Retrofitted PLC Digital Forensics: A Methodical Approach with Data Artifact Taxonomy and External Libraries

Young Park
Boise State University
Boise, ID, USA
youngpark@u.boisestate.edu

Sheikh Md. Mushfiqur Rahman
Boise State University
Boise, ID, USA
sheikhmdmushfiqu@u.boisestate.edu

Zejun Li
Boise State University
Boise, ID, USA
ZejunLi@u.boisestate.edu

ACM Reference Format:
Young Park, Sheikh Md. Mushfiqur Rahman, and Zejun Li. 2024. Retrofitted PLC Digital Forensics: A Methodical Approach with Data Artifact Taxonomy and External Libraries. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In an era characterized by the widespread integration of programmable logic controllers (PLCs) into critical infrastructure systems, the field of PLC digital forensics remains a critical yet understudied domain. As these versatile devices assume increasingly integral roles across various industries, there arises a pressing need to comprehend the digital artifacts they generate for conducting digital forensics investigation on PLC. This paper endeavors to address this challenge by proposing a method to conduct digital forensic investigation on PLC by collecting and analysing the data artifacts they produce using general purpose tools and libraries. The primary objectives encompass identifying the potential of third party libraries for PLC digital forensics investigation by using a taxonomy of PLC data types, and offering valuable insights that benefit both developers of industrial control systems (ICS) and forensic investigators.

Moreover, the research seeks to rectify a substantial gap in the current landscape of PLC digital forensics—a dearth of research and resources specific to this critical domain. Understanding and acquiring data artifacts from PLCs for forensic purposes pose distinct challenges, which we aim to unravel through this study. By presenting a case study that demonstrates an attack scenario on a Siemens PLC (SIMATIC S7-1500), we strive to contribute significantly to the enhancement of forensic readiness within heterogeneous infrastructures. Our findings have the potential to inform the development of proactive measures for forensic investigation of PLCs, further fortifying the safety of critical infrastructure systems in an increasingly digital world.

2 OBJECTIVE

The objective of the project is to use third-party tools to do a digital forensics investigation by collecting different types data artifacts (based on a taxonomy from the prior work) for Simens s7 1500 Plc and analyse them in a simulated cybercrime scenario. This objective suggests a practical and hands-on approach to understanding how the PLC's data may be affected or altered during a cybersecurity incident in addition to the usage of third party tools in digital forensics investigation which are not primarily built for digital forensics. It aligns with the broader field of digital forensics and can contribute valuable insights into the impact of cybersecurity threats on industrial control systems.

Conference'17, July 2017, Washington, DC, USA
© 2024 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 MOTIVATION

The motivation behind embarking on this research journey intersects with critical concerns in the domains of critical infrastructure and digital forensics, aligning with the following key points:

Understanding PLC Data Artifacts Across Diverse Industries: With the increasing reliance on programmable logic controllers (PLCs) across various industries, there arises an imperative to delve deeper into the digital artifacts they generate. These artifacts, often vendor-specific and unique, represent a relatively uncharted territory in the field of digital forensics.

Addressing the Need for Forensic Readiness in Heterogeneous Infrastructure: In an era marked by escalating cyber threats and evolving attack vectors, establishing forensic readiness within heterogeneous architectural landscapes becomes imperative. A robust understanding of PLC-specific digital artifacts is crucial not only for incident response but also for proactive security measures.

Making PLC digital forensics accessible: By using third party general purpose library functions, we try to simplify forensic investigation on a PLC for a broader audience.

Analyzing PLC Vendor Data Artifacts Taxonomy: The diversification of PLC vendors, each with its own distinct artifacts, poses significant challenges for forensic analysts. This research seeks to bridge these knowledge gaps by conducting an analysis of the data artifacts associated with a PLC using a Taxonomy.

Empowering Forensic Experts, Security Professionals, and Infrastructure Managers: Through this analysis, the aim is to empower forensic workers, including forensic experts, security professionals, and infrastructure managers, with valuable insights. These insights can inform the development of more effective protective measures and incident response strategies tailored to the differences of PLC digital artifacts.

Enhancing Critical Infrastructure Security and Resilience: At its core, this research is motivated by a commitment to enhance the security and resilience of critical infrastructure systems. In an ever-evolving digital landscape characterized by sophisticated cyberattacks, understanding the intricacies of PLC digital artifacts is pivotal.

In our pursuit of these objectives, the case study experiment we undertake holds the potential to inform the development of a more comprehensive matrix and taxonomy of PLC digital forensics data artifacts. These outcomes collectively represent a significant stride toward bolstering the fields of PLC digital forensics and cybersecurity, reinforcing the security of critical infrastructure systems in an era of growing digital interdependence.

4 PRIOR WORK

The field of digital forensics, particularly as it pertains to Programmable Logic Controllers (PLCs) and emerging technologies, has witnessed significant advancements in recent years. A notable contribution in this domain is by Shahbi et al. (2023) [8], who presented a comprehensive digital forensic taxonomy tailored specifically for PLC data artifacts. Their work underscores the complexities inherent in PLCs and offers a structured approach to extracting and analyzing digital evidence from these controllers, emphasizing the importance of understanding the unique characteristics of PLC data artifacts in forensic investigations. There are plenty of research in the field of digital forensics which focus on this issue. For instance, karie et al. (2017) [5] proposed a taxonomy for digital forensic evidence (DFE) that classifies DFE into a few well-defined and easily understood categories. Similarly, Azfar et al. (2016) [3] conducted a taxonomy on the data acquisition from different android application and what type of data those applications can collect. The idea behind the study is to enable forensic practitioners to acquire data from these applications quickly during investigation. Digital forensics for cloud is also receiving attention due to the popularity of cloud computing. Mishra et al. (2020) [6] proposed a taxonomy for cloud endpoint forensics tools which also focuses on data acquisition from virtual machine of a hypervisor system. The aim of the study is to make it easier for the practitioners to assist them selecting the proper tools for data acquisition for specific situations. Moreover, Stoyanova et al. (2020) [9] discussed the challenges of IoT forensics. The authors have discussed the importance of locating and identifying the evidence data and following a proper methodology of data acquisition to ensure the proper chain of custody. Cook et al.(2020) [4] introduced a taxonomy for forensics data types that can be obtained from PLCs. The taxonomy offers an organized method for collecting and analyzing digital evidence from PLCs, acting as a guide for investigators. Ahmed et al. (2017) [1] highlighted the challenges of PLC forensics, particularly given the lack of system logging. They suggest a unique methodology that combines machine learning techniques with logging pertinent memory addresses utilized by a PLC programs to detect unusual operations. This method improves the forensic examination of PLCs by giving a more effective and efficient way to examine digital evidence. Mishra et al. (2018) [7] offered a thorough taxonomy of forensic instruments made specifically for obtaining data from cloud endpoints. In order to help forensic investigators find tools that meet their technical needs during cloud forensic investigations, the taxonomy provides a searchable catalog. Alsabbagh et al. (2022) [2] conducted a practical attack on S7-1500 PLCs, showcasing a real-world application of these forensic principles. Their research not only demonstrates the practicalities of such attacks but also underscores the necessity for continued development in digital forensic methodologies and tools, particularly in the domain of PLCs. This integration of theory and practice highlights the dynamic and evolving nature of digital forensics in the face of emerging technologies and sophisticated cyber threats.

