Received: 21 December 2020 | Revised: 15 April 2021 | Accepted: 26 April 2021 | IET Cyber-Physical Systems: Theory & Applications

DOI: 10.1049/cps2.12014

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH PAPER

Man-in-the-middle attacks and defence in a power system cyber-physical testbed

Patrick Wlazlo[1] | Abhijeet Sahu[2] | Zeyu Mao[2] | Hao Huang[2] | Ana Goulart[1] | Katherine Davis[2] | Saman Zonouz[3]

[1]Electronics Systems Engineering Technology, Texas A&M University, College Station, TX, USA

[2]Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA

[3]Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ, USA

Correspondence

Ana Goulart, Electronics Systems Engineering Technology, Texas A&M University, College Station, Texas, USA.
Email: goulart@tamu.edu

Funding information

U.S. Department of Energy, Grant/Award Number: DE-OE0000895

Abstract

Man-in-The-Middle (MiTM) attacks present numerous threats to a smart grid. In a MiTM attack, an intruder embeds itself within a conversation between two devices to either eavesdrop or impersonate one of the devices, making it appear to be a normal exchange of information. Thus, the intruder can perform false data injection (FDI) and false command injection (FCI) attacks that can compromise power system operations, such as state estimation, economic dispatch, and automatic generation control (AGC). Very few researchers have focused on MiTM methods that are difficult to detect within a smart grid. To address this, we are designing and implementing multi-stage MiTM intrusions in an emulation-based cyber-physical power system testbed against a large-scale synthetic grid model to demonstrate how such attacks can cause physical contingencies such as misguided operation and false measurements. MiTM intrusions create FCI, FDI, and replay attacks in this synthetic power grid. This work enables stakeholders to defend against these stealthy attacks, and we present detection mechanisms that are developed using multiple alerts from intrusion detection systems and network monitoring tools. Our contribution will enable other smart grid security researchers and industry to develop further detection mechanisms for inconspicuous MiTM attacks.

1 | INTRODUCTION

The integration of information technology (IT) with industrial control systems (ICS) has revolutionized smart control of critical infrastructure systems such as energy, water, chemical, and transportation. Fast and accurate remote data collection and processing are helping to automate and optimize these sectors [1]. Initially, the integration of IT (cyber) and ICS (physical) systems focused on utility rather than data security. This lack of consideration for security in the design phase has permitted serious security problems such as Stuxnet malware [2], the Ukraine attacks [3, 4], and the intrusion in the European Network of Transmission System Operators in 2020 [5]. Due to the increasing awareness of such multi-stage, advanced concept-of-operations attacks, critical infrastructure stakeholders are focussing on the security of their networks, by following cyber-physical security policies that are unique to different sectors. These policies and decisions are governed by the threat models associated with the specific systems in each physical domain. In particular, the physical domain addressed in this paper is the energy sector, where we investigate man-in-the-middle (MiTM) attacks to an electrical utility's supervisory control and data acquisition system (SCADA).

A MiTM attack is one of the oldest forms of cyber intrusions, where a perpetrator positions itself in a conversation between two end points to either passively eavesdrop or actively impersonate one of the end points. MiTM attacks encompass different techniques, depending on the threat model. For example, Secure Socket Layer (SSL) hijacking is an attack where a *man-in-the-middle* intercepts a request from client to server, then continues to establish an encrypted session between itself and the server, and a regular session

between itself and the client, thus appearing to be a secure exchange between client and server.

Similar to SSL hijacking, SCADA protocols such as Distributed Network Protocol-3 (DNP3), Modbus, and IEC-61850 can also be intercepted. If this happens, there is a potential to cause severe damage to energy systems, such as these scenarios:

- False data injection (FDI) can be performed that compromises state estimations [6], which affect economic dispatch [7], generation scheduling, and load forecasting among others.
- False command injection (FCI) can be performed that can cause minor to major impacts, such as cascading failures and blackouts.
- Eavesdropping, where an intruder intercepts and reads packets, then uses this information to learn how the system operates.

Furthermore, some MiTM attacks can be stealthy enough to evade conventional intrusion detection systems (IDS). First, the added latency caused by a MiTM intercepting packets may be difficult to detect in a wide area network (WAN). ICS data packets also have different delays depending on the application [8], and polling frequencies range from milliseconds to hours. Hence, delays that would occur due to MiTM can get masked due to normal behaviour of the system. Second, most off-the-shelf security tools to detect MiTM attacks are designed for traditional Internet applications and do not support ICS protocols. There are a lot of proprietary ICS protocols, and there are vendor-specific implementations of protocols like DNP3. Moreover, a cyber-physical testbed is needed to verify the impact of MiTM attacks and ways to mitigate them.

Thus, our work addresses these gaps and challenges by focussing on MiTM attack methods that are difficult to detect within a smart grid and how to detect them. Our main contributions can be described as follows:

- We present an automated attack surface assessment solution for cyber-physical power grid infrastructures against MiTM attack vectors. It facilitates exhaustive simulation and exploration of various threats against the cyber-physical platform accurately and efficiently, and without real-world consequences and system failures. We make the proposed framework publicly available, it can be used by researchers in the field to collect comprehensive realistic datasets for purposes such as the training of machine learning models to provide security protection.
- We present a new intrusion detection solution for cyber-physical power grid platforms against the emerging MiTM attack vectors. In design of our intrusion detection framework, we leverage the test-bed we built and the automated tool we designed in our first contribution. With exhaustive attack vector simulation against the realistic cyber-physical test-bed, we evaluate the efficiency of our proposed intrusion detection solution and incrementally improve its detection accuracy.

The rest of this paper is organized into the following sections. In Section 2, we evaluate previous papers on MiTM attacks performed within a cyber-physical testbed and their limitations. Section 3 provides background on DNP3, MiTM attacks, and an overview of our testbed. In Section 4, we present different methods for MiTM attacks on DNP3, such as binary operate and analogue direct operate modifications, with detailed algorithms. We describe how an intrusion detection system can be configured to help detect two kinds of cyber attack in Section 5. In Section 6, we present four MiTM attack use cases and analyse metrics that can be used to indirectly detect a MiTM attack. Conclusions and a review of our contributions are discussed in Section 7.

2 | RELATED WORK

There are several papers that describe MiTM attacks on energy systems that have cyber-physical system (CPS) testbeds and perform experiments on industrial control protocols. Their cyber-physical testbeds differ in terms of power and network simulators or emulators, amount and type of physical devices, and which energy system they model and in how much detail (e.g. power transmission or power generation).

When comparing these papers, we observe certain limitations that are addressed by our work: (a) they target small scale systems and support limited attack scenarios, (b) they do not include details on how the MiTM attack started, and (c) they do not show how to detect stealthy MiTM attacks.

In [9], denial of service (DoS) attacks are performed in a CPS testbed which has a real-time digital simulator (RTDS) for power, network simulator-3 for communications, and devices such as phasor measurement units (PMUs) and phasor data concentrators (PDCs). They perform DoS attacks that increase delays in the communications links. The attack targets voltage stability monitoring and control in a transmission system. However, the method adopted in creating the DoS attack is not thoroughly presented in the paper.

The CPS testbed in [10] uses RTDS, Opal-RT, and a WAN emulator to demonstrate cascading failures. The failures were caused by a coordinated data integrity attack that triggered the operation of a remedial action scheme. With false measurements, the MiTM attacks manipulated the automatic generation control (AGC) algorithm to take the wrong control action. Similarly, in [11] AGC is targeted by MiTM attacks, where DNP3 packets carrying frequency and tie line flow measurements are modified using Scapy [12] tools. Although the physical scenarios in these cases are demonstrated, the precursors of the MiTM attack are not explained clearly.

A Modbus-based MiTM attack on a CPS testbed is presented in [13]. The authors use Ettercap and LibModbus libraries to poison the address resolution protocol (ARP) cache and manipulate the Modbus packets that affects the controller for a static volt-ampere reactive compensator. To simulate a communication network, the authors use the Opnet simulator that provides system-in-the-loop features to connect real devices to the simulator. Another work [14] also uses existing

libraries for performing a MiTM attack on a grid-connected photovoltaic plant using the Metasploit framework. The testbed in [14] uses Schweitzer Engineering Laboratories (SEL) PMU hardware and the IEC 60870-5-103 protocol to demonstrate the attack scenario. However, in both cases, due to the limitations of the open source Metasploit and Ettercap frameworks, only limited threat models are possible for the ICS protocols.

A multi-dimensional SCADA-specific IDS is presented in [15]. It detects MiTM attacks on IEC 61850 traffic in a real 500 kV substation. The MiTM attacks can easily be integrated in small test cases, but they have not been studied for large scale grids.

