Hindawi
Security and Communication Networks
Volume 2018, Article ID 2639750, 16 pages
https://doi.org/10.1155/2018/2639750

Research Article

Development of the ECAT Preprocessor with the Trust Communication Approach

Kevser Ovaz Akpinar and Ibrahim Ozcelik

Computer Engineering Department, Sakarya University, 54187 Sakarya, Turkey

Correspondence should be addressed to Kevser Ovaz Akpinar; kovaz@sakarya.edu.tr

Received 26 October 2017; Revised 16 February 2018; Accepted 12 March 2018; Published 18 April 2018

Academic Editor: Leandros Maglaras

In the past several years, attacks over industrial control systems (ICS) have become increasingly frequent and sophisticated. The most common objectives of these types of attacks are controlling/monitoring the physical process, manipulating programmable controllers, or affecting the integrity of software and networking equipment. As one of the widely applied protocols in the ICS world, EtherCAT is an Ethernet-based protocol; thus, it is exposed to both TCP/IP and ICS-specific attacks. In this paper, we analyze EtherCAT field-level communication principles from the security viewpoint focusing on the protocol vulnerabilities, which have been rarely analyzed previously. Our research showed that it lacks the most common security parameters, such as authentication, encryption, and authorization, and is open to Media Access Control (MAC) spoofing, data injection, and other advanced attacks, which require superior skills. To prevent, detect, and reduce attacks over the EtherCAT-based critical systems, first, we improved the open-source Snort intrusion detection/prevention system (IDS/IPS) to support packets that are not processed over transport and network layers. Second, by incorporating a vulnerability analysis, we proposed the EtherCAT (ECAT) preprocessor. Third, we introduced a novel approach called trust-node identification and applied the approach as three rules into the preprocessor. In this sense, the ECAT preprocessor differs from other supported ICS preprocessors in the literature, such as DNP3 and Modbus/TCP. Besides supporting traditional rule expansion, it is also able to handle layer 2 packets and to apply deep packet inspection on EtherCAT packets using the trust-node approach. This method first identifies engineering-station approved nodes based on EtherCAT network information (ENI) configuration files and then deeply inspects incoming packets, considering protocol specifications. The improvements and approach have been tested on the physically developed testbed environment and we have proved that proposals can detect related attacks and provide a basic level of security over the EtherCAT-implemented systems.

1. Introduction

Industrial automation systems are generally divided into three categories according to the application fields, which are factory, process, and building automation. These automation systems are designed to provide the integration between information technology (IT) communication, such as the manufacturing execution system (MES) level or enterprise resource planning (ERP) level, and field communication, such as cell, field, or sensor/actuator levels [1]. Controlling, ensuring sustainability of, and monitoring the critical system automations, which are created considering the hierarchical structure, are achieved via supervisory control and data-acquisition (SCADA) systems. Additionally, the planning and execution levels in IT communication for automation systems provide ERP and MES features and services through Ethernet and TCP/IP protocols, which are known de facto.

Field communication was previously only carried out by Profibus, Interbus, Devicenet, Controlnet, and other fieldbus protocols in the past. Due to the idea of using a single protocol for both horizontal and vertical communication, nowadays, the automation hierarchy is managed by the Ethernet and/or Ethernet-based protocols. Modbus/TCP, EtherNet/IP, PROFINET, EtherCAT, Ethernet Powerlink, Sercos III, and other protocols are examples of Ethernet-based protocols. They are directed by different manufacturers or technology groups. While this new trend offers benefits in terms of cost reduction, increased speed, and communication complexity reduction for automation systems, it also provides advantages and disadvantages regarding integration of the Ethernet-based protocols used in the field and in IT.

Eventually, the direction of the stated technological improvements led the industry to form an automation system that communicates using Ethernet or Ethernet-based protocols, contains configured IT services (web, ftp, mail, etc.), integrates with TCP/IP infrastructure, and is monitored/controlled through SCADA software [2]. Thus, even a terminal device in the field can be controlled directly or indirectly via Ethernet.

Ethernet and TCP/IP protocols are well known, and the diversity and success of the attacks are exhaustively studied in the literature [3, 4]. This duo introduces security risks and cyberthreats into industrial control systems (ICS) as well. According to a survey in 2015, over 65% of SCADA system attacks are achieved against the communication infrastructure [5]. This observation clearly shows the importance of cyberthreats for the communication network. In this context, over the last decade, various global attacks have been carried out through the communication infrastructure, such as leaking or attacking Iran's nuclear plants, the US subway collision avoidance system, Middle Eastern and North African oil plants, or the Ukrainian power grid [6–12].

The most common types of attacks are man in the middle (MITM), denial of service (DoS), distributed DoS (DDoS), cryptographic, replay, and buffer overflow attacks. The main reason behind the attack excess and diversity is that many of the protocols used in ICS do not possess encryption, authentication, and authorization services, which are considered as key parameters in the network security. Therefore, to reduce cyberrisks in the system, the proper implementation of protocols and standards and the identification and minimization of implementation-based vulnerabilities are necessary.

In addition, security teams or administrators are not aware of their control system assets and do not obtain patch feeds or track the vulnerability disclosures and harden their systems. According to a vulnerability trend report from 2016, vulnerability disclosures rapidly increased from 2014 to 2015. However, 516 of them did not have a patch at the time of disclosure [13]. This means that, out of the over 1,552 vulnerabilities analyzed, 33% are zero-day vulnerabilities and could not be fixed before exploitation. In this regard, more research and analyses are needed for zero-day or other potential vulnerabilities before they are publicly disclosed. Cheminod et al. presented a review for the current situation and management security of industrial automation control system [14].

The EtherCAT protocol studied in this paper is extensively applied in industrial automation and accepted as a hard real-time (RT) protocol where a statistical distribution of response times cannot be tolerated. The short cycle times, speed, topology flexibility, scalability, product diversity, and cost advantages, which are essential arguments in critical systems, have enabled EtherCAT to become a major protocol running on industrial automation systems compared to Modbus/TCP, EtherNet/IP, PROFINET RT, or Sercos III [15]. In addition, while PROFINET supports real-time and non-real-time communication through three structures, namely, PROFINET NRT, IRT, and RT, EtherCAT can handle this only through one protocol. Moreover, EtherCAT speed outperforms PROFINET IRT in terms of short cycle times [16]. It has a 0.1 ms response time and less than 0.1 ms jitter for a 100 Mbit/s data rate [17].

If IP routing is needed to communicate with different SCADA systems or different SCADA infrastructures within the same system, besides using the standard Ethernet frame structure defined in IEEE 802.3, it can be transferred over UDP/IP networks by adding the IP address to the frame [18]. This feature enables communication across routers in different subnets. However, it can only be used for IP routing purposes, meaning that frames cannot be used in all UDP/IP supported network devices. Although EtherCAT has advantages based on the fields of application, as in many other critical infrastructure protocols, it has a security problem of not including encryption, authentication, or authorization components in SCADA communications.

Therefore, in this study, to increase communication security and detect and prevent attacks against EtherCAT-implemented systems, the protocol structure and vulnerabilities are analyzed, and a novel solution is proposed to prevent exploitation of these weaknesses. The solution consists of the development of a new EtherCAT (ECAT) preprocessor on an open-source Snort IDS/IPS system and an application of a trust-node communication structure in the ECAT preprocessor. The proposal relies on passive monitoring; thus, real-time communication is not interrupted and does not cause extra load in the EtherCAT operation under real working conditions in an industrial environment. The novelty of our study is that it is the first research in the literature specifically focusing on EtherCAT communication vulnerabilities and introducing the trust-node approach to be used in Snort as a solution to improve EtherCAT security. In this context, the vulnerability analysis is performed by attack vectors on device-level communication, as it is responsible for carrying time-sensitive information. The analysis results proved that the EtherCAT protocol does not have any flow or connection-based security to recognize master and slaves. Thus, it is vulnerable to Media Access Control (MAC) spoofing, MITM, DoS, and TCP/IP attacks or other sophisticated attacks, such as node accessing and command injection. As part of our research, Snort is improved to support layer 2 packets for preprocessors. Later, the ECAT preprocessor is introduced with the trust-node communication approach. The approach mainly provides a basic level of prevention by recognizing each node over the EtherCAT-applied SCADA systems. Considering that most of the critical infrastructure protocols have vendor-provided eXtensible Markup Language (XML) or General Station Description- (GSD-) based ID files for almost each component, our novel proposal is a general solution and applicable to a broad range of protocols.

