Communication

Using Virtual Environment to Analyze Cyber-Attacks on Smart Grid Protocol

Shampa Banik [*] , Trapa Banik , Shudipta Banik

Posted Date: 14 September 2023

doi: 10.20944/preprints202309.0984.v1

Keywords: Smart grid (SG), communication protocols, Modbus, virtual testbed, cyber-attacks, and vulnerabilities.

Article

Using Virtual Environment to Analyze Cyber-Attacks on Smart Grid Protocol

Shampa Banik *, Mike Rogers, Rajesh Manicavasagam

Tennessee Technological University, Cookeville, USA; mrogers@tntech.edu (M.B.); rmanicava42@tntech.edu (R.M.)
* Correspondence: sbanik42@tntech.edu

Abstract: Smart grid capabilities have grown significantly in recent years. The smart grid provides advanced real-time handling of faults, advanced automatic control for efficient electricity transmission, monitoring and collection of the electrical system's capacity, and communication for information sharing. Unfortunately, its exposure to public networks makes it increasingly vulnerable to privacy breaches, vulnerabilities, and cyber-attacks. Cyber security threats and vulnerabilities in smart grid networks have become a primary concern that needs to be addressed before deploying a smart grid. Furthermore, the wide range of protocols increases the attack surface of a smart grid. This study focuses on the vulnerability of Modbus, which is regarded as one of the most prevalent protocols in smart grid communication networks. This paper presents preliminary findings of analyzing cyber-attacks against the Modbus protocol using a virtual testbed to investigate its effects on the smart grid network protocol. The concept incorporates an emulated Modbus/TCP network environment built from open-source software components that imitate fundamental industrial control features of the smart grid. Finally, we analyze the cycle of a cyber-attack leading through Reconnaissance to a DoS attack on the Modbus/TCP protocol and propose improvements to the test bed for protocol attack detection and mitigation.

Keywords: smart grid (SG); communication protocols; Modbus; virtual testbed; cyber-attacks; and vulnerabilities

1. Introduction

Smart Grid is a modern paradigm that has replaced the traditional electric grid in the energy sector. The concept of a smart grid system, as depicted in Figure 1, is the heterogeneous interconnection of dependent domains such as energy production, transmission, and distribution. The smart grid can harness programmable devices and networks to ensure two-way communication between the utility and the grid and between the utility and the consumers. Smart grids are highly automated and widely dispersed systems with advanced features such as real-time control, increased operational efficiency, grid resilience, and increased integration of renewable technology. An important impact of these features is the capability to minimize carbon footprint.

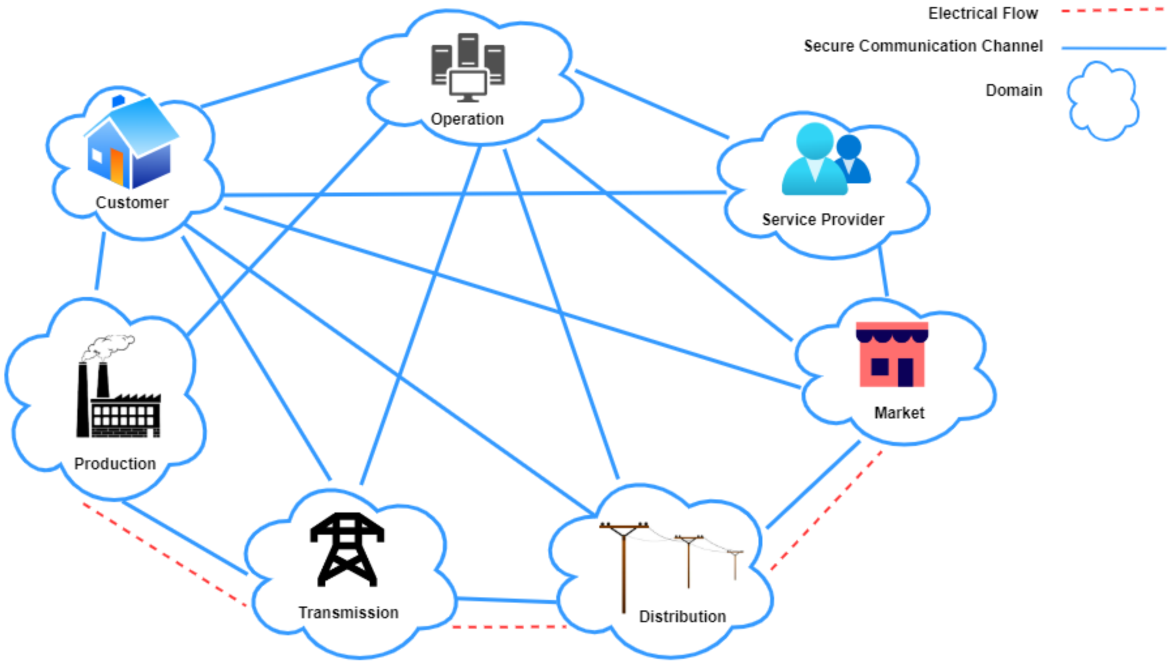


Figure 1. Smart Grid's conceptual model [1].

Smart grids combine disparate communication networks, such as those for the Internet of Things (IoT), industrial devices, wireless components, and Wireless Sensor Networks (WSN). Unfortunately, these protocols can be susceptible to various security threats. In smart grid architectures, these vulnerable protocols are typically connected to long-haul networks, such as the Internet, that expose them to malicious attackers. As a result, they pose serious security risks [2]. Network vulnerabilities could significantly impact private citizens and businesses, particularly if attackers target vital infrastructure like power plants, water distribution networks, and rail networks. Therefore, the government, businesses, and academia focus on securing smart grids.

Modbus is one of the most prominent network protocols found in smart grids. Traditional Modbus protocol cannot withstand cyberattacks as it lacks security and encryption. Information Technology (IT) and Operational Technology (OT) have seen exponential growth in security vulnerabilities for common network and SG technologies in recent years. These vulnerabilities are listed in the National Vulnerability Database (NVD) [3]. However, any vulnerability in the Modbus communication systems can affect the whole SG network.

To delve into any security research for such an Industrial Control System (ICS), particularly in the SG system, the testbed plays a pivotal role in modelling various cyber-physical system components that allow testing for security vulnerabilities and efficiency before the system is deployed [4]. Vulnerability analysis of the protocol based on testbed can mitigate the security risk *before* deployment and save money and possibly (and most importantly) lives.

This paper's contributions are as follows. First, we explore designing and developing a test bed in a virtual environment to simulate the smart grid system by mimicking the network between server and client. Second, we exploit the network vulnerabilities of Modbus/TCP by launching the Reconnaissance and DoS attacks followed by the attack cycle to monitor the effects of the system. We use Metasploit to simulate the attack on a simple test bed. We focus on Modbus and TCP protocols because they are two of the most widely used network protocols in smart grids. With this method, we analyze the results of the actions taken to compromise the smart grid's security. The findings of our study will be helpful for cyber forensics investigations into the Modbus protocol in the virtual testbed for the smart grid and projections for the future by overcoming the limitations.

In particular, the rest of this paper is organized as follows: Section 2 discusses the background literature. Section 3 outlined the Modbus/TCP protocol overview. Section 4 provides the experimental setup, followed by section 5with the experimental results and analysis. The limitation and future work appears in section 6. Finally, section 7 concludes this study.

