by exploring their replies. Another example is a python library extended from Scapy (a widely-used packet manipulation framework easy to extend and integrate with other applications) that contains Modbus specific functions to easily craft Modbus frames [22].

Next section will discuss with the execution of a series of attacks, which also served for validating the proposed DIDS.

IV. ATTACK STAGING AND EXECUTION

All the attack scenarios assumed the attacker had access to the process control network (e.g. as result of a compromised host – this step, which corresponds to the exploitation of the initial attack was intentionally omitted). For practical demonstrations, a dedicated host was deployed on the HEDVa, to serve as a base for the attacker, which could be easily relocated on the infrastructure, since it was hosted on a virtual machine. A similar attack strategy could be implemented (with the proper adjustments) to trigger an attack (for instance, forging or sending Modbus packets) directly from a compromised HMI or other component.

A three-stage attack strategy was devised, pursuing the following goals: monitoring the process values (to gain knowledge about the nature and characteristics of the controlled process), change them without being noticed in the SCADA HMI consoles and finally, induce service disruption on the energy grid. These should cover a large subset of a cyber-attack targeting a SCADA system. A by no means exhaustive list of the implemented attacks includes classical and Modbus specific scans, different variants of Denial of service attacks based on network floods, and a SCADA specific MitM specifically customized for this process environment. Next, we describe some of those attacks.

A. The HEDVa use case scenario for attack implementation

For the sake of readability, we'll describe the attacks using a subset of the HEDVa testbed, configured to emulate an electricity distribution grid composed by two energy feeders and several circuit breakers, controlled by real Modbus PLCs (see Figure 4). Several HEDVa assets, including services, equipment (such as network switches and PLCs), servers (both physical and virtualized) and networks are also part of this use case. The PLCs and the remaining elements of the SCADA infrastructure in charge of the emulated grid are connected using an Ethernet LAN infrastructure (using VLAN segmentation for domain separation).

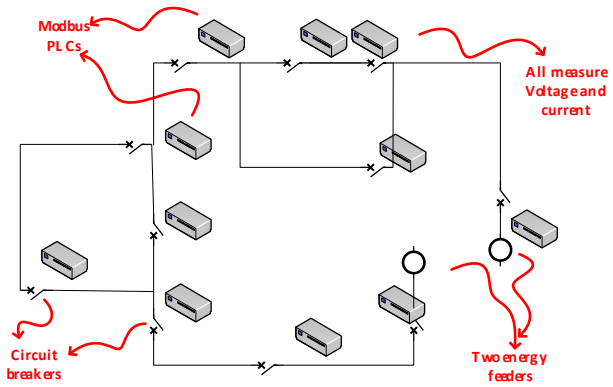

Figure 4: Representation of the electrical grid use case scenario

The scenario deployed on the HEDVa (see Figure 5) includes two Human Machine Interface (HMI) hosts, controlling and supervising the PLCs, an OPC server, a dedicated database for past events and offline analysis, and a deployment of the CockpitCI DIDS (not depicted). However, the DIDS security detection components didn't play any active role – they were used to observe and document the attacks, without interfering with the attacker's actions.

This scenario not only offered the means to validate the CockpitCI DIDS, but it also offered the opportunity to implement and analyze a series of security strategies. For the

latter purpose, and complementary to the classic penetration testing and auditing procedures, a series of team drills were executed to obtain relevant data on the most effective tactical defensive and offensive strategies.

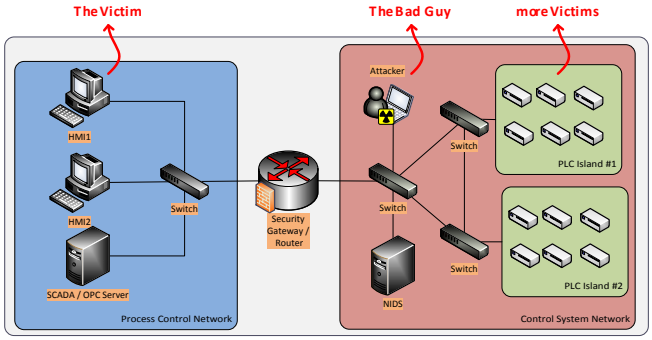

Figure 5: Reference scenario for the use cases

Besides these efforts, the acquisition of relevant datasets for development, training and offline evaluation of anomaly detection methods was also another important role of the HEDVa scenario. For capturing all the network interactions for further analysis, a centralized network point of capture was configured. This was achieved using port monitoring / mirroring in the switch layer, as opposed to a distributed packet acquisition solution to avoid all the issues with duplicated packets or timestamp synchronization.