The paper is organized as follows. In Section 2, we present the literature review based on the EtherCAT or other protocols emphasizing the vulnerabilities and related solutions. Then, in Section 3, we look closer into the EtherCAT communication principles, device protocol, and configuration files distributed by the manufacturers or generated by the configuration tool. In Section 4, we present the testbed environment and vulnerability analysis performed by the generated attack vectors. In Section 5, we propose the ECAT preprocessor and the trust-node identification approach. In Sections 6 and 7, we give some final remarks and the future directions of the research.

2. Related Work

2.1. Security Studies of Ethernet-Based Protocols (Except ECAT). Many of the Ethernet-based industrial automation protocols communicate in plaintext (without encryption) without authorization and authentication [19, 20]. One of the discussions about these protocols is that if the protocol frame structure is known, a variety of attacks could be performed aiming to debilitate the fundamental components of secure communication, which are known de facto and called CIA (confidentiality, integrity, and availability) [21].

In this regard, various attack vectors have been developed, threatening the integrity principle on Siemens S7 communication. Some of these attack types are detecting the memory addresses of data blocks and input/output units of the PLC by simple queries, identifying the module vendor information, model number, and features of the PLC via MITM attacks, manipulating the program written by the engineering station while it is downloading to the PLC or code/program injection related to the lack of authorization or encryption mechanisms [22, 23]. Other vulnerabilities, such as displaying PLC RAM or stopping/running the PLC, are also discovered by the attack vectors.

The exploits of these vulnerabilities are posted to the Metasploit framework, which is a tool widely used for exploitation or testing purposes [24–26]. For PROFINET communication, the session ID and session information are transferred without encryption. Moreover, this ID can be used multiple times, unless the ID is not being used by another session on the server side.

Taking advantage of these vulnerabilities, session hijacking attacks and privilege escalation in the wake of hijacking are carried out [27]. In addition, DoS attacks are generated again on PROFINET communication, aiming to fuzz the protocol or stop the service. These types of attacks do not necessarily have to be complicated. For instance, one of the attacks relies on sending crafted PROFINET packets to convey the invalid/nonsense information to a PROFINET device using the DCP services [28]. This vulnerability was discovered in 2014 and added to the vulnerability database as "CVE-2014-2252" [29].

Intrusion detection systems and monitoring systems are important in terms of early detection of potential attacks on both the protocol and the system [30, 31]. Within this context, Ntalampiras et al. developed a hidden Markov model based on a fault-monitoring system for independent critical infrastructures [32]. The authors categorized critical system component data according to their distance from the training data. They generated two types of outcomes: DoS and replay attacks.

Besides the general-purpose research, there are studies focused on protocol specifications as well. For instance, Goldenberg and Wool modeled the Modbus/TCP protocol for intrusion detection. The researchers identified each communication channel between the HMI and PLC using deterministic finite automata (DFA) [33]. They achieved high accuracy on abnormal detection and faster detection of improper HMI configurations.

In 2014, Siemens S7 SCADA communication was modeled as an intrusion detection and monitoring system [34]. However, the proposed models had only periodic communication and client-server connections. Peer communications and aperiodic communication were not considered.

In addition to the intrusion detection system studies, there are other solutions in the literature. For instance, Cook et al. assigned identification to the attacks based on the fact that each attack possesses a unique behavior [35]. Then, the authors evaluated digital monitoring, malware analysis, network monitoring, honeypot, or trace monitoring methods, which are commonly used for identifying attacks.

Another technique used for attack detection was introduced in 2015 as an agent-node addition [36]. This study first spotted critical nodes of the power system by applying spanning tree algorithms. Then, it places an agent node between the two most critical nodes. The virtual node always generates the same virtual data and tries to mislead the attacker. The system is monitored by a master station, and in case of an attempt to change the agent-node data, an alert is generated.

Byres et al. investigated the difficulty levels of Modbus-implemented SCADA system attacks, vulnerabilities exploited by the attacks, and attack goals by attack trees [4]. They also evaluated the security risks and key parameters that need to be considered on protocol specification.

Finally, Ramachandruni and Poornachandran developed a honeypot system that simulates the Modbus and S7 PLC [37]. They connected the system to the external network and gathered real data for 30 days. Then, they identified the attack vector types on critical systems.

As stated, the research in the literature focuses on subjects such as monitoring systems, mathematical modeling of attacks/protocols for abnormal detection, or simulating the attacks on systems for threat identification. The future direction of studies confirms the need for a structure that monitors the critical infrastructure systems and prevents the attacks at the same time. This type of solution, which detects potentially malicious probes without generating any artificial test traffic, is called passive monitoring. It basically monitors the system inactively, that way preserving the real-time properties of the systems, and takes actions in the case of abnormal situations.

Another research field is IDS/IPS improvement to support critical infrastructure protocols. Snort, as a commonly used IDS/IPS system, provides preprocessors for analyzing incoming network packets. Recently, only Modbus/TCP and DNP3 preprocessors have been introduced by Snort developers to identify SCADA protocol packets. The ECAT preprocessor has not been developed yet. In addition, it is a challenge to develop an ECAT preprocessor because EtherCAT has a different communication structure from the other two protocols. Even though the released preprocessors have different characteristics, they have a similarity in that both of them transfer packets over TCP or UDP. The main reason for this is that the preprocessor base structure given by Snort only supports packets on layer 3. However, EtherCAT does not have a TCP/UDP header in the field and factory levels (except IP-tunneled communication) and requires further investigation by Snort for forwarding packets at layer 2.

2.2. Security Studies of ECAT. EtherCAT is one of the most widely applied protocols in the critical infrastructures. It has many features, such as hardware and software support diversity, as well as topology independency and performance. It supports lower cycle times since it processes packets in a byte-by-byte manner, namely, on the fly [38]. However, there are a few studies done in the literature focusing on the security aspect. In this section, research is grouped into three categories, that is, monitoring/data-gathering proposals, environmental developments, and protocol improvements.

The industrial network security book states that the EtherCAT protocol is open to DoS/DDoS attacks [39]. The author also emphasized the need for both MAC-based and master/slave communication-based systems or products to prevent unwanted EtherCAT flows. Related to this, a real-time data-acquisition system is introduced for gathering EtherCAT data securely with high speed [40]. Similarly, a proposal is presented for data acquisition on distributed EtherCAT-based systems. The study, which targets high accuracy and speed of data collection, is evaluated on the FPGA environment [41]. Another study is about the performance and design analysis of the data-acquisition systems [42]. The main problem regarding these studies is that the data-acquisition proposals can be considered a substitution for conventional data-acquisition cards, so that they do not possess any threat detection or protection properties on behalf of system security.

However, performance analysis studies have been done to improve the protocol. Knezic et al. proposed an algorithm that utilizes the frame size by examining data patterns within the EtherCAT frame [43]. In addition, researchers have recently evaluated the protocol efficiency using a simulation on MATLAB or by measuring hardware latencies on EtherCAT switches [44, 45].

Likewise, BeStorm developed a dynamic testing tool supporting EtherCAT [46]. This tool is commercial and only performs black-box fuzzing, which is a basic-level fuzzing approach that can be applied when the target is unknown. The recent study on EtherCAT security is presented in [47]. The research, however, does not mention the preprocessor structure, and only Snort rules are used for attack detection without proposing a novel attack-detection mechanism.

As stated above, during the literature review, various studies have been introduced on ICS and protocols used for system communications. They essentially focus on attack or threat detection, system or protocol optimization, or testing new approaches. However, as we outlined in this section, industrial automation system research, which specifically studies protocol-based security issues, is inadequate in the literature.

Moreover, there are many other factors that affect the security of these systems. Even small-scale attacks may cause catastrophic results due to the strategic location of critical systems. Mandatory controls applied on other systems, such as penetration tests, cannot be applied or are not advised for these systems because of their critical infrastructure. Thus, existing vulnerabilities cannot be explored.

In addition, these systems do not have fundamental security parameters, such as encryption, authorization, or authentication, so that if the frame structure and communication patterns are known, exposure to attacks is inevitable. All these security aspects influence the control systems in that they are open to attacks. From this point of view, to increase security against internal or external attacks, original proposals, such as passive monitoring, which detects malicious actions without creating any load on the system, can be introduced based on observational studies. In this paper, to contribute to the security of EtherCAT-implemented industrial automation systems, we took the intrusion prevention and detection research as a lodestar and conducted research on protocol vulnerability detection and attack prevention.