2. Related Works

Recent, pertinent research on security on Modbus in the smart grid has been studied in depth. We attempted to categorize our findings in this work from many research viewpoints to explore the Modbus cyber attacks in the SG. The most relevant work to our research is in the categories of Modbus cyber-attacks, various detection techniques, including Intrusion Detection Techniques that can be deployed upon Modbus and testbed-based countermeasures to prevent cyber attacks on Modbus.
Modbus cyber attacks. A complete theoretical analysis of the multiple cyber-attacks against the Modbus protocol was published in [5]. The attacks' threat categories, targets, and control system asset impacts are summarized. The attack taxonomies define Modbus's control system and network security risks, making formal risk analysis easier. In [6], the authors conducted four distinct cyber-attacks on the Modbus protocol to compile pertinent data that machine learning algorithms may use. The authors identified intrusions that share traits with well-known assaults. They employed neural networks and decision trees to categorize the traffic produced by this simulated environment. Another review [7] presents different attack taxonomy as an intrusion on various protocols of the SG communication protocol with a recommendation of mitigation technique. The authors in [8] list 17 assaults on SCADA control systems. Attacks fall into four categories: reconnaissance, response and measurement injection, command injection, and denial of service.
Detection Techniques. T. Morris et al. published Modbus-related rules in [9], and popular signature-based IDS like Snort and Suricata can use these. Researchers developed the rule set for intrusion detection systems during a vulnerability study of the Modbus/TCP and Modbus over Serial Line networks. A Modbus/TCP fuzzer called MTF was designed and implemented by Voyiatz et al. [10]. Reconnaissance is a step in the MTF testing process that helps map the capabilities of the tested device and modify the attack vectors. In [11], the authors describe a series of experiments demonstrating the effectiveness of a signature-based Snort threshold module and an anomaly-based change detection method for spotting Modbus flooding attacks. In the research [12], the authors analyzed 37 cases and then looked at the Implementation and Detection of Modbus in the smart grid paradigm. The IDPS systems are an additional line of defence that strengthens cryptographic operations by promptly identifying and preventing potential security breaches. The paper's authors [13] proposed a two-fold IDS. They started by researching and improving the cyber-attacks offered by the Smod pen-testing program. The second step is introducing an intrusion detection system (IDS) based on anomalies that might spot intrusions that result in Modbus/TCP denial of service (DoS). The authors in [14] suggested an intrusion detection technique that uses a honeypot for the Modbus TCP protocol. The proposed solution uses an unsupervised clustering technique based on similarities between log sequences and cluster attacking patterns.
Testbed Based Solution. A CPS testbed called [15] was used to develop the Modbus/TCP Attack-Tree model to exploit several assaults. This model includes attacks such as replay, man-in-the-middle, denial-of-service, and reconnaissance. With the integration of assault time, detection time, and plant-hazard generating time, this serves as a formal risk assessment model. The approach in [16] is the development of a test bed based on the Modbus Protocol of nuclear power plants for cybernetic attacks analysis. The concept incorporates a simulated Modbus/TCP network environment with fundamental industrial control features created using free and open-source software. The paper [17] presented a virtual testbed for implementing a Man-in-The-Middle (MITM) attack on the Modbus protocol to analyze Modbus vulnerabilities, practical limitations of modern smart grid systems, and relevant prevention measures. The authors of [4,18] conducted an in-depth analysis of the known and reported network vulnerabilities in the Modbus protocol. They reviewed numerous attempts at exploiting such network vulnerabilities in creating the virtual test bed. Since the main concept of the most deployed testbed is a cyber-physical system that incorporates many simulation tools, Typhoon

HIL, unfortunately, lacks many important features; for example, TYfoon is not worried about the network as to why it has no packet analyzed. This is one of the crucial aspects of choosing a VM with all the necessary tools to assist in conducting any network security-related study or analysis.
Alternative Countermeasures. An alternative method is [19], which developed the MODLSTM Neural Network architecture, comprising input preprocessing, feature recording, and traffic classification to identify Modbus-related DoS assaults. A case study presents various countermeasures that can enhance security for the Modbus/TCP protocol in SCADA systems by studying and testing the current security countermeasures specific to SCADA systems [20]. A model for organizing information security in Modbus TCP was presented in [21]. The paper [22] surveys detection and mitigation techniques against denial-of-service (DoS) attacks on the Modbus protocol in the smart grid. It evaluates the effectiveness of the current approach and outlines the prospects for future research. Different machine learning-based anomaly detection techniques were applied to an industrial Modbus/TCP data set to detect fraudulent traffic in a synthetically generated Modbus/TCP communication data set in a hypothetical industrial setting. In [23], S. Anton et al. utilized and evaluated various machine learning classification techniques for identifying Modbus threats. An approach has been proposed to cluster cyber attacks targeting Modbus-enabled SCADA and ICS devices. The network telescope sensors have been deployed to empirically evaluate the approach on one month of real network data. Fachkha. et al. [24] presents a thorough investigation to determine cyber scanning activities into SCADA Modbus communication.

To our knowledge, there has not been enough published research on deploying a virtual environment as a testbed to carry out cyber analysis on any crucial protocol of the SG system and the literature on simulating testbed; nobody is concerned about the network security. After digging deeper into the recent studies on Modbus security-related works, we focused on conducting the cyber investigation on Modbus/TCP in a virtual testbed. Regarding network architecture and security concerns, our work significantly differs from others, which could open a new avenue to research in this domain.

The following aspects of a virtual machine (VM) have facilitated our work to serve as a promising virtual testbed for Modbus/TCP that imitates the real-time scenario.

- VM facilitates the real protocol setup into the Network architecture according to the experimental requirements. This setup has many tools that allow us immediate visualization and automation to carry on large-scale experiments.
- VMs allow automated experimental setup given dynamic parameters.
- VMs allow analysis both inside the VM as well as outside. The attacker can compromise analysis tools inside the VM but allow more specific and detailed analysis. Analysis outside the VM protects the analysis tools from the attacker. Can reference existing general-purpose black box tools.
- VMs isolate the network from dangerous attacks by simulating the network instead of sending the packets over the actual network.
- The VM has many tools available to do analysis, compared to (examples: packet analyzers, process image analyzers, memory analyzers, etc.).
- VM can be remotely operated, making the experiments far more flexible.

This testbed is developed to resolve the following security issues in Modbus protocol research.

- Establish a communication network between the Modbus components like the real-time testbed
- Analysis of the level of vulnerability of Modbus/TCP protocol
- Estimating the security impact on the system due to this exploitation
- Discussing the defence mechanism

3. Modbus/TCP Protocol

Modbus was initially designed for communication in industrial control systems and was developed more than 40 years ago by PLC manufacturer Modicon™, which is now Schneider Electric™.

It continues to be widely used for SCADA/ICS applications such as smart grid systems. Its use is widespread for several reasons: it is an open standard, simple to apply, and practically all commercial automation equipment comes with implementing it. Modbus is an OSI model level seven application layer messaging protocol that enables client/server communication between devices connected to various buses or networks [25]. A master/slave architecture is employed in Modbus to communicate with multi-drop devices. The Modbus TCP protocol enables communication between a master and its slaves in a LAN-based Modbus network and between multiple masters in IP-connected Modbus networks (each with multiple slaves). By allowing a slave to communicate concurrently with many masters and a master to have several transactions open simultaneously, Modbus TCP expands the capabilities of its serial counterpart [5]. Modbus/TCP master-slave, in other words, client-server communication, is illustrated in Figure 2.

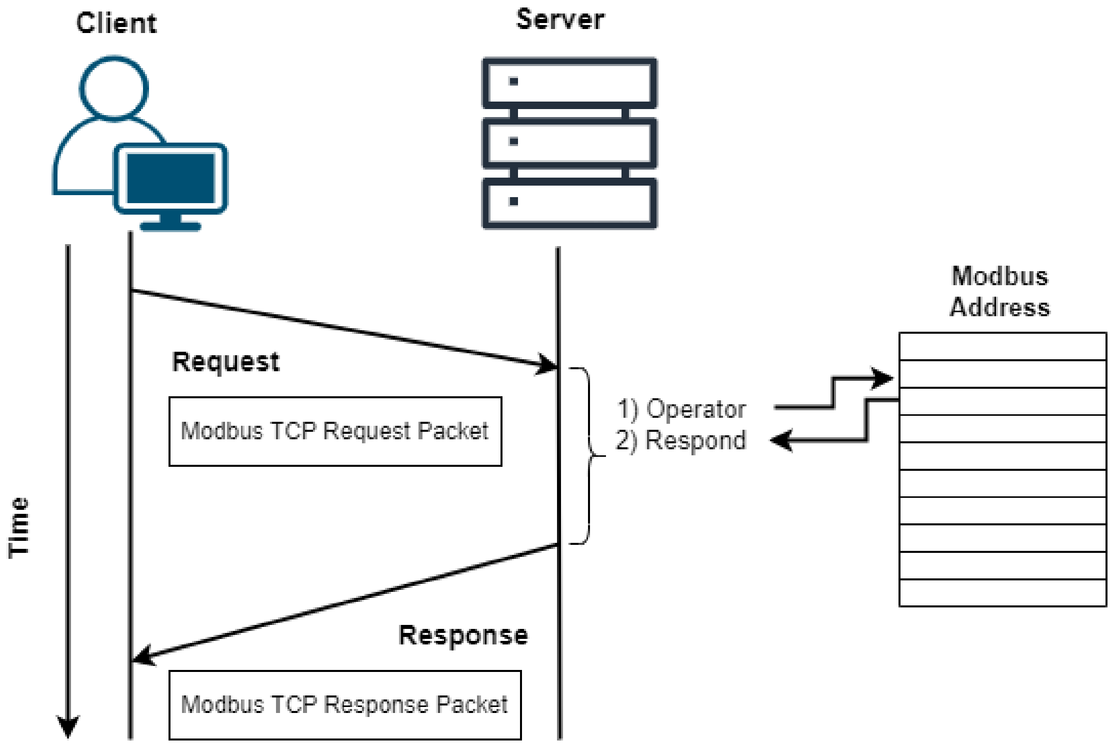


Figure 2. Modbus/TCP client-server communication.