B. Network Reconnaissance

Network scouting is one of the first steps of an attack, meant to gather information about all the components of the target environment, to discover and identify topologies, hosts and services. For instance, traditional network components such as HMIs are identified by IP and MAC addresses, operating system versions and a set of services (using techniques such as FIN scans, see Figure 6) – in such cases, the specific service footprint, together with TCP fingerprinting data is useful to identify specific components or software implementations.

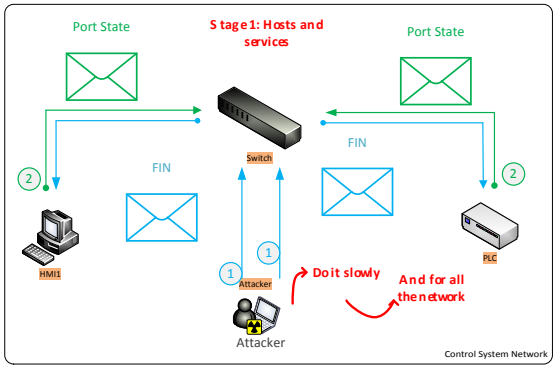

Figure 6: First step of a Network/Modbus scan

In addition to that, each PLC is also identified and addressed by the unitID field, part of the Modbus frame (see Figure 7). For simple scenarios where one IP address correspond to one PLC, the unitID can be set to a fixed known value (typically "1") or may be "ignored" by the Modbus implementation. Nevertheless, a Modbus gateway, using only one IP address, may hide several PLCs with different unitIDs. As part of an attack, a Modbus request with a wrong unitID, blindly used by an attacker, may be discarded or easily flagged with proper security mechanisms. Thus, and for Modbus over TCP, it is critical to perform a Modbus enumeration on top of the traditional TCP/IP scans. Both types of scans are relevant as they can be used not only to discover devices and types of services but also to perform fingerprinting and discover PLCs behind gateways.

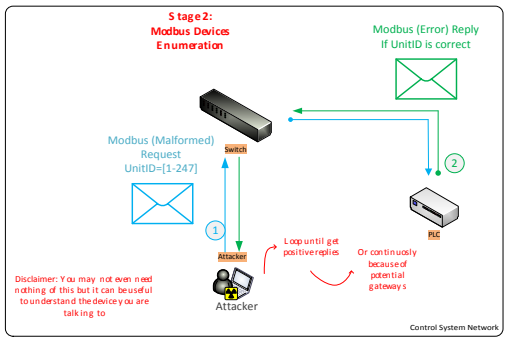

Figure 7: Modbus Device Scan / Enumeration

Network scouting provides a perspective on the target infrastructure from the network point-of-view, corresponding to the layers 2-4 of the OSI model. Despite its usefulness as a tool to identify and enumerate devices and services it doesn't provide process-level information, which is required to implement sophisticated attacks. The next subsection will present the technique that was used to obtain such information.

C. Using ARP poisoning to implement a MitM attack

The concept of a ARP poisoning MitM attack usually comprises two parts: an ARP spoofing and a communication hijacking step. In the first stage, the idea is to spoof the ARP cache of both target devices, belonging to the same link, by sending malicious and unsolicited ARP "is-at" messages to the network (see Figure 8) to force both devices to send the packets through the attacker MAC address. This requires the attacker to know at least the IP and MAC addresses of the victims and the link they are connected to. As soon as the ARP cache of each victim is spoofed, the traffic gets redirected through the attacker.

