Forensic Analysis of Industrial Network Data

1 Overview

Analysis of packet captures from a network monitor in an industrial control system (ICS) is an important step towards understanding what has transpired on the control network. Captured network data between a supervisory computer (e.g., a Human-Machine Interface (HMI) system or a Historian system) and a field device such as a programmable logic controller (PLC) can provide a record of commands to field devices, malware payloads, and exfiltration of field data during a breach. In addition to Ethernet-based industrial protocols such as Common Industrial Protocol (CIP) [1] and Ethernet/Industrial Protocol (EtherNet/IP) [2], most PLCs support common TCP/IP application protocols such as HTTP, FTP, SNMP, etc. for system configuration and management purposes. These protocols can be exploited by a rogue machine on the network.

The learning objective of this lab is to introduce students to common vulnerabilities in an industrial network and a commercial EtherNet/IP implementation, and to demonstrate the importance of industrial network data analysis in forensics investigations.

2 Lab Environment

The lab environment is provided by the Labtainer framework, which can be installed and run as described in the Labtainer Student Guide. Additionally, you will need to use a version of Wireshark that can extract object attributes in a CIP message (e.g., Version 2.2.0). Wireshark should be installed on the Linux host on which Labtainer was installed.The lab is started by typing

```
$ ./start.py plc-forensics
```

at your Labtainer working directory.

2.1 Lab Configuration

The resulting virtual terminal labeled “investigator” is connected to a Labtainer computer that you will use to access the vulnerable PLC in a manner similar to that followed by the attacker. But first, you will need to analyze a set of PCAP files to determine what the attacker did.

This lab uses the Wireshark network protocol analyzer tool to examine packet traces in the lab exercises described in Section 3.

The PCAP files are automatically copied into the “plc-forensics” directory relative to your Labtainer working directory (where you typed “start.py”). Please note that these PCAP files are

specific to your instance of the lab, and thus should not be replaced by PCAP files from any other source.

2.2 ICS Test Environment

The simulated ICS environment used to generate the PCAPs for this lab consists of a Windows HMI system, an industrial network switch, a Linux system used to launch attacks on the PLC, and a modular PLC rack. This ICS environment is illustrated in Figure 1. The network monitor, industrial switch, remote terminal unit (RTU), and physical devices shown in the diagram are informational only.

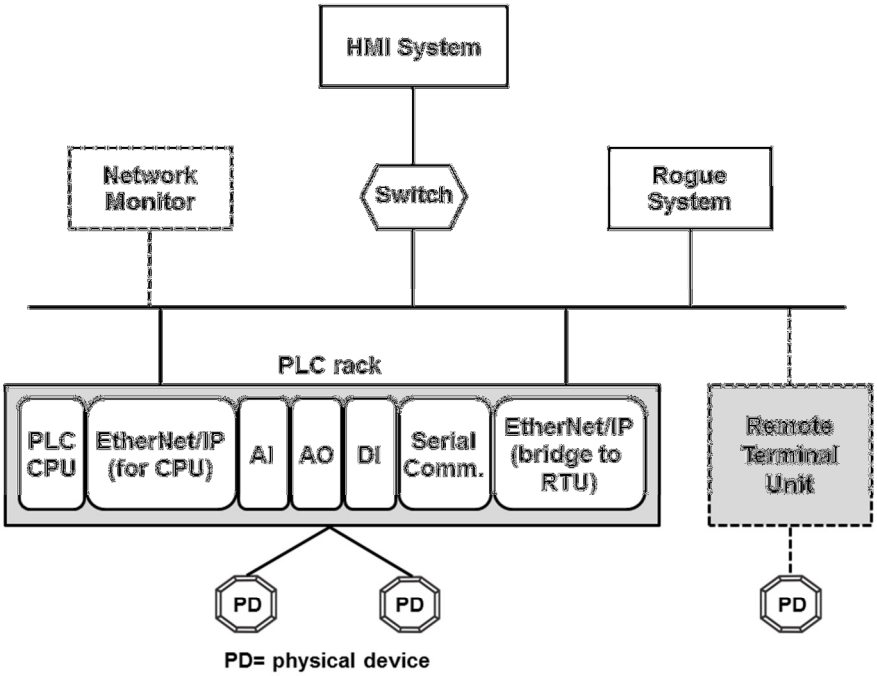

Figure 1. ICS environment used to generate PCAP files

The IP addresses that are relevant to the lab exercises are shown in Table 1.