4. Experimental Setup

In this section, we outline the specifications when developing the suggested testbed and discuss the principle that was the foundation for validating our design. The simulation strategy is software-based, and this virtual testbed component is described and discussed in detail below.

This paper presents a real-time virtual test bed as shown in Figure 3. Our testbed has been developed in a virtual environment integrating the Modbus server, client, and attacker.

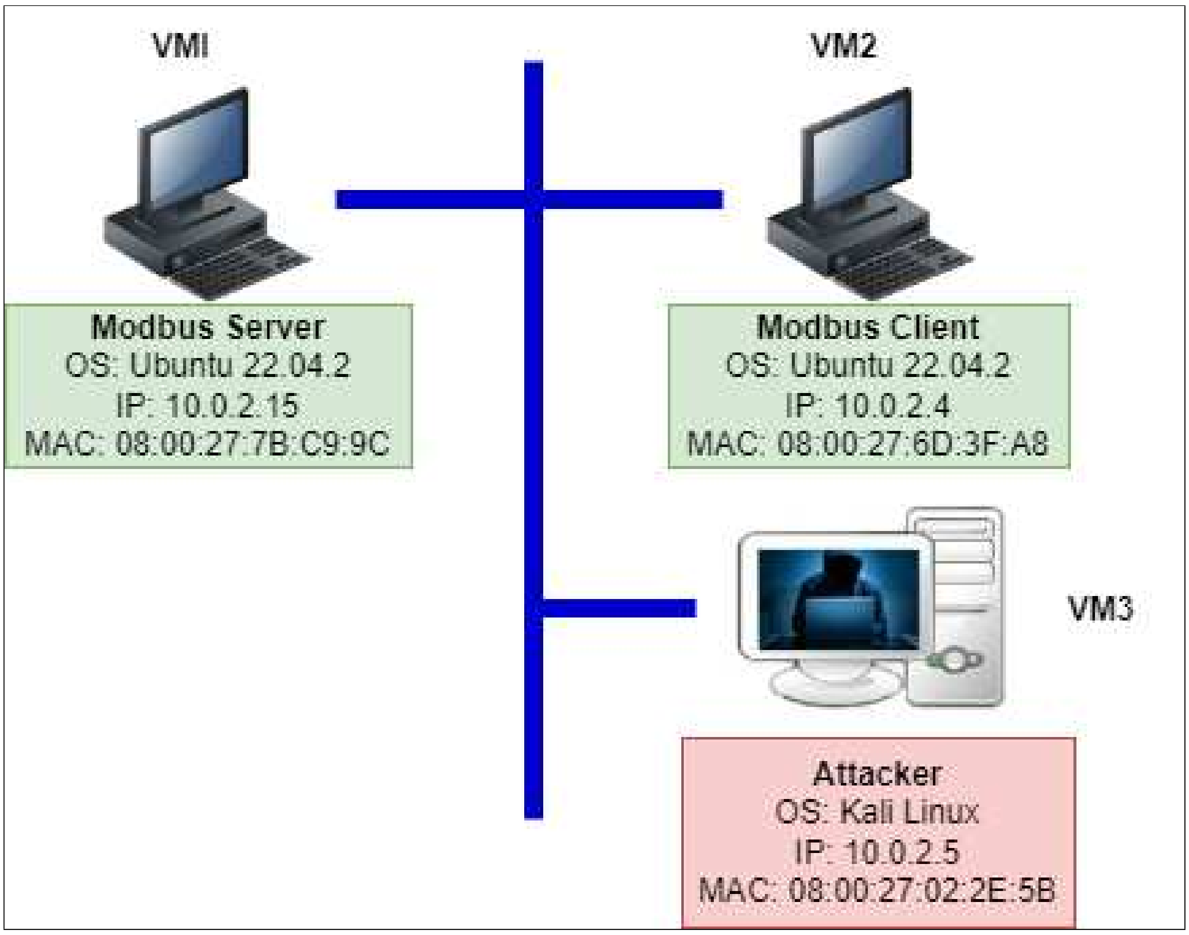


Figure 3. The Virtual Testbed Setup.

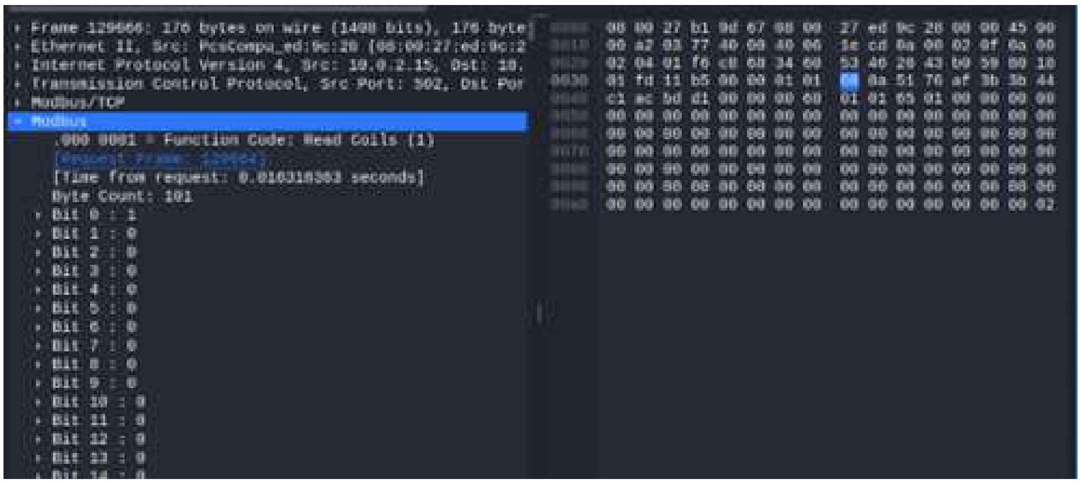
We employed a virtual machine to run the Modbus Master Simulator for the master (VM2). First, we create a simulated Modbus environment and test the connectivity between the server and the client. The emulated Modbus Master can access and control those Modbus Slaves through the Modbus Gateway as long as virtual machines 1 and 2 (VM1 and VM2) are connected. Numerous Modbus Slave Simulators are available. ModbusPal, a Java-based Modbus Slave emulator that works with any operating system that supports Java, is used in this project [26]. This software provides a graphical user interface; therefore, creating many Modbus slaves is simple.

We execute ModbusPal on Modbus Slave on one virtual machine (VM1) (which accepts TCP connections on port 502). It emulates a Modbus Gateway with the virtual machine's IP address while running ModbusPal on just one virtual machine (VM1) (accepts TCP connections on port 502). Each Slave we build using ModbusPal is identifiable by its Unit ID (ranging from 1 to 247).

Modbus Master and Slave Communication Environment Setup

Modbus is a master-slave protocol where they communicate with each other through IP addresses [11]. Therefore, attackers can use software such as Shodan [27] to find and remotely target Modbus installations for Internet-connected devices. Additionally, because various PLC manufacturers integrate the protocol into their devices as an option, and those devices respond to external commands without requiring authentication, they are vulnerable to injection attacks [9].