3. EtherCAT Protocol in Industrial Automation

In automation hierarchy, protocols used for fieldbus communication provide real-time criteria by keeping the setpoints for cycle time and cycle-time delay variance parameters. This situation also applies to Ethernet-based protocols. Therefore, Ethernet-based protocols have made significant modifications to meet with the real-time requirements on protocol structure and hardware used for executing protocols. Real-time Ethernet protocols are divided into two categories, namely, soft real time (RT) and hard RT. While Modbus/TCP is a soft RT protocol, protocols such as CC-Link, PROFINET, Sercos III, and EtherCAT are considered as hard RT. In this research, a hard RT protocol EtherCAT, which communicates over a frame carried by the modified Ethernet header, is studied.

The EtherCAT protocol has been managed by the EtherCAT organization since 2003. In comparison with other hard RT protocols, it has several advantages. For instance, it usually communicates over a standard Ethernet frame without any IP addressing and supports up to the data-link layer. However, if an EtherCAT network needs controlling through other subnets, the EtherCAT frames can be addressed and routed by adding UDP/IP headers. Moreover, it provides a short cycle time, topology flexibility, product variety, scalability, and low cost and supports real-time features through only one protocol (e.g., EtherCAT has better performance on real-time synchronous communication compared to the PROFINET IRT protocol [27]). The short cycle-time property of the EtherCAT is provided by on-the-fly technology [48]. The EtherCAT processes each frame in bytes, and this makes it faster even than Sercos III, which has a similar feature but processes input and output data individually (dual processing) on the fly.

Moreover, [15] stated that EtherCAT is the fastest industrial Ethernet technology with low update and response times. The response time of the EtherCAT is 0.1 ms, which is better than Ethernet/IP, Ethernet Powerlink, PROFINET IRT, and Sercos III [17]. Since these properties are the key points in critical systems, EtherCAT has become one of the major protocols in the automation sector.

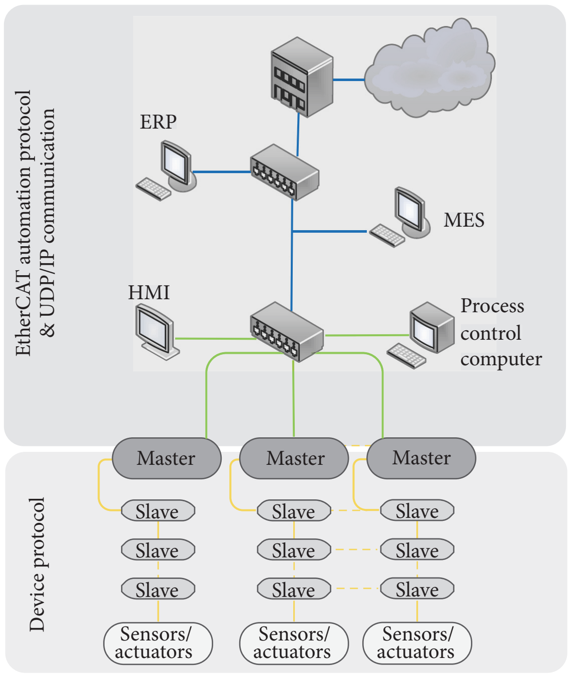


Figure 1: EtherCAT communication at all levels.

The founder and biggest supporter company of EtherCAT is Beckhoff. Since its establishment, Beckhoff has focused on computer-based automation applications and thus progresses in a different area from other automation companies. Beckhoff PLCs have the Windows operating system installed. Some properties of EtherCAT by Beckhoff, such as being Ethernet-based, the on-the-fly feature, the ability to integrate with UDP/IP, and the manageability of PLCs via Windows operating systems, provide incontrovertible advantages for providing the hard RT feature in the automation industry; however, they also bring many handicaps from a security viewpoint.

3.1. EtherCAT Communication at All Levels and Device Protocol. The TwinCAT program environment provides PLC programming and system configurations in the EtherCAT-applied systems. TwinCAT has a unique property and PLC programming, program/configuration downloading, and hardware configuring features. If PLC does not exist in the system or if there is a need for a PLC for testing purposes, the TwinCAT-installed engineering station can act as a PLC. The EtherCAT protocol supports typical master-master, slave-master, and slave-slave communications of critical systems. These communications are performed over three protocols: device, automation (EAP), and UDP protocols by integrating IP (Figure 1). The point to be noted here is that all types of communication are realized via a single type of cable: the Ethernet cable.

The EAP communicates via two subprotocols, namely, mailbox and process data protocols. The EAP transfers EtherCAT packets between master terminals and offers communication among cell, MES, and ERP levels in the automation hierarchy.

Another type of EtherCAT communication is outer systems communication. In this type of communication, the IP address is used, and transport is done over the UDP protocol. Therefore, by defining the extra 28 bytes of UDP and IP headers, accessing EtherCAT applications through other subnets is supported.

Since EtherCAT is an Ethernet-based standard, each EtherCAT packet is encapsulated by the Ethernet header. The EtherType of EtherCAT is defined as 0x88A4 in the Ethernet header. The type field in the header identifies the type of the data carried and may contain different values, as shown in Figure 2 [43]. In our research, we specifically focused on the field-level communication so that IP-based and EAP-based communication levels are out of the scope of this study.

3.1.1. Device Protocol. The device protocol is essentially responsible for handling field or sensor/actuator level communications. It exchanges data between slaves and masters and has its own frame structure. If the carried payload belongs to the device protocol, the value “1” is written to the type field in the main EtherCAT header. As illustrated in Figure 3, following the outer EtherCAT header, each EtherCAT datagram has its own header. This way, as an Ethernet packet passes through the slave stations like cars of a train, each station recognizes its own datagram. Each EtherCAT datagram has 2 bytes of “working counter” values at the end. During the data exchange inside each slave, this counter is incremented by 1 for read or write access and by 3 for read/write command executions in the memory [28].

If there are more datagrams following the current datagram, the M bit is set to 1 in the datagram header. In each datagram header, there are 32 bits of address field. This field indicates the unique slave addresses produced in a specific pattern. The address field can be filled in three different ways:

(i) If station-address assignments will be done according to their positions, automatic addresses are defined consisting of 16-bit position and 16-bit offset (automatically incremented by 1). In this case, the cmd command field will take APxx-type commands.

(ii) If user-defined addresses will be used, 16-bit addresses and 16-bit offsets are used, and the command field will take FPxx-type commands.

(iii) If logical addressing will be used, all 32 bits are addressed as logical, and the command field will take Lxx-type commands.

Command types can be used as follows:

(i) Automatically incremented and assigned addresses have read (R), write (W), read and write (R/W), and read and multiple write (RMW) access.

(ii) Logically assigned addresses only have R, W, and R/W access.

(iii) The NOP command is used to pass without any execution or change in status.

(iv) The broadcast command (Bxx) is used for unacknowledged transmission to an unspecified number

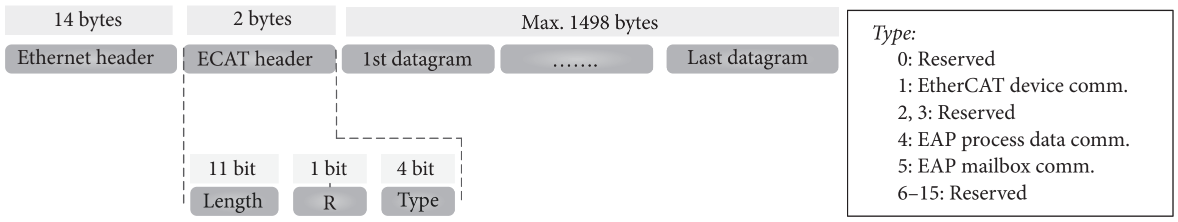


Figure 2: EtherCAT header.