Recent work on the Idaho CPS testbed [16] presents MiTM attacks on IEC C37.118, IEC 61850, and DNP3. However, the impact of the attacks on the power system is not presented, and the strategy adopted by incorporating the MiTM attacks is not clear. Another MiTM attack using DNP3 is presented in [17], which also does not clearly illustrate the physical side threat model.

Authors in [1] present a MiTM attack on PMUs and PDCs by generating IEEE C37.118 packets using Wireshark. The use of Wireshark for creating a MiTM attack is unrealistic because it adds latency to the system that can be easily detected.

On the topic of latency and the impact of delays on the power system, the delay for ICS packets was studied in [18]. The authors found that for a normal relay's *TRIP/CLOSE* command the maximum tolerable delay was between 3 and 16 ms. The maximum delay for a human machine interface workstation to receive updates was between 16 and 100 ms. This is a considerable short time frame for a MiTM intrusion to modify packets.

Also, the authors in [19] address delays in a power generation system by varying the transmission delay of the communication links that carry the AGC data packets. When the transmission delay was increased, the overshoot amplitude and delay time of the generators both increased. This in turn caused a temporary increase in the frequency of the transmission grid. This demonstrates the importance of timely delivery of SCADA packets.

A more recent work [20], which covers generation systems in microgrids, shows that when the delay of a communication network for a photo-voltaic system increases past the maximum tolerable communication delay, more power is produced than the load demands. This again shows that the delay of generator control commands for a power generation system must be promptly received and in regular intervals to avoid over-voltage.

To complement these previous works, our contribution is to investigate MiTM threat scenarios in the context of SCADA systems and how compromised DNP3 packets can cause failures to the physical system. Using our own libraries and scripts to emulate the MiTM attacks, we implement and analyse use case scenarios in a CPS testbed that simulates a large scale synthetic electric grid based on the Texas footprint [21, 22], where the attacker's stealthiness and impact on various scenarios, in the context of 5 to 10 simulated substations, are evaluated. Such evaluations play a major role in exploring the threat space and proposing detection mechanism using network metrics such as retransmission rate and network delays. We present detection mechanisms that will enable other smart grid security researchers and industry stakeholders to detect similar MiTM attacks.

3 | BACKGROUND

This section gives a background on the implementation of multi-stage MiTM attacks for DNP3 in our emulation-based cyber-physical power system testbed. First, we present an overview of DNP3 and MiTM attack types. Our testbed, which allows us to model these MiTM threat and defence scenarios, is also explained in this section.

3.1 | DNP3

DNP3 [23] is a protocol used in SCADA systems for monitoring and controlling field devices. The protocol was released in 1993 for RS-485 serial links but has been upgraded to work with TCP/IP networks. It can have multiple network setups using a master/outstation architecture. One example is a multi-drop network, where a DNP3 master communicates with more than one DNP3 outstation. Another example is a one-on-one network where a DNP3 master communicates with only one outstation.

There are three layers in the DNP3 protocol:

1. The *data link layer* ensures the reliability of the physical link by detecting errors and duplicate frames. As shown in the example DNP3 packet in Figure 1, the DNP3 header has 10 bytes, or octets, including two synchronization octets (\x05 \x64), followed by a frame length, data link control information field, and source and destination device addresses. At the end, there is a cyclic redundancy error (CRC) code to detect any bit errors in the header.
2. The *transport layer* supports fragmentation and reassembly of large application payloads. Using one octet, it stores FIR (1 bit), FIN (1 bit) and sequence number (6 bits), where FIR and FIN determine if the fragment is the first or the last fragment. The sequence number identifies each fragment so

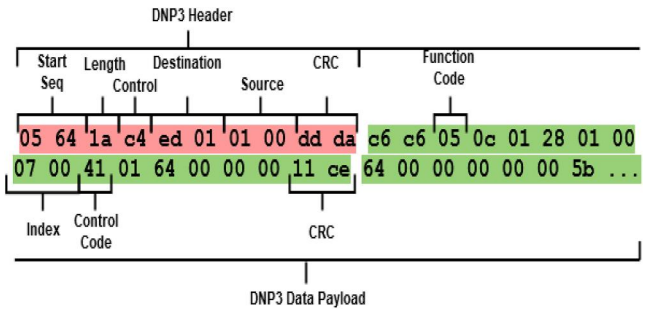

FIGURE 1 Hexadecimal representation of a DNP3 packet structure

that they can be reassembled in the correct order before they are sent to the application layer.

3. The *application layer* provides services to the DNP3 user software so that DNP3 devices can send and receive messages. First, the application layer deals with DNP3 devices, known as DNP3 points, and then groups them according to their type: binary inputs (*BI*), binary outputs (*BO*), analogue inputs (*AI*), analogue outputs (*AO*), and counter input. Each group is identified by an index. Also, the application layer organizes static data and events into classes, where Class 0 means static data and Classes 1, 2, and 3 correspond to events with different priorities. Static data means the state of a DNP3 point, whereas an event means a change in the current state. To indicate the purpose of the DNP3 message, the application layer header has a function code (FC) octet (Table 1). There are two ways to send a command from the master to the outstation: the SELECT OPERATE where the master sends a select packet to a device in the outstation (FC:03), followed by the operation it should perform (FC:04); or the DIRECT OPERATE (FC:05), where one packet contains the device address with the operation it should perform.

Similar to other Internet protocols, the DNP3 packet contains a header and a payload. The DNP3 payload has multiple *data chunks*, consisting of 16-octet data blocks followed by a two-octet CRC to ensure each data block's integrity. Inside the payload, a function code is used to identify the operation the outstation should perform, as in Table 1. The index will tell the outstation which device within the outstation the master is requesting the operation to be performed on, or retrieve data from. As shown in the sample packet in Figure 1, we can see the hexadecimal representation of a binary DIRECT OPERATE DNP3 packet (FC:05). This packet is indexed to close breaker seven in a substation of the Texas synthetic grid model. This is indicated by the index 0700, along with the 41 control code to close the breaker.

TABLE 1 Function codes for DNP3 packets [24]

Function code (Hex)	Operation
0 × 00	Confirm
0 × 01	Read
0 × 02	Wire
0 × 03	Select
0 × 04	Operate
0 × 05	Direct operate with acknowledge
0 × 06	Direct operate without acknowledge
0 × 07	Freeze with acknowledge
0 × 08	Immediate freeze – No acknowledge
0 × 09	Freeze and clear with acknowledge
0 × 10	Freeze and clear – No acknowledge
0 × 13	Cold restart
0 × 14	Enable spontaneous messages
0 × 15	Disable spontaneous messages
0 × 16	Assign classes
0 × 17	Delay measurement
0 × 81	Solicited response
0 × 82	Unsolicited response

As for confidentiality, DNP3 is a clear text, unencrypted, protocol with no inherent security mechanism [25]. For this reason, the DNP3 protocol is susceptible to MiTM attacks, where an outsider can eavesdrop the communication between two nodes and modify the content of the packets. There have been numerous studies that try to incorporate encryption onto the DNP3 protocol using Transport Layer Security encryption. However, this has not been widely adopted since maintaining time-sensitive public-key certificate server available for the DNP3 server and client requires costly upgrades to existing field equipment.

3.2 | ARP cache poisoning

The first step of the MiTM attack is for the adversary to impersonate a network device. This is done using ARP spoofing, or ARP cache poisoning, where the adversary sends an unsolicited ARP message to the targeted node. These messages are used to link the adversary's hardware address, or media access control (MAC) address, with the Internet protocol (IP) address of the targeted device.

As illustrated in Figure 2, the adversary sends an unsolicited ARP frame to the substation router telling the router to correlate the outstation's IP address with the adversary's MAC address. Thus, when the substation router needs to deliver a packet to the outstation, the router will instead forward it to the adversary. Similarly, the adversary sends an unsolicited ARP packet to the outstation node so that it maps the router's IP address to the adversary's MAC address. After these unsolicited ARP packets, the adversary receives all packets sent between outstation and router. The adversary can now read or modify the packets' contents, before it forwards the packets to the correct device. The adversary in the example in Figure 2 changes a DIRECT OPERATE command from *CLOSE* to *TRIP*, as well as a the response from *TRIP* to *CLOSE*. As a result, the DNP3 communication channel remains open, and neither router nor outstation suspects that their packets are being intercepted.

The MiTM time diagram in Figure 2 shows that the majority of the delay is at the adversary node, where the DIRECT OPERATE command is modified. This processing delay, or data injection delay, is the amount of time the adversary needs to filter the DNP3 packets, modify the packets' contents, and recalculate the DNP3 layers' CRC and TCP header checksum. A small portion of the delay time is due to the longer route the packet must travel after the ARP cache poisoning, since the packet is going through an extra node, the adversary.

