Received February 21, 2019, accepted March 12, 2019, date of publication March 22, 2019, date of current version April 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2906926

A Comprehensive Security Analysis of a SCADA Protocol: From OSINT to Mitigation

LUIS ROSA[1], MIGUEL FREITAS[1], SERGEY MAZO[2], EDMUNDO MONTEIRO[1], (Senior Member, IEEE), TIAGO CRUZ[1], (Senior Member, IEEE), AND PAULO SIMÕES[1], (Member, IEEE)

[1]Centre for Informatics and Systems, University of Coimbra, 3030-290 Coimbra, Portugal
[2]Israel Electric Corporation, Haifa 31000, Israel

Corresponding author: Luis Rosa (lmrosa@dei.uc.pt)

This work was supported by the ATENA H2020 EU Project (H2020-DS-2015-1 Project) under Grant 700581.

ABSTRACT It is an established fact that the security of Industrial Automation and Control Systems (IACS) strongly depends on the robustness of the underlying supervisory control and data acquisition (SCADA) network protocols (among other factors). This becomes especially evident when considering the extent to which certain protocols, designed with poor or nonexistent security mechanisms, have led to a considerable number of past incident reports affecting critical infrastructures and essential services. Considering the current situation, it is rather obvious why the proper auditing and analysis of SCADA protocols are considered as key when it comes to design and/or protect IACS infrastructures. However, while the security of some protocols, such as Modbus or DNP3, has already been extensively analyzed, the same cannot be said for other protocols and technologies being used in the same domain that have not received the same amount of attention. In this paper, we provide a comprehensive security analysis of the PCOM SCADA protocol, including a dissection of PCOM, a demonstration of several attacks scenarios on PCOM-based systems, and also an analysis of possible mitigation strategies against these potential attacks. Moreover, this paper also describes a number of open-source tools that we developed for further analysis and research of PCOM security aspects, including a PCOM Wireshark dissector, a Nmap NSE PCOM scan, multiple Metasploit PCOM modules, a set of Snort PCOM rules, and several network traffic datasets containing multiple samples of different types of PCOM operations.

INDEX TERMS SCADA, security, PCOM, ICS, IACS.

I. INTRODUCTION

Supervisory control and data acquisition (SCADA) systems are a subset of the Industrial Automation and Control Systems (IACS) that monitor and control Essential Services such as power grids, water distribution facilities and automated factories. They include sensors and actuators, Programmable Logic Controllers (PLCs) / Remote Terminal Units (RTUs), Human-Machine Interfaces (HMI)s and all the related servers, arranged in several network segments and levels. Different types of communications, using specific SCADA network protocols, occur between those levels, both vertically and horizontally [1].

While SCADA systems were originally confined to an isolated domain, several factors – such as the introduction of Manufacturing Execution Systems (MES) – have pushed for further integration of the process control and IT domains within organizational infrastructures. Consequently, the IACS vulnerability and threat profiles have increasingly become similar to their IT counterparts, a situation further aggravated by the fact that most SCADA communication protocols still lack adequate security mechanisms. An often mentioned example is the Modbus protocol [2], which was only recently updated to include security mechanisms [3].

The IACS/SCADA market fragmentation makes it further difficult to address all the security needs for each vendor and communication protocol. Many protocols are found in the field, often vendor-specific. Some are closed or not properly

The associate editor coordinating the review of this manuscript and approving it for publication was Noor Zaman.

documented, therefore their security enforcing mechanisms are unknown, not extensively discussed nor validated. Even Modbus, now an open and widely accepted protocol (despite not being a formal standard), still accommodates user-defined and reserved function codes that may differ from vendor to vendor. While such features may increase flexibility, they pose additional security problems if not properly implemented and documented.

There is now extensive literature addressing the security of SCADA protocols, but it clusters around a very small subset of the most used and well-known protocols. Many real-world SCADA systems include devices and protocols left outside the focus of existing literature. However, the lack of discussion, public exploits or tools for those less known protocols does not mean the protocol is safer.

Such a security analysis is important to continuously raise awareness for the security issues affecting SCADA systems and to prevent their appearance in the next generation of SCADA communication protocols. For instance, and to a certain extent, the recent security-oriented revision of the Modbus protocol [3] can be considered to be the ultimate result of a broad discussion involving security researchers, equipment providers and end-users - this would not be possible without creating interest and awareness about the existing security issues and their implications.

In this paper we provide a comprehensive security assessment of the PCOM SCADA communication protocol [4]. PCOM is used by a wide range of PLC devices and has not yet been covered by the literature, therefore constituting a good example of the aforementioned range of unaddressed but relevant SCADA protocols. More specifically, this paper provides the following contributions:

- A security analysis of the PCOM protocol and a discussion on how such a SCADA protocol can be subverted and used by a malicious attacker to ultimately gain full control over the physical processes and underlying infrastructures.
- A description of multiple open-source contributions to security-related tools, directly deriving from this PCOM analysis, including contributions to widely used tools [5] such as Wireshark [6], Metasploit [7] and Snort [8] and Nmap [9].
- Datasets containing several samples of PCOM network traffic for various operation and attack scenarios, which can be used for future research.

Besides the inclusion of a comprehensive security analysis of the PCOM protocol, which was started from scratch and combining the attacker and the defendant viewpoints, this paper also provides a step-by-step overview of the methodology that was adopted in the process, constituting, to the best of our knowledge, one of the first tutorial-like descriptions of such procedures and a valuable guide to anyone planning to perform a similar study of other SCADA protocols.

The remainder of this document is structured as follows. Section II provides an overview of the existing literature and SCADA related security tools. Section III starts with a PCOM primer, followed by a description of the Wireshark PCOM dissector that was implemented to support the protocol analysis process. Section IV goes through a set of cyberattack scenarios that explore the security gaps found in PCOM, while Section V explains how automated fuzzing strategies might be used to further assess the PCOM protocol flaws. Section VI discusses different types of strategies to mitigate the disclosed security issues. Section VII presents the PCOM capture dataset that was generated for evaluation purposes, being also contributed to the public domain. Finally, section VIII draws a series of conclusions, resulting from the effort hereby documented.

II. RELATED WORK

SCADA security has been widely discussed [10] [11] [12] and several security design flaws have already been identified. The lack of authorization, authentication and encryption in popular SCADA protocols such as Modbus, DNP3, EthernetIP/CIP or IEC 60870-5-104 have been one of the main discussion points. Several types of attacks have been referred in the literature [13], focused mainly on intelligence gathering, network reconnaissance, accessing and changing restricted process parameters and, finally, disrupting the physical processes under control (with its inherent consequences). Moreover, since most of the protocols are based on plain-text communications, such tasks become a matter of using the right function codes and either establishing direct connections or hijacking TCP/IP sessions on-the-fly. Other types of incidents against different levels of the SCADA reference architecture [14] have also been reported.

Win32/Industroyer, a major publicly-disclosed real-world malware [15], provides a good example of how recent malware combines the use of multiple SCADA protocols. Win32/Industroyer was specifically built for affecting electric power systems, focusing on specific SCADA protocols. It made use of a modular design, in order to accommodate four different protocols: IEC 60870-5-101 (for serial connections); IEC 60870-5-104 (for TCP/IP connections); IEC 61850; and OPC. Three SCADA-specific features should also be highlighted in the scope of that malware: the network device enumeration capability by parsing device responses of different protocols, the capability of accessing field values leveraging SCADA protocols that do not enforce proper security mechanisms and, finally, a DoS tool to send specifically crafted packets against Siemens SIPROTEC devices.

On the other hand, most of the SCADA-related security management solutions focus on monitoring such protocols to detect unwanted communications, either by (i) narrowing the allowed functions and remote communications or by (ii) detecting suspicious traffic patterns via statistic and machine learning algorithms.

In the following sections we provide an overview of the existing PCOM-related tools, followed by a broader analysis of tools and strategies that have been used in other SCADA protocols – which might also apply to PCOM.

A. PCOM-RELATED TOOLS

To the best of our knowledge, the security of PCOM is not discussed in the available literature, even though PCOM suffers from security issues similar to other SCADA protocols. To the best of our knowledge, the tools that support the PCOM protocol, such as Visilogic [16] and Crimson [17], are not designed with security assessment in mind. They are both full graphical products for Microsoft Windows. Crimson only allows reading/writing registers from/to PCOM-enabled devices. Visilogic allows additional administrative remote operations such as starting/stopping PLCs, but is not suitable for discovering devices on the network and cannot be used for detecting or blocking PCOM traffic in the network. A .Net driver that implements some additional internal functions, which are not part of the original PCOM specifications [4], is also available [18]. However, it lacks functions such as downloading/uploading ladder logic to/from PLCs or device enumeration. More recently, a basic Python implementation of PCOM for supporting the development of PCOM-enabled applications has been released [19].

B. TYPICAL ANATOMY OF SCADA ATTACKS

One of the first steps performed during a typical cyber-attack is *intelligence gathering* – collecting as much detail as possible for each asset in the target system. For networks in general, and more specifically for SCADA systems, this means discovering and enumerating devices such as PLCs and collecting their specific characteristics (manufacturer, model, firmware version, etc.) to look for known vulnerabilities. A classic network scan is useful to find responding IPs and open ports, but collecting details about PLC models and versions requires additional investigation. Moreover, in SCADA systems, a PLC might be configured to bridge serial segments which may hide additional PLCs that will not be disclosed, for instance, by TCP SYN Scans.