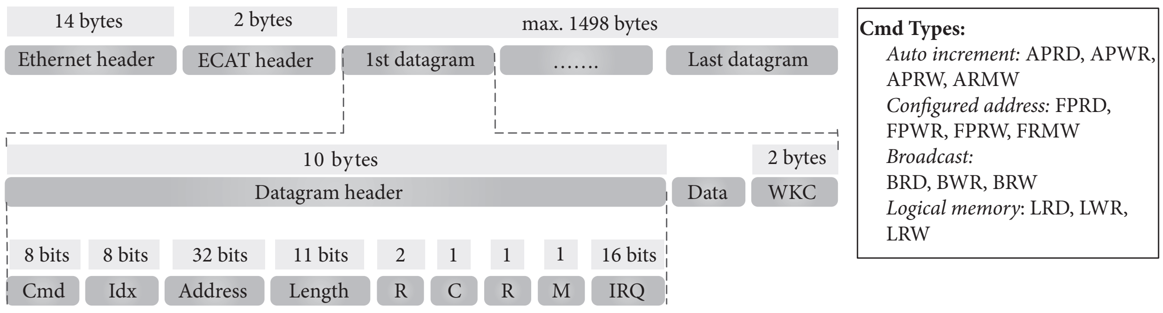


Figure 3: Device protocol.

of receivers, where all the stations share part of the frame. It could be sent for initialization or checking status of all the slaves. It has R, W, and R/W access.

3.2. EtherCAT Slave Information/EtherCAT Network Information Files. In EtherCAT-based systems, the EtherCAT network information (ENI) and EtherCAT slave information (ESI) files indicate a trust relationship between slave or master terminals. These files are in XML format and contain startup configurations of communications. Each slave has an ESI file, which includes factory default properties of the slave created by the manufacturer. The ESI files specifically contain manufacturer information, device information, such as the module, group, or order number, and the default parameters used during the communication as presented in Figure 4. These files are distributed during manufacturing and are stored in the TwinCAT installation directory. All ESI information and/or online information sent from the slave EEPROMs is extracted and combined as one ENI file that describes the transmission details between masters and slaves. The ENI file consists of factory-assigned PLC information, such as the MAC address, PLC configuration information, synchronous data exchange information, the mailbox and its subprotocol information, slave information, and process data identification information. The ENI.xml file given in Figure 4 presents our actual testbed configurations. If necessary, the ENI file can be exported by the EtherCAT engineering station or other third-party configuration tools. In this paper, the ENI files generated by the engineering station are considered.

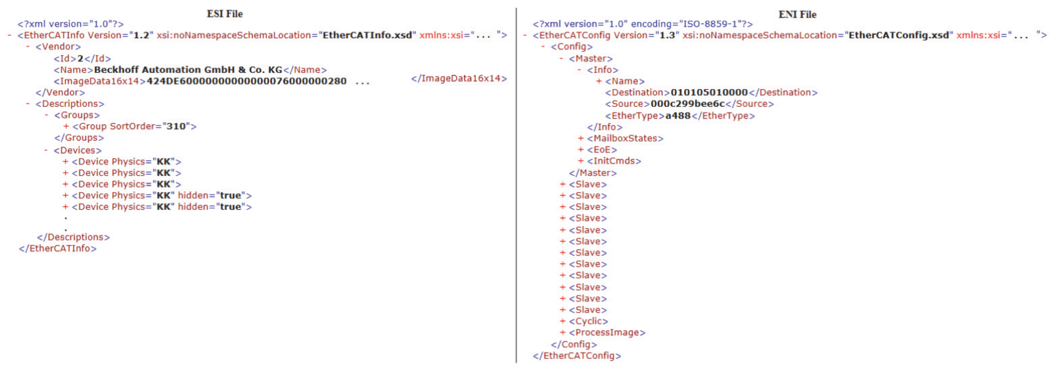
Section 5.2.2 introduces the trust-node identification approach, which is designed using the ESI and ENI configuration file data. For the trust-node identification approach, we used the MAC address information of master stations and the PhysAddr, AutoIncAddr, command, order of command, and data-size information of each slave.

4. EtherCAT Vulnerability Research

In this section, protocol vulnerabilities that are caused by exploiting the protocol weaknesses caused by the absence of encryption, authentication, and authorization features in the EtherCAT protocol, similar to other industrial protocols, are analyzed. As a result of analysis, predicted vulnerabilities are evaluated on the device level by attack vectors so that protocol weaknesses are identified. To prevent exploitation of these proved vulnerabilities, a basic-level EtherCAT protocol decoder, preprocessor, and trust-node identification approach within the preprocessor are developed on an open-source IDS/IPS system named Snort.

4.1. Testbed. The test environment is created by real hardware at Cyber Security Laboratory, Sakarya University, as illustrated in Figure 5. Vulnerabilities are examined by applying the specified attack vectors on this testbed. The test environment has a Windows XP installed PLC, several digital I/O units, a network tap for performing MITM attacks, a TwinCAT-installed virtual computer to download the configuration/program, and a Snort IDS/IPS-installed virtual computer for implementation of our proposal.

4.2. Attack Vector Generation on Device-Level Communication. A widely accepted definition of vulnerability is a fault or weakness that decreases or restricts the ability of a system to withstand a threat or resume a new stable condition [49]. Similarly, EtherCAT vulnerabilities are weaknesses from the nature of the protocol, such as its frame structure, intra- and inner-level communications, or the absence of encryption,


ESI File

```
  <?xml version="1.0"?>
- <EtherCATInfo Version="1.2" xsi:noNamespaceSchemaLocation="EtherCATInfo.xsd" xmlns:xsi="... ">
  - <Vendor>
      <Id>2</Id>
      <Name>Beckhoff Automation GmbH & Co. KG</Name>
      <ImageData16x14>424DE6000000000000076000000280  ...      </ImageData16x14>
    </Vendor>
  - <Descriptions>
    - <Groups>
      + <Group SortOrder="310">
      </Groups>
    - <Devices>
      + <Device Physics="KK">
      + <Device Physics="KK">
      + <Device Physics="KK">
      + <Device Physics="KK" hidden="true">
      + <Device Physics="KK" hidden="true">
        .
        .
    </Descriptions>
  </EtherCATInfo>
```

ENI File

```
  <?xml version="1.0" encoding="ISO-8859-1"?>
- <EtherCATConfig Version="1.3" xsi:noNamespaceSchemaLocation="EtherCATConfig.xsd" xmlns:xsi="...  ">
  - <Config>
    - <Master>
      - <Info>
        + <Name>
          <Destination>010105010000</Destination>
          <Source>000c299bee6c</Source>
          <EtherType>a488</EtherType>
        </Info>
      + <MailboxStates>
      + <EoE>
      + <InitCmds>
      </Master>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Slave>
    + <Cyclic>
    + <ProcessImage>
    </Config>
  </EtherCATConfig>
```

Figure 4: Sample ENI.xml and ESI.xml (for EL1xxx module).

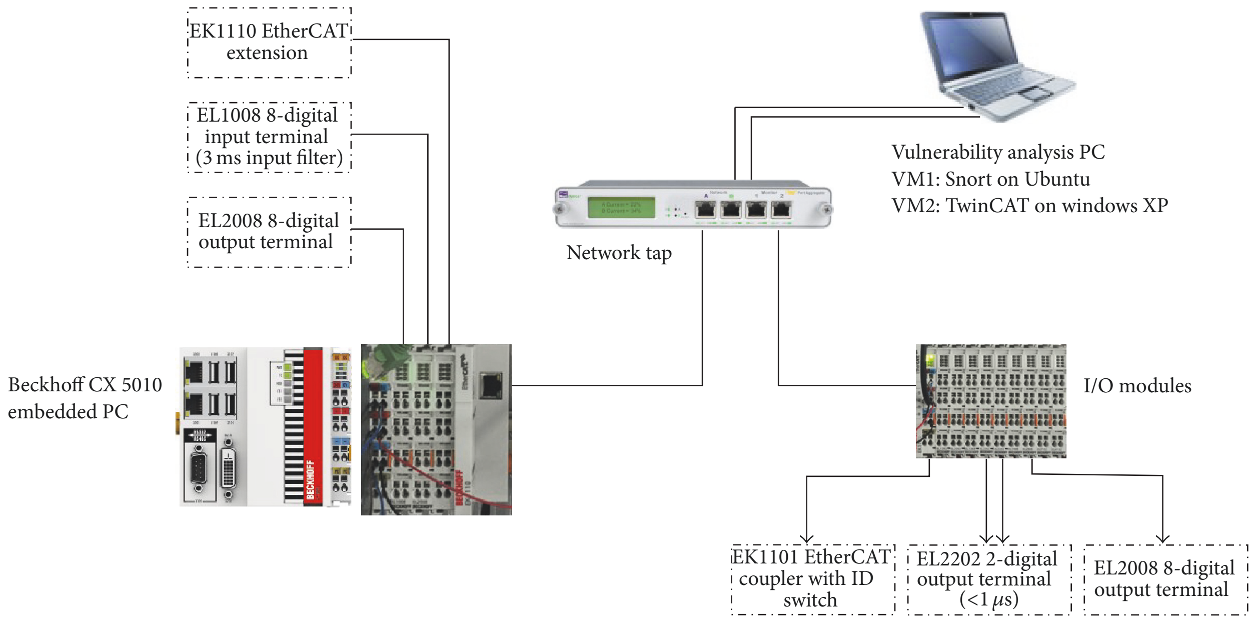

EK1110 EtherCAT extension
EL1008 8-digital input terminal (3 ms input filter)
EL2008 8-digital output terminal
Vulnerability analysis PC
VM1: Snort on Ubuntu
VM2: TwinCAT on windows XP
Network tap
Beckhoff CX 5010 embedded PC
I/O modules
EK1101 EtherCAT coupler with ID switch
EL2202 2-digital output terminal (<1 μs)
EL2008 8-digital output terminal