3.3 | Integration of MiTM attacks in RESLab testbed

The MiTM attacks in our work are programmed to perform a staged intrusion, by trespassing into the broadcast domain of one substation's local area network (LAN). The trespassing could be a result of the adversary getting physical access to the substation site or by getting the credentials and remote access of one of the local devices.

To simulate a substation LAN and control centre, we use the RESLab testbed [26], shown in Figure 3, which is a CPS testbed comprised of the following components:

- *Network Emulator* – the Common Open Research Emulator (CORE) [27] emulates the communication network. CORE is a Linux-based application maintained by the U.S. Naval Research Laboratory that uses FreeBSD jails configured as routers, switches, servers, and personal computers to create various emulated network nodes.
- *DNP3 Master* – there are two different DNP3 masters in the RESLab testbed: the OpenDNP3 and the SEL-3530 real-time automation controller (RTAC). The OpenDNP3 is a DNP3 master application with a command line interface that is used to remotely operate outstation devices. The SEL-3530 RTAC is a cyber-physical component connected

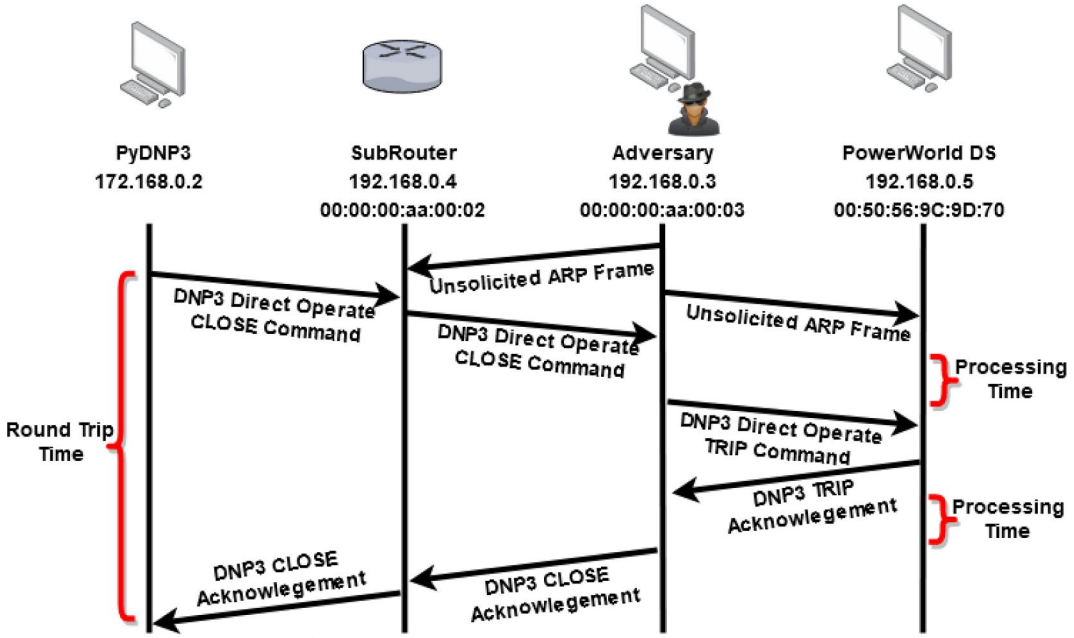

FIGURE 2 Timing diagram for address resolution protocol cache poisoning of the substation router (*SubRouter*) and *DNP3 Outstation* prior to the MiTM attack to modify the *DIRECT OPERATE CLOSE* command

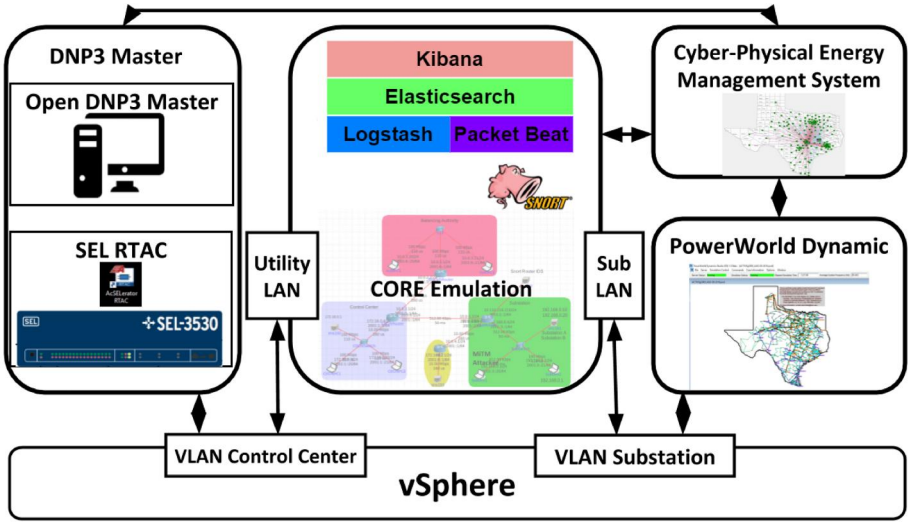

FIGURE 3 The RESLab emulation-based testbed architecture

to the testbed that in this instance is being used as a DNP3 master.

- *DNP3 Outstation* – PowerWorld Dynamic Studio (PWDS) is a real-time simulation engine for high voltage power systems [28]. In this paper, we use PWDS to simulate the synthetic Texas 2000-bus model [22] as our exemplar power system.
- *Intrusion Detection System* – Snort [29] is being used in RESLab as the rule-based, open-source IDS. It is configured to generate alerts for ARP cache poisoning and FDI attacks.
- *Storage and Visualization* – The Elasticsearch, Logstash, and Kibana (ELK) [30] stack probes and stores all virtual and physical network interfaces' traffic, in addition to storing all Snort alerts generated during each use case. This data can be queried using Lucene queries to perform in depth visualization and cyber data correlation.
- *Cyber-Physical Resilient Energy Systems (CYPRES) Application* – CYPRES aggregates information, that is, from the cyber side CORE emulation environment, from the power side PWDS, as well as from the DNP3 masters regarding the communication status of DNP3 packets. All these datasets are then analysed.

The RESLab testbed components are hosted in different virtual machines with the vSphere virtualization platform, as illustrated in Figure 3. In the middle, we have CORE which emulates a communication network that allows the DNP3 masters and outstation to interact. On the left side, CORE connects the utility control centre, where there is a DNP3 master modelled using OpenDNP3 libraries and SEL-RTAC. On the right side, CORE connects the DNP3 outstations running in PWDS.

The network topology of CORE is shown in detail in Figure 4, where the OpenDNP3 master and SEL-RTAC are connected to the CORE's network through virtual port *ens192*. To emulate the outstations, the synthetic power system in PWDS is connected through virtual port *ens224*. When the DNP3 communication link is not ARP cache poisoned, the traffic flows directly from *ens192* to *ens224*. However, when the adversary cache poisons the DNP3 communication link, all traffic between the control centre and outstation passes through the adversary node, as shown by the dotted arrows.

4 | MiTM ATTACKS ON DNP3

In this section four different MiTM attack algorithms are described. Each algorithm is used in a different sequence to generate the FDI and FCI use cases.

Table 2 shows the order in which each algorithm is used for the four use cases.

For example, in *Use Case* 3, the adversary script calls Algorithms 3, 4, and 2. The function of each algorithm is described in the proceeding paragraphs.

4.1 | Scapy MiTM script

The MiTM attack scripts are programmed using the Scapy Project Python wrapper [12]. Scapy is a powerful library that

TABLE 2 The sequence of function calls to the MiTM algorithms for each use case

Use case	Sequence of function calls
Use Case 1	Algorithm 1
Use Case 2	Algorithm 2
Use Case 3	Algorithm 3 → Algorithm 4 → Algorithm 2
Use Case 4	Algorithm 3 → Algorithm 4 → Algorithm 2 → Alg. 4

Abbreviation: MiTM, Man-in-The-Middle.