The Modbus packet has been captured to ensure the communication between the master and slave employing wireshark in Figure 4.



```
› Frame 129666: 176 bytes on wire (1408 bits), 176 byte
› Ethernet II, Src: PcsCompu_ed:9c:28 (08:00:27:ed:9c:2
› Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.
› Transmission Control Protocol, Src Port: 502, Dst Por
› Modbus/TCP
⌄ Modbus
     .000 0001 = Function Code: Read Coils (1)
     [Request Frame: 129664]
     [Time from request: 0.016316363 seconds]
     Byte Count: 101
   › Bit 0 : 1
   › Bit 1 : 0
   › Bit 2 : 0
   › Bit 3 : 0
   › Bit 4 : 0
   › Bit 5 : 0
   › Bit 6 : 0
   › Bit 7 : 0
   › Bit 8 : 0
   › Bit 9 : 0
   › Bit 10 : 0
   › Bit 11 : 0
   › Bit 12 : 0
   › Bit 13 : 0
   › Bit 14 : 0
```

```
0000   08 00 27 b1 9d 67 08 00  27 ed 9c 28 08 00 45 00
0010   00 a2 03 77 40 00 40 06  1e cd 0a 00 02 0f 0a 00
0020   02 04 01 f6 c8 68 34 60  53 40 20 43 b0 59 80 18
0030   01 fd 11 b5 00 00 01 01  08 0a 51 76 af 3b 3b 44
0040   c1 ac bd d1 00 00 00 00  01 01 65 01 00 00 00 00
0050   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0070   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00a0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 02
```

Figure 4. Wireshark Captured Modbus Packets.

5. Results and Analysis

The cyberattack scenario on the Modbus protocol is shown in Figure 5. The malicious hackers or attackers frequently follow four stages of the attack cycle to attack and acquire control of the system through the Modbus protocol. Those stages are *unauthorized access*, *reconnaissance*, *scanning*, and *exploitation*.

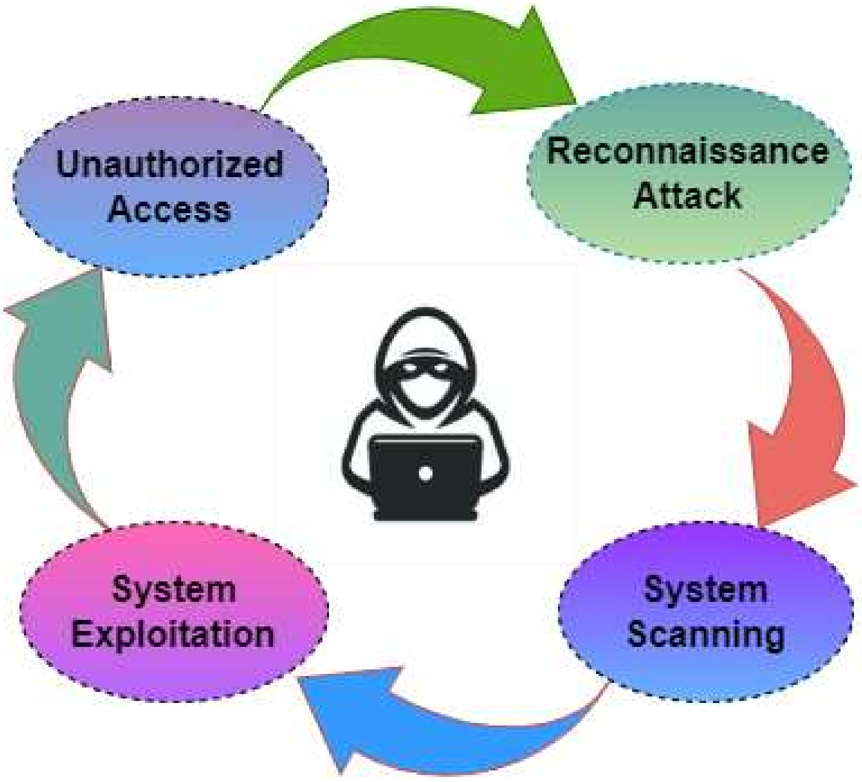


Figure 5. Attacking cycle followed by attackers to exploit the system.

Unauthorozed Access: In the first step, the attacker can get unauthorized access to the system by several means, such as manipulating any user login, temping user login, or installing covert and undetected software.
Reconnaissance Attacks & System Scanning: After obtaining unauthorized access to the system, reconnaissance is the second phase in which the attacker learns everything about the target. In the third stage, scanning, the attacker hunts for system network weaknesses. When an attacker targets a victim's system to learn about its weaknesses, this is known as a reconnaissance attack. These activities strive to discover the newly opened ports and learn about the services provided by each port, as well as their shortcomings. During the exploitation phase, the attacker seeks to compromise and gain complete control of the target by performing a wide range of potential cyber-attacks. Once the attacker starts exploiting the victim, the fourth step begins. In this virtual testbed, the Modbus/TCP has been the target of automated cyber attacks scripted in a scripting language.

The two components of this analysis to plan and carry out a reconnaissance attack on Modbus protocol are as follows: 1. Performing a network scan (IP address & MAC address) to discover our virtual network's IP addresses, MAC addresses, and open ports using NMAP. 2. To determine the registers, coils, and function codes available in the Modbus protocol using S-MOD.

In a reconnaissance assault, by using 'nmap', the attacker can only get to know the data and cannot alter or block packets as shown in Figures 6 and 7.