Table 1. IP address allocation.

Component	IP address
HMI system	10.1.30.1
Rogue system	10.1.40.1
PLC rack (EtherNet/IP for CPU)	10.1.100.2

The PLC rack is a Rockwell Automation/Allen-Bradley (RA/AB) 1756 ControlLogix system [3] that consists of a controller (CPU) module and multiple I/O modules—an EtherNet/IP communication module used to communicate with the HMI system, an analog input (AI) module, an analog output (AO) module, a digital input (DI) module, a serial

communication module, and a second EtherNet/IP module used to communicate with the RTU. These I/O modules communicate with the controller module via a proprietary backplane. For this lab, the controller runs a ladder logic application that controls a number of physical devices[1].

The HMI system runs the RA Studio 5000 Logix Designer software [4], which is used to develop and run ladder logic applications on the controller. The HMI system regularly asks the controller for I/O statuses and the controller's current operating mode. The HMI system and the controller communicate via EtherNet/IP and CIP.

2.3 Overview of CIP

The Common Industrial Protocol (CIP) models each node in the network as a set of objects (Figure 2). The following definitions are taken from the CIP specification [1]. An *object* provides "an abstraction of a component within a product." A *class* is a set of objects that "are identical in form and behavior, but may contain different attribute values." An *object instance* is "the actual representation of a particular object within a class." An *instance of a class* shares "the same set of attributes, but has its own particular set of attribute values." An *attribute* describes "an externally visible characteristic or feature" of an object.

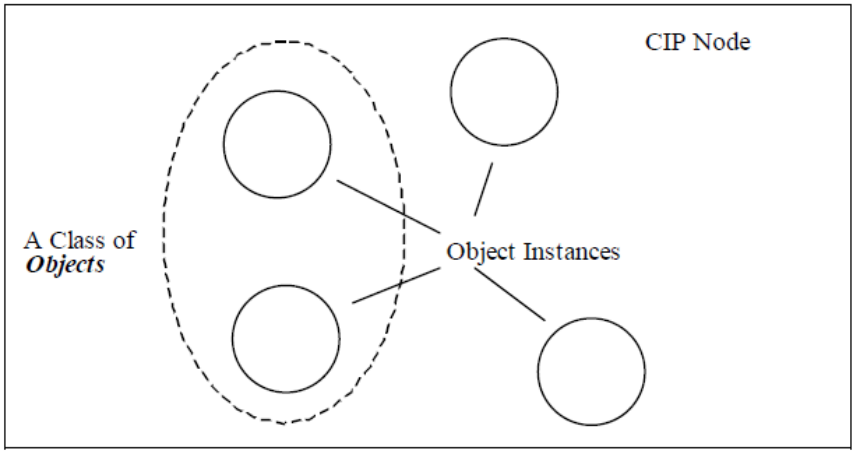

Figure 2. A CIP node with multiple object instances. From [1].

Each object or class supports a set of *common services* (defined in Appendix A of [1]), and in certain products, a number of vendor-specific services. This lab requires a basic understanding of the Identity object and two common services— Get Attribute List (GAL) and Multiple Service Packet (MSP). The GAL service returns the value(s) of the requested object attribute(s) or class attribute(s). The MSP service allows a CIP client (e.g., the HMI system) to request a CIP server (e.g., the controller) a number of services in a single CIP message. An "embedded" service request can be a GAL service. The Identity object "provides identification of and general information about the device," and the Status attribute (attribute ID=5) of the Identity Object provide the "current status of the entire device" [1]. See Section A.2 for more information on the Status attribute.

[1] You don't need to know the application or physical devices to do the lab.

2.4 Overview of EtherNet/IP

The Ethernet/Industrial Protocol (EtherNet/IP), also known as "CIP over Ethernet", uses standard Ethernet and TCP/IP protocols to transport CIP messages [2]. EtherNet/IP supports two types of communications: UDP-based implicit messaging for time-critical operations and TCP-based explicit messaging for operations that are not time-sensitive. An implicit message can be multicast or unicast, and requires a CIP connection to be established between the two devices; an explicit message does not require a CIP connection. EtherNet/IP uses the same port number (44818 or 0xAF12) for both UDP and TCP connections.

The EtherNet/IP encapsulation message inside a TCP/UDP packet consists of a 24-byte header and a command-specific data portion (Figure 3) [2]. There are both pre-defined commands and vendor-specific commands. The same packet format is used for both requests and replies.