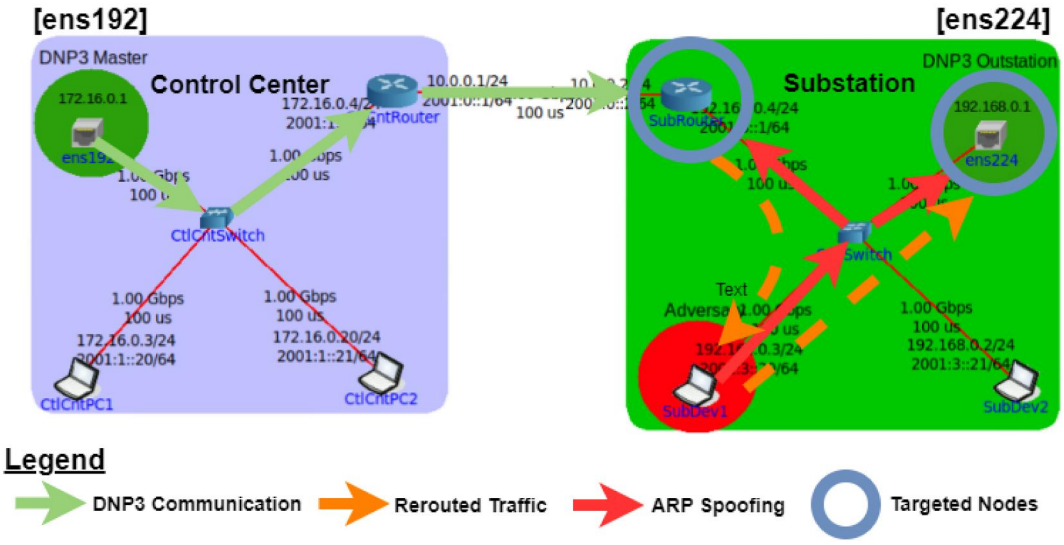

FIGURE 4 The RESLab testbed network topology emulated in CORE. The arrows show the data flows and virtual machine interconnections for our use cases (Section 4)

can modify frames and/or packets in real-time. The library has many built-in packet dissectors for applications using either user datagram protocol or transmission control protocol (TCP). However, DNP3 is not one of the supported TCP applications. Fortunately, [31] introduces a Scapy extension for DNP3, which we use to dissect and filter DNP3 packets.

Here is how it works. Scapy reads all traffic sent to the adversary once the route is ARP cache-poisoned. Then, the DNP3 extension along with Scapy's native libraries filters traffic based on IP, TCP, and DNP3 header information. For instance, if the function code of a captured DNP3 packet is (FC:05), it indicates an analogue or a binary DIRECT OPERATE command. Thus, the adversary needs to have its value modified before it forwards the packet to the original destination. If the TCP packet does not have a function code, it is not a DNP3 packet and the adversary only forwards it to the destination. This process is not as straightforward as it may seem. First, the total process must be optimized to take the least amount of processing time as possible. Second, the DNP3 traffic is filtered by function code and then by control code to determine if the payload is an analogue or binary command. Third, in order to keep the operator unaware that the wrong command has been sent to the outstation, the acknowledgement number of each DNP3 command packet is used to filter the appropriate response packet that is changed to match the original command sent by the control centre.

4.2 | Binary operate

Substation breakers are represented by binary points (*BI* or *BO*) that can have their states updated remotely by DNP3 binary DIRECT OPERATE packets. Each point can either be opened (tripped) or closed. The *TRIP* action will disconnect or de-energize a line, a *CLOSE* will energize an open a line. The MiTM script inverts the binary command. In other words, a *TRIP* command is forced to a *CLOSE* command, or vice-versa.

Algorithm 1 describes the process to invert a binary DIRECT OPERATE packet. After a received packet (*recv_pkt*) is identified as a DNP3 binary DIRECT OPERATE packet, its TCP header checksum is removed, because Scapy automatically recalculates the TCP header checksum if there is not one detected when forwarding the frame. The *recv_pkt*'s acknowledgement number is stored as *binary_operate_ack*, so the response packet can be changed to the original binary value. Then, the DNP3 header is stored as *dnp_header*. The same is done for the packet's payload or *dnp_pl*. Next, the payload is bisected around the *control code*, as in Figure 1. The first half of the payload is stored under the *dnp_front*. The second half is stored under the *dnp_end*. If the packet's control code is a *CLOSE* command ('41'), it is modified to be a *TRIP* command ('81'), or vice-versa.

Algorithm 1 MiTM attack on binary control commands

```
function modify_binary_direct_operate(recv_pkt)
  binary_operate_ack = recv_pkt[TCP].ack
  mod_pkt = recv_pkt[TCP]
  Delete mod_pkt.checksum
  dnp_header = mod_pkt.pl[: dnp_hdr_size]
  dnp_pl = mod_pkt.pl[dnp_hdr_size:]
  dnp_front = mod_pkt.pl[dnp_hdr_size: bin_loc]
  dnp_end = mod_pkt.pl[bin_loc + 1:]
  dnp_mid = mod_pkt.pl[bin_loc]
  if dnp_mid == b'\x41' then dnp_mid = b'\x81'
  else dnp_mid = b'\x41'
  end if
  merged_pl = Join[dnp_front, dnp_mid, dnp_end]
  pl_with_crc = update_crc_payload(merged_pl)
  mod_pkt.pl = Join[dnp_header, pl_with_crc]
  mod_pkt = send_to_outstation(mod_pkt)
return mod_pkt, binary_operate_ack
end function
```

Then, the DNP3 payload is reassembled by joining the *dnp_front*, *dnp_mid*, and *dnp_end* together as *merged_pl*. The reassembled payload is passed to the *update_crc_payload* function, which comes from the DNP3 Scapy extension [31]. Finally, the MAC address in the frame's header is updated to the MAC address of the outstation, and the adversary forwards the frame to the outstation.

4.3 | Analogue direct operate

The setpoints of generators and other controls are represented by analogue points (*AI* or *AO*) in the DNP3 protocol. Each setpoint can be varied by the DNP3 master. In the MiTM script, any analogue DIRECT OPERATE setpoint is forced to a lower value. The lower value ramps down the generator without tripping it. Algorithm 1 inverts the control code for a binary DIRECT OPERATE command. Algorithm 2 changes analogue values instead of binary values. The main difference between the two algorithms is *dnp_mid* which is an analogue value that is modified in Algorithm 2. The analogue value is a four-octet float value that is encoded with a one-octet control status. In the *update_new_val*() function, the analogue value in the original *recv_pkt* is changed to a forged value. Once the forged value is placed in the correct position, the DNP3 payload CRC, TCP header checksum, and the MAC address are updated before the packet is forwarded to the outstation.

Algorithm 2 MiTM attack on analogue control commands

```
1: function modify_analog_direct_operate(recv_
   pkt)
2:   analog_operate_ack = recv_pkt[TCP].ack
3:   mod_pkt = recv_pkt[TCP]
```

```
 4:   Delete mod_pkt.checksum
 5:   dnp_header = mod_pkt.pl[: dnp_hdr_size]
 6:   dnp_pl = mod_pkt.pl[dnp_hdr_size:]
 7:   dnp_front = mod_pkt.pl[dnp_hdr_size:
      alg_loc]
 8:   dnp_end = mod_pkt.pl[anlg_loc + 5:]
 9:   dnp_mid = mod_pkt.pl[anlg_loc + 1:
      anlg_loc + 5]
10:   dnp_mid = update_new_val(dnp_mid)
11:   merged_pl = Join[dnp_front, dnp_mid, dnp_end]
12:   pl_with_crc = update_crc_payload(merged_pl)
13:   mod_pkt.pl = Join[dnp_header, pl_with_crc]
14:   mod_pkt = send_to_outstation(mod_pkt)
15: return mod_pkt, analog_operate_ack
16: end function
```

4.4 | Polled measurement sniff and store

To target the intended DNP3 packet, the MiTM script must first sniff through all the network traffic between the substation gateway and PWDS. Then, each packet the MiTM script receives is filtered by its function code. For every fifth packet with a DNP3 function code of '81', the analogue and binary data value is stored in the adversary's machine. Not every read response packet is stored since the processing time for these packets is relatively high and would lead to more retransmitted packets.

Algorithm 3 describes how the read response packets are stored in a *dnpDatabase*. A DNP3 response payload for poll request consists of collection of the DNP3 points stored in the *data chunks* with the *chunk_size* of 18 bytes (16 bytes of payload and two bytes of CRC). First, the packet data *dnp_pl* is checked to see if it contains one or more *datachunks*. For the payload with at least one *datachunk*, each *dnp_chunk*'s CRC is removed and its contents are concatenated into contiguous bytes of *BI*, *AI*, *BO*, and *AO* points. Then, based on the header information *bi_hdr*, *ai_hdr*, *bo_hdr*, *ao_hdr*, the number of DNP3 points under each category *bi_count*, *ai_count*, *bo_count*, *ao_count* is extracted. The information such as the *pointIndex*, *value*, *chunkIndex*, *pointType* of each DNP3 point is stored in a *dnpDatabase* classified by the source address of the packet, which is unique to each outstation number. The *pointIndex* stores the actual DNP3 index. The *value* stores the value associated with the DNP3 point. The *pointType* indicates the type of the DNP3 point. The purpose of storing *chunkIndex* is to identify the location of the DNP3 point in the *datachunks*. These attributes are further used by the intruder in Algorithm 4 to modify the measurement in a specific location, which results in a faster modification of the DNP3 payload.