In [20], a Nmap NSE script is used to identify and enumerate Modbus devices, including Modbus slaves (please note that multiple Modbus slaves may be behind a single IP address). For the EtherNet/IP protocol, another Nmap NSE script, from [21], explores the lack of authentication and, by sending a Request Identification packet, is capable of retrieving multiple information from a device, such as the model, firmware, OS and hardware versions, serial numbers, etc. Other examples, such as those found in [22] and [23], may be used to find and identify specific details for several Siemens PLC models.

After the enumeration phase, since several SCADA protocols still use unencrypted TCP connections, it is typically possible to hijack TCP sessions or to simply establish new connections to PLCs, in order to *access or modify sensitive data*. For instance, a metasploit module available in [24] allows reading and writing different types of registers using standard Modbus functions. In addition to reading and writing registers, a PLC may sometimes be remotely shutdown, either by sending a valid command (from a malicious actor) or by exploring vulnerabilities in the PLC input validation. A metasploit module for Modbus [25], for instance, allows remotely starting and stopping a PLC using Modbus requests. For Ethernet/IP CIP, there are also several modules [26] [27] that explore application layer issues in the packet handling, that eventually lead to *Denial-of-Service (DoS)* conditions.

Reprogramming the entire logic of a PLC might also be possible, as demonstrated by [28], a metasploit module that allows downloading and uploading ladder logic code from/to Schneider Modicon PLCs.

C. MITIGATION STRATEGIES

Since the aforementioned attack steps require access to the SCADA process control network, it makes sense to consider the protection of the communications infrastructure as part of a mitigation strategy. One of the possible solutions for detecting, alerting or blocking such attacks is to use Network Intrusion Detection Systems (NIDS) or firewalls. Nevertheless, in order to have a perspective of what is going on the network, such solutions must be able to perform Deep Packet Inspection (DPI). For instance, read operations might be allowed but write and administrative operations might be blocked and reported. Repositories such as [29] contain collections of Snort rules for several SCADA protocols (Modbus, DNP3 and S7, among others) that might be used for detecting several types of operations, including network scans and Modicon PLC reprogramming attempts. Those rules specify portions of the TCP payload (based on the `offset` and the `depth` Snort keywords) and match them against hard-coded values (the signatures of each protocol).

For more complex protocols or interactions, it is recommended to have specific preprocessors able to decode specific protocol values that may happen to be not always at the same offset. Jordan [30] details how specific preprocessors for Modbus and DNP3 protocols allow creating syntax-richer rules by providing an extended list of keywords. Furthermore, a set of Ethernet/IP rules to be used with Suricata is described in [31].

D. WRAP-UP

To the best of our knowledge, there is a gap regarding the existence of specific studies or any sort of analysis about the PCOM protocol, from a security-oriented standpoint. This is a relevant issue if we take into account that both theoretical scenarios and real-world incidents, such as Win32/Industroyer, have demonstrated the extent to which the lack of security in SCADA protocols constitutes a relevant problem.

To address this lack of information regarding PCOM, the next sections provide a security analysis focused on understanding how the vulnerabilities and/or techniques identified in this section may apply to PCOM, allowing an attacker to disclose classified process data, to disrupt physical processes or even to introduce subtle process changes, by reprogramming PLCs.

Throughout the following sections, the aforementioned security tools (that were built for other SCADA protocols) were used as an inspiration for us to develop several open-source contributions, which can be used to replicate our experiments and for further studying the security of PCOM.

III. ANALYSIS OF THE PCOM PROTOCOL

This section introduces the main concepts of the PCOM network protocol and explains how we started from a protocol specification and ended up developing a protocol dissector, that was later used by in the context of this research to debug and understand additional details of the protocol.

A. A BRIEF PRIMER ON PCOM

PCOM [4] is a protocol that enables applications to communicate with PLC devices. Similar to Modbus, the protocol is based on requests and responses using command codes. Such codes identify the type of operation (e.g. reading a Memory Integer (MI) operand). This way, it can be used to continuously poll (or change) the values of a given set of PLC registers (e.g. process monitoring values), as well as for implementing other remote administrative interfaces.

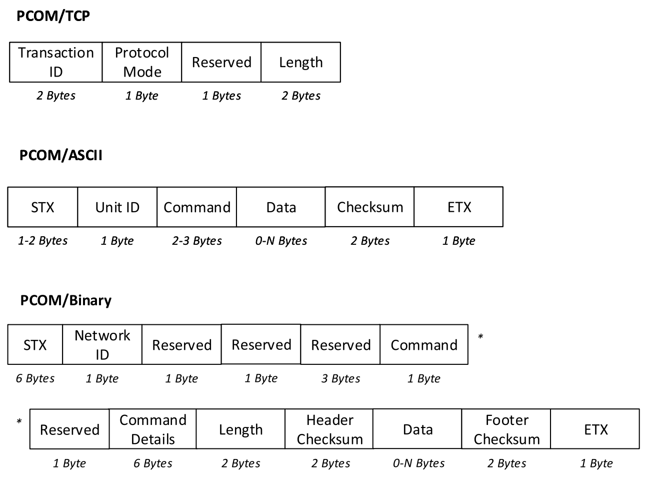


FIGURE 1. PCOM/TCP, PCOM/ASCII and PCOM/binary protocol structure (based on Unitronics specifications).

The communication may take place on top of different physical layers and field buses, including CAN (Controller Area Network) bus, RS-485 or Ethernet. It is also possible to have inter-PLC communication in master-slave schemes, where the master PLC acts as a bridge, forwarding all the requests and replies to/from the slave PLCs. A Unit ID field is used to uniquely identify a device on a network. For Ethernet networks, a special zero Unit ID value indicates a direct connection. Moreover, in such Ethernet networks, PCOM works on top of TCP sessions by adding an extra 6 bytes header in between the TCP header and original PCOM messages (PCOM/TCP). The protocol also supports two different modes: *ASCII* and *Binary*, hereafter referred as PCOM/ASCII or PCOM/Binary. Fig. 1 illustrates the structure of both modes. In PCOM/TCP, the communication socket in the PLC side defaults to TCP port 20256.

PCOM/ASCII allows reading and writing not just memory addresses (which are usually mapped from inputs and outputs), but also other types of operands and reserved values, such as System Bits (SB), System Integers (SI) and System Longs (SL). PCOM/Binary allows composed requests such as querying more than one type of operand in the same packet, using multiple data request blocks – opposed to only one type of operand per request in PCOM/ASCII. It also allows reading and writing PLC Data Tables – an operation not supported in the PCOM/ASCII mode.

PCOM capabilities go beyond reading and writing values. Remote administrative operations via specific command codes and parameters (e.g. reset a PLC, set the RTC value) are also possible and can even be used to reprogram the entire ladder logic of a PLC. Nevertheless, not all PLC models support PCOM/Binary or even PCOM/TCP.

Specialized applications such as Visilogic use PCOM to access and manage field devices such as PLCs.

B. BUILDING A DISSECTOR FOR PCOM MESSAGES

In order to develop an application, troubleshoot an erroneous behavior or even reverse-engineer the PCOM protocol to better understand the underlying communication, it is necessary to have a way of quickly parsing and analyzing PCOM messages.

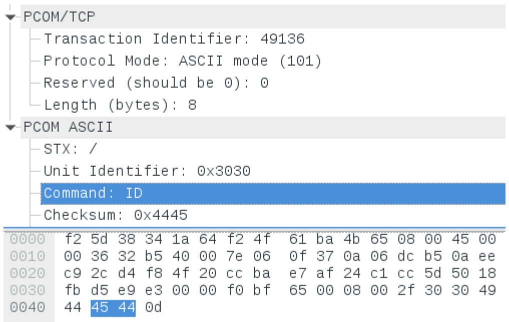
Distinguishing the protocol mode (i.e. ASCII or Binary) is based on the value of the third byte of the PCOM message, and the remainder of the packet may be parsed accordingly, since quickly spoting and grouping different types of operations depends on the protocol mode. For instance, in PCOM/ASCII the command code of PCOM responses is truncated to the first two chars, whereas in PCOM/Binary the same operation uses a different code, depending on whether it is a request or a response – the \x80 value is added to the command code value in replies.

Distinguishing between PCOM/ASCII and PCOM/Binary is also necessary to automatically handle and decode different field formats and endianness (e.g. the same Unit ID value is specified differently in PCOM/ASCII and PCOM/Binary) or even to compute and validate checksums (since requests containing a bad checksum are automatically discarded by the PLC).

A PCOM dissector may help to detect hidden features. For instance, based on the analysis of captured PCOM messages, we observed that the 9th to 11th bytes of PCOM/Binary messages, which according to the specification [4] should always be zero, are in fact not always zero. In PCOM/Binary messages with command code 41 (a request to a PLC, part of a multi-message operation, cf. section IV) those bytes start from zero but are incremented by one at each message. Our observations also showed that other reserved fields from the PCOM/Binary structure are sometimes used to specify data and do not always contain the values described in the specification.