5 PROPOSED METHODOLOGY

A matrix-based approach is one possible systematic approach to comparing and contrasting the data artifacts acquired from S7-1500 PLC. This approach would involve creating a matrix that lists the data artifacts relevant to PLC digital forensics. Figure 1 illustrates the categories of data artifacts in PLC which are suitable for digital forensics. Each proposed matrix will focus on a specific category of data type and contain different columns for S7-1500 PLC, and each row would represent a different type of data artifact from a specific category. We would then extract and analyze the data artifacts for each vendor and fill in the corresponding cells in the matrix. This approach would also allow us to identify gaps in the taxonomy and refine them based on the comparison findings. The main contribution of this experiment will be to provide digital forensics practitioners a systematic approach to acquire any analyse data artifacts from specific PLCs which can reduce the cost of time and effort in a digital forensics investigation. The provided illustration in Figure 3 serves as a sample matrix designed for a particular type of data elements referred to as "Process Variable Content Values." These data elements encompass time-stamped values of different variables, updated during each execution cycle to manage logical operations.

For collecting each types of data artifact and creating the matrix, we will first run S7-1500 PLC with simple ladder logic programs so that data artifacts can be generated for collection. In this experiment we will mainly focus on data acquisition using two separate ways: 1) use manufacturer specific software, 2) Third-party open-source communication library. The methodology for this experiment can be summarized as follows:

1. Initiation of PLCs with Ladder Logic Program: The process begins by creating a basic ladder logic program. This program is then downloaded onto the PLCs and run for a certain duration to generate data artifacts.

2. Collection of Data Artifacts: The focus shifts to gathering essential data artifacts which include the Operating Systems, User Control Logic Code, PLC Device Meta-Data, Built-in Diagnostic Logs, and Process Variable Content Values.

3. Techniques for Data Acquisition: Two primary methods are employed for data acquisition. The first involves using manufacturer-specific software, tailored for S7-1500 PLCs. The second method utilizes a third-party communication library, offering a broader range of data collection capabilities.

4. Matrix Creation: For each category of data artifacts, a distinct matrix is created. This matrix is structured with a column dedicated to the S7-1500 PLC and rows corresponding to each type of data artifact within the selected category.

5. Populating the Matrix: Data artifacts collected from the S7-1500 PLCs are then systematically entered into the matrix. Each entry in the matrix represents a specific category of the collected data artifact.

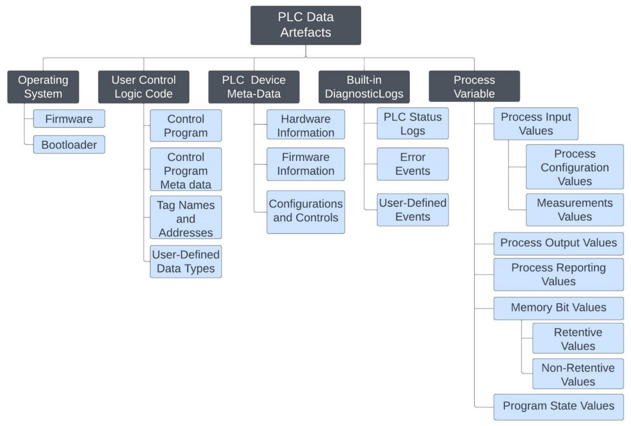


Figure 1: Data Artifact Categories

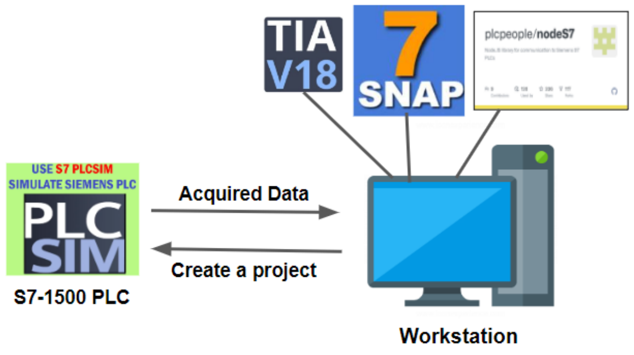


Figure 2: Illustration of Methodology

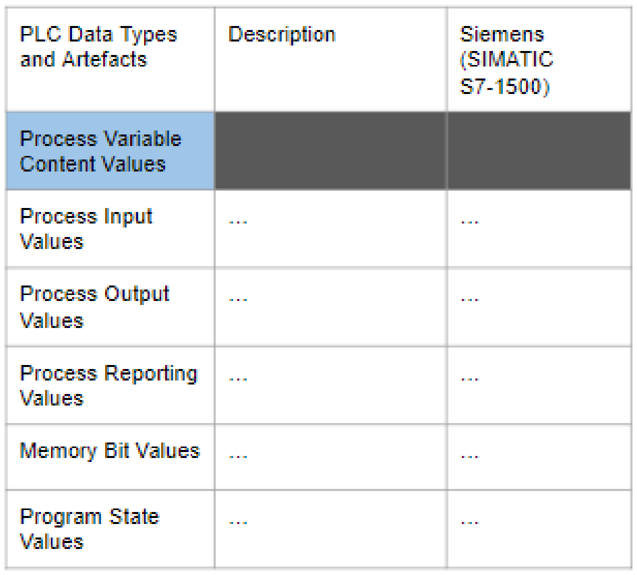
6. Taxonomy Refinement: Based on the comparative analysis of the data in the matrices, the taxonomy is refined. This refinement may include the addition of new categories or subcategories to better classify the data artifacts.