Algorithm 3 MiTM attack on sniffing measurements

```
 1:  function sniff_read_response(recv_pkt,
     outstation)
 2:   mod_pkt = recv_pkt[TCP]
 3:   Delete mod_pkt.checksum
 4:   dnp_header = mod_pkt.pl[: dnp_hdr_size]
 5:   dnp_pl = mod_pkt.pl[dnp_hdr_size:]
 6:   chunk_size = 18
 7:   dnp_chunks = len(dnp_pl)/chunk_size
 8:   if dnp_chunks == 0 then
 9:     send_to_master(mod_pkt) return mod_pkt
10:   end if
11:   Store each data chunks in dnp_chunks_pl
12:   dnp_pl_without_crc = Remove 2 bytes of CRC from
      each dnp_chunks_pl
13:   dnp_reassembled = Join(dnp_pl_without_crc)
14:   Obtain bi_hdr, bi_pl, ai_hdr, ai_pl, bo_hdr,
      bo_pl, ao_hdr, ao_pl using bi_count,
      ai_count, bo_count, ao_count
15:   Obtain start and end index for BI, AI, BO, AO
      group types
16:   Store DNP3Points for each point types
17:   Create dnpDatabase for outstation using each
      point in DNP3Points for BI, AI, BO, AO types
        return dnpDatabase
18: end function
```

4.5 | Polled measurements modification

Periodically, the master polls each outstation that is connected to it for updates on the binary and analogue points. In the RESLab testbed, the polling interval varies from 30 to 60 s. When polled, an outstation responds with a list of all the binary and analogue points housed within that outstation. There are multiple ways this data can be manipulated. For instance, the poll measurements can be spoofed to a wrong value, causing the operator to send the incorrect command to the outstation. Or these updates can be forged so the operator will choose not to send a command to an outstation, when he/she should.

Algorithm 4 MiTM attack on modifying measurements

```
 1:  function modify_read_response(dnpDatabase,
     ModPoints)
 2:   mod_pkt = recv_pkt[TCP]
 3:   Delete mod_pkt.checksum
 4:   for dnp3Point in ModPoints do
 5:     dnp_header = mod_pkt.pl[: dnp_hdr_size]
 6:     dnp_pl = mod_pkt.pl[dnp_hdr_size:]
 7:     if dnp3Point.pointType is BI ∨ BO then
 8:       bin_loc = using dnp3Point.chunkIndex and
      dnp3Point.pointIndex
 9:       Follow steps 7 to 15 in Algorithm 1
10:     end if
11:     if dnp3Point.pointType is AI ∨ AO then
12:       anlg_loc =  using dnp3Point.chunkIndex
      and dnp3Point.pointIndex
13:       if len(dnp3Point.chunkIndex) > 1 then
14:         Process the dnp3Point.chunkIndex[0]and
      dnp3Point.chunkIndex[1]separately
15:       end if
16:       if len(dnp3Point.chunkIndex) == 1
     then
17:         Follow steps 7 to 14 in Algorithm 2
18:       end if
19:     end if
```

```
20:   end for
21:   mod_pkt = send_to_master(mod_pkt)
22: return mod_pkt
23: end function
```

The *dnpDatabase* containing all the datapoints captured in Algorithm 3 is passed to the *modify_read_response*() function, shown in Algorithm 4. A list of *BI*, *BO*, *AI*, and *AO* points that the MiTM attack intends to use are contained in the variable *ModPoints* that is passed to the function. Each point listed in *ModPoints* is modified in the *dnpDatabase*, and the CRC for each *datachunk* is updated. *BI* and *BO* points have only one-octet and therefore can be contained in one *datachunk* (lines 7–10 in Algorithm 4). However, *AI* and *AO* are five octets long and can be split between two datachunks (lines 11–19 in Algorithm 4). Each *datachunk* that is modified must have its CRC recalculated for the master to accept the packet.

4.6 | Acknowledgements modification

After a binary or analogue DIRECT OPERATE packet is received at an outstation, a DNP3 acknowledgement packet is returned stating the action was performed. When the MiTM script changes the binary or analogue command or value to perform an incorrect action, the outstation's response is a telltale sign that the DNP3 communication channel is compromised. For the MiTM to remain unnoticed by the control centre's operator, the DNP3 acknowledgement from the outstation must be modified.

For example, when the operator sends a binary *CLOSE* command, the MiTM scrip changes the command to a binary *TRIP* and then forwards it to the outstation. Correspondingly, in the acknowledgement, the outstation sends a binary response stating that the breaker is opening. The intruder then modifies the true response into a binary *CLOSE* response and forwards it to the operator. This leaves the DNP3 master unaware that the wrong action has been sent to the outstation. In this way, Algorithm 1 modifies the binary operate response packet. Similarly, Algorithm 2 modifies the analogue operate response packet.

4.7 | Graphical user interface for MiTM attacks

To simplify the execution of each use case of Table 2, we designed a graphical user interface (GUI) using Python's Tkinter libraries [32]. The GUI allows a user to specify the gateway, the IP addresses of the outstation and DNP3 master, and which use case to run. Once the user makes the selections and presses the *Enter* button, as shown in Figure 5, the route is ARP cache-poisoned and the MiTM program starts sniffing traffic. It modifies the DNP3 traffic according to the chosen use case. While running, the *textbox* at the bottom displays where the packets are coming from and which values have been changed. To help debug the application or the network

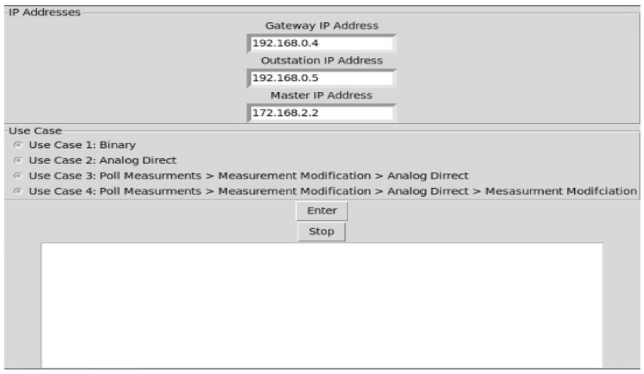

FIGURE 5 Graphical User Interface used in the REStab testbed to launch each Use Case

setup, the *textbox* also displays any errors that may occur. The MiTM program runs until the user clicks the *Stop* button. When it is pressed, both the gateway and outstation return to their previous configuration and the traffic flows between the gateway and DNP3 master as before the attack.

5 | SNORT CONFIGURATION FOR MiTM ATTACK

It is essential for utility companies to monitor and secure their networks from various forms of cyber threats. Generally, this comes in the form of an IDS that can detect vulnerabilities in a network and generate alarms. Snort is a open-source IDS [29] that can be configured to dissect Ethernet packets to monitor for a variety of attacks. Each type of attack has a pre-processor which can be enabled in the Snort's configuration file. Then, rules based on the data the pre-processors collects are created to generate alerts. The alerts can be displayed in real-time or saved to a file.

During each trial, Snort is running in the substation gateway or *SubRouter* (IP: 192.168.0.4), shown in Figure 4. The Snort ARP and DNP3 pre-processors are used. In the Snort configuration file, the MAC addresses of the *SubRouter* and *DNP3 Outstation* (IP: 192.168.0.5) are white-listed (Listing 1), where a list of known IP addresses and their MAC addresses within a LAN is maintained by Snort [33]. This allows it to detect if the MAC address has changed from the listed IP address, which indicates an attempt to poison the ARP table of the router. The DNP3 pre-processor (Listing 2) detects a DNP3 packet and checks if the CRC is correct; if not, an alert is generated. The pre-processor is configured to detect when a DNP3 DIRECT OPERATE packet is sent.

Listing 1: ARP pre-processor enabled in Snort configuration file

```
preprocessor arpspoof_detect_host:
  ↪ 192.168.0.4 00:00:00:aa:00:02
preprocessor arpspoof_detect_host:
  ↪ 192.168.0.5 00:50:56:9c:9d:70
```

Listing 2: DNP3 pre-processor enabled in Snort configuration file

```
preprocessor dnp3: ports {20000}
    memcap 262144 \
    check_crc
```

With the pre-processors enabled, custom rules are created. Rules are added to generate logs that can be used to alert the operator. The first three alerts, $R1$, $R2$, and $R3$, indicate an ARP cache poisoning. The next two alerts, $R4$ and $R5$, notify when a DNP3 DIRECT OPERATE packet is sent to the outstation.