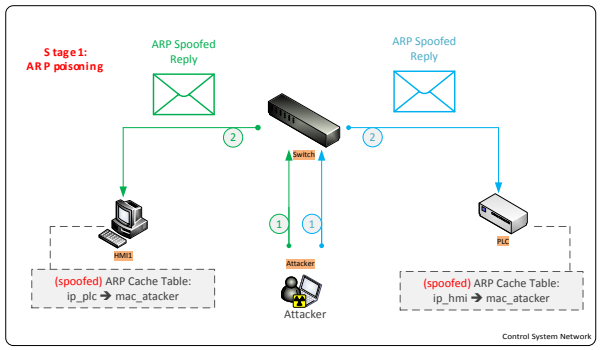

Figure 8: ARP poisoning attack

In the second attack stage (see Figure 9), when the traffic is already being redirected, the attacker can choose to read the messages and forward them, or actively change them.

Depending on the type of TCP connection, its payload and the actual data the attacker is interested in, the process may get

complex. For persistent TCP connections, as opposed to one TCP connection per data request (Modbus can be implemented using the two communication models), the attacker will need to keep the TCP fields consistent (e.g. sequence and acknowledgement numbers) and the connection open (e.g. TCP keep-alive packets).

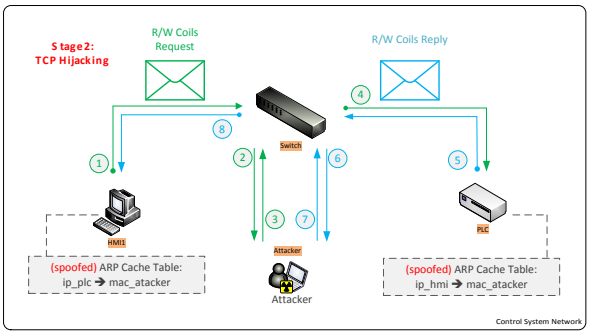

Figure 9: TCP hijacking

Moreover, in the case of Modbus, the requested values typically change in real-time and some of them are directly changed by the SCADA operator (e.g. Modbus writes), this means the attacker needs to somehow keep track not only of all the interactions but also compute and reproduce the effects in the physical process (e.g close of a circuit breaker in electric path may change the physical values such as current and voltage in other parts of the circuit). The complexity of this increases as the number of elements, relations and interdependencies increases.

D. Attack strategy and execution

The objective of the attacker can be summarized as such: hijack the entire grid in such a way that the main HMI (HMI1) has no clue about the ongoing attack. Moreover, the attack goal should be accomplished by the attacker while going unnoticed.

One of the first challenges faced by the attacker has to do with understanding the network topology and communication flows. For instance, the HMI1 host (one of the victims) is not part of the same network link as the PLCs, requiring the attacker to implement an ARP spoof targeting the gateway interface of the network link where the attacker is placed instead of the HMI1 (see Figure 10).

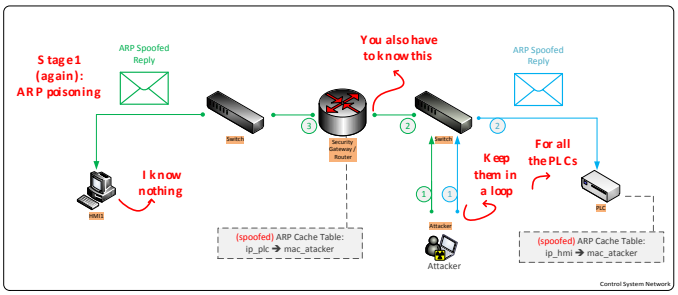

Figure 10: ARP poisoning for the implemented attack

Besides HMI1, there is a second HMI (HMI2) developed to observe and validate the attack, which was not spoofed. HMI1 uses TCP persistent connections to control several PLCs (11, to be more precise). Thus, the attacker needs to know how to handle or forward any spoofed packets in real-time, while avoiding TCP connection drops, to prevent any suspicious behaviour on the HMI console that could unveil his presence (see Figure 11). Packet drops automatically raise an alarm and change the view of the HMI for the corresponding PLC after a couple of seconds, indicating a potential issue. A TCP connection lost or a lack of a Modbus reply from the PLC is also visible from the HMI console. The second HMI did not use persistent connections. Later, during the trials, it was discovered that each PLC only supported a maximum of two simultaneous TCP connections. This may limit the way TCP connections are handled and redirected by the attacker.