Since we found no publicly available tool able to perform the dissection of PCOM messages, we developed our own Wireshark built-in PCOM/TCP dissector, which was later used to support the development and validation steps by helping to flag malformed packets and reveal undocumented features. This tool can dissect the PCOM/TCP header, as well as the header structure for both PCOM/ASCII and PCOM/Binary modes. It can also translate over 25 PCOM command codes into meaningful descriptions and dissect the details of PCOM/ASCII read and write content of `inputs`, `outputs`, `SBs`, `MBs`, `MIs`, `SIs`, `MLs`, `SLs`, `MDW`, `SDW` operands. Some fields were also defined as expert fields [32], in order to easily spot protocol violations, based on the same concept Wireshark already uses for flagging TCP anomalies based on sequence and acknowledgement numbers. The code of our tool has been integrated into the upstream Wireshark repository [33] [34] [35].

Developing this built-in dissector for Wireshark provided us a graphical and a command line interface to visualize the flows of PCOM messages, their structure and their content. Fig. 2 provides a snapshot of a PCOM packet dissection based on this tool.



```
▾ PCOM/TCP
   ├ Transaction Identifier: 49136
   ├ Protocol Mode: ASCII mode (101)
   ├ Reserved (should be 0): 0
   └ Length (bytes): 8
▾ PCOM ASCII
   ├ STX: /
   ├ Unit Identifier: 0x3030
   ├ Command: ID
   └ Checksum: 0x4445
0000  f2 5d 38 34 1a 64 f2 4f  61 ba 4b 65 08 00 45 00
0010  00 36 32 b5 40 00 7e 06  0f 37 0a 06 dc b5 0a ee
0020  c9 2c d4 f8 4f 20 cc ba  e7 af 24 c1 cc 5d 50 18
0030  fb d5 e9 e3 00 00 f0 bf  65 00 08 00 2f 30 30 49
0040  44 45 44 0d
```

FIGURE 2. **Snapshot of a PCOM command packet using the wireshark.**

From a security perspective, a protocol dissector also helps understanding policy violations such as abnormal and unauthorized messages (e.g. reflecting malicious network scouting or even PLC reprogramming attempts). It also becomes possible to use specific PCOM filters, such as `pcomascii.command == "RC"`, to search and understand the flow of communications and the used PCOM messages.

In the field of forensics, such protocol dissection framework helps to gather, filter and reconstruct evidences from network attacks. For instance, when programming a PLC, a set of specific PCOM commands (as described with detail in Section IV) are used to transfer files between an application and the PLC. The PCOM dissector enables quick detection by searching for file signatures, using Wireshark filters (e.g. `pcombinary.data contains "\x50\x4b\x03\x04"` for a PKZIP file format) and then reconstructing the file by looking at the content between the signature and the end of file (e.g. `pcombinary.data contains "\x50\x4b\x05\x06"` for a PKZIP file format). Generic tools to recover files from network traces, using TCP or HTTP payloads, would fail, since the final file results from a concatenation of the bytes of a specific PCOM field of a specific PCOM operation across multiple packets rather the entire TCP or HTTP payload.

IV. REFERENCE VALIDATION SCENARIO

Real world attacks often depend on specific details of the target system. For the sake of simplicity, in this paper we adopted a reference validation scenario which is simple enough to be quickly understood while still being fairly representative of larger and more complex scenarios.

Starting from a literature review including not only research works but also post-mortem analysis reports of past incidents, the validation scenario hereby presented was designed for threat modeling, vulnerability analysis and exploitation purposes. Regarding the latter point, we intend to demonstrate specific attack use cases which take advantage of PCOM's characteristics, including device scanning techniques, the possibility of accessing all the PLC data using unauthorized requests, how to disrupt the physical process under control via remote commands, or even how to reprogram the PLC to introduce subtle changes that might only be perceived in long term.

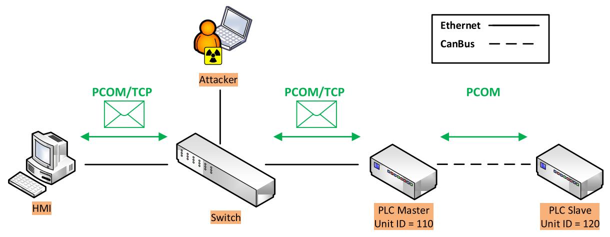


FIGURE 3. **Reference scenario for the validation tests.**

Fig. 3 illustrates the reference scenario that was implemented on a testbed, being used to perform attacks, to develop the auxiliary tools, and to collect the respective datasets. Two different physical layers are used: an Ethernet segment and a **CAN** [36] bus. The scenario is composed of two Unitronics V130 PLCs, one of them connected only in the CAN bus and the other deployed in both segments, acting as a bridge. An HMI with Visilogic 9.8.65 plays the role of an authorized device/operator. Finally, it is assumed that the attacker has access to a compromised device in the same network as the HMI and the PLC Master – for instance, an operator workstation, a historian database or a network printer able to reach the field network segment.

This paper is focused only on PCOM network communications between HMIs and PLCs (i.e. corresponding to levels 0, 1 and 2, according to IEC 62443 standard), and no other protocol or service vulnerabilities are addressed. Therefore, we make no assumptions on how the attacker compromised that device. Nevertheless, incident records show that getting control of such devices is fairly common as an intermediate step of successful attacks.

The PLC runs a minimal working ladder logic example where two sockets were initialized, one for PCOM/TCP and the other for Modbus. A single ladder `OR` element (cf. Fig. 4), representing the process logic, was used to compute the input of two `MIs` (`MI1` and `MI2`), into a third one `MI3`, representing the field data.

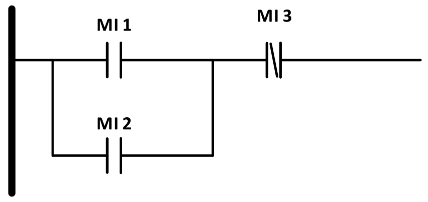


FIGURE 4. Ladder logic encoding/representation of the logical process.

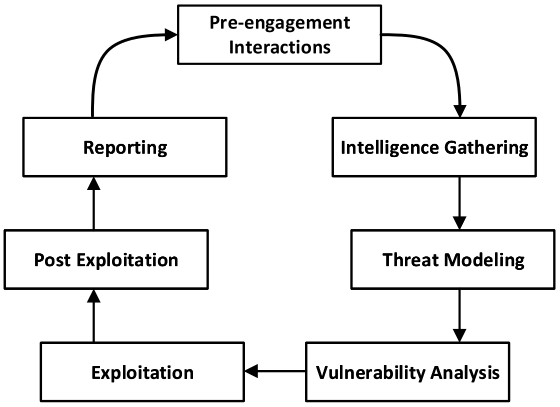


FIGURE 5. PTES assessment main steps (according to [37]).

In a real SCADA system, such registers may have different interpretations, from circuit switch states on a electrical distribution grid to reservoir levels in water treatment systems. Even though in real SCADA environments the number of used registers is larger and a significantly more complex functional logic is expected to be found, our reference scenario allow us to verify whether we can access and/or change register values or change process logic.

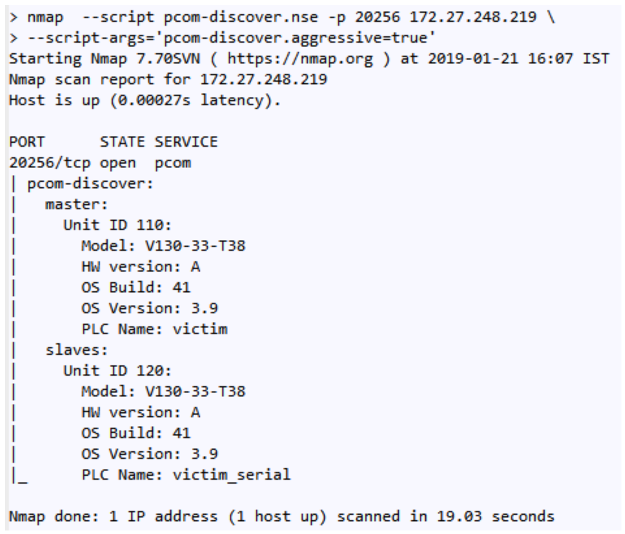
Given the lack of authentication, authorization and encryption, PCOM-based systems are vulnerable at least to the following main classes of attacks: direct connection and session hijacking attacks. The following sections discuss how each of the security issues of PCOM might be explored by a malicious attacker in the context of a SCADA system.

A. NETWORK SCOUTING

Formal security assessment methodologies such as the Penetration Testing Execution Standard (PTES) [37], illustrated in Fig. 5, typically specify information gathering as one of the first steps for understanding the environment and collecting as much information as possible. Structured cyber-attacks usually start by collecting as much information as possible about target systems. Typical network scans such an SYN Scans are good at identifying hosts on a network, but they fail to identify specific host details. Therefore, they must be complemented with fingerprinting checks and additional scripts to collect specific host characteristics.

The PCOM protocol allows unauthenticated queries to PLCs that can be used to retrieve, among others, the PLC model, the hardware version, the OS build and OS version, the PLC name and the UnitID value. During our research, we developed a NMAP NSE script to collect such detailed information using PCOM/ASCII and PCOM/Binary commands [38].