Listing 3: ARP and DNP3 specific alerts configured in Snort

```
R1 alert (msg: "
  ↪ ARPSPOOF_ETHERFRAME_ARP_MISMATCH_SRC"
  ↪ ; sid: 2; gid: 112; rev: 1; metadata:
  ↪ rule-type preproc; classtype: bad-
  ↪ unknown;)
R2 alert (msg: "
  ↪ ARPSPOOF_ETHERFRAME_ARP_MISMATCH_DST";
  ↪ sid: 3; gid: 112; rev: 1; metadata:
  ↪ rule-type preproc; classtype: bad-
  ↪ unknown;)
R3 alert (msg: "
  ↪ ARPSPOOF_ARP_CACHE_OVERWRITE_ATTACK";
  ↪ sid: 4; gid: 112; rev: 1; metadata:
  ↪ rule-type preproc; classtype:bad-
  ↪ unknown;)
R4 alert tcp $EXTERNAL_NET any ->
  ↪ 192.168.0.5 20000 (msg: "DNP3_Snort_
  ↪DIRECT_OPERATE"; flow:established,
  ↪ to_server; dnp3_func: direct_operate;
  ↪ sid:123000;)
R5 alert tcp $EXTERNAL_NET any ->
  ↪ 192.168.0.5 20000 (msg:"DNP3_Snort_
  ↪ OPERATE"; flow: established, to_server;
  ↪dnp3_func: operate; sid:123002;)
```

6 | RESULTS AND ANALYSIS

This section briefly describes the experiment setup in the RESLab testbed, then presents the results of four MiTM use cases. In addition, we show how Snort alerts can be used to detect the MiTM attack. Snort is operating in a Network Intrusion Detection System mode at the substation router, protecting the substation's LAN.

6.1 | Experimental setup

As shown in RESLab testbed in Figure 3, the DNP3 master and the SEL RTAC are connected through vSphere's control centre virtual local area network (VLAN) to the CORE emulator. The DNP3 outstations, modelled in PWDS, are connected through vSphere's substation VLAN to the CORE network. In most known cyber attacks on an ICS network, the intruder had to perform multi-stage intrusions to reach the targeted grid components. Since this work focuses on the dynamics of MiTM attacks, the prior stages do not play a major role. Hence, we assume the intruder, after a reconnaissance stage, has remote access to one of the computer nodes in the substation LAN, which in this instance is the adversary node.

6.2 | Evaluation metrics

The strength of the MiTM attack is determined by analysing the average round trip time (RTT), retransmission rate, and average processing time of DNP3 packets, as described below.

6.2.1 | Retransmission rate

When a TCP packet is sent, the sender starts a variable-length retransmission timer, and waits for the acknowledgement. If it does not receive an acknowledgement before the timer expires, the sender assumes the packet is lost and retransmits it. During the MiTM attack, the number of retransmissions increases, because packets may not be successfully forwarded to the outstation and the DNP3 master may not receive the acknowledgement. This may also happen if the adversary cannot forward the acknowledgement it receives from the outstation. Since the duration of each use case varies, retransmission rate is used as a metric instead of the number of retransmissions. The retransmission rate R_R is computed using Equation (1),

$$R_R = N_R / T_R \tag{1}$$

where N_R is the number of retransmitted packets during the MiTM, and T_R is the time interval between the first and the last retransmitted packet in seconds.

6.2.2 | Average round trip time

The round trip time (RTT) can be seen in the time diagram in Figure 2. It includes the network's propagation delay due to the distance between nodes, the added transmission delays as the packet travels through the adversary node, and the processing time the adversary takes to modify the commands and response. Hence, we evaluate the impact of a MiTM attack on the RTT.

6.2.3 | Processing time

The processing time depends on the type of DNP3 traffic the intruder modifies. The processing time for modifying

outstation polled responses can vary based on the outstation data that is polled. The outstation's read response depends on the number of DNP3 points housed at a particular outstation.

The retransmission rate and average RTT are extrapolated by analysing Wireshark packet captures (PCAP) data from the *SubRouter*'s network interface. The processing delay is automatically calculated by the MiTM attack script.

6.3 | Modifying measurements and commands

The objective of the intruder is to disrupt grid operations. Details on the sequence of actions that create the FCI and FDI attacks and how they impact the physical components of the power system are presented in detail in [26]. Here in this paper we focus on the impact of the attacks on the communications network, or cyber telemetry. These are our four use cases:

6.3.1 | Use Case 1: Branch Control modifications

Each binary DIRECT OPERATE command is changed from a *CLOSE* to a *TRIP* command, with any other traffic simply forwarded. The change in the binary operate command introduces some processing delay, which may cause the packet to be retransmitted.

6.3.2 | Use Case 2: Generator set-point modification

When the MiTM script is running, the analogue point for the generator is set to a lower value, in some cases 20 MW, which will decrease the generator setpoint from its current value down to 20 MW.

6.3.3 | Use Case 3: Measurement and status modification

Use Case 3 is a combination of FCI and FDI attacks. After each polling interval, the DNP3 master will send a read request packet to each outstation, which then sends a read response packet back to the master. This read response is filled with the all the binary input, analogue input, binary output, and analogue output DNP3 points. Next, analogue input points in the read response packet are changed to a lower value lower of 20 or 0 MW. The operator controlling the DNP3 master is then forced to send an analogue DIRECT OPERATE command to bring the generators back to their original loaded set points. However, when the operator sends this original set point value to the generator, the MiTM script is programmed to change the setpoint to 20 or 0 MW.

6.3.4 | Use Case 4: measurement and status modification

The adversary first follows the steps of Use Case 3, then modifies the read response packet of the preceding packets, based on the actual set point given by the master. Thus, the master is unaware of the contingency created.

6.4 | Use cases implementation

For each use case, we alter the polling intervals and the number of polled DNP3 outstations. The polling intervals tested were 30 and 60 s, while the number of polled DNP3 outstations were 5 and 10. For instance, the scenario *UC*1 10 *OS* 30 means that we implemented *Use Case* 1 with 10 outstations and a polling interval of 30 s. In each scenario, the normal operation is conducted first without the MiTM attack. Then, the operation is conducted again with the attack to analyse its impact. Finally, the attack is stopped and the network restored.

The main reason for choosing polling intervals of 30 and 60 s is that most DNP3 masters have polling rates of 30 s, 1 min, or 5 min, with a maximum of 15 min. A polling interval of more than two minutes has little impact on attack strength because the adversary processing time is less than 60 to 70 ms (see Section 6.7).

Similarly, we choose outstation numbers of 5 and 10 since our objective is to study the communication dynamics of an impacted outstation, and how the number of outstations becomes a limitation on the attack success probability. The numbers of 5 and 10 coincide with our use cases in the Texas 2000-bus model where each utility control centre communicates with at least two and at most 25 substations. Because the RESLab testbed uses CORE, which is an emulator and not a simulator, there are practical limitations to the number of substations that can be modified by the MiTM script. This number in our testbed is about 50 substations; however, this depends on the amount of memory and capacity of the network interface card's buffer that is allocated to CORE. Since CORE is an emulator, it demonstrates more realistically the physical limitations an actual adversary would have to experience in order to create a successful MiTM attack.

6.5 | Impact of polling rates and number of outstations on retransmission

High polling rates, or low polling intervals, result in packet losses during an attack due to the limitations of an adversary's resources to process all the command and response DNP3 traffic. Hence, we study the impact of polling rates on the number of retransmissions. Figure 6 shows the impact of different scenarios on the retransmission rate. Scenarios with 60 s polling intervals result in less retransmissions in comparison to 30 s scenarios. For example, in *UC*1 10 *OS* 30 the retransmission rate is almost four times that of *UC*1 10 *OS* 60.