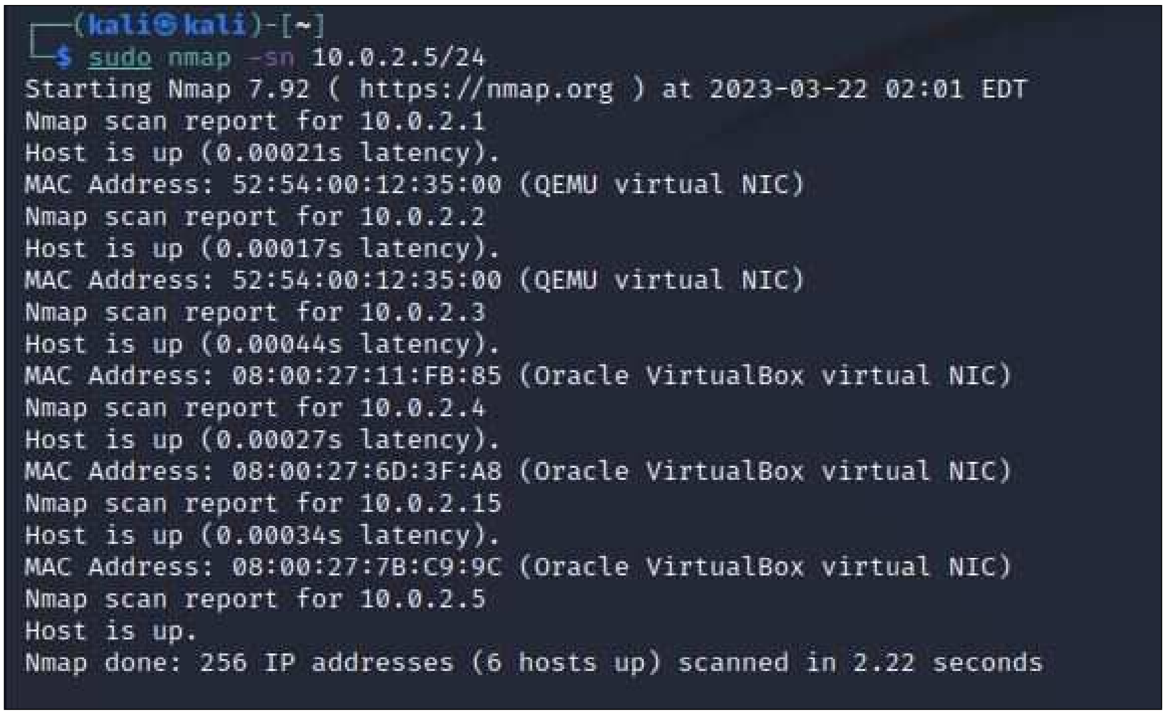
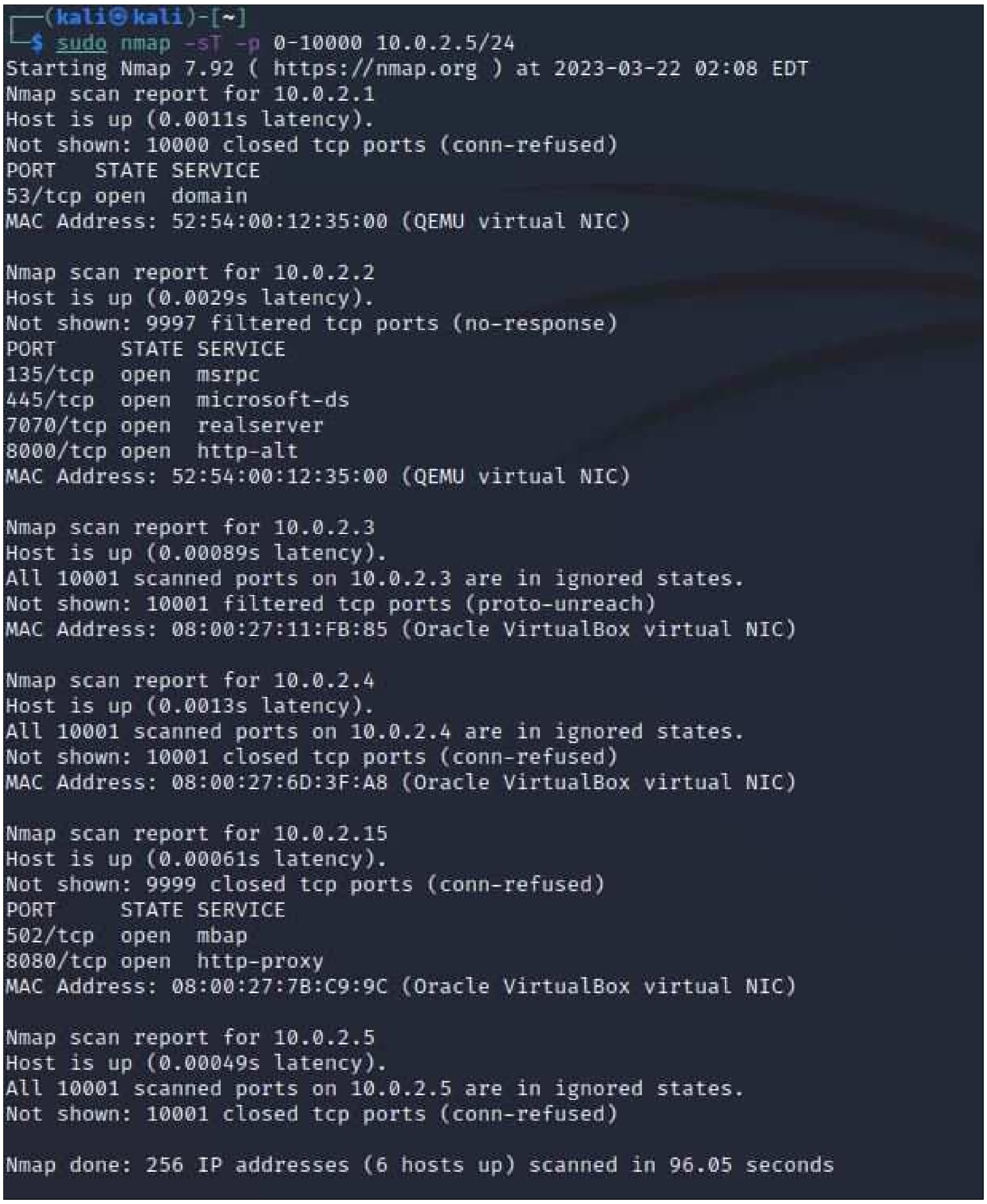

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sn 10.0.2.5/24
Starting Nmap 7.92 ( https://nmap.org ) at 2023-03-22 02:01 EDT
Nmap scan report for 10.0.2.1
Host is up (0.00021s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.00017s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00044s latency).
MAC Address: 08:00:27:11:FB:85 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4
Host is up (0.00027s latency).
MAC Address: 08:00:27:6D:3F:A8 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.15
Host is up (0.00034s latency).
MAC Address: 08:00:27:7B:C9:9C (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.5
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 2.22 seconds
```

Figure 6. NMAP scanning from KALI VM.

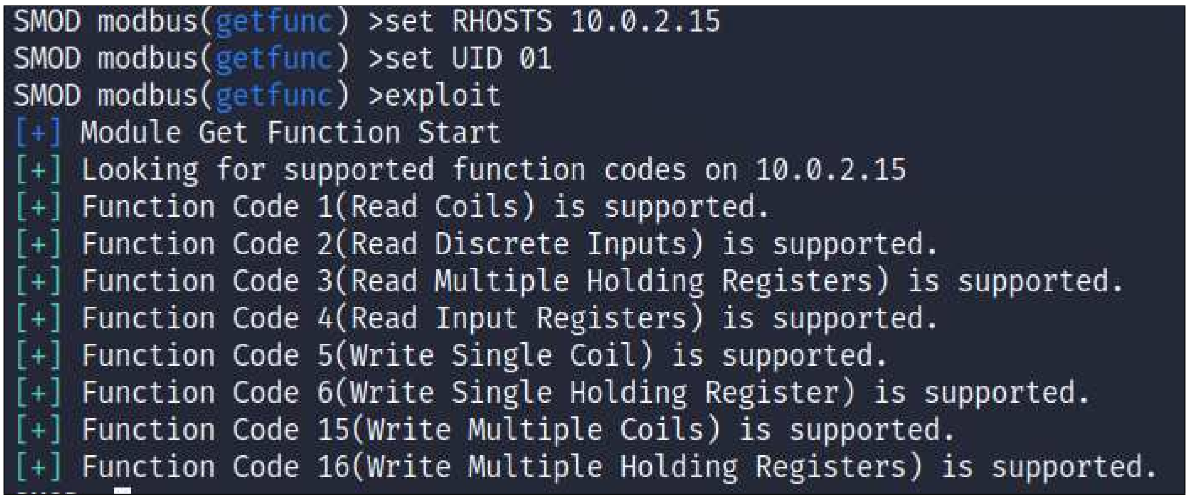


```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sT -p 0-10000 10.0.2.5/24
Starting Nmap 7.92 ( https://nmap.org ) at 2023-03-22 02:08 EDT
Nmap scan report for 10.0.2.1
Host is up (0.0011s latency).
Not shown: 10000 closed tcp ports (conn-refused)
PORT   STATE SERVICE
53/tcp open  domain
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)

Nmap scan report for 10.0.2.2
Host is up (0.0029s latency).
Not shown: 9997 filtered tcp ports (no-response)
PORT     STATE SERVICE
135/tcp  open  msrpc
445/tcp  open  microsoft-ds
7070/tcp open  realserver
8000/tcp open  http-alt
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)

Nmap scan report for 10.0.2.3
Host is up (0.00089s latency).
All 10001 scanned ports on 10.0.2.3 are in ignored states.
Not shown: 10001 filtered tcp ports (proto-unreach)
MAC Address: 08:00:27:11:FB:85 (Oracle VirtualBox virtual NIC)

Nmap scan report for 10.0.2.4
Host is up (0.0013s latency).
All 10001 scanned ports on 10.0.2.4 are in ignored states.
Not shown: 10001 closed tcp ports (conn-refused)
MAC Address: 08:00:27:6D:3F:A8 (Oracle VirtualBox virtual NIC)

Nmap scan report for 10.0.2.15
Host is up (0.00061s latency).
Not shown: 9999 closed tcp ports (conn-refused)
PORT     STATE SERVICE
502/tcp  open  mbap
8080/tcp open  http-proxy
MAC Address: 08:00:27:7B:C9:9C (Oracle VirtualBox virtual NIC)

Nmap scan report for 10.0.2.5
Host is up (0.00049s latency).
All 10001 scanned ports on 10.0.2.5 are in ignored states.
Not shown: 10001 closed tcp ports (conn-refused)