7. Application of an Attack Scenario to the Taxonomy: The final step involves applying a simulated attack scenario to the taxonomy. This step is crucial to compare the data artifacts within the taxonomy against those generated in an attack scenario. It also serves to check if the taxonomy comprehensively includes all artifacts pertinent to such scenarios. This application ensures that the taxonomy is robust and effectively covers a wide range of possible forensic scenarios.

6 PROPOSED CASE STUDY

6.1 S7-1500 PLC Architecture

Siemens’ SIMATIC S7 family includes several PLC product lines such as the S7-300, S7-400, S7-1200, and the S7-1500, which is the specific model utilized in our context. Each of these models shares a common architectural framework. The standard architecture of an S7-1500 PLC, as depicted in Fig. 4, consists of essential components such as input and output modules, a power supply, and various forms of memory including Random Access Memory (RAM) and



PLC Data Types and Artefacts	Description	Siemens (SIMATIC S7-1500)
Process Variable Content Values		
Process Input Values	...	...
Process Output Values	...	...
Process Reporting Values	...	...
Memory Bit Values	...	...
Program State Values	...	...

Figure 3: The Matrix Format

Electrically Erasable Programmable Read-only Memory (EEPROM). The firmware or Operating System (OS) of the PLC, along with user-specific programs, is stored in the EEPROM. In the operation

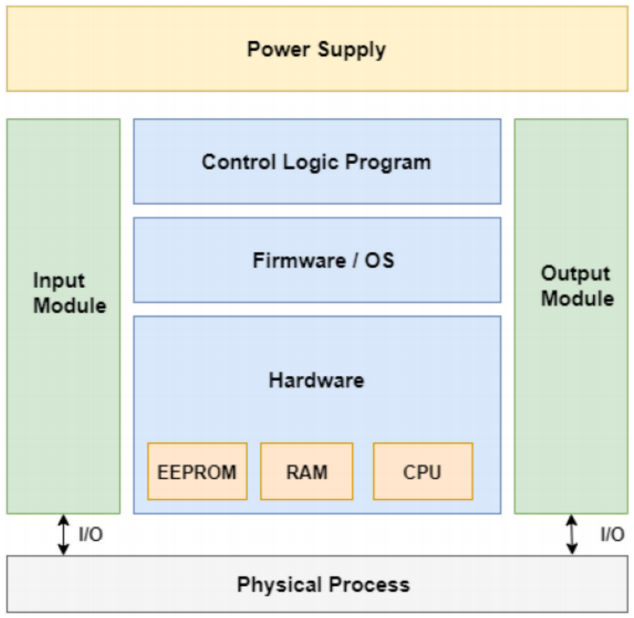


Figure 4: A S7-1500 PLC Architecture

of the S7-1500 PLC, input and output devices like sensors, switches, relays, and valves play a crucial role. These devices are connected to the PLC’s input and output modules. The PLC is an integral part of a physical process; it receives data about the current state of the process from the input devices, processes this information through its control logic, and then controls the physical process accordingly through the output devices.

The control logic for the S7-1500 PLC is programmed and compiled into a lower-level code representation, specifically into MC7+ bytecode. This compilation is done by the engineering station. Once the code is compiled, its blocks in the MC7/MC7+ format are downloaded and installed into the PLC. This process is facilitated by Siemens' s7CommPlus protocol, which is specifically used for S7-1200 and S7-1500 PLCs. Following installation, the MC7/MC7+ virtual machine within the S7-1500 PLC dispatches, interprets, and executes the bytecode blocks, ensuring the PLC functions as programmed to control and interact with the connected physical processes.

Siemens PLCs are equipped with a real-time operating system (OS) that initiates cycle time monitoring, a process detailed in [Figure 5]. This operating cycle involves four key steps executed by the CPU.

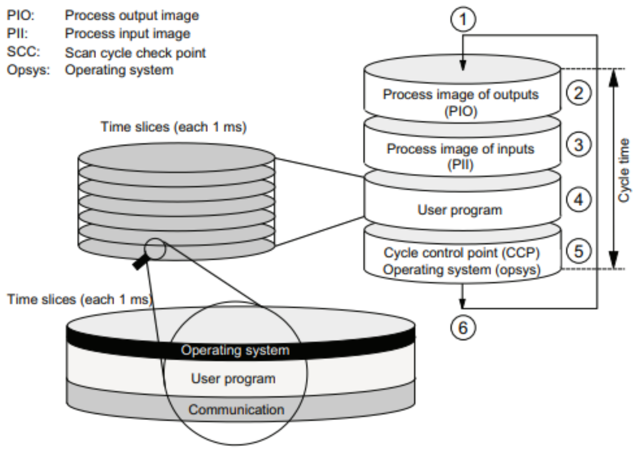


Figure 5: A overveiw of cyclic program processing

In the first step, the CPU transfers the values from the process image of the outputs to the output modules. This step ensures that the output modules accurately reflect the current state of the process image.

The second step involves the CPU reading the status of the input modules. After reading, it updates the process image with the input values. This step is crucial for ensuring that the PLC accurately reflects the real-time status of the inputs.

In the third step, the user program is run in specific time slices, each lasting 1 millisecond (ms). These time slices are further divided into three distinct parts, which are executed one after the other: the operating system, the user program, and communication tasks. The way these parts are sequenced and managed within each time slice ensures efficient processing.