Fig. 6 shows the output of a scan (using the aforementioned PCOM NMAP NSE script) on our scenario. The script sends PCOM requests with command code `ID` and the UnitID field set to `00`, exploring two weaknesses, namely: (1) no authentication is required to communicate with a PLC and (2), the use of UnitID `00`, will make any connected PLC to respond, no matter with which UnitID was configured. The PCOM/ASCII command code `UG` is then used to retrieve the actual PLC Unit ID, and PCOM/Binary command value `0x0C` is used to retrieve the PLC name. Algorithm 1 shows the simplified pseudo-code behind the aforementioned described scan script.



```
> nmap  --script pcom-discover.nse -p 20256 172.27.248.219 \
> --script-args='pcom-discover.aggressive=true'
Starting Nmap 7.70SVN ( https://nmap.org ) at 2019-01-21 16:07 IST
Nmap scan report for 172.27.248.219
Host is up (0.00027s latency).

PORT      STATE SERVICE
20256/tcp open  pcom
| pcom-discover:
|   master:
|     Unit ID 110:
|       Model: V130-33-T38
|       HW version: A
|       OS Build: 41
|       OS Version: 3.9
|       PLC Name: victim
|   slaves:
|     Unit ID 120:
|       Model: V130-33-T38
|       HW version: A
|       OS Build: 41
|       OS Version: 3.9
|_      PLC Name: victim_serial

Nmap done: 1 IP address (1 host up) scanned in 19.03 seconds
```

FIGURE 6. Snapshot of the NMAP output using the PCOM Nmap NSE script.

Algorithm 1 PCOM/TCP Network Scan

```
1:  $unitId \leftarrow 0$
2:  $master_t \leftarrow$ getUnitID($unitId$)            ▷ PCOM/ASCII
3:  $master_t \leftarrow$ getID($unitId$)                ▷ PCOM/ASCII
4:  $master_t \leftarrow$ getPlcName($unitId$)           ▷ PCOM/Binary
5:  if $aggressiveMode$ then
6:      for $unitId \leftarrow 1, 127$ do
7:          if $unitID \neq master_{id}$ then
8:              $slaves_t \leftarrow$ getID($unitId$)
9:              $slaves_t \leftarrow$ getPlcName($unitId$)
10:         end if
11:     end for
12: end if
```

As a technical remark, it should be noted that, at each step of the loop, the script needs to adjust to the fact that even though the PLC replies with a TCP acknowledgement to all received requests (including those for non-existent UnitID's), it does not send any PCOM reply if it doesn't hold the matching UnitID. It is therefore necessary to iterate across the whole range of possible UnitID values, which is fairly short anyway (1 to 127). Moreover, if several requests are sent in a burst they might be rejected, therefore it is possible to adjust the rate of this interaction.

The information that was gathered allows an attacker to look-up for potential vulnerabilities and exploits applicable only to certain models or firmware versions. It may be used, for instance, to identify which commands a specific PLC may accept – accordingly to the PCOM specification [4] not all models support both PCOM modes or operands.

Both the Visilogic software and the PCOM .Net driver support the `ID` command used during the scan. Nevertheless, Visilogic is not suitable for performing enumeration tasks, and using a NMAP script eliminates the need for developing and packaging an entire application from scratch based on the .Net driver. Moreover, by using NMAP, we get convenient added features such as portability, a powerful command-line interface and the possibility of integration with other toolchains, on a single, well maintained open-source tool that can be used to assess multiple SCADA devices from different vendors, regardless the communication protocol.

B. ACCESSING SENSITIVE DATA

PCOM is vulnerable to all sorts of layer 2 and layer 3 attacks, and such vulnerabilities can be used to access all the sensitive data hold by a PLC. Fig. 7 illustrates two possible classes of attacks against PCOM, discussed next.

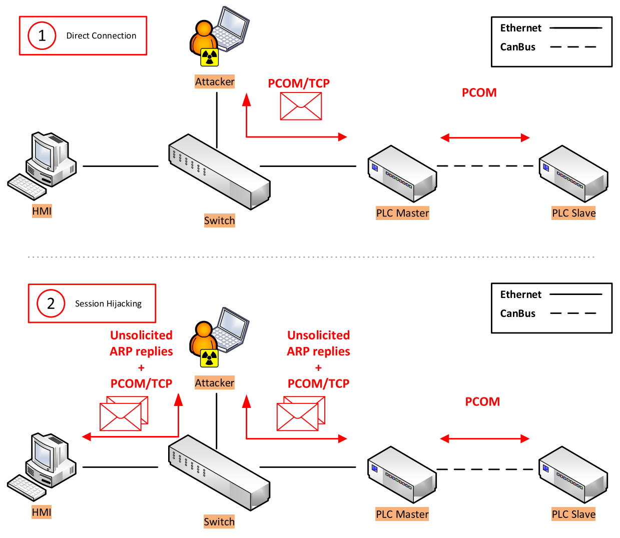


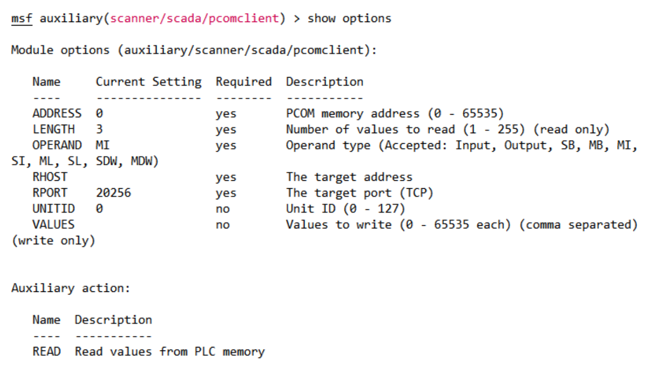
FIGURE 7. Direct TCP connection and Man-in-Middle attacks.

Direct TCP connections can be used to query not only process-related values but also all types of operands, without restriction, including inputs, outputs, System Bits (SIs), Memory Integers (MIs), etc. This allows not only to access values that are related to the process under control, but also to change configuration parameters (such as the network settings, in `SI[101-148]` or the info mode password, in `SI[253]`) or even to disrupt the PLC operation, by setting system registers (e.g. `SB[314]`) which can be used to block the communications between legitimate nodes and the PLC).

Complex main-in-the-middle (MITM) scenarios where a PCOM/TCP session is intercepted and redirected via an attacker-compromised device are also possible, for instance, by means of an ARP poisoning attack against the communication endpoints. Such attacks are not specifically related to PCOM, but can be used against PCOM communications due to their lack of confidentially and integrity protection. In [39] we provide a detailed discussion (although based on the Modbus protocol) of how such MITM attacks can be deployed on SCADA systems.

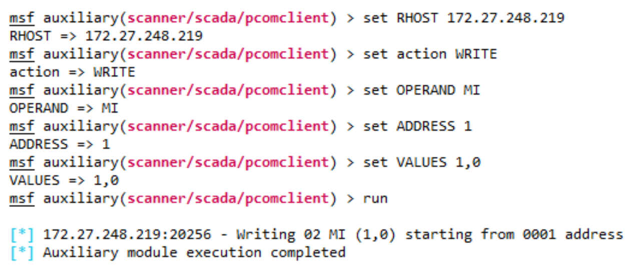
Moreover, and similarly to what was done in the scanning step, an attacker positioned in an Ethernet segment can also reach the PLC in the CAN bus via the master PLC, simply by specifying its Unit ID in the PCOM/TCP requests.

In the course of our work, we developed a Metasploit auxiliary module that can be used to read and write PLC registers by selecting the Unit ID, the operand type, the address and the number of values to read (or by providing the values to write) [40]. This module supports the operands `inputs`, `outputs`, `SBs`, `MBs`, `MIs`, `SIs`, `MDW`, `SDW`, `MLs` and `SLs`. Moreover, other commands are also supported by providing the raw hexadecimal of the PCOM/ASCII data payload, with the other fields being automatically computed – including the PCOM/ASCII checksum the and PCOM/TCP header. Fig. 8 showcases the main options of the Metasploit PCOM module.



```
msf auxiliary(scanner/scada/pcomclient) > show options

Module options (auxiliary/scanner/scada/pcomclient):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   ADDRESS  0                yes       PCOM memory address (0 - 65535)
   LENGTH   3                yes       Number of values to read (1 - 255) (read only)
   OPERAND  MI               yes       Operand type (Accepted: Input, Output, SB, MB, MI,
SI, ML, SL, SDW, MDW)
   RHOST                     yes       The target address
   RPORT    20256            yes       The target port (TCP)
   UNITID   0                no        Unit ID (0 - 127)
   VALUES                    no        Values to write (0 - 65535 each) (comma separated)
(write only)


Auxiliary action:

   Name  Description
   ----  -----------
   READ  Read values from PLC memory
```

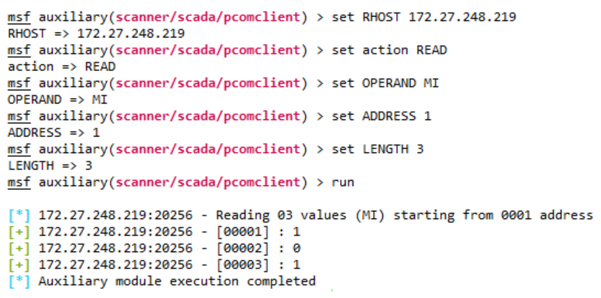
FIGURE 8. Snapshot of Metasploit auxiliary PCOM client options.



```
msf auxiliary(scanner/scada/pcomclient) > set RHOST 172.27.248.219
RHOST => 172.27.248.219
msf auxiliary(scanner/scada/pcomclient) > set action WRITE
action => WRITE
msf auxiliary(scanner/scada/pcomclient) > set OPERAND MI
OPERAND => MI
msf auxiliary(scanner/scada/pcomclient) > set ADDRESS 1
ADDRESS => 1
msf auxiliary(scanner/scada/pcomclient) > set VALUES 1,0
VALUES => 1,0
msf auxiliary(scanner/scada/pcomclient) > run

[*] 172.27.248.219:20256 - Writing 02 MI (1,0) starting from 0001 address
[*] Auxiliary module execution completed
```

FIGURE 9. Writing registers of the reference scenario PLC using Metasploit PCOM client.