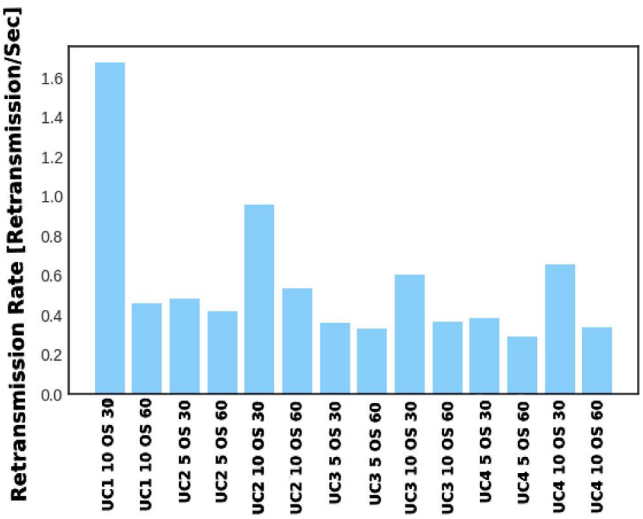

FIGURE 6 Retransmission rate of DNP3 traffic for each scenario

Note that the *UC*1 5 *OS* 30 and *UC*1 5 *OS* 60 are not included in Figure 6, because Use Case 1 requires eight or more outstations to have their binary operate packets inverted in order to generate a cascading failure in the Texas 2000-bus topology.

The number of DNP3 outstations polled also affects the amount of traffic the intruder is required to process. A larger number of outstations causes more traffic. The network buffer temporarily stores incoming packets before they are processed. Due to the limitation on the buffer size, the intruder may not be able to process all the traffic that traverses through it, which results in some attacks failing. Hence, the number of retransmissions increases. For example, from Figure 6 we can observe the retransmission rate for *UC*2 10 *OS* 30 with 10 polled outstations is almost 2.5 times that of the *UC*2 5 *OS* 30 case with five polled outstations.

6.6 | Impact on average RTT

The RTT can be an indicator that an intruder is intercepting network traffic. RTT is also affected by the number of outstations polled and how often each is being polled. Figure 7 shows the DNP3 packets that resulted in a high RTT and had to be retransmitted for different use cases. The majority of the DNP3 traffic is received before the retransmission timer expires, shown by the dashed-line. The dashed-line shows the cut-off time for the DNP3 retransmission timer, which was set to 7.0 s. We have observed that there is a longer delay when 10 outstations are polled, compared with the number of retransmissions for the scenarios with five outstations.

6.7 | Processing time by packet type

The processing time at the adversary node is different for each packet type, as shown in Figure 8, where we measure the

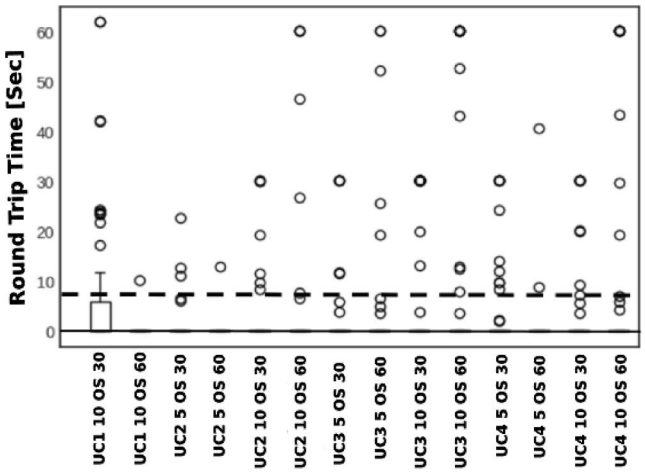

FIGURE 7 Round trip time for DNP3 traffic for each scenario

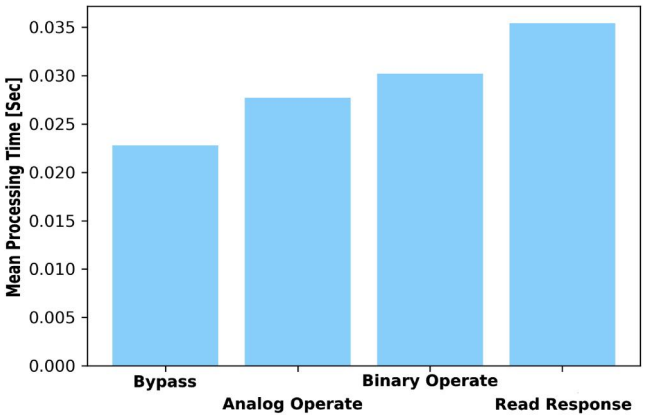

FIGURE 8 Average processing time by packet type at the adversary node

average time to forward packets between the substation gateway and outstation by packet type. The lowest forwarding time and therefore most difficult packet type to detect the MiTM attack were the bypass packets, which took 22.775 ms to process. In the bypass packets, the adversary only modifies the source and destination MAC addresses of the frame. Next, the average processing time for an analogue DIRECT OPERATE packet is higher at 27.693 ms, followed by binary DIRECT OPERATE taking an average of 30.217 ms. The highest processing time is for the read response packets, which take an average of 35.415 ms.

The processing time is directly correlated to the amount of traffic and the number of operations the MiTM script has to perform on each packet. For this reason, the bypass traffic took the least amount of time, as the adversary only updates the MAC address and forwards it to the original destination. Next, the operations to the binary and analogue DIRECT OPERATE packets include modifying the value sent by the operator, recalculating the CRC, and forwarding the forged packet to the outstation. The read response packets take the most number of operations: two to three analogue/binary

values are modified, then the CRC for multiple data blocks are calculated and updated, and the packet is forwarded to the master.

6.8 | Snort detection

In the RESLab testbed, the Snort logs and alerts are collected by Logstash, an open source tool that is used with Kibana to create dashboards for data analysis and visualization. After the Snort log data is processed by Kibana, the frequency of the ARP and DNP3 alerts show which DNP3 packets are being compromised at what time.

During each use case, the PCAP data containing unsolicited ARP and DNP3 traffic, and the Snort's alert log files are collected by Logstash, which filters and formats the data. Then, as shown in Figure 9, the data is stored in the Elasticsearch database. Kibana acts as the front-end for the Elasticsearch database, by allowing us to create graphs that show the correlation between Snort alerts, ARP frames, and DNP3 packets.

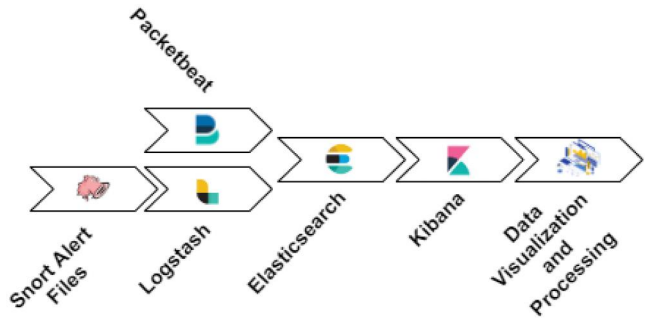

FIGURE 9 Data flow pipeline for Kibana data processing and graph generation

To illustrate what is displayed in RESLab testbed's dashboard, Figure 10 shows the ARP alerts that are generated when an ARP cache poisoning is detected based on the rules $R1$, $R2$, and $R3$, illustrated in Listing 3 in Section 5. In addition, DNP3 alerts are generated when DNP3 DIRECT OPERATE packets are detected as per $R4$ and $R5$ rules.

We can conclude that during the time-period an ARP spoof is detected, the DNP3 DIRECT OPERATE packet is rerouted to the adversary's node, which indicates that the MiTM script is modifying the operation. By monitoring the RESLab's dashboard, it is possible to determine with a one-minute resolution which DNP3 packets are potentially compromised and should be discarded. The one-minute intervals that contain the DNP3 packets that should be discarded are shown by the blue lines in Figure 10.

7 | CONCLUSION

Non-stealthy MiTM attacks can be developed and detected from features such as CRC mismatch, acknowledgements, and round trip times. However, if the intruder is stealthy enough to forge the CRCs, modify the acknowledgement packets, and reduce processing time by modifying selected DNP3 points in a payload, it can be difficult to detect such FDIs or FCIs.