The number of time slices allocated for the execution of the user program varies. It depends on factors such as the complexity of the current user program and the occurrence of any events that might interrupt the program's execution. This dynamic allocation of time slices allows the PLC to manage its tasks efficiently, adjusting to the demands of the user program and external events as needed.

User Programs designed for Siemens S7 PLCs are structured into various units, including Organization Blocks (OBs), Functions (FCs), Function Blocks (FBs), Data Blocks (DBs), System Functions (SFCs), System Function Blocks (SFBs), and System Data Blocks (SDBs), as illustrated in Fig. 6.

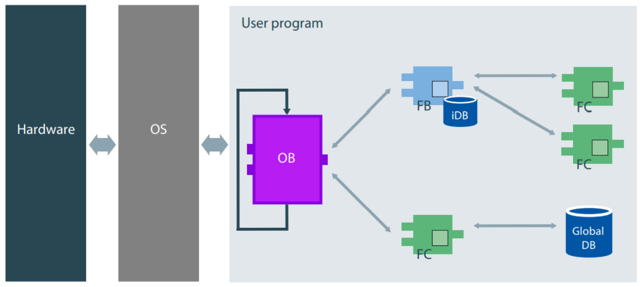


Figure 6: A S7 PLC's controller programming

The core of the program's code is found in OBs, FCs, and FBs. DBs are used to store data structures, while SDBs hold information regarding the PLC's current configurations. The internal data storage within the PLC is addressed using the prefix 'M', denoting memory.

A basic S7 PLC program typically includes at least one Organization Block, referred to as OB1. This OB1 is analogous to the main() function in a traditional C program. For more complex programming needs, engineers may encapsulate code using functions and function blocks. The primary distinction when using a Function Block, as opposed to a Function, lies in the inclusion of an additional DB as a parameter during the FB call.

SFCs and SFBs are integrated into the PLC's firmware. The operating system of the PLC is responsible for cyclically calling the OB, which in turn triggers the cyclic execution of the user's program. This systematic approach to program structure and execution allows for organized and efficient operation of the PLC, catering to both simple and complex programming requirements.

6.2 Testbed Setup

For the test set up, we are focusing on Siemens S7 1500 PLC by using Totally Integrated Automation Portal (TIA Portal) V18 and SIMATIC S7-PLCSIM Advanced 5 to create and simulate the project. We created a project that included two devices: CPU 1515-2 PN and SIMATIC PC Station with an IP Address of 192.168.1.1.

6.2.1 ***Coding Information.*** For the external data acquisition S7-1500 PLC, Python code and JavaScript code have been used. These scripts require an understanding of the S7-1500 PLC architecture and programming principles. The following is an overview of the implemented Python code using the Snap7 library:

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++
import snap7
import time
from snap7.util import *
from snap7.types import *

# PLC connection settings
```

```
IP = "192.168.1.1"
RACK = 0
SLOT = 1
DB_NUMBER = 1
START_ADDRESS = 0
SIZE = 260

# Establish connection to the PLC
plc = snap7.client.Client()
plc.connect(IP, RACK, SLOT)

# Function definitions for reading PLC data
def read_output_bit(byte, bit):
    ...

def read_bit(db_number, byte_number, bit_number):
    ...

# Displaying various information from the PLC
print("++ PLC CPU Date and Time ++")
print(f"PLC CPU Date and Time: {plc.get_plc_datetime()}")
...

# Code for reading and displaying block information
block_list = plc.list_blocks()
...

# Displaying the state of various motors
print("++ PLC Program Data Block Information ++")
...

# Reading and displaying process variable content values
addresses_with_names = [...]
...

# Displaying PLC CPU operation status and security information
state = plc.get_cpu_state()
print("++ PLC CPU Operation Status Mode ++")
print(f"Current Mode is: {state}")
...
++++++++++++++++++++++++++++++++++++++++++++++++++++
```

This Python code is a simplified representation of the full implementation and is intended to illustrate the process of interfacing with the S7-1500 PLC.

The JavaScript code was implemented using the NodeS7 library to facilitate communication with the S7-1500 PLC. The code is responsible for reading and writing data to the PLC. Below is the key part of the script:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++
var nodes7 = require('nodes7'); // NodeS7 package
var conn = new nodes7;
var doneReading = false;
var doneWriting = false;

// Define PLC variables
var variables = {
  MOTOR1_OUTPUT: 'Q0.0',     // Output at byte 0, bit 0
  MOTOR1_START: 'DB2,X0.0',   // DB2, byte 0, bit 0
  ...
  PLC_DATE_TIME: 'DB2,DT0'
};

// Initiate connection to the PLC
conn.initiateConnection({
  port: 102,
  host: '192.168.1.1',
  rack: 0,
  slot: 1,
  debug: false
}, connected);

// Connected callback
function connected(err) {
  if (typeof(err) !== "undefined") {
    console.log(err);
    process.exit();
  }
  ...
}

// Function to handle read values
function valuesReady(anythingBad, values) {
  ...
}

// Function to handle write confirmation
function valuesWritten(anythingBad) {
  ...
}
++++++++++++++++++++++++++++++++++++++++++++++++++++
```

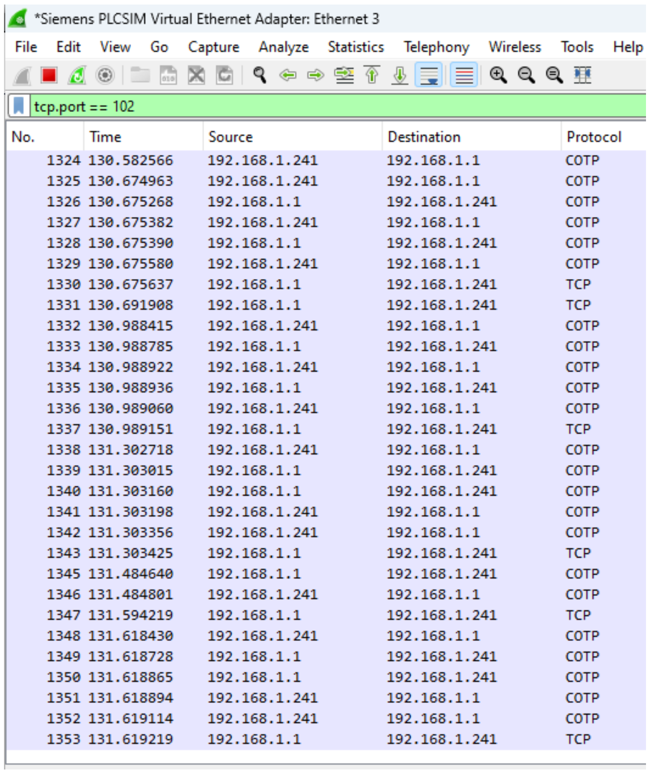
This Javascript code demonstrates the essential parts of the JavaScript code, including PLC connection setup, variable definition, and functions for reading and writing PLC data. The full code contains additional details and error handling mechanisms.