Fig. 9 and Fig. 10 illustrate the usage of the Metasploit PCOM module to write (and, later, to read) the values of the reference scenario via PCOM/ASCII requests. In the write operation it is necessary to specify the operand type, the starting address and the values for the first two registers (the third



```
msf auxiliary(scanner/scada/pcomclient) > set RHOST 172.27.248.219
RHOST => 172.27.248.219
msf auxiliary(scanner/scada/pcomclient) > set action READ
action => READ
msf auxiliary(scanner/scada/pcomclient) > set OPERAND MI
OPERAND => MI
msf auxiliary(scanner/scada/pcomclient) > set ADDRESS 1
ADDRESS => 1
msf auxiliary(scanner/scada/pcomclient) > set LENGTH 3
LENGTH => 3
msf auxiliary(scanner/scada/pcomclient) > run

[*] 172.27.248.219:20256 - Reading 03 values (MI) starting from 0001 address
[+] 172.27.248.219:20256 - [00001] : 1
[+] 172.27.248.219:20256 - [00002] : 0
[+] 172.27.248.219:20256 - [00003] : 1
[*] Auxiliary module execution completed
```

FIGURE 10. Reading registers of the reference scenario PLC using Metasploit PCOM client.

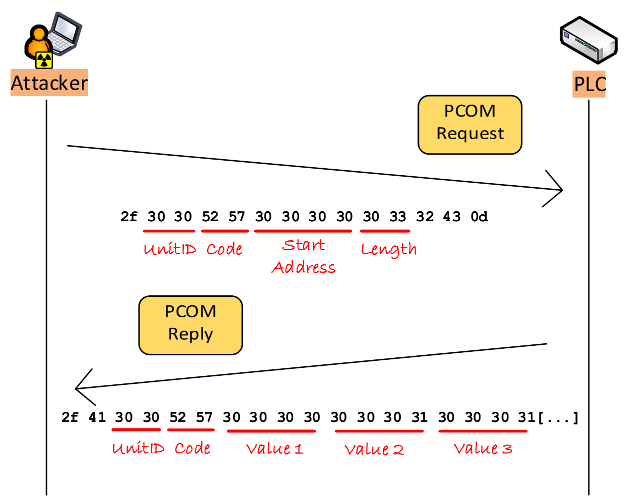

Attacker
PLC
PCOM Request
2f 30 30 52 57 30 30 30 30 30 33 32 43 0d
UnitID
Code
Start Address
Length
PCOM Reply
2f 41 30 30 52 57 30 30 30 30 30 30 30 31 30 30 30 31[...]
UnitID
Code
Value 1
Value 2
Value 3

FIGURE 11. Example of PCOM/ASCII request and reply in the reference scenario.

one is always rewritten according to the OR operation). In the read mode it is necessary to specify the number of registers to read from the starting address.

Fig. 11 depicts the PCOM/ASCII request and reply messages used to read the three MI PLC operands of our reference scenario.

C. DENIAL OF SERVICE (DOS) ATTACKS

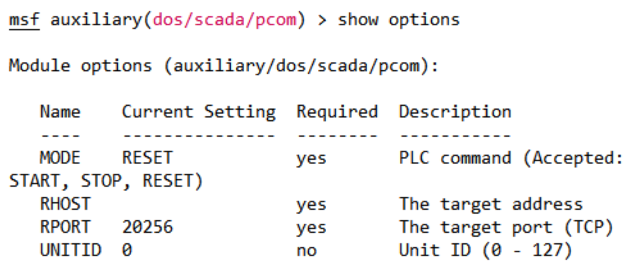
Responsible pentesting procedures for ICS [41] often recommend special caution when handling PLCs. This is due to the fact that it is not uncommon to find devices which are vulnerable against diverse situations, which may or not be intentionally triggered, such as single packets triggering input handling bugs (eventually generated by device fingerprinting procedures) or even brute force attacks causing resource exhaustion (such as SYN floods). This may lead to DoS scenarios, with manifold consequences.

DoS attacks are especially relevant in the Critical Infrastructure domain, due to their potential impact. An ICS-targeted DoS may have disastrous consequences, ranging from service or production interruptions (availability is considered a topmost priority for most automation infrastructures) to physical damage or even loss of human life [42].

By analyzing the network traffic generated by OEM applications such as [16] [17], we were able to identify a series of reserved commands used for PLC administrative purposes, which are not part of the command set documented in [4]. When used for their legitimate purpose, these commands provide a convenient way to remotely control a PLC, allowing for an operator to recover it from a failure or to reset the device after a reprogramming operation.

However, due to the lack of authentication and authorization mechanisms, an attacker can abuse administrative commands for malicious purposes. In particular, the STOP or the RESET commands can be used to prevent a PLC from communicating with the HMI or other PLCs. Other scenarios are also possible: for instance, an attacker may hijack a TCP connection to block RESET or START operations triggered by the legitimate operator. In this case, on top of traditional Address Resolution Protocol (ARP) poisoning, the attacker should acknowledge such PCOM/TCP requests so the sender believes they were successful, while preventing them from reaching the PLC.

In order to study these vulnerabilities, we developed another Metasploit auxiliary module that is able to send several PCOM/ASCII administrative commands – namely start, stop and reset operations [43]. While we managed to discover and explore these non-disclosed commands in PCOM/ASCII mode, our research showed no evidence of similar commands in the PCOM/Binary mode. Fig. 11 illustrates the options available in the Metasploit module.



```
msf auxiliary(dos/scada/pcom) > show options

Module options (auxiliary/dos/scada/pcom):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   MODE     RESET            yes       PLC command (Accepted:
START, STOP, RESET)
   RHOST                     yes       The target address
   RPORT    20256            yes       The target port (TCP)
   UNITID   0                no        Unit ID (0 - 127)
```

FIGURE 12. Snapshot of available options in the admin pcom metasploit module.

D. REPROGRAMMING THE PLC

Another interesting attack strategy is to reprogram the PLC, in order to modify its behavior, for instance, to induce permanent damage to the physical equipment or process under control. Moreover, a skilled attacker may use this technique as part of a long-term strategy, by introducing subtle changes to the process control tasks that can go unnoticed for a long time, eventually only being detected when permanent damage is already unavoidable (as demonstrated by the well-known Stuxnet incident [44]). This is a sharp contrast when compared to the almost immediately noticeable effects of other attacks, such as DoS.

PCOM supports several options for pushing the ladder logic project to the PLC RAM or flash memory. Although actively used by applications [16] [17], such options and their specifications in terms of the PCOM packet structure are not publicly documented.

Nevertheless, there is an option that allows pushing (and, later, recover) the PLC project, which is used by [16]. If the PLC was programmed with that option, the lack of authentication and authorization makes it possible to use a rogue setup to retrieve the original project, change it as desired, and download it again to the PLC. Moreover, since no encryption is used, it is also possible to reconstruct the entire packet sequence or even to change such packets on-the-fly to recreate a MITM attack.

From the network traffic, we observed a multi-part PCOM/Binary operation (with command code `0x41`) was used to transfer a relatively large data block to the PLC. Moreover, it was possible to group all the message parts, contained in the PCOM/Binary data field, since each one is identified by little-endian sequential value (starting from 0 and incremented by 1) at bytes 16th to 18th. Additionally, it was also possible to identify the data format, a plain PKZIP file by looking for its signature (e.g. using PCOM wireshark dissector `pcombinary.data contains "\x50\x4b\x03\x04"`). Similarly, when uploading the PLC project to Visilogic, a PKZIP file is also transferred from the PLC. Since PCOM communications are not encrypted, as long as the malicious attacker is able to access the communication, he can reconstruct this file.

Moreover, the PKZIP file, not encrypted, contains a single file inside, a Microsoft Jet4 Database. The database is protected by a master password. Nevertheless, Jet4 databases are known to obfuscate the password in the file header, so the database password can be easily guessed. In the aforementioned scenario, as an example, we were able to change the `XOR` ladder element to an `AND` element simply by updating its element type within the database, after guessing the database password. This required experimentation with several undocumented functions and values – something which might break things, if not done correctly. A simpler and more viable path in some attack scenarios, would be to use a rogue application deployment to retrieve and reprogram the PLC ladder logic.

V. FUZZING THE PCOM PROTOCOL

In the previous section several attack cases were described, using our reference PCOM scenario to illustrate their fundamental operation and deployment procedures. These are classic "textbook" attacks which we were able to perform after conducting an investigation about the inner workings of the PCOM protocol, based on publicly available documentation and analysis of network traffic.