Nmap done: 256 IP addresses (6 hosts up) scanned in 96.05 seconds
```

Figure 7. NMAP Port scanning from KALI VM.

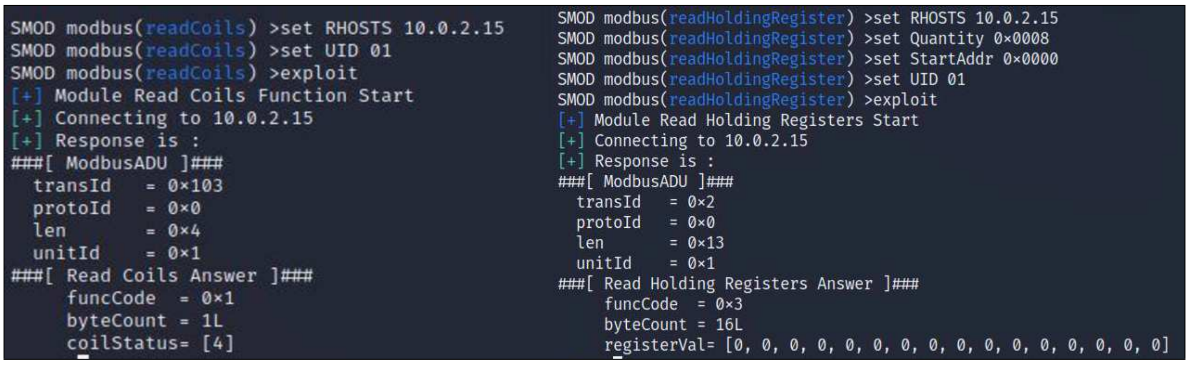
In the second attempt of reconnaissance assault, by using 'smod', the attacker is only able to observe registers, coils, and function codes that are available in the Modbus protocol as shown in both of Figures 8 and 9.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0984.v1



```
SMOD modbus(getfunc) >set RHOSTS 10.0.2.15
SMOD modbus(getfunc) >set UID 01
SMOD modbus(getfunc) >exploit
[+] Module Get Function Start
[+] Looking for supported function codes on 10.0.2.15
[+] Function Code 1(Read Coils) is supported.
[+] Function Code 2(Read Discrete Inputs) is supported.
[+] Function Code 3(Read Multiple Holding Registers) is supported.
[+] Function Code 4(Read Input Registers) is supported.
[+] Function Code 5(Write Single Coil) is supported.
[+] Function Code 6(Write Single Holding Register) is supported.
[+] Function Code 15(Write Multiple Coils) is supported.
[+] Function Code 16(Write Multiple Holding Registers) is supported.
```

Figure 8. Read the function code of Modbus protocol.



```
SMOD modbus(readCoils) >set RHOSTS 10.0.2.15
SMOD modbus(readCoils) >set UID 01
SMOD modbus(readCoils) >exploit
[+] Module Read Coils Function Start
[+] Connecting to 10.0.2.15
[+] Response is :
###[ ModbusADU ]###
  transId   = 0×103
  protoId   = 0×0
  len       = 0×4
  unitId    = 0×1
###[ Read Coils Answer ]###
     funcCode  = 0×1
     byteCount = 1L
     coilStatus= [4]
```

```
SMOD modbus(readHoldingRegister) >set RHOSTS 10.0.2.15
SMOD modbus(readHoldingRegister) >set Quantity 0×0008
SMOD modbus(readHoldingRegister) >set StartAddr 0×0000
SMOD modbus(readHoldingRegister) >set UID 01
SMOD modbus(readHoldingRegister) >exploit
[+] Module Read Holding Registers Start
[+] Connecting to 10.0.2.15
[+] Response is :
###[ ModbusADU ]###
  transId   = 0×2
  protoId   = 0×0
  len       = 0×13
  unitId    = 0×1
###[ Read Holding Registers Answer ]###
     funcCode  = 0×3
     byteCount = 16L
     registerVal= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Figure 9. Reading Holding Register and Coil Status Values of the Modbus protocol.

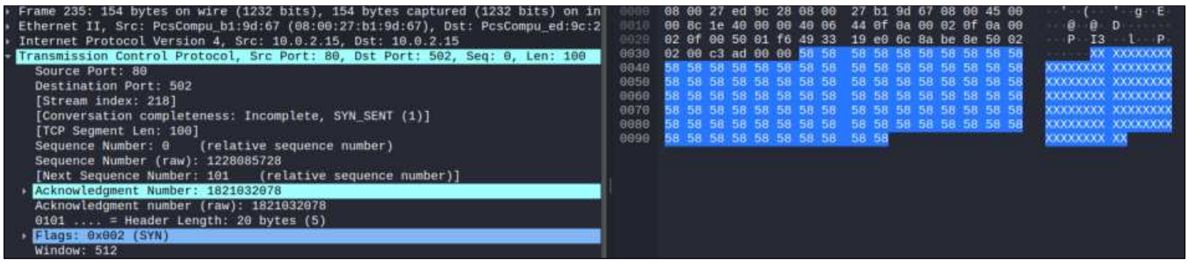
It is evident that function codes 1 through 4 allow the attacker to construct the reconnaissance attack. Thus, it is possible to access the network, allowing the attacker to see the resource data. However, in a reconnaissance attack, the function codes 5 through 16 cannot be used by the attacker to alter the values of registers or coils (because of write type data).

System Exploitation: The control centre will receive delayed information from IEDs during the DoS attack. If the DoS attack blocks the measurement, the control centre will not have an accurate image of the system's status. Smart grid applications in the control centre that rely on IED data will malfunction.

In our denial-of-service attacks, the *LAND assault* and *SYN Flood attacks* were developed in Part 1.

The LAND and SYN Flood assaults are two separate kinds of Denial of Service (DoS) attacks that aim to block access to a system by authorized users. Several attack strategies and weaknesses are used in each situation, which makes them distinct. In Part 1 of the DoS attack, we employed the LAND attack by the following command in the terminal of the Kali Linux machine.

Figure 10 illustrates the outcome of the LAND assault in Wireshark. When the LAND attack was successful, we discovered that the PLC's IP address was faked and that both the source and destination showed the same IP address, as shown in Figure 2. Data transfer uses the TCP protocol and the 502 receiving port. Additionally, it is demonstrated that repetitive network packets are sent to establish a connection and cause a system crash and denial of service to legitimate users.



```
▸ Frame 235: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on in
▸ Ethernet II, Src: PcsCompu_b1:9d:67 (08:00:27:b1:9d:67), Dst: PcsCompu_ed:9c:2
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.15
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 502, Seq: 0, Len: 100
    Source Port: 80
    Destination Port: 502
    [Stream index: 218]
    [Conversation completeness: Incomplete, SYN_SENT (1)]
    [TCP Segment Len: 100]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1228085728
    [Next Sequence Number: 101    (relative sequence number)]
  ▸ Acknowledgment Number: 1821032078
    Acknowledgment number (raw): 1821032078
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x002 (SYN)
    Window: 512
```

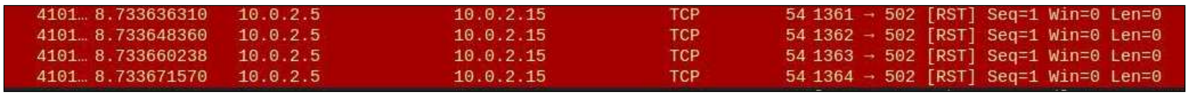
```
0000  08 00 27 ed 9c 28 08 00  27 b1 9d 67 08 00 45 00   ··'··(··'··g··E·
0010  00 8c 1e 40 00 00 40 06  44 0f 0a 00 02 0f 0a 00   ···@··@·D·······
0020  02 0f 00 50 01 f6 49 33  19 e0 6c 8a be 8e 50 02   ···P··I3··l···P·
0030  02 00 c3 ad 00 00 58 58  58 58 58 58 58 58 58 58   ······XX XXXXXXXX
0040  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0050  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0060  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0070  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0080  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0090  58 58 58 58 58 58 58 58  58 58                     XXXXXXXX XX
```

Figure 10. Packet captured in Wireshark after LAND command.

In Part 2 of the DoS attack, we performed the SYN flood attack by writing the following command in the terminal of the Kali Linux machine.

TCP SYN flood attack tactic was used to flood the Modbus Master to test the delay-sensitive nature of Modbus TCP [28]. A TCP SYN Flood Attack uses the TCP three-way connection handshake method to create a dependable session between a sender and a receiver [29]. The Modbus Master is subjected to a TCP SYN flood attack, which floods it with TCP 0 connection requests from possible Modbus clients with forged source IP addresses and arbitrary destination TCP ports, according to Breaker Status. The Modbus master was targeted by fake TCP SYN requests sent using the Hping tool [30].