Figure 5: Testbed.

authentication, and authorization mechanisms, as many other industrial automation protocols do not include these. Therefore, as stated in the literature review, many attacks, such as code/program injection, PLC stop/run, displaying RAM contents, replay attack, MITM attack, DoS attack, and querying address or model information, are attempted to weaken the CIA components of other protocols. Given that all of these attacks take place through the communication protocol, it is likely that the potential vulnerabilities stem from the EtherCAT protocol structure as well.

The vulnerability analysis stage of this research was examined only for field-level (device protocol) communication, and the attack vectors were generated by programming each attack individually. We have performed, succeeded in, and later evaluated MAC spoofing, data injection, and slave-address attack vectors, which are given in detail below.

The MAC Spoofing Attack. This attack is based on manipulating the MAC address to an unauthorized master MAC and consists of four steps:

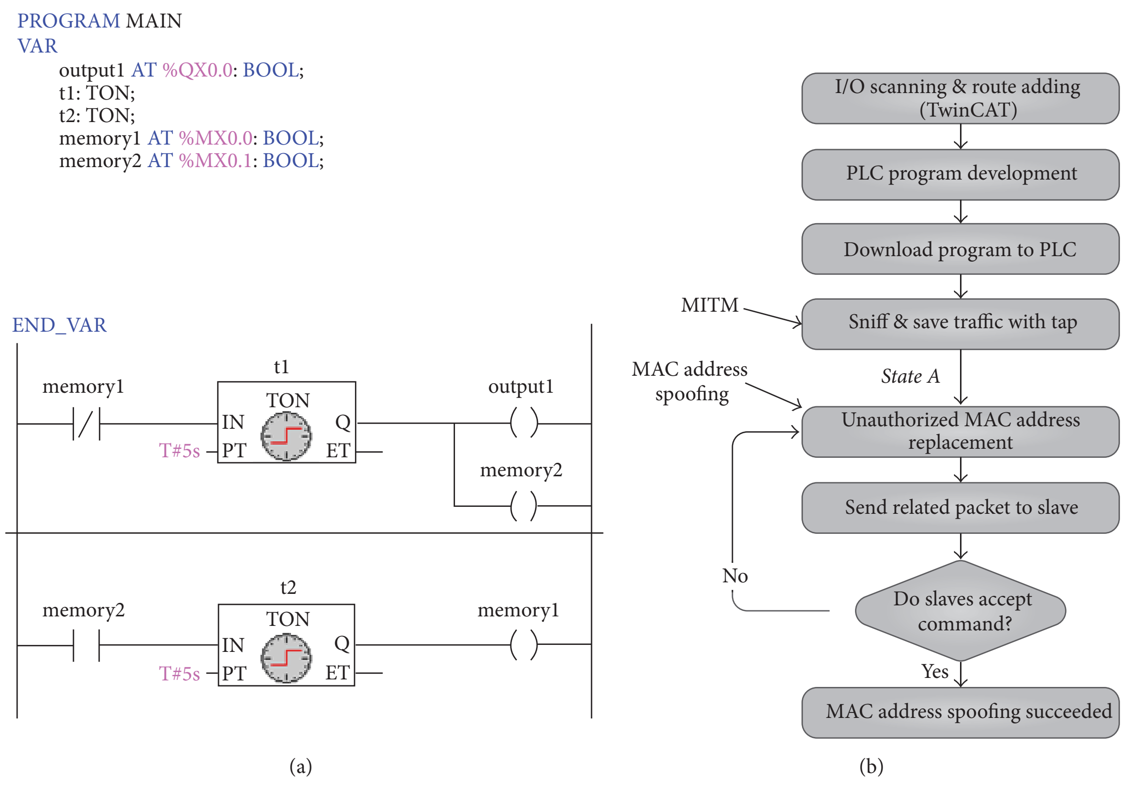
(I) PLC program development and capturing communication packets

(II) Analyzing the captured packets

(III) Manipulating the MAC address

(IV) Execution of the attack

The first step is programming the PLC located in the testbed environment. We have developed a simple PLC program on the TwinCAT-installed computer, which turns on an LED of an output for 5 seconds and then turns it off for 5 seconds (Figure 6(a)). While programming was completed, the program outputs were mapped to the first output on the EL2008 terminal of the EK1101 module on the I/O units in Figure 5. After this operation, the program was loaded to the PLC and run. Communication patterns between the PLC-I/O units were captured by the MITM technique using the network tap device presented in Figure 5.



```
PROGRAM MAIN
VAR
      output1 AT %QX0.0: BOOL;
      t1: TON;
      t2: TON;
      memory1 AT %MX0.0: BOOL;
      memory2 AT %MX0.1: BOOL;
END_VAR
```

memory1
t1
TON
IN
Q
T#5s
PT
ET
output1
memory2
memory2
t2
TON
IN
Q
T#5s
PT
ET
memory1
(a)
I/O scanning & route adding (TwinCAT)
PLC program development
Download program to PLC
MITM
Sniff & save traffic with tap
State A
MAC address spoofing
Unauthorized MAC address replacement
Send related packet to slave
No
Do slaves accept command?
Yes
MAC address spoofing succeeded
(b)

Figure 6: (a) Developed simple PLC program and (b) MAC spoofing attack flow diagram.

Second, we analyzed the network traffic. We have observed that multiple commands (NOP, logical write [LWR], logical read [LRD], broadcast [Bxx], and autoincrement physical read multiple write [ARMW]) were used synchronously. In addition, the commands for setting the output were sent synchronously in each cycle. They were not only sent when the LED needs to be turned on. In other words, the value “0” was sent in each cycle for turning off the output, while “1” was sent to turn it on. Besides the EtherCAT field-level communication, there were a few Link Layer Discovery Protocol (LLDP) and Multicast Domain Name System (MDNS) protocol packets. To generate the MAC spoofing attack, the communication packets were first exported in the K12 text file format by Wireshark, which is a packet-sniffing program. Second, the MAC address of each packet was replaced by a counterfeit MAC address of an authorized member of the network (Figure 6(b)). It is observed that as long as the PLC power is not interrupted, slaves accept the incoming packets. We proved this statement by examining the accepted working counter values of the packets and monitoring the output lights. The outgoing working counter value of the packets was 1, whereas the same incoming working counter packet was incremented to 5. In addition, the master PLC did not present any errors; thus, the system administrator cannot realize that a MAC spoofing attack is generated. The main reason for this is the absence of an authentication mechanism between the master stations and the slaves and the presence of only a basic level of identification, which is performed during the configuration and device scan phases.

Alternatively, due to the erased configurations of the I/O units, the MAC spoofing attack cannot be performed if the PLC power is interrupted. Thus, a network scan must be done by TwinCAT in advance, and a network route must be created between the PLC and I/O units. A related system configuration must be completed, and the PLC must be in a running state.

Data Injection Attack. This attack is developed by first capturing the packets from the system through a MITM approach. Since the LWR commands were responsible for writing to the output, the data field of the LWR command is manipulated (Figure 7(a)). The data length of the LWR containing datagram was 16 bytes. The payload of the datagram is manipulated to 20 bytes. In addition, the datagram length and frame length fields of the packet are modified. The datagram length is set to 20, and the packet length is set to 150 bytes. As there are no authentication or integrity controls, these frames are accepted by the slave stations. Depending on the state of the network, this type of attack could disrupt the real-time capability or even lead to physical damage.