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	Encapsulation command
	Length	UINT	Length, in bytes, of the data portion of the message, i.e., the number of bytes following the header
	Session handle	UDINT	Session identification (application dependent)
	Status	UDINT	Status code
	Sender Context	ARRAY of octet	Information pertinent only to the sender of an encapsulation command. Length of 8.
	Options	UDINT	Options flags
Command specific data	Encapsulated data	ARRAY of 0 to 65511 octet	The encapsulation data portion of the message is required only for certain commands

Figure 3. EtherNet/IP packet format. From [2].

This lab requires a basic understanding of the pre-defined RegisterSession command, which must be executed to establish an EtherNet/IP session prior to any CIP communications between two devices. The session handle returned in a RegisterSession reply is used in all subsequent EtherNet/IP messages until the session is terminated. The session handle is a 32-bit unique value generated by the target device. See Section 2-4.4 of the EtherNet/IP specification [2] for more information about the RegisterSession command.

2.5 Note for Students

This lab assumes the following:

1. The student has taken TCP/IP networking course(s);
2. The student has working knowledge of HTTP and FTP application protocols, and HTML syntax;
3. The student is familiar with basic ICS terminology and concepts;
4. The student has hands-on experience with Wireshark. Minimally, the student must know how to: filter a particular packet type, obtain protocol-specific statistics, follow a TCP stream, customize display columns, and set up TCP preferences such as turning off TCP

streams reassembly feature.
5. The student can independently look up vendor information available on the Internet.

2.6 Note for Instructors

We suggest that this lab be conducted in a supervised lab environment, and that the following materials be covered at the beginning of the lab session:

1. Labtainer installation, including installation of Virtual Box and a Linux VM (if a Linux system is not already available). Refer to the Labtainer Student User Guide.
2. Review of HTTP and FTP protocol, and HTML format. Only need to cover the basic structure of HTTP authentication and GET method, basic FTP commands, and basic structure of a webpage and HTML elements.
3. Review of ICS fundamentals.
4. Explanation of CIP and EtherNet/IP protocols. Only need to cover the material discussed in Section 2.3 and 2.4 above.

3 Lab Tasks

This lab consists of five tasks. Tasks 1 and 2 cover reconnaissance activities. Task 3 addresses data exfiltration. Task 4 demonstrates an insider threat scenario. Task 5 examines the evidence of an EtherNet/IP attack.

The following list provides some useful hints for Wireshark:

1. When working with a particular protocol, set the display filter to only show traffic for that protocol, e.g., set the display filter to only show HTTP traffic when working with web requests.
2. When working with HTTP, enable the HTTP header and body reassembly options. These options are usually enabled by default.
3. Disable the TCP streams reassembly option if packet data span multiple TCP segments. This option is usually enabled by default. See https://wiki.wireshark.org/TCP_Reassembly for more information about the TCP reassembly option.
4. Use the Follow TCP Stream feature to see application-specific data that span multiple packets, e.g., a web page or an FTP operation.
5. For tasks 4 and 5, set WireShark to display the EtherNet/IP session handle, CIP attribute, and returned value of the Status attribute of the Identity object.
6. The packet capture file for task 5 is large so be patient when you ask Wireshark to process it, e.g., filtering for a specific EtherNet/IP response.

3.1 Task 1: Reconnaissance Activity – Part 1

Reconnaissance is the first phase of an attack progression. This activity is often difficult to detect if it was done using the same tools and processes prescribed for regular system maintenance, e.g., reviewing component configuration and status via a web browsing interface. The ControlLogix EtherNet/IP modules include an integrated web server that allows remote systems to monitor and manipulate controller data. When contacted, the web server returns a home page that includes a

list of available operations (Figure 4), some of which require login with appropriate access permissions.

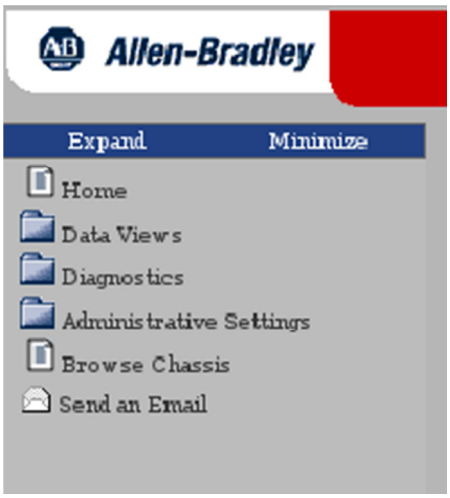

Figure 4. Operations supported by the web server.