6.2.2 ***S7CommPlus Protocol.*** S7CommPlus Protocol was also utilized to aquire the data from S7-1500 PLC externally. In our attempt to analyze data using Wireshark with the s7commPlus protocol, we encountered an absence of the application layer in the network traffic. We can get the only TCP and COTP (Connection-Oriented Transport Protocol). This was due to the fact that while the application layer was designed to support s7commPlus, our specific setup did not generate any s7commPlus protocol traffic. This limitation arose because we utilized an s7 emulator, which did not support the protocol as expected [Figure 7].

6.3 Data Artifact Collection

6.3.1 ***Operating System.*** For the information about the Operating System, neither Python code nor Javascript code is supported to collect the data artifacts for the firmware version and bootloader.

6.3.2 ***User Control Logic Code.*** To find the information about the User Control Logic Code, we implemented Python code and


*Siemens PLCSIM Virtual Ethernet Adapter: Ethernet 3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 102

No.	Time	Source	Destination	Protocol
1324	130.582566	192.168.1.241	192.168.1.1	COTP
1325	130.674963	192.168.1.241	192.168.1.1	COTP
1326	130.675268	192.168.1.1	192.168.1.241	COTP
1327	130.675382	192.168.1.241	192.168.1.1	COTP
1328	130.675390	192.168.1.1	192.168.1.241	COTP
1329	130.675580	192.168.1.241	192.168.1.1	COTP
1330	130.675637	192.168.1.1	192.168.1.241	TCP
1331	130.691908	192.168.1.1	192.168.1.241	TCP
1332	130.988415	192.168.1.241	192.168.1.1	COTP
1333	130.988785	192.168.1.1	192.168.1.241	COTP
1334	130.988922	192.168.1.241	192.168.1.1	COTP
1335	130.988936	192.168.1.1	192.168.1.241	COTP
1336	130.989060	192.168.1.241	192.168.1.1	COTP
1337	130.989151	192.168.1.1	192.168.1.241	TCP
1338	131.302718	192.168.1.241	192.168.1.1	COTP
1339	131.303015	192.168.1.1	192.168.1.241	COTP
1340	131.303160	192.168.1.1	192.168.1.241	COTP
1341	131.303198	192.168.1.241	192.168.1.1	COTP
1342	131.303356	192.168.1.241	192.168.1.1	COTP
1343	131.303425	192.168.1.1	192.168.1.241	TCP
1345	131.484640	192.168.1.1	192.168.1.241	COTP
1346	131.484801	192.168.1.241	192.168.1.1	COTP
1347	131.594219	192.168.1.1	192.168.1.241	TCP
1348	131.618430	192.168.1.241	192.168.1.1	COTP
1349	131.618728	192.168.1.1	192.168.1.241	COTP
1350	131.618865	192.168.1.1	192.168.1.241	COTP
1351	131.618894	192.168.1.241	192.168.1.1	COTP
1352	131.619114	192.168.1.241	192.168.1.1	COTP
1353	131.619219	192.168.1.1	192.168.1.241	TCP

Figure 7: Emulator S7 PLC in Wireshark

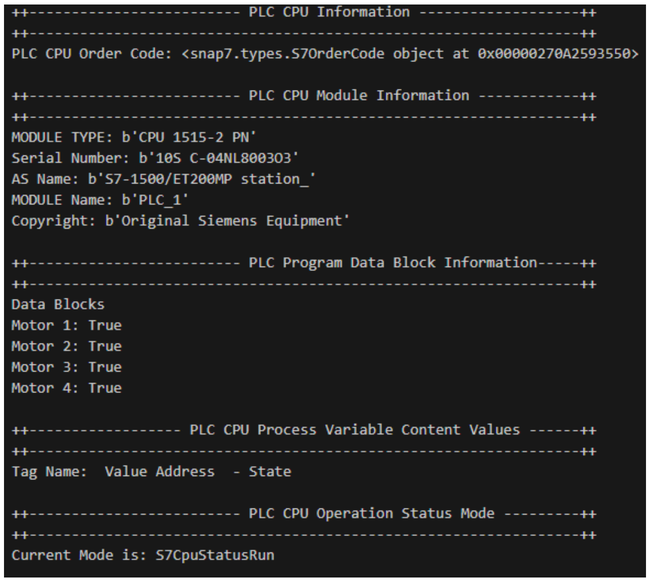
Javascript code to collect the data artifacts for the Control Program, Control Program Meta data, Tag Names and Addresses, and User-Defined Data Types. However, due to the limitation of PLC simulation, the Python code is able to find the information to an actual plc but not with plc simulation.

6.3.3 ***PLC Device Meta-Data.*** To collect the PLC Device Meta-Data, we implemented the Python code and JavaScript code for the Hardware Information, including Module Type Name, Serial Number, AS Name, Module Name, and Copyright. We can acquire some data [Figure 8] from the Python code and the Javascript.

For the data artifacts for Firmware, as what we showed below, neither the Python code nor Javascript we used are supported.