Figure 11 shows the data collected in Wireshark after a successful SYN flood attack by initializing the target IP address and port number.

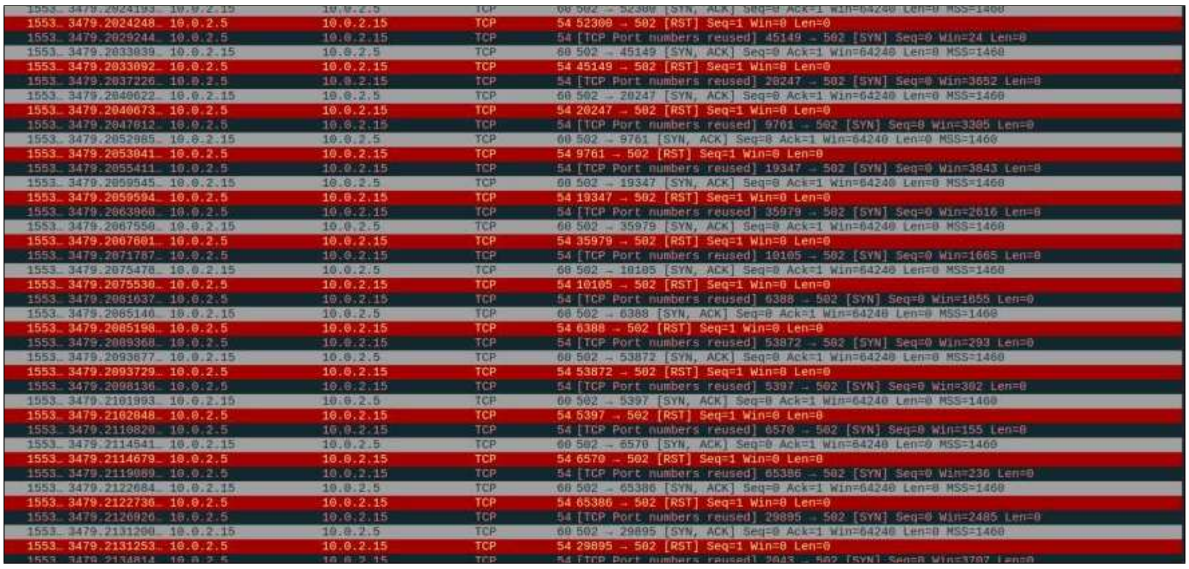


```
4101… 8.733636310  10.0.2.5   10.0.2.15  TCP  54 1361 → 502 [RST] Seq=1 Win=0 Len=0
4101… 8.733648360  10.0.2.5   10.0.2.15  TCP  54 1362 → 502 [RST] Seq=1 Win=0 Len=0
4101… 8.733660238  10.0.2.5   10.0.2.15  TCP  54 1363 → 502 [RST] Seq=1 Win=0 Len=0
4101… 8.733671570  10.0.2.5   10.0.2.15  TCP  54 1364 → 502 [RST] Seq=1 Win=0 Len=0
```

Figure 11. The captured packet in Wireshark after SYN flood attack.

Using Metasploit to perform a DoS Attack

In reality, Metasploit is a specific type of DoS attack in which the attacker takes advantage of flaws in the target system to make it crash or stop working. Penetration testing and exploitation are the primary uses of this Metasploit attack. It is distinguished by a unique traffic pattern shown in Figure 12.



```
1553… 3479.2024193… 10.0.2.15  10.0.2.5   TCP  60 502 → 52300 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2024248… 10.0.2.5   10.0.2.15  TCP  54 52300 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2029244… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 45149 → 502 [SYN] Seq=0 Win=24 Len=0
1553… 3479.2033039… 10.0.2.15  10.0.2.5   TCP  60 502 → 45149 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2033092… 10.0.2.5   10.0.2.15  TCP  54 45149 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2037226… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 20247 → 502 [SYN] Seq=0 Win=3652 Len=0
1553… 3479.2040622… 10.0.2.15  10.0.2.5   TCP  60 502 → 20247 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2040673… 10.0.2.5   10.0.2.15  TCP  54 20247 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2047612… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 9761 → 502 [SYN] Seq=0 Win=3305 Len=0
1553… 3479.2052985… 10.0.2.15  10.0.2.5   TCP  60 502 → 9761 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2053041… 10.0.2.5   10.0.2.15  TCP  54 9761 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2055411… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 19347 → 502 [SYN] Seq=0 Win=3843 Len=0
1553… 3479.2059545… 10.0.2.15  10.0.2.5   TCP  60 502 → 19347 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2059594… 10.0.2.5   10.0.2.15  TCP  54 19347 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2063960… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 35979 → 502 [SYN] Seq=0 Win=2616 Len=0
1553… 3479.2067550… 10.0.2.15  10.0.2.5   TCP  60 502 → 35979 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2067601… 10.0.2.5   10.0.2.15  TCP  54 35979 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2071787… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 10105 → 502 [SYN] Seq=0 Win=1665 Len=0
1553… 3479.2075478… 10.0.2.15  10.0.2.5   TCP  60 502 → 10105 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2075530… 10.0.2.5   10.0.2.15  TCP  54 10105 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2081637… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 6388 → 502 [SYN] Seq=0 Win=1855 Len=0
1553… 3479.2085140… 10.0.2.15  10.0.2.5   TCP  60 502 → 6388 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2085198… 10.0.2.5   10.0.2.15  TCP  54 6388 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2089368… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 53872 → 502 [SYN] Seq=0 Win=293 Len=0
1553… 3479.2093677… 10.0.2.15  10.0.2.5   TCP  60 502 → 53872 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2093729… 10.0.2.5   10.0.2.15  TCP  54 53872 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2098136… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 5397 → 502 [SYN] Seq=0 Win=302 Len=0
1553… 3479.2101993… 10.0.2.15  10.0.2.5   TCP  60 502 → 5397 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2102048… 10.0.2.5   10.0.2.15  TCP  54 5397 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2110820… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 6570 → 502 [SYN] Seq=0 Win=155 Len=0
1553… 3479.2114541… 10.0.2.15  10.0.2.5   TCP  60 502 → 6570 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2114679… 10.0.2.5   10.0.2.15  TCP  54 6570 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2119089… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 65386 → 502 [SYN] Seq=0 Win=236 Len=0
1553… 3479.2122684… 10.0.2.15  10.0.2.5   TCP  60 502 → 65386 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2122736… 10.0.2.5   10.0.2.15  TCP  54 65386 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2126926… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 29895 → 502 [SYN] Seq=0 Win=2485 Len=0
1553… 3479.2131200… 10.0.2.15  10.0.2.5   TCP  60 502 → 29895 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1553… 3479.2131253… 10.0.2.5   10.0.2.15  TCP  54 29895 → 502 [RST] Seq=1 Win=0 Len=0
1553… 3479.2134814… 10.0.2.5   10.0.2.15  TCP  54 [TCP Port numbers reused] 2043 → 502 [SYN] Seq=0 Win=3707 Len=0
```

Figure 12. The captured packet in Wireshark after Metasploit attack.

We carried out the LAND attack in Part 1 and then the SYN flood attack in Part 2. They both represent distinct Denial of Service (DoS) attacks. The LAND attack involves the attacker spoofing the victim's IP address as the packet source, sending packets to the victim's system with the SYN flag set, and setting the destination port to the same port as the source port. As a result, the victim's system constantly tries to connect with itself, using up resources and possibly resulting in system crashes. Furthermore, the SYN flood attack causes the target system to allocate resources for each connection attempt. It keeps the resources busy until they time out, which causes the system to become unresponsive or crash. The attacker floods the server with numerous SYN packets with bogus IP addresses, making it challenging for the server to discriminate between real and phony connection requests. Another observation is that the SYN flood attack executes much more quickly than the LAND attack. The main distinction between a LAND assault and a SYN flood attack is that the former overloads the server with connection requests as traffic. At the same time, the latter uses a flaw in the victim's operating system by delivering packets with a fake IP address. Lastly, the experiment has been accomplished successfully, proving the virtual simulation is good enough not to hamper any hardware in the loop.