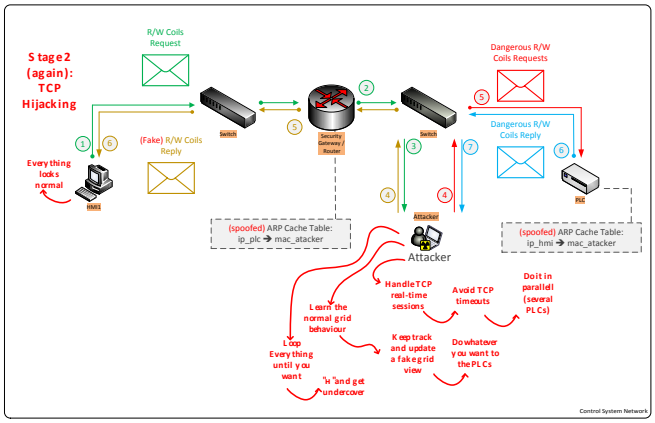

Figure 11: TCP hijacking for the implemented attack

At first, the main concern was to place the attacker in the middle of the communication between the HMI1 and the PLCs to capture and analyze relevant process information. This allowed the attacker to gather more detailed information about the communications and the controlled process, learning how each Modbus register value affected the others (e.g. circuit breakers, current and voltage ranges).

Once the attacker was able to figure out the basic behavior of the controlled process, it was time to step up the challenge and hijack the entire process. This required forging the entire grid state in such a way that any HMI interaction may produce a realistic state update, while decoupling HMI-PLC interactions. For this purpose, the attacker needs to reply to the Modbus requests in real-time. Moreover, TCP session hijacking requires the attacker to maintain the integrity of the TCP connection (such as TCP sequence numbers) to avoid a connection drop.

Then, the following task is crafting the Modbus frames and recreate a fake view of the entire scenario in real-time. This task was implemented using a in-house application on the top of Scapy framework [22] – since common open-source tools normally used for this sort of attacks are not SCADA/Modbus aware and did not fulfill the project needs, either by not offering an integrated solution for all the steps or by lacking flexibility to adjust settings to the HEDVa scenario.

After the ARP spoofing, the attacker first starts by capturing the current state of the grid. This is achieved by dumping and decoding one complete interaction cycle (i.e. the set of Modbus request-reply transactions) between the HMI1 and all PLCs. This represents the initial state of the simulated view and it allows to restore the previously grid state after stopping the attack (in case the attacker wants to do so). The attacker is also responsible to perform deep inspection of each packet and selectively intercept all the TCP connections from

the HMI1 to the PLCs while forwarding the others (i.e. the communications between HMIs and PLCs).

When requests from the HMI1 are received, the attacker will compute the responses based on its own replica of the model (obtained during the process analysis stage). This effectively decouples the HMI1 from the PLCs, creating two distinct communication flows: one between the HMI1 and the attacker and the other one between the attacker and each PLC. This allows not only to hijack the data exchanged between them but also trigger any kind of service disruption against the PLCs compromising the physical process behind them.

Since the true state of the PLCs is hidden from HMI1, the attacker is free to do whatever he wants without the knowledge of the legit SCADA operator. Moreover, all the changes performed by the SCADA operator such as opening or closing a breaker are properly intercepted and handled by the attacker. Finally, whenever the attacker decides to stop the attack, he only needs to perform the inverse of the first steps, dumping the values of the simulated HMI1 view to the PLCs, so that there is no difference between the HMI1 and PLC states, also restoring the ARP caches by sending additional unsolicited ARP replies with the correct associations between MAC and IP addresses.

V. Conclusions and Future Work

The attack procedures here described illustrate a complete intrusion procedure applied to a specific IACS use case. The reconnaissance step is like other types of network scans, the main difference is the Modbus unitID field, depending on the components and how they are deployed. The service disruption is also straight-forward since as soon as the attacker has access to the network, it is simple to redirect Modbus traffic (causing the disruption) or even flood the PLCs, as they typically have moderate / small amount of resources available. The communication hijacking attack that was implemented has proven to be considerably more complex and tightly coupled to the field processes in the SCADA environment than, for instance, a HTTP hijacking attempt. This is due to several reasons, such as the need to reproduce part the physical process behavior without getting detected.