The objective of this task is to determine how much information about the PLC rack a rogue machine on the network can discover from browsing the web server. Your task is to use Wireshark to inspect the `Task1-trace.pcap` file and work through work through the questions below.

Question 1.1: Which browser (user-agent) was used to collect controller data?

Question 1.2: What is the name of the web server?

Question 1.3: How many HTTP requests are contained in the PCAP file?

Question 1.4: Which top-level operation shown in Figure 4 was performed?

Question 1.5: What subsequent actions were performed to get information about the PLC rack after the top-level operation was selected? Describe your method for producing your answer to this question and provide Wireshark output you use for your answer.

Question 1.6: What information about the PLC rack and each module in the rack was obtained from the first `chassisWho.asp` query? Provide all data that are relevant to the chassis and each module.

Question 1.7: Show how the information obtained from the first `chassisWho.asp` query corresponds to the PLC rack configuration shown in Figure 1. You can use the following table for your answer:

Question 1.8: For each module, how many subsequent queries were made to obtain additional information about the module? Describe the method you use to find these queries and identify the packet used to send each query.

Question 1.9: What kinds of information were returned from the queries identified in Question 1.7? Describe your method for producing your answer and list the returned information for the following modules: controller, EtherNet/IP module for the controller, and serial communication module.

Question 1.10: Examine the returned data for other modules. What similarities and differences between the data returned for those modules and the data for the three modules covered in Question 1.8 do you observe? You don't need to list specific values.

3.2 Task 2: Reconnaissance Activity – Part 2

RA documentation (available on the Internet) indicates that the ControlLogix EtherNetIP modules support several TCP/IP application services. The objective of this task is to determine what information about these services a rogue machine can retrieve from the web server. Your task is to use Wireshark to inspect the `Task2-trace.pcap` file and work through the questions below.

Question 2.1: Starting at which packet did the rogue machine discover that certain web pages are protected? Describe your method for producing your answer.

Question 2.2: Describe the actions performed by the rogue machine after this discovery? Describe your method for producing your answer and show the result of each action.

Question 2.3: What information did the rogue machine discover? Describe your method for producing your answer and show the returned information.

Action 2.1 Using information derived from the PCAP, use the "investigator" terminal and "wget" to retrieve the protected web pages.

3.3 Task 3: Data Exfiltration

After finding out which application services are running on a controller, the next step for the attacker is to exploit those services to obtain high-value control data. The objective of this task is to determine what information a rogue machine can retrieve from an integrated application server in a ControlLogix system. Your task is to use Wireshark to inspect the `Task3-trace.pcap` file and work through the questions below.

Question 3.1: Excluding the two CIP services, how many application services discovered in Task 2 were exploited and what were they? Show evidence from the PCAP file that supports your answer.

Question 3.2: Provide the name, version number, and information about the operating system of an exploited server. Show evidence from the PCAP file that supports your answer.

Question 3.3: Did the exploited service(s) require user authentication? If yes, was the user credential protected during transit? Show evidence from the PCAP file that supports your answer.

Question 3.4: What information did the rogue machine find out about the directory structure of the exploited server(s)? Show evidence from the PCAP file that supports your answer.

Question 3.5: Provide a high level description of the actions performed by the rogue machine to obtain data from the exploited server(s). Show evidence from the PCAP file that supports your answer.

Question 3.6: Describe the information contained in the retrieved data? Show evidence from the PCAP file that supports your answer. See Chapter 7 of the CIP specification [1] for more information on the retrieved data. Note that EtherNet/IP vendor ID for RA/AB is 1, and thus vendor-specific information will have a "1_" prefix.

Question 3.7: Discuss a hypothetical attack scenario based on your understanding of the retrieved data.

Action 3.1 Using information derived from the PCAP, use the "investigator" terminal to access the service as it was accessed by the attacker and retrieve the information that the attacker retrieved.

3.4 Task 4: Controller Manipulation

An insider had gained physical access to the PLC rack and was able to manipulate the controller via the mode switch on the front of the controller. The objective of this task is to analyze EtherNet/IP and CIP messages to determine the actions taken by the insider. Your task is to use Wireshark to inspect two PCAP files, `Task4-trace1.pcap` (normal traffic) and `Task4-trace2.pcap` (traffic with abnormal activities), and work through the questions below. You need to read Appendix A before starting this task.