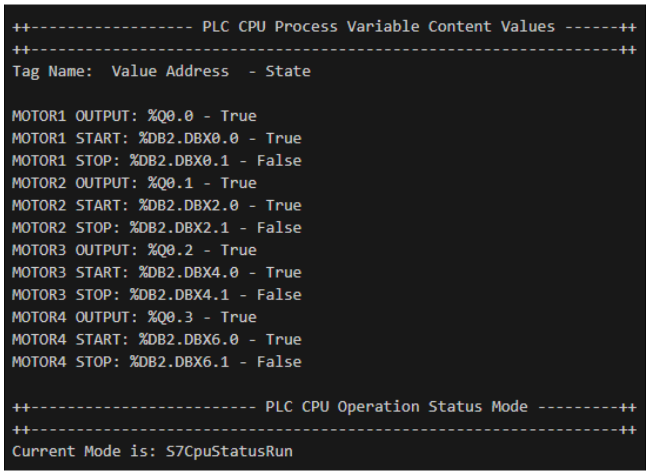
6.3.4 ***Built-In Diagnostic Logs.*** For the data artifacts for Built-In Diagnostic Logs, we collected the data PLC Status Logs by using Python code, the following picture shows that our machine is in RUN mode. However, neither Python code nor JavaScript code have built-in support for finding specific error events or user-defined events due to primarily focusing on low-level communication with PLCs.

6.3.5 ***Process Variable Content Values.*** By implementing the Python code, we successfully captured process variable content values [Figure 9]. Analyzing these data can provide insight into the typical operating patterns of the PLC, allowing forensic investigators to detect any anomalous behavior. However, we cannot capture the data with JavaScript code as much as the Python code.



```
++------------------------ PLC CPU Information -------------------++
++----------------------------------------------------------------++
PLC CPU Order Code: <snap7.types.S7OrderCode object at 0x00000270A2593550>

++------------------------ PLC CPU Module Information ------------++
++----------------------------------------------------------------++
MODULE TYPE: b'CPU 1515-2 PN'
Serial Number: b'10S C-04NL800303'
AS Name: b'S7-1500/ET200MP station_'
MODULE Name: b'PLC_1'
Copyright: b'Original Siemens Equipment'

++------------------------ PLC Program Data Block Information-----++
++----------------------------------------------------------------++
Data Blocks
Motor 1: True
Motor 2: True
Motor 3: True
Motor 4: True

++----------------- PLC CPU Process Variable Content Values ------++
++----------------------------------------------------------------++
Tag Name:  Value Address  - State

++------------------------ PLC CPU Operation Status Mode ---------++
++----------------------------------------------------------------++
Current Mode is: S7CpuStatusRun
```

Figure 8: Meta-Data

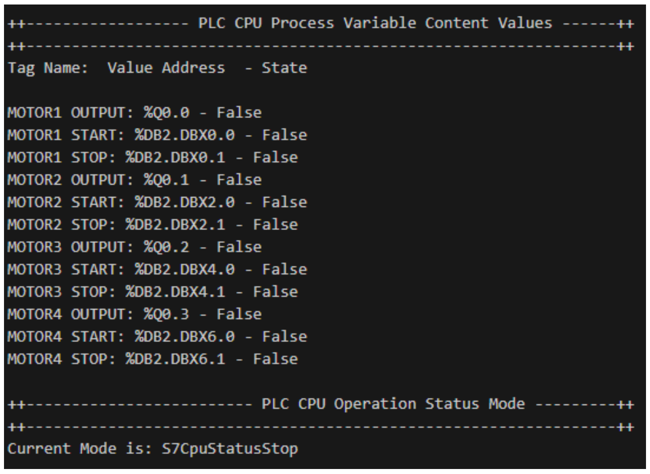


```
++----------------- PLC CPU Process Variable Content Values ------++
++----------------------------------------------------------------++
Tag Name:  Value Address  - State

MOTOR1 OUTPUT: %Q0.0 - True
MOTOR1 START: %DB2.DBX0.0 - True
MOTOR1 STOP: %DB2.DBX0.1 - False
MOTOR2 OUTPUT: %Q0.1 - True
MOTOR2 START: %DB2.DBX2.0 - True
MOTOR2 STOP: %DB2.DBX2.1 - False
MOTOR3 OUTPUT: %Q0.2 - True
MOTOR3 START: %DB2.DBX4.0 - True
MOTOR3 STOP: %DB2.DBX4.1 - False
MOTOR4 OUTPUT: %Q0.3 - True
MOTOR4 START: %DB2.DBX6.0 - True
MOTOR4 STOP: %DB2.DBX6.1 - False

++------------------------ PLC CPU Operation Status Mode ---------++
++----------------------------------------------------------------++
Current Mode is: S7CpuStatusRun
```