In order to go beyond, a network fuzzer can be used. Network fuzzers are tools that automate and test different types of input conditions by generating random or semi-randomized values. For the PCOM protocol, known functions could be validated, for instance, against malformed packets such as protocol violations, bad check-sums, unexpected fields values, etc.

Fuzzing the PCOM protocol might also be useful to explore undocumented functions, protocol vulnerabilities or simply to discover specific inputs that might crash a PLC (e.g. a buffer overflow), using an automated procedure. This automation is also valuable if we intend to assess how different PLC models handle exceptional conditions or to discover supported PCOM functions in each model. Although not part of our research, those vulnerabilities might apply not only to PCOM PLCs but also to software applications used in the HMI side.

In the context of our research, we developed a PCOM fuzzer, in the form of a Metasploit auxiliary module. This module allows specific testing options such as sending randomized values of some specific protocol fields (e.g. PCOM/ASCII command codes) or completely random messages.

TABLE 1. Summary of results of fuzzing.

ID	Mode	Description	Result
1	Both	Invalid PCOM/TCP Length	PLC accept requests
2	ASCII	Consecutive connections	PLC rejects too fast connections
3	Both	Bad check-sum	No reply from PLC
4	ASCII	Non-existent Unit IDs	No reply from PLC
5	ASCII	Random command codes	No reply
6	ASCII	R&W with random params	No reply or invalid

Table 1 provides a brief summary of the tests and results performed during the fuzz experiments we conducted. Based on test 1, we observed the tested PLCs accept and reply to malformed-packets containing invalid PCOM/TCP length value. Nevertheless, based on tests 3 and 4, it appears that the tested PCOM PLCs acknowledge all the TCP requests but do not respond to bad PCOM requests (e.g. they do not respond to packets with bad checksums or non-existent Unit IDs). Opposed to other protocols, PCOM does not seem to return any exception or code to invalid requests. This makes it harder to identify all the hidden functions and features of a given PLC – which is positive from a security point-of-view. Moreover, the tested PCOM PLCs seem to reject consecutive connections if they are sent too fast (test 2), forcing scouting processes to slow down. Moreover, we also tested invalid combinations of both command codes and command data fields. We observed non-conclusive results, as the tested PLCs sometimes do not reply or reply with a invalid or empty results. Nevertheless, none of the combinations we tried resulted in a PLC DoS condition – which is also positive from a security point-of-view.

VI. MITIGATION STRATEGIES

Several strategies might be pursued to address the security shortcomings of the protocols such as PCOM. One of the first approaches, if not the most obvious, would imply a complete redesign of the protocol to add the missing security features.

As several SCADA vendors are already moving away from legacy protocols, this could be an opportunity to pursue a balanced approach, by keeping most of the PCOM semantics and adding the missing authentication, access control, confidentiality and integrity functionalities. For instance, the Modbus

Organization released in October 2018, a new Modbus/TCP Security protocol specification [3]. New conforming devices should use TLS 1.2 or better to achieve confidentiality and data integrity on the top of TCP sessions avoiding all sort of replay and MITM attacks. Whereas the authentication and authorization rely on x.509v3 certificates and a combination of an AuthZ function and a Roles-to-Rights Rules Database to determine if the requested Modbus functions code is authorized or not. Both, authorization algorithm and database, are vendor specific.

Unfortunately, the authors have no evidence pointing towards a possible improvement, redesign or replacement of the PCOM protocol. Considering this situation, several strategies may be considered to mitigate the aforementioned issues, which will be described next. Rather than definitive solutions (something that would imply revising the protocol), these proposals intend to provide the means to detect and block potential unwanted communications in existing scenarios.

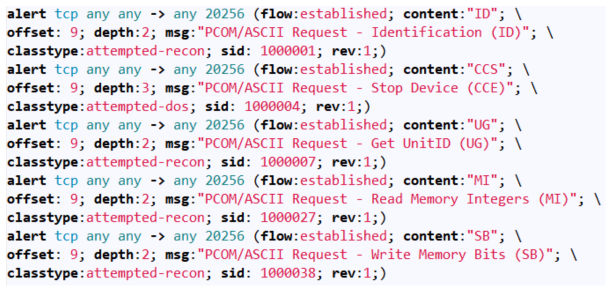
A. IMPROVING DETECTION CAPABILITIES

Network IDS (NIDS) are among the most effective tools when it comes to deal the security issues of legacy or insecure protocols (as it is the case for PCOM). The role of a NIDS is to distinguish legitimate traffic patterns from malicious ones, offering a complementary approach to the fundamental role performed by traditional firewalls. Firewalls are helpful to shield the trusted perimeter and block the traffic based on TCP/IP header fields, such as an IP address or a TCP port, mostly playing a preventive role. Nevertheless, a fine control based on protocol features (e.g. determine which PCOM functions should be blocked and which ones are allowed) would require Deep Packet Inspection (DPI) and custom PCOM parsing.

But even when supporting DPI, classic firewalls often fall short when it comes to offer comprehensive protection against more sophisticated attacks. For instance, because PCOM is vulnerable to spoofing attacks, the single reliance on classic firewalls for protection is insufficient, requiring complementary solutions (e.g. ARP monitoring solutions such as Snort's ARP preprocessor, to detect ARP poisoning attacks). Moreover, because of the specific nature of the SCADA ecosystem, where availability has been traditionally favored over confidentially and integrity, the decision to specify traffic blocking or throttling reactions for specific rules is something often questioned, due to the potentially negative impact in terms of operational and safety levels, opting instead for simple event logging.

A NIDS such as Snort might be used to detect and report unwanted communications, but to be able to effectively block them it must be deployed on the middle of the communications path, providing intrusion prevention capabilities – this is referred as *inline mode* in the official Snort documentation. Inline mode has its own fair share of issues which makes it unpopular among SCADA operators: besides introducing a single point of failure in the communications path (by placing the Snort host in a traffic mediation point), there is also the possibility that a knowledgeable attacker may try to abuse it to deliberately drop legitimate traffic. On the other hand, if the NIDS deployment is only passive (offering pure detection capabilities, by just reporting alarms), specific PCOM attacks that only require a single PCOM packet to go through (like remotely shutdown a PLC) might be successful, despite being detected and reported. In the end, it all comes down to a tradeoff between prioritizing availability (passive mode) or having an effective reaction (inline mode) capabilities.

Independently of the deployment options, a Snort instance (or any other NIDS, for that purpose) can only be effective for a specific environment on condition that it is loaded with an adequate ruleset covering the protocol mix being used. But, to the extend of the authors' knowledge, PCOM was not supported by any mainstream NIDS. To address this gap, the authors released a set of Snort Rules to detect PCOM/TCP packets [45]. Each rule was designed to match a single command code, so that it is possible not only to distinguish administrative remote commands but also reads and writes of different types of operands. For instance, we might want to block system related operands (i.e. `SB`, `SI`, `SL`) and allow the access to other operands. We might also want to distinguish between read and writes in some of the operations. Each rule searches for specific byte values corresponding to the command codes using hard-coded positions. To improve the readability, those command values were specified using their ASCII representation for PCOM/ASCII and hexadecimal values for PCOM/Binary rules. Fig. 13 shows an excerpt of those rules.



```
alert tcp any any -> any 20256 (flow:established; content:"ID"; \
offset: 9; depth:2; msg:"PCOM/ASCII Request - Identification (ID)"; \
classtype:attempted-recon; sid: 1000001; rev:1;)
alert tcp any any -> any 20256 (flow:established; content:"CCS"; \
offset: 9; depth:3; msg:"PCOM/ASCII Request - Stop Device (CCE)"; \
classtype:attempted-dos; sid: 1000004; rev:1;)
alert tcp any any -> any 20256 (flow:established; content:"UG"; \
offset: 9; depth:2; msg:"PCOM/ASCII Request - Get UnitID (UG)"; \
classtype:attempted-recon; sid: 1000007; rev:1;)
alert tcp any any -> any 20256 (flow:established; content:"MI"; \
offset: 9; depth:2; msg:"PCOM/ASCII Request - Read Memory Integers (MI)"; \
classtype:attempted-recon; sid: 1000027; rev:1;)
alert tcp any any -> any 20256 (flow:established; content:"SB"; \
offset: 9; depth:2; msg:"PCOM/ASCII Request - Write Memory Bits (SB)"; \
classtype:attempted-recon; sid: 1000038; rev:1;)
```

FIGURE 13. Excerpt of PCOM rules for Snort NIDS.

In ASCII mode, the command code of a request may have 2 or 3 bytes, whereas the replies will always have only 2 bytes. In the reply, the command code appears at a different offset because of the `STX` marker. Such variations in PCOM/ASCII can be accommodated by using Snort `offset` and `depth` keywords. In both modes, the keyword `byte_test` was used to verify whether the payload is encoded in PCOM/ASCII or PCOM/Binary mode, based on the value of the third byte.

Nevertheless, in PCOM/Binary mode such hard-coded rules might only be able to detect the operation type (e.g. byte 12th at 77 for reading operands), being unable to distinguish the operand types and addresses (which are structured in data blocks with a variable size), because in this mode a single packet might request more than one operand type. This calls for a dedicated PCOM Snort preprocessor, not only to unlock more complex DPI logic but also to improve the range of available keyword options to further improve the readability of such rules.