Attack on Slave Stations. When packet patterns of the communication are observed, datagrams consisting only of the LWR command are responsible for turning on the LED of an output on the EL2008 module, which is a slave I/O unit. Using this information, the detected datagrams are extracted from the packets, and their address structure is examined (Figure 7(b)). Analysis showed that individual I/O units of the same terminals (EK1101) share the same logical addresses.

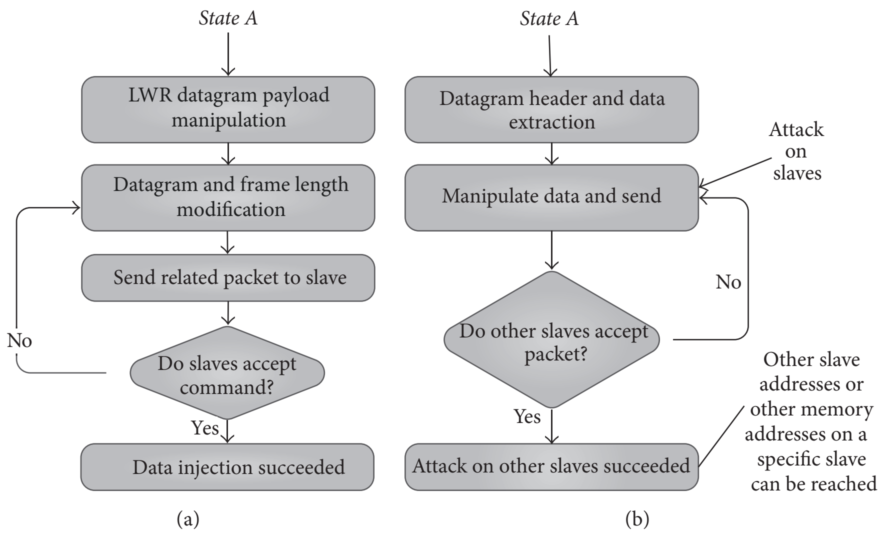

State A
LWR datagram payload manipulation
Datagram and frame length modification
Send related packet to slave
Do slaves accept command?
No
Yes
Data injection succeeded
(a)
State A
Datagram header and data extraction
Manipulate data and send
Attack on slaves
Do other slaves accept packet?
No
Yes
Attack on other slaves succeeded
Other slave addresses or other memory addresses on a specific slave can be reached
(b)

Figure 7: (a) Data injection attack flow diagram and (b) slave access attack flow diagram.

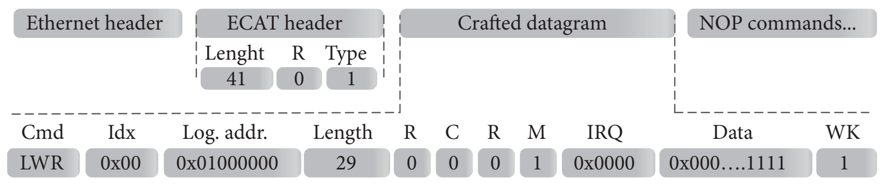

Ethernet header
ECAT header
Crafted datagram
NOP commands...
Lenght R Type
41 0 1
Cmd Idx Log. addr. Length R C R M IRQ Data WK
LWR 0x00 0x01000000 29 0 0 0 1 0x0000 0x000....1111 1

Figure 8: Sample crafted datagram with LWR command.

Therefore, a new crafted packet, which contains a datagram with an LWR command and has the same length as the original datagram (29 bytes), was generated by a program developed in C. The NOP commands were appended to achieve the minimum frame length (60 bytes). The data field of the LWR command was manipulated by changing the last 2 bytes (little endian) to 255 (Figure 8). After that, packets were sent back to the network.

The last 16 outputs of the same module flashed. Slaves do not have any security profiling identification to distinguish masters, and they cannot control the access of other memory addresses. Therefore, we have found that slaves, commonly known as dummy devices, allow access to their unauthorized memory addresses and do not authenticate masters for each flow. Thus, if the memory address map of a slave is correctly detected, it will immediately respond to incoming packets.

5. Snort IDS/IPS on Industrial Automation Systems

Snort, commonly accepted as an IDS/IPS system, is open-source software used for detecting or preventing anomalies within the network. It works in a signature-based manner. It can be installed either on a virtual host or on a firewall as add-on for deep packet inspection. The network to be investigated should be passed through the Snort software. The chosen signatures can be added to the rule database via Oinkcode provided by the Snort developers. Traffic/flow/packets that match these rules are processed by various actions, such as log, block, or ignore.

Snort rules can be implemented either by using the rule database in its repository or by writing the requested rules manually. The Snort system identifies packets at the second layer of TCP/IP protocol suite using the Libpcap library. The system first forwards an incoming packet to the *decoder* module for extracting third- and fourth-layer headers. Later, it sends the packet to the default preprocessors, which are previously activated by the user in the *snort.conf* file (Figure 9). Rule-based matching can be done during this stage as well. Consecutively, if other related preprocessors that can handle the packet up to the application layer are found, the packet is forwarded to them; otherwise, the packet is handled by the *detection engine* module at the fourth-layer protocol level with processing rules and options. Eventually, it is saved to the database in XML or other formats using alert, log, ignore, or block actions.

Early versions of Snort were able to extract up to a third or fourth layer, but newer versions can recognize the application layer as well. The received packets are first extracted from the Ethernet headers. During this step, preextraction of the next protocol is also performed by examining the type field of the header. The rest of the packet is forwarded to the related module for the next protocol, such as TCP/UDP, ARP, VLAN, and PPP processing. At last, the packet data is extracted with the help of transport layer protocols. It should be noted that the data can be handled or processed as default if and only if it is carried by the supported level protocols. For instance,

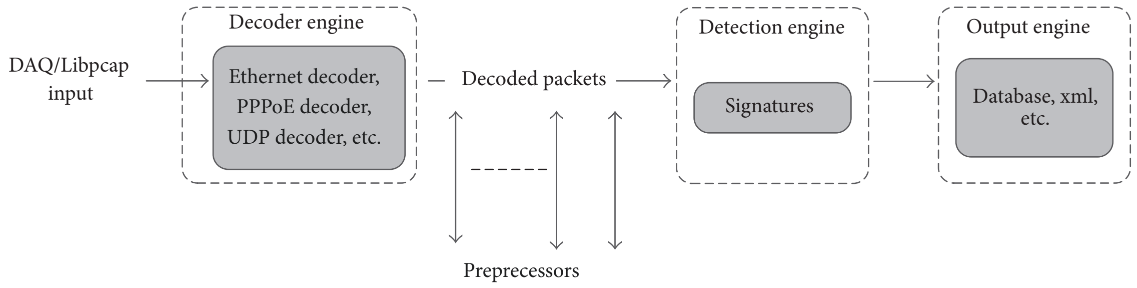


Figure 9: Packet flows in Snort.

if an analysis will be done for protocols over the application layer or other protocols over the transport layer, additional preprocessors are needed for extracting.

These preprocessors can identify alert, pass, drop, sdrop, or reject rule actions. While some preprocessors come as default with early versions of Snort, some are loaded dynamically in runtime with recent versions. These preprocessors are called dynamic preprocessors. Snort commonly uses DNS, ARPSpoof, FTP/Telnet, SSH, and other protocol preprocessors. In addition, it has a few automation system preprocessors, such as DNP3 or Modbus preprocessors.

Automation system protocols that come with recent versions are very few and have a limited number of rules. For instance, the Modbus preprocessor contains only three rules, and these are defined for very simple packet analysis, such as function code or protocol ID field checking. The common property of these automation system preprocessors is that their underlying protocol communication is achieved over the TCP/IP protocol suite. In other words, all of these protocols process over the transport layer.

The main reason for this is related to a principle of Snort, which is that Snort always extracts incoming packets until the transport layer. After that, if any preprocessor will be used, it forwards the packets to the preprocessor. Therefore, all the preprocessors must receive the remaining part of the packets after the transport layer extraction. However, instead of having the transport layer over the data-link layer, EtherCAT uses its own frame structure for device and EAP protocols. This type of packet flow challenges the ECAT-preprocessor development. The next section presents the solution developed within the *decoder engine* of Snort.