This paper presented an automated tool that allows us to assess different cyber attack scenarios based on MiTM intrusions. We described this framework with step-by step algorithms in Section 4, including a graphical user interface to help users run different use cases. Additionally, we introduced a method to detect MiTM attack traffic by correlating Snort IDS alerts with ARP and DNP3 packet data, using network metrics such as retransmission rate and average RTT. It is important to monitor these metrics to detect the signature of a MiTM attack. The processing time at the adversary causes the RTT to increase, and the increased RTT causes

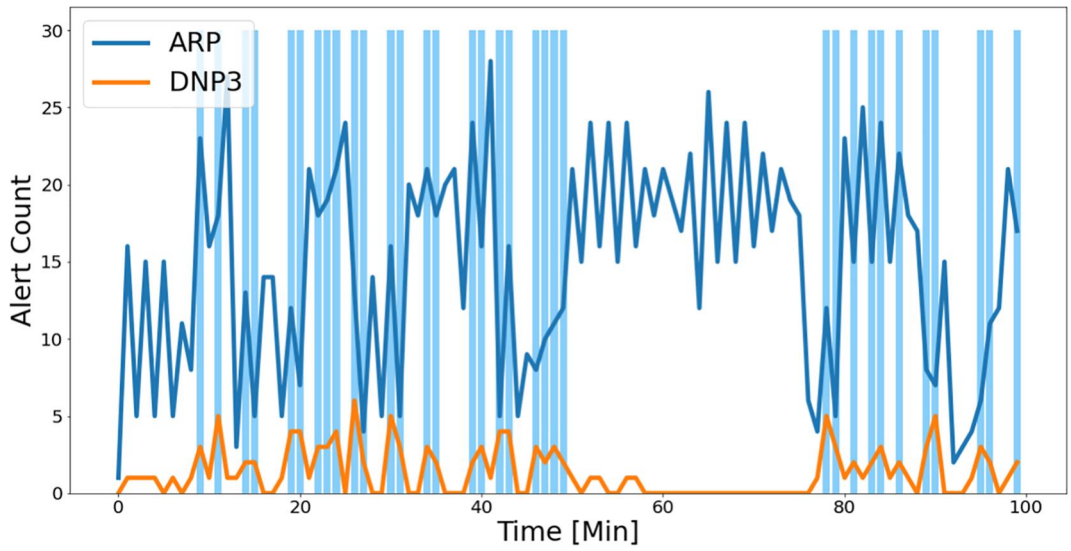

FIGURE 10 Address resolution protocol and Distributed Network Protocol-3 Snort alerts show potentially compromised DNP3 packets

retransmissions. These causal behaviours can be extracted in the form of timestamped features for training machine-learning-based detection algorithms.

Our results show that as the number of polled outstations increases, the DNP3 packets are delayed, and the efficiency of the MiTM attack decreases. This causes more DNP3 retransmissions, because consecutive packets from different outstations arrive at the adversary faster than the MiTM script can modify and forward the first packet. Also, we observe different processing times at the adversary for different types of DNP3 traffic. Read response packets had the longest processing time. Then, based on these results, we present defence recommendations such as showing how cyber telemetry can be used to detect stealthy MiTM attacks. Results show that while rule-based IDS such as Snort can detect ARP spoofs using existing pre-processors, they can still result in higher false positives due to rule selection criteria. Hence, this work also provides recommendations for future work to incorporate metrics such as average RTT and retransmission rate into security-centric data analysis on anomaly based attack detection.

ACKNOWLEDGEMENTS

This research is supported by the US Department of Energy's (DoE) Cybersecurity for Energy Delivery Systems program under award DE-OE0000895.

ORCID

Abhijeet Sahu https://orcid.org/0000-0002-7647-3758
Ana Goulart https://orcid.org/0000-0001-7184-7485

REFERENCES

1. Fritz, J.J., et al.: Simulation of man in the middle attack on smart grid testbed. In: 2019 SoutheastCon, pp. 1–6 (2019)
2. Langner, R.: Stuxnet: dissecting a cyberwarfare weapon. IEEE Security & Privacy. 9(3), 49–51 (2011)
3. Lee, R.M., Assante, M.J., Conway, T.: Analysis of the cyber attack on the Ukrainian power grid: defense use case by SANS ICS. https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf
4. Liang, G., et al.: The 2015 Ukraine blackout: implications for false data injection attacks. IEEE Transactions on Power Systems. 1–1, 11 (2016)
5. Targett, E.: High voltage attack: EU's power grid organisation hit by hackers. (2020). [Online]. Available: https://www.cbronline.com/news/eu-power-grid-organisation-hacked
6. Kundu, A., et al.: A3d: Attention-based auto-encoder anomaly detector for false data injection attacks. Elect Power Syst Res. 189, 106795 (2020). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378779620305988
7. Yuan, Y., Li, Z., Ren, K.: Modeling load redistribution attacks in power systems. IEEE Transactions on Smart Grid. 2(2), 382–390 (2011)
8. Wang, W., Lu, Z.: Cyber security in the smart grid: survey and challenges. Comput Network. 57, 1344–1371 (2013)
9. Liu, R., et al.: Analyzing the cyber-physical impact of cyber events on the power grid. IEEE Transactions on Smart Grid. 6(5), 2444–2453 (2015)
10. Kezunovic, M., et al.: The use of system in the loop, hardware in the loop, and co-modeling of cyber-physical systems in developing and evaluating new smart grid solutions. In Proceedings of the 50th Hawaii International Conference on System Sciences (2017)
11. Ashok, A., et al.: Experimental evaluation of cyber attacks on automatic generation control using a cps security testbed. In: 2015 IEEE Power & Energy Society General Meeting, pp. 1–5. IEEE (2015)
12. Burns, B., et al.: Security Power Tools. O'Reilly Media, Inc. (2007)
13. Chen, B., et al.: Implementing a real-time cyber-physical system test bed in rtds and opnet. In: 2014 North American Power Symposium (NAPS), pp. 1–6 (2014)
14. Yang, Y., et al.: Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid scada systems. In: International Conference on Sustainable Power Generation and Supply (SUPERGEN 2012), pp. 1–8 (2012)
15. Yang, Y., et al.: Multidimensional intrusion detection system for iec 61850-based scada networks. IEEE Trans Power Deliv. 32(2), 1068–1078 (2017)
16. Oyewumi, I.A., et al.: Isaac: the Idaho cps smart grid cybersecurity testbed. In: 2019 IEEE Texas Power and Energy Conference (TPEC), pp. 1–6 (2019)
17. Darwish, I., Saadawi, T.: Attack detection and mitigation techniques in industrial control system -smart grid dnp3. In: 2018 1st International Conference on Data Intelligence and Security (ICDIS), pp. 131–134 (2018)
18. Lu, X., et al.: On network performance evaluation toward the smart grid: a case study of dnp3 over tcp/ip. In: 2011 IEEE global telecommunications conference – GLOBECOM, pp. 1–6 (2011)
19. Zhang, Z., et al.: An event-triggered secondary control strategy with network delay in islanded microgrids. IEEE Systems Journal. 13(2), 1851–1860 (2019)
20. Zhang, Z., et al.: Delay-tolerant predictive power compensation control for photovoltaic voltage regulation. IEEE Transactions on Industrial Informatics. 17(7), 4545–4554 (2021)
21. Birchfield, A.B., et al.: Grid structural characteristics as validation criteria for synthetic networks. IEEE Trans Power Syst. 32(4) (2017)
22. Wlazlo, P., et al.: A cyber topology model for the Texas 2000 synthetic electric power grid. In: 2019 Principles, Systems and Applications of IP Telecommunications (IPTComm), pp. 1–8 (2019)
23. Clarke, G., Reynders, D., Wright, E.: Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems. Newnes (2004)
24. DNP Function Code Descriptions, 2020. [Online]. Available: https://www.proface.com/support/index?page=content&country=APS_GLOBAL&lang=en&locale=en_US&id=FA222356&prd=
25. Rosborough, C., Gordon, C., Waldron, B.: All about eve: comparing DNP3 secure authentication with standard security technologies for SCADA communications. In: 13th Australasian Information Security Conference, vol. 161 (2019)
26. Sahu, A., et al.: Design and evaluation of a cyber-physical resilient power system testbed. (2020). [Online]. Available: http://arxiv.org/abs/2011.13552
27. Ahrenholz, J., et al.: Core: a real-time network emulator. In: MILCOM 2008 – 2008 IEEE Military Communications Conference, pp. 1–7 (2008)
28. Glover, T.: Overbye, and sarma, 'powerworld simulator'. [Online]. Available: https://www.powerworld.com/products/simulator/overview
29. Orebaugh, A.D., Biles, S., Babbin, J.: Snort Cookbook. O'Reilly Media, Inc. (2005)
30. Elasticsearch, Logstash, Kibana (ELK). https://www.elastic.co/what-is/elk-stack
31. Rodofile, N., Radke, K., Foo, E.: Real-time and interactive attacks on dnp3 critical infrastructure using scapy. In: Proceedings of the 13th Australasian Information Security Conference (AISC 2015), pp. 67–70 (2015)
32. Van Rossum, G.: The Python Library Reference, Release 3.8.2. Python Software Foundation (2020)
33. Hawthorne, Mel: What is Mac Address Filtering? (2020). [Online]. Available: https://www.technipages.com/what-is-mac-address-filtering

How to cite this article: Wlazlo, P., et al.: Man-in-the-middle attacks and defence in a power system cyber-physical testbed. IET Cyber-Phys. Syst., Theory Appl. 6(3), 164–177 (2021). https://doi.org/10.1049/cps2.12014