Despite new infection paths, types of attacks or strategies to get unnoticed, further efforts and research should focus on improving the process of recreate and maintain the fake views used by the attacker during the communication hijacking and for specific known domains like energy grids.

This work is part of a wider effort where multiple cyber detection technologies are being researched to understand how these types of cyber security events could be adequately handled. Moreover, this effort also intends to alleviate the lack of open available datasets (such as raw traces from SCADA IACS) allowing to further explore and research new security approaches and detection mechanisms.

Acknowledgment

This work was partially funded by the CockpitCI European Project (FP7-SEC-2011-1 Project 285647) and by the ATENA European Project (H2020-DS-2015-1 Project 700581).

References

[1] ISA, “ISA-62443-1-1 security for industrial automation and control systems part 1: Terminology, concepts, and models draft 5,” International Society for Automation, 2015.

[2] NIST, “800-82,” Guide to Industrial Control Systems (ICS) Security, Rev. 2, National Institute of Standards and Technology, 2015.

[3] ISA-99.00.01, Security for Industrial Automation and Control Systems - Part 1: Terminology, Concepts, and Models, American National Standard. 2007.

[4] FP7 CockpitCI Research Project, https://www.cockpitci.eu/

[5] H2020 ATENA Research Project, https://www.atena-h2020.eu/

[6] T. Cruz, L. Rosa, J. Proenca, L. Maglaras, M. Aubigny, L. Lev, J. Jiang, P. Simoes, A cyber security detection framework for supervisory control and data acquisition systems, IEEE Transactions on Industrial Informatics, Preprint. doi:10.1109/TII.2016.2599841

[7] T. Cruz, J. Proença, P. Simões, M. Aubigny, M. Ouedraogo, A. Graziano, L. Maglaras, A Distributed IDS for Industrial Control Systems, International Journal of Cyber Warfare and Terrorism, 4(2), 1-22, April-June 2014. DOI: 10.4018/ijcwt.2014040101

[8] C. Queiroz, A. Mahmood, J. Hu, Z. Tari, and X. Yu, “Building a scada security testbed,” in Network and System Security, 2009. NSS’09. Third International Conference on, pp. 357–364, IEEE, 2009.

[9] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri, “A testbed for analyzing security of SCADA control systems (tasscs),” in Innovative Smart Grid Technologies, 2011 IEEE PES, pp. 1–7, 2011.

[10] S. Bhatia, N. Kush, C. Djamaludin, J. Akande, and E. Foo, “Practical modbus flooding attack and detection,” in Proceedings of the 12th Australasian Information Security Conference-Volume 149, pp. 57–65, Australian Computer Society, Inc., 2014.

[11] 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 Comm. Quality and Reliability, 2015 IEEE International Workshop Technical Committee on, pp. 1–6, 2015

[12] E. E. Miciolino, G. Bernieri, F. Pascucci, and R. Setola, “Communications network analysis in a SCADA system testbed under cyber-attacks,” in Telecommunications Forum (TELFOR) 2015 23rd, pp. 341–344, 2015.

[13] D. Chen, Y. Peng, and H. Wang, “Development of a testbed for process control system cybersecurity research,” in 3rd International Conference on Electric and Electronics, Atlantis Press, 2013.

[14] R. Langner, “To kill a centrifuge a technical analysis of what stuxnet’s creators tried to achieve,” The Langner Group, November 2003.

[15] Laboratory of Cryptography and System Security (CrySyS), Duqu: A Stuxnet-like malware found in the wild, http://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf.

[16] “Blackenergy & quedagh: the convergence of crimeware and apt attacks.”,https://www.fsecure.com/documents/996508/1030745/blackenergy_whitepaper.pdf.

[17] Modbus Organization, “Modbus protocol specification,”

[18] Modbus Organization, “Modbus messaging on TCP/IP implementation guide”

[19] IMS Research, “The World Market for Industrial Ethernet – 2013 Edition”.

[20] “Nmap scripting engine-modbus-discover nse script.”, https://nmap.org/nsedoc/scripts/modbus-discover.html.

[21] Mark Bristow, Modscan, https://code.google.com/archive/p/modscan/

[22] A. Gervais, “Modbus/TCP library for scapy 0.1.