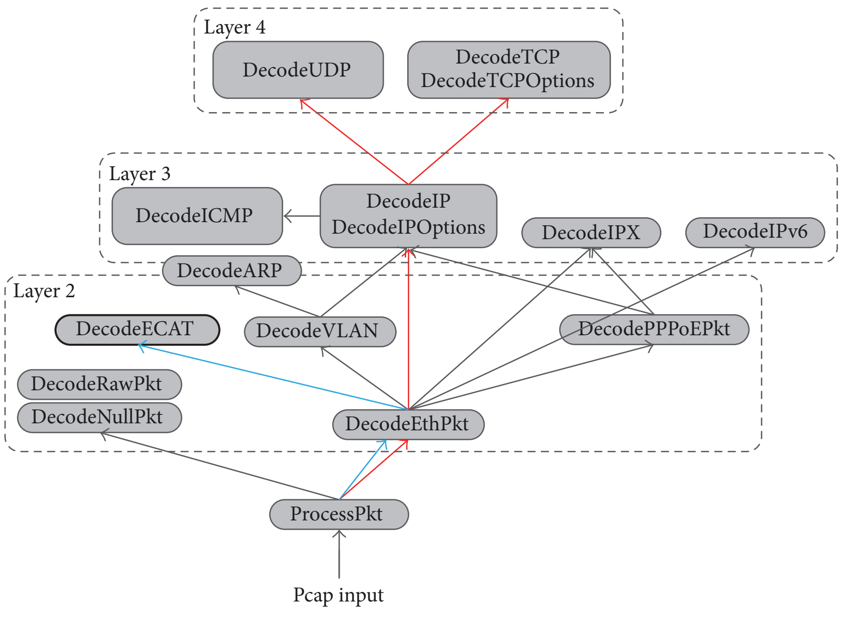
5.1. Layer 2 EtherCAT Decoder for Snort. Each packet received by Snort is handled within the *decoder module*. Once extraction is done, it is forwarded to the corresponding activated preprocessor using the transport layer protocols. Packet flows inside this module are presented in Figure 10. Previously developed industrial automation system protocol preprocessors, such as DNP3 and Modbus/TCP, communicate over TCP/IP and follow the path shown with red arrows. As seen, incoming packets are delivered to preprocessors once the layer 4 extraction is completed. Thus, during the preprocessor development, the preprocessor must be registered as either TCP or UDP protocols regarding the transportation needs of the packets.

As mentioned, EtherCAT communication does not contain the IP address or any transport layer protocols in the factory and field levels. In this respect, it works differently from the DNP3 or Modbus/TCP standards. Even if preprocessors of industrial automation system protocols that process only over Ethernet without using TCP/IP are developed, they will not be supported by the Snort decoder structure; thus, they will not function. Therefore, prior to the preprocessor development, we first developed the *EtherCAT decoder (DecodeECAT)* on the Snort *decoder module*. Then, we delivered the remaining packet to the ECAT preprocessor (Figure 10). An incoming EtherCAT packet follows the blue path after the *Ethernet decoder* and reaches the *EtherCAT decoder*. The *DecodeECAT* computes some important parameters, such as the EtherCAT header structure, number of datagrams within the packet, or the beginning of the payload, which will be later used by the preprocessor, statistics, and rules. It then locates the first bit of the data field and forwards the packet to the ECAT preprocessor. To register this type of preprocessor to the Snort system, the "none" value is defined for the transport layer protocol field. This solution is an improvement to Snort in the sense of supporting protocols that transport over layer 2. Thus, the ECAT preprocessor in Snort can process accurately.

5.2. ECAT-Preprocessor Development and Trust-Node Communication Approach

5.2.1. ECAT Preprocessor. As stated, an ECAT preprocessor that decodes EtherCAT protocol packets has not been introduced by Snort yet. Therefore, any signature of EtherCAT packets cannot be defined. In this context, the second proposal on Snort, which now can identify and support the coming layer 2 packets, is the development of a new preprocessor that can process actions and define rules, particularly for EtherCAT packets.

The previously activated ECAT preprocessor in *snort.conf* first completes the registration and preloading stages (initialization in Figure 11). These processes are handled during the startup configuration loading. During registration, the *process* function, which will be called for each received packet, must be specified. This function essentially checks the rules defined with the preprocessor's ID number for each incoming packet, dynamically. Alternatively, for each ECAT-preprocessor rule, one control function is created as well.



Red: TCP/IP flows
Blue: EtherCAT flows
Grey: other flows

FIGURE 10: Packet flows in the Snort decoder engine.

Once related functions operate, the attack or detected anomaly packets are saved in different log formats using the *_dpd* structure. After checking the rules, the *detection engine* is also called with the corresponding log and alert commands. The *detection engine* is responsible for parsing the dynamic preprocessor rule files. The preprocessor rules are written, and the matching rules with the alarms are forwarded to the *output module* (Figure 11).

5.2.2. Trust-Node Communication Approach. Snort preprocessors provide IDS/IPS features by checking the defined signatures. Each preprocessor has a unique generator ID *(gid)*, and each rule has a Snort ID *(sid)*. The *sid* shows the rule number for a particular preprocessor. The preprocessor rules are defined differently from the common rule syntax, as follows: *rule_type_or_action (message_to_be_printed; snort_ID; generator_ID; revision_num; metadata; classtype;)*.

Supported industrial automation preprocessors of Snort contain simple rules, such as checking the type field, packet length, or protocol ID values of incoming frames. However, the ECAT preprocessor provides deep packet inspection. This process is based on the idea of detecting existing nodes in the EtherCAT-applied system and accepting them as secure nodes. The proposal is introduced as an outcome of the vulnerability analysis performed in the previous section.

As identified in the EtherCAT vulnerability analysis, there is no authentication mechanism between the master and slave terminals. Taking advantage of this weakness, various attacks can be applied. One of these attacks is collecting a sufficient number of packets over the system and detecting a slave. After this, to investigate other nodes or slaves, the attacker alters the address of the slave and sends similar packets to different or consecutive addresses. This way, network map, address, and configuration information about the nodes of the network can be predicted.

To prevent these types of vulnerabilities, we detected all devices within the EtherCAT system and loaded them during the startup configuration of the preprocessor of Snort. These nodes are called trust nodes, and any communication with a different slave address, master address, command type, data size, or command order is detected. This approach is achieved by creating an advanced rule that saves the log or alert with 146 ECAT-preprocessor ID numbers.

Each received packet that has an EtherCAT type value is delivered to the preprocessor after decoding the Ethernet and EtherCAT headers. Here, the preprocessor checks every packet to determine whether the address of each datagram, command, command order, data size, and MAC address of master devices matches the trust communication identified value. For signature-based detection, three rules are generated, namely, untrusted slave, untrusted master, and data injection. According to the following defined rule, for the addresses not included in the trust communication, an alarm is generated and saved into the log file (Figure 11): *alert (msg: "ECAT_UNTRUSTED_SLAVE"; sid: 1; gid: 146; rev: 1; metadata: rule-type preproc; classtype: protocol-command-decode;)*.