Figure 9: Process Variable Content Values

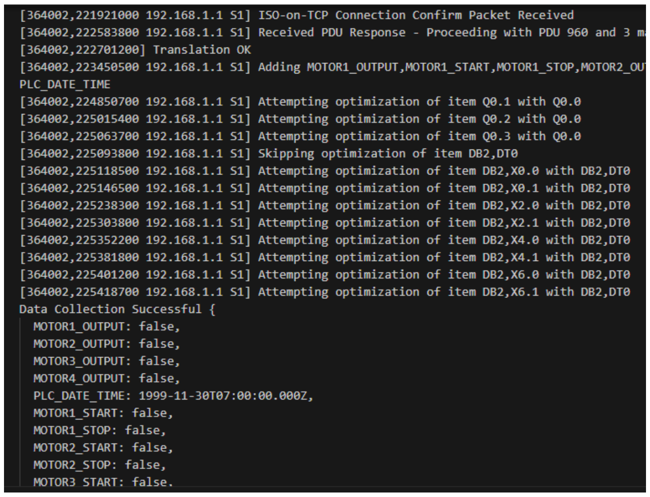
6.4 Attacking Scenario

To have a further understanding of digital forensics analysis, we conduct an attacking scenario to compare the difference in the data artifacts after the PLC was attacked. As the attacker, we changed the operation mode from RUN to STOP, and process values that were true to false in the TIA Portal. As the investigator, we can detect the changes by implementing the Python code [Figure 10] and Javascript code [Figure 10]. However, due to the S7-1500 emulator's lack of an application layer in its network traffic, the S7CommPlus Protocol is unable to access the data from the S7 PLC emulator. However, when using an actual physical S7 PLC, the S7CommPlus protocol could successfully acquire the data, including firmware information, PLC model details, and data information, as shown in [Figure 12].



```
++------------------ PLC CPU Process Variable Content Values ------++
++----------------------------------------------------------------++
Tag Name:  Value Address  - State

MOTOR1 OUTPUT: %Q0.0 - False
MOTOR1 START: %DB2.DBX0.0 - False
MOTOR1 STOP: %DB2.DBX0.1 - False
MOTOR2 OUTPUT: %Q0.1 - False
MOTOR2 START: %DB2.DBX2.0 - False
MOTOR2 STOP: %DB2.DBX2.1 - False
MOTOR3 OUTPUT: %Q0.2 - False
MOTOR3 START: %DB2.DBX4.0 - False
MOTOR3 STOP: %DB2.DBX4.1 - False
MOTOR4 OUTPUT: %Q0.3 - False
MOTOR4 START: %DB2.DBX6.0 - False
MOTOR4 STOP: %DB2.DBX6.1 - False

++------------------------- PLC CPU Operation Status Mode ---------++
++----------------------------------------------------------------++
Current Mode is: S7CpuStatusStop
```

Figure 10: Changed Values detected in Python code

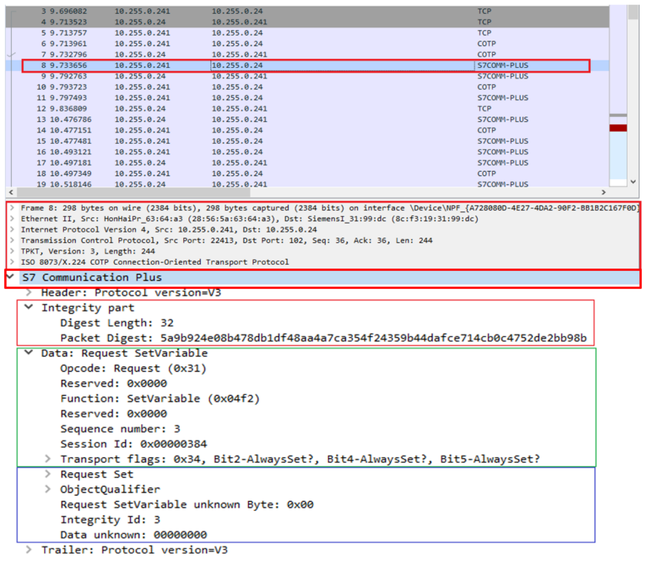


```
[364002,221921000 192.168.1.1 S1] ISO-on-TCP Connection Confirm Packet Received
[364002,222583800 192.168.1.1 S1] Received PDU Response - Proceeding with PDU 960 and 3 m
[364002,222701200] Translation OK
[364002,223450500 192.168.1.1 S1] Adding MOTOR1_OUTPUT,MOTOR1_START,MOTOR1_STOP,MOTOR2_OU
PLC_DATE_TIME
[364002,224850700 192.168.1.1 S1] Attempting optimization of item Q0.1 with Q0.0
[364002,225015400 192.168.1.1 S1] Attempting optimization of item Q0.2 with Q0.0
[364002,225063700 192.168.1.1 S1] Attempting optimization of item Q0.3 with Q0.0
[364002,225093800 192.168.1.1 S1] Skipping optimization of item DB2,DT0
[364002,225118500 192.168.1.1 S1] Attempting optimization of item DB2,X0.0 with DB2,DT0
[364002,225146500 192.168.1.1 S1] Attempting optimization of item DB2,X0.1 with DB2,DT0
[364002,225238300 192.168.1.1 S1] Attempting optimization of item DB2,X2.0 with DB2,DT0
[364002,225303800 192.168.1.1 S1] Attempting optimization of item DB2,X2.1 with DB2,DT0
[364002,225352200 192.168.1.1 S1] Attempting optimization of item DB2,X4.0 with DB2,DT0
[364002,225381800 192.168.1.1 S1] Attempting optimization of item DB2,X4.1 with DB2,DT0
[364002,225401200 192.168.1.1 S1] Attempting optimization of item DB2,X6.0 with DB2,DT0
[364002,225418700 192.168.1.1 S1] Attempting optimization of item DB2,X6.1 with DB2,DT0
Data Collection Successful {
  MOTOR1_OUTPUT: false,
  MOTOR2_OUTPUT: false,
  MOTOR3_OUTPUT: false,
  MOTOR4_OUTPUT: false,
  PLC_DATE_TIME: 1999-11-30T07:00:00.000Z,
  MOTOR1_START: false,
  MOTOR1_STOP: false,
  MOTOR2_START: false,
  MOTOR2_STOP: false,
  MOTOR3_START: false,
```