Question 4.1: How many EtherNet/IP packets were captured in the `Task4-trace1.pcap` file? Show evidence from the PCAP file that supports your answer.

Question 4.2: How many EtherNet/IP sessions were captured in the `Task4-trace1.pcap` file, and what are the values of their session handles? Show evidence from the PCAP file that supports your answer.

Question 4.3: List the CIP services found in the `Task4-trace1.pcap` file. Describe the CIP service request/response pattern that you observe. Indicate the object class IDs associated with the services. Show evidence from the PCAP file that supports your answer.

Question 4.4: What is the returned Status value for attribute 5 of the Identity object in the `Task4-trace1.pcap` file, as reported by Wireshark? Using the information provided in Appendix A, describe what this value represents. Show evidence from the PCAP file that supports your answer.

Question 4.5: Summarize the events associated with the Identity object that were captured in the `Task4-trace2.pcap` file, and provide an analysis of each event according to the information provided in Appendix A. Show evidence from the PCAP file that supports your answer.

Question 4.6: What unusual CIP behavior (compared with the normal traffic) did you observe in the `Task4-trace2.pcap` file? Show evidence from the PCAP file that supports your answer.

3.5 Task 5: EtherNet/IP Attack

The HMI system reported that EtherNet/IP communications with the PLC had been lost, but there was no error indication on the controller module or the EtherNet/IP communications module, i.e., the status LEDs on the front panel of those modules still indicated that the modules were running normally. The HMI system could still `ping` the EtherNet/IP communications module, however subsequent requests to establish EtherNet/IP sessions continued to fail. This problem persisted until the PLC system was reset.

The objective of this task is to analyze EtherNet/IP traffic to determine the cause of the problem. Your task is to use Wireshark to inspect the `Task5-trace.pcap` file, and work through the questions below. You should review Section 2-4.4 of the EtherNet/IP specification [2] before starting this task.

Question 5.1: How many EtherNet/IP sessions were requested? Show evidence from the PCAP file that supports your answer.

Question 5.2: How many EtherNet/IP sessions were successfully established (i.e., the EtherNet/IP module returned a non-zero session handle)? Show evidence from the PCAP file that supports your answer.

Question 5.3: How many established EtherNet/IP sessions were requested by the rogue system? Show evidence from the PCAP file that supports your answer.

Question 5.4: What is the difference between the answer for Question 5.2 and the answer for Question 5.3? Discuss why you think it is so. (Hint: Review your answer for question 4.2).

Question 5.5: How many requests for EtherNet/IP sessions did not have a response? Explain your calculation.

Question 5.6: Build a time line of events and describe what you think happened on the network that caused the observed malfunction. You should note similar activities but you don't need describe them in details. Show evidence from the PCAP file that supports your answer.

Question 5.6: What type of attack was used? Describe the attack in terms of mechanism(s) and resource(s) that were exploited by the rogue system.

Question 5.7 (hard): Can the attacker gain any information about how session handles are allocated? You don't need to come up with a mathematical formula for the allocation algorithm.

4 Submission

You need to submit a detailed lab report to describe what you have done and what you have observed. Please provide details using screen shots. You also need to provide explanation to the observations that are interesting or surprising.

References

[1] ODVA & ControlNet International Ltd, "The CIP Networks Library Volume 1, Common Industrial Protocol (CIP)," Edition 3.3, November, 2007. http://www.tud.ttu.ee/im/Kristjan.Sillmann/ISP0051%20Rakenduslik%20Andmeside/CIP%20docs/CIP%20Vol1_3.3.pdf

[2] ODVA & ControlNet International Ltd, "The CIP Networks Library Volume 2, EtherNet/IP Adaptation of CIP," Edition 1.4, November 2007. http://www.tud.ttu.ee/im/Kristjan.Sillmann/ISP0051%20Rakenduslik%20Andmeside/CIP%20docs/CIP%20Vol2_1.4.pdf

[3] Allen-Bradley, "ControlLogix System," User Manual, Rockwell Automation Publication 1756-UM001O-EN-P, October 2014. http://literature.rockwellautomation.com/idc/groups/literature/documents/um/1756-um001_-en-p.pdf