6. Limitation and Future Work

Even though our strategy entails creating a test bed replicating a real-time smart grid system and studying how the cyberattack on Modbus affects it, it requires adding more hardware to the loop to identify the attack at a different network layer. Another limitation is that it is not simulating full devices but only simulating the mathematical properties of the respective devices. So, the problem is scalability, as the VM is heavyweight. If I have an SG with many devices and I spawn the VM for each device, my host will eventually be unable to simulate all the VMs. It can be overcome by simulating multiple devices within a single guest VM or having multiple network hosts running multiple guest VMs. The assessment was only done on a limited scale; it must also be verified on a larger size. We intend to extend our work performing with more potential cyber-attacks, such as man-in-the-middle (MiTM) and false data injection (FDI) on different protocols, for instance, on WiMax, DNP3 in the future, so that we may look into the system how it responds to an attack or after the attack has occurred. To validate, we can use some machine learning tools to verify our work in the following stage.

7. Conclusion

This paper identifies Modbus protocol vulnerabilities, exploits them to launch cyber-attacks and analyzes their influence on the SG system stability based on the virtual testbed. It is worth mentioning that during our experiment, we noticed that none of the packets entered our campus network, which indicates the virtual testbed can be considered a secured platform to conduct such a security-related experiment. Aggregating or visualizing data after running the experiment overnight is the nicest aspect of the virtual environmental setup. Simulating a network in such an environment is helpful for security, unlike other software simulations. Our findings showed that the Reconnaissance and DoS attacks can severely affect a system's operation and stability. The attacker can learn the system data and control commands into numerous smart grid applications; as a result, the system's security and stability are compromised. Any attacker who gains knowledge of the IP address of the Modbus master can target it. Case studies demonstrate that the Modbus protocol could be more trusting, as it lacks access control lists and a trusted domain.

References

1. Avi Gopstein, Avi Gopstein, Cuong Nguyen, Danielle Sass Byrnett, Kerry Worthington, and Christopher Villarreal. *Framework and Roadmap for Smart Grid Interoperability Standards regional Roundtables Summary Report*. US Department of Commerce, National Institute of Standards and Technology, 2020.
2. Adam Hahn, Aditya Ashok, Siddharth Sridhar, and Manimaran Govindarasu. Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid. *IEEE Transactions on Smart Grid*, 4(2):847–855, 2013.
3. Harold Booth, Doug Rike, and Gregory Witte. The national vulnerability database (nvd): Overview. 2013.

4. Ayesha Rahman, Ghulam Mustafa, Abdul Qayyum Khan, Muhammad Abid, and Muhammad Hanif Durad. Launch of denial of service attacks on the modbus/tcp protocol and development of its protection mechanisms. *International Journal of Critical Infrastructure Protection*, 39:100568, 2022.
5. Peter Huitsing, Rodrigo Chandia, Mauricio Papa, and Sujeet Shenoi. Attack taxonomies for the modbus protocols. *International Journal of Critical Infrastructure Protection*, 1:37–44, 2008.
6. Szu-Chuang Li, Yennun Huang, Bo-Chen Tai, and Chi-Ta Lin. Using data mining methods to detect simulated intrusions on a modbus network. In *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)*, pages 143–148. IEEE, 2017.
7. Shampa Banik, Trapa Banik, and Shudipta Banik. Intrusion detection system in smart grid-a review. Preprints, 2023.
8. Thomas H Morris and Wei Gao. Industrial control system cyber attacks. In *1st International Symposium for ICS & SCADA Cyber Security Research 2013 (ICS-CSR 2013) 1*, pages 22–29, 2013.
9. Thomas H Morris, Bryan A Jones, Rayford B Vaughn, and Yoginder S Dandass. Deterministic intrusion detection rules for modbus protocols. In *2013 46th Hawaii International Conference on System Sciences*, pages 1773–1781. IEEE, 2013.
10. Artemios G Voyiatzis, Konstantinos Katsigiannis, and Stavros Koubias. A modbus/tcp fuzzer for testing internetworked industrial systems. In *2015 IEEE 20th conference on emerging technologies & factory automation (ETFA)*, pages 1–6. IEEE, 2015.
11. Sajal Bhatia, Nishchal Singh Kush, Chris Djamaludin, Ayodeji James Akande, and Ernest Foo. Practical modbus flooding attack and detection. In *Proceedings of the Twelfth Australasian Information Security Conference (AISC 2014)[Conferences in Research and Practice in Information Technology, Volume 149]*, pages 57–65. Australian Computer Society, 2014.
12. Panagiotis Radoglou-Grammatikis, Ilias Siniosoglou, Thanasis Liatifis, Anastasios Kourouniadis, Konstantinos Rompolos, and Panagiotis Sarigiannidis. Implementation and detection of modbus cyberattacks. In *2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4. IEEE, 2020.
13. Panagiotis I Radoglou-Grammatikis and Panagiotis G Sarigiannidis. Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems. *IEEE Access*, 7:46595–46620, 2019.
14. Pin-Han Wang, I-En Liao, Kuo-Fong Kao, and Jyun-Yao Huang. An intrusion detection method based on log sequence clustering of honeypot for modbus tcp protocol. In *2018 IEEE International Conference on Applied System Invention (ICASI)*, pages 255–258. IEEE, 2018.
15. May Bashendy, Sohaila Eltanbouly, Ashraf Tantawy, and Abdelkarim Erradi. Design and implementation of cyber-physical attacks on modbus/tcp protocol. In *World Congress on Industrial Control Systems Security (WCICSS-2020)*, 2020.
16. Israel Barbosa de Brito and Rafael T de Sousa Jr. Development of an open-source testbed based on the modbus protocol for cybersecurity analysis of nuclear power plants. *Applied Sciences*, 12(15):7942, 2022.
17. Shampa Banik, Trapa Banik, SM Hossain, and Sohag Kumar Saha. Implementing man-in-the-middle attack to investigate network vulnerabilities in smart grid test-bed. *arXiv preprint arXiv:2306.00234*, 2023.
18. Abebe Tesfahun and D Lalitha Bhaskari. A scada testbed for investigating cyber security vulnerabilities in critical infrastructures. *Automatic Control and Computer Sciences*, 50:54–62, 2016.
19. Hao Zhang, Yuandong Min, Sanya Liu, Hang Tong, Yaopeng Li, and Zhihan Lv. Improve the security of industrial control system: A fine-grained classification method for dos attacks on modbus/tcp. *Mobile Networks and Applications*, pages 1–14, 2023.
20. John Luswata, Pavol Zavarsky, Bobby Swar, and Davison Zvabva. Analysis of scada security using penetration testing: A case study on modbus tcp protocol. In *2018 29th Biennial symposium on communications (BSC)*, pages 1–5. IEEE, 2018.
21. FR Ametov, EA Bekirov, and MM Asanov. Organizing the information security in modbus tcp interfaces for use in the energy complex. In *IOP Conference Series: Materials Science and Engineering*, volume 1089, page 012007. IOP Publishing, 2021.
22. Ines Ortega-Fernandez and Francesco Liberati. A review of denial of service attack and mitigation in the smart grid using reinforcement learning. *Energies*, 16(2):635, 2023.

23. Simon Duque Anton, Suneetha Kanoor, Daniel Fraunholz, and Hans Dieter Schotten. Evaluation of machine learning-based anomaly detection algorithms on an industrial modbus/tcp data set. In *Proceedings of the 13th international conference on availability, reliability and security*, pages 1–9, 2018.
24. Claude Fachkha. Cyber threat investigation of scada modbus activities. In *2019 10th IFIP international conference on new technologies, mobility and security (NTMS)*, pages 1–7. IEEE, 2019.
25. IDA Modbus. Modbus application protocol specification v1. 1a. *North Grafton, Massachusetts (www. modbus. org/specs. php)*, page 51, 2004.
26. Admin. Modbuspal - a java modbus simulator, 10-15-2019.
27. Search Engine for the Internet of Everything. https://www.shodan.io/. Accessed: 2023-09-04.
28. Bo Chen, Nishant Pattanaik, Ana Goulart, Karen L Butler-Purry, and Deepa Kundur. Implementing attacks for modbus/tcp protocol in a real-time cyber physical system test bed. In *2015 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pages 1–6. IEEE, 2015.
29. Tcp syn flooding and ip spoofing attacks.
30. Hping3 packet generator and analyzer.