B. OTHER APPROACHES

Besides the traditional strategies recommending the introduction of segregated network domains with disallow-by-default traffic control policies, or the use of bump-in-the-wire Virtual Private Networks to provide encryption with role-based authentication, there are other alternatives for which this research may also be relevant, namely the implementation of PCOM-aware data diodes and honeypots.

Data diodes provide a mechanism to enforce strict one-way communications between two networks or devices, being also known as unidirectional gateways, since they allow for data to be securely transferred from a restricted access domain (such as a process control network) to a less secure network tier, but not in the opposite direction [46]. Such gateways require the use of additional software components in each side of the unidirectional link in order to support the conversion of TCP/IP SCADA protocol requests into unidirectional data streams - this due to the fact these protocols (such as Modbus/TCP or DNP3) were conceived for bidirectional operation, relying on a three-way handshake and continuous acknowledgments between peers. To the best of our knowledge, there is no PCOM support for existing commercial data diode offers – however, we consider that the information provided in this paper will contribute to ease the task of creating the required protocol conversion components.

Another interesting mechanism for defensive purposes are honeypots. SCADA honeypots have existed for some time, such as conpot [47] or the one proposed by [48], but none has PCOM support. While such a honeypot would hardly constitute an attack deterrent, it could play a relevant role in helping disclose attackers at early phases, providing a means not only to detect them but also to profile their strategy, by gathering as much evidence as possible. The research hereby documented could also be used to create a PCOM device emulator that would share a significant deal of its code basis with the software components required for data diode support.

Finally, the PCOM dissection provided in this paper is also instrumental to develop SCADA-specific security detection solutions based on statistics and/or machine learning approaches, which are useful to monitor and detect process value deviations from a normal behavior.

VII. PCOM DATASETS - A MANIFOLD CONTRIBUTION

Datasets are vital to study and comprehend how protocols work, as well as for the development and validation of security tools, being used for model training and validation (for instance, in the case of Machine Learning classifiers) or for the establishment of nominal parameters for statistical analysis. As far as the authors know, when it comes to the PCOM protocol, there are no publicly available datasets.

In the particular case of the research effort hereby documented, a PCOM dataset was considered instrumental to validate the correct behavior of the previously discussed tools (e.g. assess the Snort PCOM rules), but also to support further research and development of new tools. Moreover, the authors believe there might be several undocumented and undiscovered PCOM/Binary functions used during the reprogramming step – having a dataset would be instrumental towards disclosing their existence, semantics and structure.

To fill this gap, the authors released a set of labeled individual PCOM/TCP captures in libpcap format [49], each one associated to a single PCOM operation, whether reading or programming. The description of each one is enumerated in table 2.

TABLE 2. **PCOM datasets released within this research.**

ID	Mode	Description
1	ASCII	Getting PLC Versions
2	ASCII	Getting PLC Unit ID
3	Binary	Getting PLC Name
4	ASCII	Reading Inputs
5	ASCII	Reading Outputs
6	ASCII	Writing Outputs
7	ASCII	Reading System Bits
8	ASCII	Writing System Bits
9	ASCII	Reading Memory Bits
10	ASCII	Writing Memory Bits
11	ASCII	Reading System Integers
12	ASCII	Writing System Integers
13	ASCII	Reading Memory Integers
14	ASCII	Writing Memory Integers
15	ASCII	Reading System Longs
16	ASCII	Writing System Longs
17	ASCII	Reading Memory Longs
18	ASCII	Writing Memory Longs
19	Binary	Reading Data Table Value
20	ASCII	Stop PLC command
21	ASCII	Reset PLC command
22	ASCII	Start PLC command
23	Both	PCOM Scan

VIII. CONCLUSION

The main objective of this article was to create awareness for the security issues found in PCOM, a rather unexplored SCADA protocol, while at the same time showcasing the use of a structured methodology to disclose these same issues. By striking a contrast with the conventional approach to this kind of work, often based on ad-hoc procedures mostly focused on filling a CVE report, the effort hereby documented goes full-circle by providing several public contributions that are instrumental for infrastructure operators, security pentesters, auditors and researchers alike.

Starting from a purpose-built research testbed and following the PTES methodology, the authors were able to scout for, pinpoint, identify and explore several vulnerabilities of the PCOM protocol, implementing and testing several attack use cases. Together with an effective OSINT (Open Source Intelligence) effort, which provided information about the core PCOM functions, the authors were able to pursue a security analysis procedure, in this case to showcase how valid PCOM messages can be leveraged by a malicious actor to ultimately gain full control over a controlled process. Finally, several mitigation strategies were proposed, albeit a more definitive solution would imply an entire redesign of the protocol, similarly to what is happening with other SCADA protocols.

Overall, and similarly to other SCADA protocols, it was found that PCOM lacks security features such as confidentiality or integrity and is vulnerable to several types of network attacks that might be used to disclose information about the process under control, affect the integrity of inflight data, manipulate runtime device registers or even disrupt the process. Despite the importance of such issues, it must be clearly stated that PCOM is no worse than its contemporary SCADA protocol counterparts.

However, not all is done yet. Some of the proposed solutions, such as the use of inline NIDS providing automatic reaction mechanisms, are not compatible with the fundamental SCADA ecosystem premise that considers availability a topmost priority. Support for PCOM-aware DPI capabilities on firewalls or the development of a dedicated Snort preprocessor capable of properly handle multiple requests in a single PCOM/Binary packet are examples of how the information that was gathered about PCOM may be used to improve the security of existing production environments.

Finally, several open-source contributions were released, in an intent to engage more discussion around the PCOM protocol from a security perspective, namely:

- A Wireshark PCOM dissector, which constitutes an extra step to better understand the protocol, while providing a starting point for further research.
- A set of Metasploit modules, used to implement several proof-of-concept attacks and to test our Snort rules. They can also be used for security auditing procedures in future security assessments.
- A set of Snort rules, published in order to provide more solid detection capabilities and eventually help deciding which PCOM options are to be allowed for a given scenario.
- A complete PCOM dataset, generated to support the internal protocol analysis effort and tool development processes, which was also donated to the public domain.

Nevertheless, there is a lot of space for improvement, including new detection and preventing mechanisms or exploring other undocumented functions and features.

REFERENCES

[1] ENISA. (2017). *Communication Network Dependencies for ICS/SCADA Systems*. [Online]. Available: https://www.enisa.europa.eu/publications/ics-scada-dependencies/at_download/fullReport

[2] Modbus Organization. (2006). *Modbus Application Protocol Specification V1.1b*. [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

[3] Modbus Organization. (2018). *Modbus/TCP Security Protocol Specification*. [Online]. Available: http://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf

[4] Unitronics. *Communication With the Vision PLC*. Accessed: Jan. 4, 2019. [Online]. Available: https://unitronicsplc.com/Download/SoftwareUtilities/Unitronics%20PCOM%20Protocol.pdf

[5] Nmap Project. *Sectools.Org Top Network Security Tools*. Accessed: Jan. 5, 2019. [Online]. Available: https://sectools.org/

[6] Wireshark Foundation. *WiresharkÂů Go Deep*. Accessed: Jan. 6, 2019. [Online]. Available: https://www.wireshark.org

[7] Rapid7. *Metasploit | Penetration Testing Software, Pen Testing Security*. Accessed: Jan. 7, 2019. [Online]. Available: https://www.metasploit.com/

[8] Cisco. *Snort–Network Intrusion Detection & Prevention System*. Accessed: Jan. 11, 2019. [Online]. Available: https://www.snort.org/

[9] G. Lyon. *NMAP: The Network Mapper–Free Security Scanner*. Accessed: Jan. 13, 2019. [Online]. Available: https://nmap.org/

[10] S. Nazir, S. Patel, and D. Patel, "Assessing and augmenting SCADA cyber security: A survey of techniques," *Comput. Secur.*, vol. 70, pp. 436–454, Sep. 2017.

[11] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security—A survey," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1802–1831, Dec. 2017.

[12] E. D. Knapp and J. T. Langill, *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Boston, MA, USA: Syngress, 2014.

[13] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on SCADA systems," in *Proc. IEEE Int. Conf. Internet Things, Cyber, Phys. Social Comput.*, Oct. 2011, pp. 380–388.

[14] S. D. Antón, D. Fraunholz, C. Lipps, F. Pohl, M. Zimmermann, and H. D. Schotten, "Two decades of SCADA exploitation: A brief history," in *Proc. IEEE Conf. Appl., Inf. Netw. Secur. (AINS)*, Nov. 2017, pp. 98–104.

[15] A. Cherepanov, "WIN32/INDUSTROYER, a new threat for industrial control systems," ESET, White Paper, Jun. 2017. [Online]. Available: https://www.welivesecurity.com/wp-content/uploads/2017/06/Win32_Industroyer.pdf

[16] Unitronics. *Visilogic for Vision and Samba*. [Online]. Available: https://unitronicsplc.com/software-visilogic-for-programmable-controllers

[17] R. Lion. *Crimson 3.0*. Accessed: Jan. 4, 2019. [Online]. Available: http://www.redlion.net/crimson-30

[18] Unitronics. *.Net Driver*. Accessed: Jan. 4, 2019. [Online]. Available: https://unitronicsplc.com/Download/SoftwareUtilities

[19] J. Thériault. *PCOM—A Basic PCOM Implementation in Python*. Accessed: Feb. 4, 2019. [Online]. Available: https://pypi.org/project/pcom/

[20] A. Rudakov. *Enumerates SCADA Modbus Slave IDS (SIDS) and Collects Their Device Information*. Accessed: Feb. 12, 2019. [Online]. Available: https://svn.nmap.org/nmap/scripts/modbus-discover.nse

[21] S. Hilt. *ENIP NMAP Scan*. Accessed: Feb. 15, 2019. [Online]. Available: https://svn.nmap.org/nmap/scripts/enip-info.nse

[22] T. Deneut. *S7 NMAP Scan*. Accessed: Feb. 17, 2019. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/scada/profinet_siemens.rb

[23] S. Hilt. *S7 NMAP Scan*. Accessed: Feb. 17, 2019. [Online]. Available: https://svn.nmap.org/nmap/scripts/s7-info.nse

[24] EsMnemon, A. Soullie, A. Torrents, and M. Chevalier, *Modbus Client Utility*.