[4] Allen-Bradley, "Logix5000 Controllers Ladder Diagram," Programming Manual, Rockwell Automation Publication 1756- PM008F-EN-P, June 2016. http://literature.rockwellautomation.com/idc/groups/literature/documents/pm/1756-pm008_-en-p.pdf

[5] Allen-Bradley, "Troubleshoot EtherNet/IP Networks," Application Technique, Rockwell Automation Publication ENET-AT003B-EN-P, June 2014. http://literature.rockwellautomation.com/idc/groups/literature/documents/at/enet-at003_-en-p.pdf

[6] Allen-Bradley, "ControlLogix System," User Manual, Rockwell Automation Publication 1756-UM001O-EN-P, October 2014. http://literature.rockwellautomation.com/idc/groups/literature/documents/um/1756-um001_-en-p.pdf

[7] Allen-Bradley, "Logix5000 Controllers General Instructions Reference Manual," Reference Manual, Rockwell Automation Publication 1756-RM003Q-EN-P, July 2016. http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm003_-en-p.pdf

Appendix A Supplemental Material

A.1 Key Switch and Device Mode

The mode switch on the front of a ControlLogix 1756-L7x controller is shown in Figure 5.

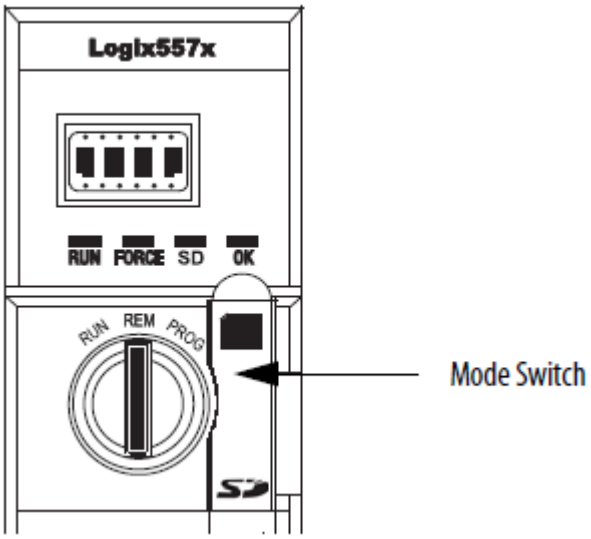

Figure 5. Controllogix 1756-L7x mode switch. From [6].

Depending on the position of the mode switch, a ControlLogix 1756-L7x can be in one of the following modes [6]:

1. RUN position: the controller mode is Run. In this mode, the controller continuously monitors and controls the I/O modules. While in this mode, the loaded applications cannot be modified.

2. REM position: the controller can be in one of three modes:

    a. Remote Run: This mode is identical to the Run mode except applications can be modified remotely.

    b. Remote Program: This mode is identical to the Program mode except its operations can be invoked remotely.

    c. Remote Test: The controller continues executing the loaded application code, stops controlling the I/O modules, and provides limited editing capability.

3. PROG position: the controller mode is Program. In this mode, the controller does not execute application code, stops controlling the I/O modules. Applications can be modified or loaded from the controller's user memory.

A.2 Status Attribute of Identity Object

The Identity Object is described in Chapter 5, Section 5-2 of the CIP specification [1]. For this lab, the most relevant attribute of this object is the Status attribute (attribute ID=5). The values of this attribute "changes as the state of the device changes" [1]. The bit definition of the Status attribute is shown in Figure 6.

Bit (s)	Called	Definition
0	Owned	TRUE indicates the device (or an object within the device) has an owner. Within the Master/Slave paradigm the setting of this bit means that the Predefined Master/Slave Connection Set has been allocated to a master. Outside the Master/Slave paradigm the meaning of this bit is TBD.
1		Reserved, shall be 0
2	Configured	TRUE indicates the application of the device has been configured to do something different than the "out–of–box" default. This shall not include configuration of the communications.
3		Reserved, shall be 0
4 – 7	Extended Device Status	Vendor–specific or as defined by table below. The EDS shall indicate if the device follows the public definition for these bits using the DeviceStatusAssembly keyword in the [Device] section of the EDS. If these bits are vendor specific then they shall be enumerated in the EDS using the Assembly and Parameter sections.
8	Minor Recoverable Fault	TRUE indicates the device detected a problem with itself, which is thought to be recoverable. The problem does not cause the device to go into one of the faulted states. See Behavior section.
9	Minor Unrecoverable Fault	TRUE indicates the device detected a problem with itself, which is thought to be unrecoverable. The problem does not cause the device to go into one of the faulted states. See Behavior section.
10	Major Recoverable Fault	TRUE indicates the device detected a problem with itself, which caused the device to go into the "Major Recoverable Fault" state. See Behavior section.
11	Major Unrecoverable Fault	TRUE indicates the device detected a problem with itself, which caused the device to go into the "Major Unrecoverable Fault" state. See Behavior section.
12 - 15		Reserved, shall be 0

Figure 6. Bit definition of Status attribute of Identity object. From [1].