Figure 11: Changed Values detected in JavaScript code

7 DISCUSSION

Our research delves into the identification and definition of data artifacts retrievable from the Siemens S7-1500 PLC, particularly in the context of digital forensics. This exploration not only contributes to the understanding of data types specific to this PLC model but also enhances the broader knowledge in the field of digital forensic investigations involving PLC systems. By pinpointing and categorizing these data artifacts, we bridge a critical gap in current forensic methodologies, offering a refined lens through which to view and analyze PLC-related data. Our study stands out by providing a more detailed and specific taxonomy of PLC data types, a step beyond what current research offers. This detailed taxonomy is tailored to assist forensic investigators and developers in navigating the complex landscape of PLC systems. By comparing our findings with the existing taxonomy, we uncover subtle differences and similarities, paving the way for a more sophisticated and



```
 3 9.696082   10.255.0.241   10.255.0.24    TCP
 4 9.713523   10.255.0.24    10.255.0.241   TCP
 5 9.713757   10.255.0.241   10.255.0.24    TCP
 6 9.713961   10.255.0.241   10.255.0.24    COTP
 7 9.732796   10.255.0.24    10.255.0.241   COTP
 8 9.733656   10.255.0.241   10.255.0.24    S7COMM-PLUS
 9 9.792763   10.255.0.24    10.255.0.241   S7COMM-PLUS
10 9.793723   10.255.0.241   10.255.0.24    COTP
11 9.797493   10.255.0.241   10.255.0.24    S7COMM-PLUS
12 9.836809   10.255.0.24    10.255.0.241   TCP
13 10.476786  10.255.0.24    10.255.0.241   S7COMM-PLUS
14 10.477151  10.255.0.241   10.255.0.24    COTP
15 10.477481  10.255.0.241   10.255.0.24    S7COMM-PLUS
16 10.493121  10.255.0.241   10.255.0.24    S7COMM-PLUS
17 10.497181  10.255.0.24    10.255.0.241   S7COMM-PLUS
18 10.497349  10.255.0.241   10.255.0.24    COTP
19 10.518146  10.255.0.24    10.255.0.241   S7COMM-PLUS

> Frame 8: 298 bytes on wire (2384 bits), 298 bytes captured (2384 bits) on interface \Device\NPF_{A728080D-4E27-4DA2-90F2-BB1B2C167F0D}
> Ethernet II, Src: HonHaiPr_63:64:a3 (28:56:5a:63:64:a3), Dst: SiemensI_31:99:dc (8c:f3:19:31:99:dc)
> Internet Protocol Version 4, Src: 10.255.0.241, Dst: 10.255.0.24
> Transmission Control Protocol, Src Port: 22413, Dst Port: 102, Seq: 36, Ack: 36, Len: 244
> TPKT, Version: 3, Length: 244
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
v S7 Communication Plus
  > Header: Protocol version=V3
  v Integrity part
      Digest Length: 32
      Packet Digest: 5a9b924e08b478db1df48aa4a7ca354f24359b44dafce714cb0c4752de2bb98b
  v Data: Request SetVariable
      Opcode: Request (0x31)
      Reserved: 0x0000
      Function: SetVariable (0x04f2)
      Reserved: 0x0000
      Sequence number: 3
      Session Id: 0x00000384
    > Transport flags: 0x34, Bit2-AlwaysSet?, Bit4-AlwaysSet?, Bit5-AlwaysSet?
    > Request Set
    > ObjectQualifier
      Request SetVariable unknown Byte: 0x00
      Integrity Id: 3
      Data unknown: 00000000
  > Trailer: Protocol version=V3
```

Figure 12: S7CommPlus Protocol of Physical S7 PLC in Wireshark

targeted approach in forensic investigations. This comparative analysis not only underscores the uniqueness of our methodology but also marks a significant advancement in the precision and efficacy of forensic practices in the realm of PLC systems.

8 LIMITATIONS

One of the limitation of the proposed experiment is that we have conducted the process only on a simulated PLC as we could not collect physical PLC for the research work. In the both Python code and JavaScript code, simulated PLCs have restrictions about the usage of certain functions which are only available for physical PLC. Another known limitation is that we have conducted the case study on a single PLC, although we have discussed about conducting the experiment of different PLCs in the future work section.

9 FUTURE WORK

In terms of future research direction, we will develop code for different PLCs, such as Rockwell Allen-Bradley ControlLogix 1756-L71/B and Siemens S7-300, with a more standardized codebase. In addition, one of our goals is to develop integrated codes and an approach method for all the PLC types for digital forensics investigations so that even individuals without technical expertise can undertake the investigation.

10 CONCLUSION

In conclusion, our paper presents a novel approach to conducting digital forensic investigations on Siemens S7 1500 simulated programmable logic controller (PLC) by analyzing the data artifacts under a simulated attack scenario based on a taxonomy from prior work using digital forensics tools and libraries. By doing so, we hope to contribute to the development of a more comprehensive matrix and taxonomy of PLC digital forensics data artifacts, which can help bolster the fields of PLC digital forensics and cybersecurity, and reinforce the security of critical infrastructure systems in an era of growing digital interdependence. While our approach has some limitations and challenges compared to the prior work due to the restriction of a simulated PLC, we believe that it can be helpful for vendors to have a deeper understanding of the PLC under a cyber-security attack scenario. We hope that our work will inspire further research and innovation in this area, and help improve the security and resilience of critical infrastructure systems worldwide.

REFERENCES

[1] I. Ahmed, S. Obermeier, S. Sudhakaran, and V. Roussev. Programmable logic controller forensics. *IEEE Security & Privacy*, 15(6):18–24, 2017.

[2] W. Alsabbagh and P. Langendoerfer. A new injection threat on s7-1500 plcs - disrupting the physical process offline. *IEEE Open Journal of the Industrial Electronics Society*, 3:1–1, 01 2022.

[3] A. Azfar, K.-K. R. Choo, and L. Liu. An android communication app forensic taxonomy. *Journal of forensic sciences*, 61(5):1337–1350, 2016.

[4] M. Cook, I. Stavrou, S. Dimmock, and C. Johnson. Introducing a forensics data type taxonomy of acquirable artefacts from programmable logic controllers. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–8. IEEE, 2020.

[5] N. M. Karie, V. R. Kebande, and H. Venter. Taxonomy for digital forensic evidence. 2017.

[6] A. K. Mishra, M. Govil, and E. Pilli. A taxonomy of hypervisor forensic tools. In *Advances in Digital Forensics XVI: 16th IFIP WG 11.9 International Conference, New Delhi, India, January 6–8, 2020, Revised Selected Papers 16*, pages 181–199. Springer, 2020.

[7] A. K. Mishra, E. Pilli, and M. Govil. A taxonomy of cloud endpoint forensic tools. In *Advances in Digital Forensics XIV: 14th IFIP WG 11.9 International Conference, New Delhi, India, January 3-5, 2018, Revised Selected Papers 14*, pages 243–261. Springer, 2018.

[8] F. Shahbi, J. Gardiner, S. Adepu, and A. Rashid. A digital forensic taxonomy for programmable logic controller data artefacts. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 320–328. IEEE, 2023.

[9] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis. A survey on the internet of things (iot) forensics: challenges, approaches, and open issues. *IEEE Communications Surveys & Tutorials*, 22(2):1191–1221, 2020.