[25] R. Wightman. *Schneider Modicon Remote START/STOP Command*. Accessed: Feb. 18, 2019. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/admin/scada/modicon_command.rb

[26] R. Santamarta and R. Wightman. *Allen-Bradley/Rockwell Automation EtherNet/IP CIP Commands*. Accessed: Feb. 18, 2019. [Online]. Available: https://www.rapid7.com/db/modules/auxiliary/admin/scada/multi_cip_command

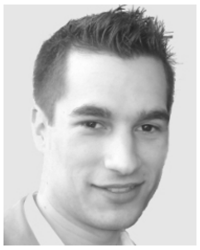
[27] J. D. Monteiro, L. Rosa, and M. B. de Freitas. *DoS Exploitation Of Allen-Bradley's Legacy Protocol (PCCC)*. Accessed: Feb. 5, 2019. [Online]. Available: https://github.com/rapid7/metasploit-framework/pull/11106

[28] R. Wightman. *Schneider Modicon Ladder Logic Upload/Download*. Accessed: Feb. 5, 2019. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/admin/scada/modicon_stux_transfer.rb

[29] S. Hilt. *Digital Bond's IDS/IPS Rules for ICS and ICS Protocols*. Accessed: Feb. 17, 2019. [Online]. Available: https://github.com/digitalbond/Quickdraw-Snort/blob/master/all-quickdraw.rules

[30] R. Jordan. *Snort 2.9.2: SCADA Preprocessors*. Accessed: Feb. 17, 2019. [Online]. Available: https://blog.snort.org/2012/01/snort-292-scada-preprocessors.html

[31] Digital Bond, Inc. and N-Dimension Solutions, Solana Networks. *A Set of ICS IDS Rules for Use With Suricata*. Accessed: Feb. 7, 2019. [Online]. Available: https://github.com/digitalbond/Quickdraw-Suricata/blob/master/enip-rules
[32] Wireshark Foundation. *Wireshark–7.4. Expert Information*. Accessed: Feb. 16, 2019. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/ChAdvExpert.html
[33] L. Rosa. *PCOMTCP: New Built-in Dissector for PCOM Protocol*. Accessed: Feb. 20, 2019. [Online]. Available: https://code.wireshark.org/review/#/c/30823/
[34] L. Rosa. *PCOMTCP: Dissection of Additional PCOM/ASCII Fields*. Accessed: Feb. 20, 2019. [Online]. Available: https://code.wireshark.org/review/#/c/31467/
[35] L. Rosa. *PCOMTCP: PCOM/Binary Command to Descriptions*. Accessed: Feb. 20, 2019. [Online]. Available: https://code.wireshark.org/review/#/c/31858/
[36] R. Bosch *et al.*, "Can specification version 2.0," Rober Bousch GmbH, Postfach, Gerlingen, Germany, Tech. Rep. 300240, 1991, p. 72.
[37] Penetration Testing Execution Standard Group. *Penetration Testing Execution Standard*. [Online]. Available: http://www.pentest-standard.org/index.php/Main_Page
[38] L. Rosa. *SCADA Scan to Collect Information From Unitronics PLCS Via PCOM Protocol*. Accessed: Feb. 20, 2019. [Online]. Available: https://github.com/nmap/nmap/pull/1445
[39] L. Rosa, T. Cruz, P. Simões, E. Monteiro, and L. Lev, "Attacking SCADA systems: A practical perspective," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Lisbon, Portugal, May 2017, pp. 741–746. Accessed: Feb. 6, 2019. [Online]. Available: http://ieeexplore.ieee.org/document/7987369/
[40] L. Rosa. *New Module Pcomclient*. Accessed: Feb. 20, 2019. [Online]. Available: https://github.com/rapid7/metasploit-framework/pull/11219/
[41] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "NIST special publication 800-82, revision 2: Guide to industrial control systems (ICS) security," in *Proc. NIST*, 2014.
[42] T. Cruz *et al.*, "A cybersecurity detection framework for supervisory control and data acquisition systems," *IEEE Trans. Ind. Informat.*, vol. 12, no. 6, pp. 2236–2246, Dec. 2016.
[43] L. Rosa. *New PCOM Module to Send Admin Commands*. Accessed: Feb. 20, 2019. [Online]. Available: https://github.com/rapid7/metasploit-framework/pull/11220/
[44] R. Langner, "To kill a centrifuge: A technical analysis of what Stuxnet's creators tried to achieve," The Langner Group, Hamburg, Germany, 2013.
[45] L. Rosa. *New Snort Rules for PCOM Protocol*. Accessed: Feb. 20, 2019. [Online]. Available: https://marc.info/?l=snort-sigs&m=154746968717558
[46] M. B. de Freitas, L. Rosa, T. Cruz, and P. Simões, "SDN-enabled virtual data diode," in *Computer Security*. Cham, Switzerland: Springer, 2018, pp. 102–118.
[47] L. Rist, J. Vestergaard, D. Haslinger, A. Pasquale, and J. Smith. (2013). *CONPOT ICS/SCADA Honeypot*. Honeynet Project.
[48] P. Simões, T. Cruz, J. Proença, and E. Monteiro, "Specialized honeypots for SCADA systems," in *Cyber Security: Analytics, Technology and Automation*. Cham, Switzerland: Springer, 2015, pp. 251–269.
[49] L. Rosa. *Pcom PCAP Captures*. Accessed: Feb. 20, 2019. [Online]. Available: https://github.com/lmrosa/pcom-misc/tree/master/pcaps

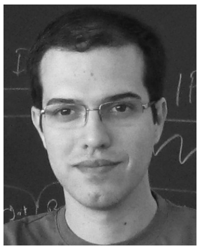


LUIS ROSA received the M.Sc. degree in informatics engineering from the Higher School of Technology and Management, Polytechnic Institute of Coimbra, Coimbra, Portugal, in 2013. He is currently pursuing the Ph.D. degree in informatics engineering with the University of Coimbra, where he is also a Junior Researcher with the Centre for Informatics and Systems and participates in several research projects in the those fields. His research interests include security, event management, and critical infrastructure protection.

MIGUEL FREITAS received the M.Sc. degree in chemical engineering from the Technical University of Lisbon, in 2016, and the M.Sc. degree in informatics engineering from the University of Coimbra, in 2018, where he is currently a Junior Researcher with the Center for Informatics and Systems. He is also a collaborator for some open-source projects. His research interests include software-defined networking, cybersecurity, and critical infrastructure protection.

SERGEY MAZO received the M.Sc. degree in electricity from Ivanovo State Power University, Russia, in 2003. He is currently an Engineer with Israel Electric Corporation. He is also a specialist in several supervisory control and data acquisition (SCADA)-related software and network communication protocols. His research interests include security, SCADA systems, critical infrastructure protection, penetration testing, and computer forensics.

EDMUNDO MONTEIRO is currently a Full Professor with the University of Coimbra, Portugal. He has more than 30 years of research experience in the field of computer communications, wireless networks, quality of service and experience, network and service management, and computer and network security. He participated in many Portuguese, European, and international research projects and initiatives. His publication list includes over 200 publications in journals, books, and international refereed conferences. He has co-authored nine international patents. He is a member of the Editorial Board of *Wireless Networks* (Springer) journal and is involved in the organization of many national and international conferences and workshops. He is also a Senior Member of the IEEE Communications Society and the ACM Special Interest Group on Communications. He is also a Portuguese Representative in IFIP TC6 (Communication Systems).



TIAGO CRUZ (SM'18) received the Ph.D. degree in informatics engineering from the University of Coimbra, Coimbra, Portugal, in 2012, where he has been an Assistant Professor with the Department of Informatics Engineering, since 2013. He is the author of more than 40 publications, including chapters in books, journal articles, and conference papers. His research interests include the areas of management systems for communications infrastructures and services, critical infrastructure security, broadband access network device and service management, the Internet of Things, software-defined networking, and network function virtualization. He is a member of the IEEE Communications Society.

PAULO SIMÕES (M'10) received the Ph.D. degree in informatics engineering from the University of Coimbra, Coimbra, Portugal, in 2002, where he is currently a Tenured Assistant Professor with the Department of Informatics Engineering. He regularly collaborates with the Instituto Pedro Nunes as a Senior Consultant, leading technology transfer projects for industry partners, such as telecommunications operators and energy utilities. He has more than 150 publications in refereed journals and conferences. His research interests include the future Internet, networks, and infrastructure management, security, critical infrastructure protection, and virtualization of networking and computing resources. He is also a member of the IEEE Communications Society.