Regarding the Extended Device Status bits, it is unclear if the controller used in the test environment follows the public definition shown in Figure 7, or uses a vendor-specific definition since the controller's EDS file obtained from the vendor's web site contains neither the Device Status Assembly keyword nor the Assembly and Parameter sections.

Bits 4 - 7:	Extended Device Status Description
0 0 0 0	Self-Testing or Unknown
0 0 0 1	Firmware Update in Progress
0 0 1 0	At least one faulted I/O connection
0 0 1 1	No I/O connections established
0 1 0 0	Non-Volatile Configuration bad
0 1 0 1	Major Fault – either bit 10 or bit 11 is true (1)
0 1 1 0	At least one I/O connection in run mode
0 1 1 1	At least one I/O connection established, all in idle mode
1 0 0 0 1 0 0 1	Reserved, shall be 0
1 0 1 0 thru 1 1 1 1	Vendor/Product specific

Figure 7. Bit definition of Extended Device Status of Status attribute. From [1].

The ControlLogix controller used in the test environment supports the Get System Value (GSV) instruction, which returns system data stored in ControlLogix objects [7]. One of the ControlLogix objects supported by the GSV instruction is the ControllerDevice object, which "identifies the physical hardware of the controller" [7]. The information returned includes the device name, product code, product revision number, device serial number, device type, vendor ID, and a 16-bit Status value as shown in Figure 8.

For this lab, you can assume the following:

- The Extended Device Status bits of the Status attribute of the Identity object (bits 7-4) corresponds to the Device Status Bits (bits 7-4) of the GSV Status value.
- The Reserved bits of the Status attribute of the Identity object (bits 15-12) corresponds to the Controller Status Bits (bits 15-12) of the GSV Status value, which corresponds to the Extended Device Status 2 value of the Status attribute of the Identity object reported by Wireshark (see Figure 9).

Status	INT	GSV	Device Status Bits 7...4 Meaning 0000 Reserved 0001 Flash update in progress 0010 Reserved 0011 Reserved 0100 Flash is bad 0101 Faulted modes 0110 Run 0111 Program Fault Status Bits 11...8 Meaning 0001 Recoverable minor fault 0010 Unrecoverable minor fault 0100 Recoverable major fault 1000 Unrecoverable major fault Controller Status Bits 13...12 Meaning 01 Keyswitch in run 10 Keyswitch in program 11 Keyswitch in remote 15...14 Meaning 01 Controller is changing modes 10 Debug mode if controller in run mode
Type	INT	GSV	Identifies the device as a controller. Controller = 14.
Vendor	INT	GSV	Identifies the vendor of the device. Allen-Bradley = 0001.

Figure 8. Bit definition of Status value of GSV instruction. From [7].

```
▲ Common Industrial Protocol
  ▲ Service: Get Attribute List (Response)
      1... .... = Request/Response: Response (0x1)
      .000 0011 = Service: Get Attribute List (0x03)
  ▷ Status: Success:
    [Request Path Size: 2 (words)]
  ▷ [Request Path: Identity, Instance: 0x01]
  ▲ Get Attribute List (Response)
      Attribute Count: 1
    ▲ Attribute List
      ▲ Attribute: 5 (Status)
          General Status: Success (0x0000)
        ▲ Status: 0x1060
            .... .... .... ...0 = Owned: 0
            .... .... .... .0.. = Configured: 0
            .... .... 0110 .... = Extended Device Status: 0x6
            .... ...0 .... .... = Minor Recoverable Fault: 0
            .... ..0. .... .... = Minor Unrecoverable Fault: 0
            .... .0.. .... .... = Major Recoverable Fault: 0
            .... 0... .... .... = Major Unrecoverable Fault: 0
            0001 .... .... .... = Extended Device Status 2: 0x1
```

Figure 9. Returned value of Status attribute of Identify object.