The trust-node identification approach is a solution introduced for preventing vulnerabilities and detecting attacks on EtherCAT field-level communication. This method is executed on the initialization and rule-matching stages of the preprocessor. The method is as follows. After the network

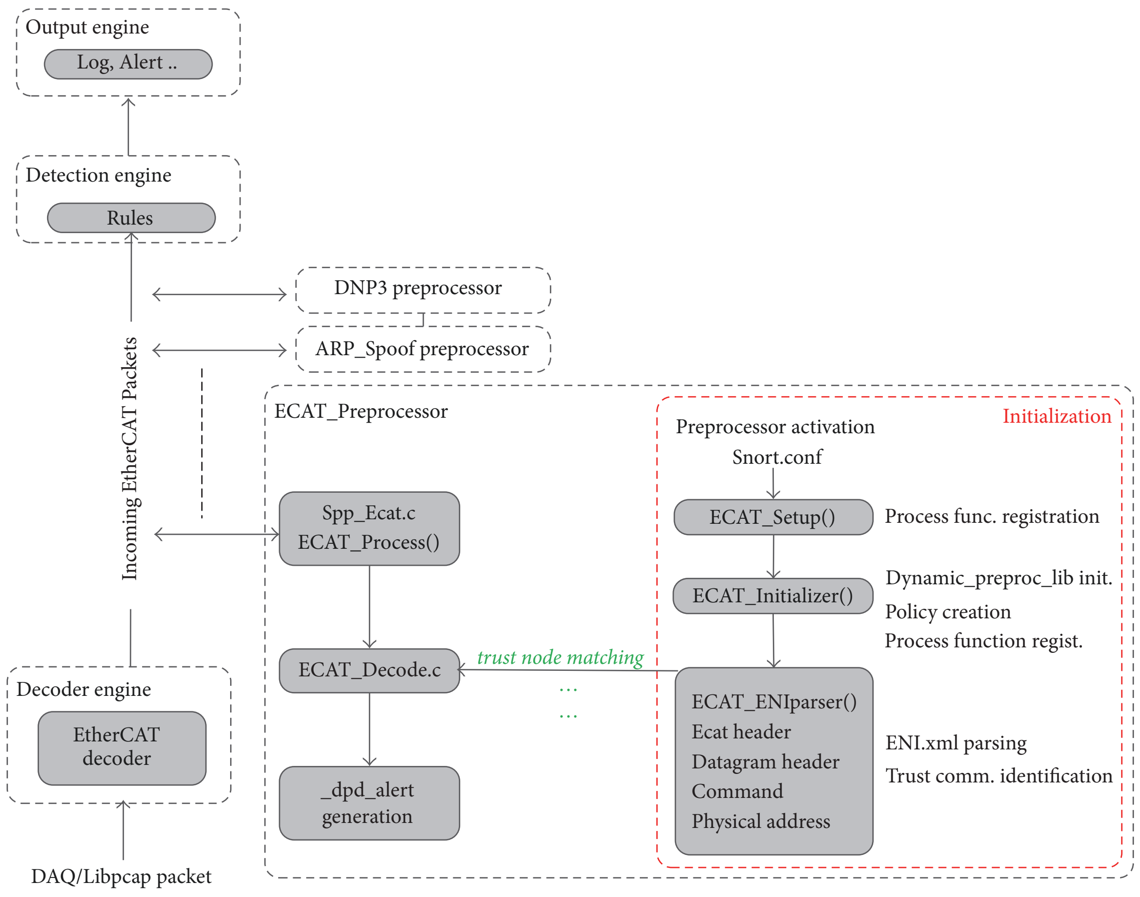


FIGURE 11: The ECAT preprocessor.

scan and initial configuration stage are completed on TwinCAT, each ESI file of the slave is loaded to the EtherCAT configurator tool, which connects to the master terminals. This ESI file contains essential information about a specific slave during EtherCAT communication. Slave ESI files and/or online slave information is combined on the master side, and an XML-based ENI file is created. This file contains some startup configuration data, such as the master information, slave information, cycle, and process image data (Figure 12). The ECAT preprocessor decodes the ENI file, which is generated during the critical system runtime using the parser presented in Figure 11. Decoding is completed during Snort runtime at the initialization stage. This way, we identify the master and slave nodes, data size, commands, and even the command orders included during the usual EtherCAT communication approved by the engineering station so that only the flows that fit with these attributes are accepted as trusted communication. Eventually, each packet that is sourced from the decoder engine during runtime and is destined to the ECAT preprocessor dynamically is checked, whether it is trusted or not; thus, a basic level of security mechanism is provided over the network.

5.3. Preprocessor Testing. To test the ECAT preprocessor, we used the testbed environment presented in Figure 5. The virtual machine with Snort installed is located between the master and slaves, as shown in Figure 12. The ECAT-preprocessor registration and initialization can be seen in Figures 13(a), 13(b), and 13(c). The ECAT preprocessor reads the ENI.xml file exported from the TwinCAT-installed engineering station and extracts available trusted nodes. Later, it loads all activated preprocessors with version and build numbers into Snort and waits for incoming packets. When an EtherCAT packet is received, the preprocessor extracts all datagrams and headers, such as EtherCAT and datagram headers. A decoded random EtherCAT packet is printed in Figure 13(d) for testing purposes.

For the attack on slave stations, crafted EtherCAT datagrams with the LWR command are generated with an untrusted node (address: 257) and sent to the network. The packet in Figure 13(d) includes one for those datagrams since the untrusted slave access alarms are triggered.

For a MAC address spoofing attack, the frame MAC addresses are replaced by unapproved master hardware addresses and sent to the network. The flagged alarm is given in Figure 13(e). It is observed that the ECAT preprocessor can detect defined attacks and log them properly as alarms into the *alert file* presented in Figure 13(f).

For data injection attacks, some other fields extracted from the ENI file, such as command names, the command order, and the expected data length of each command, are used. These fields are under the cyclic section of the ENI file. If

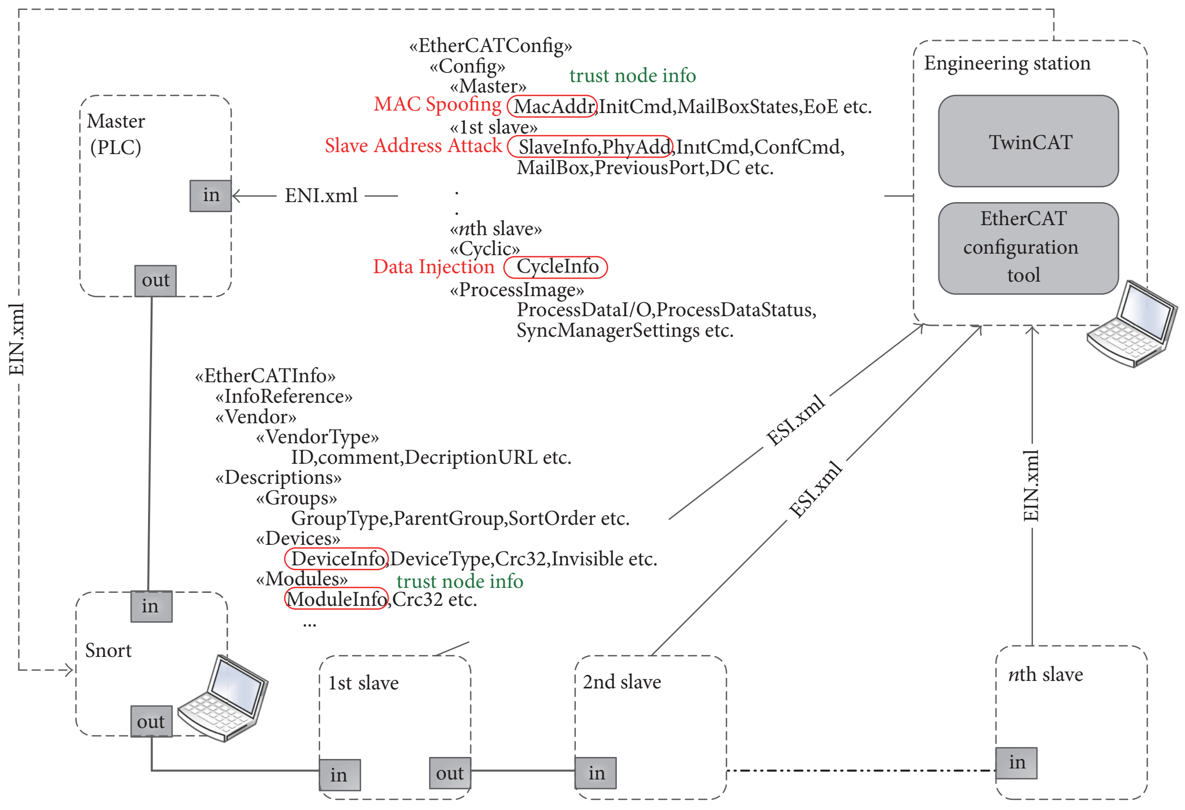


FIGURE 12: The process of ESI-ENI files.

any of the attributes, such as command order, does not match, an alarm is generated. For instance, when the ENI file in the testbed is analyzed, the ARMW command is expected as the fourth frame in the package with 4-byte data. For testing purposes, order and data length of the ARMW contained datagram are modified. It is observed that any change in the order of the command, data length, or data field is detected, and an alarm is generated in the alarm file (Figure 13(f)). The attack vectors mentioned in Section 4.2 can be simply prevented by the features of the ECAT preprocessor.

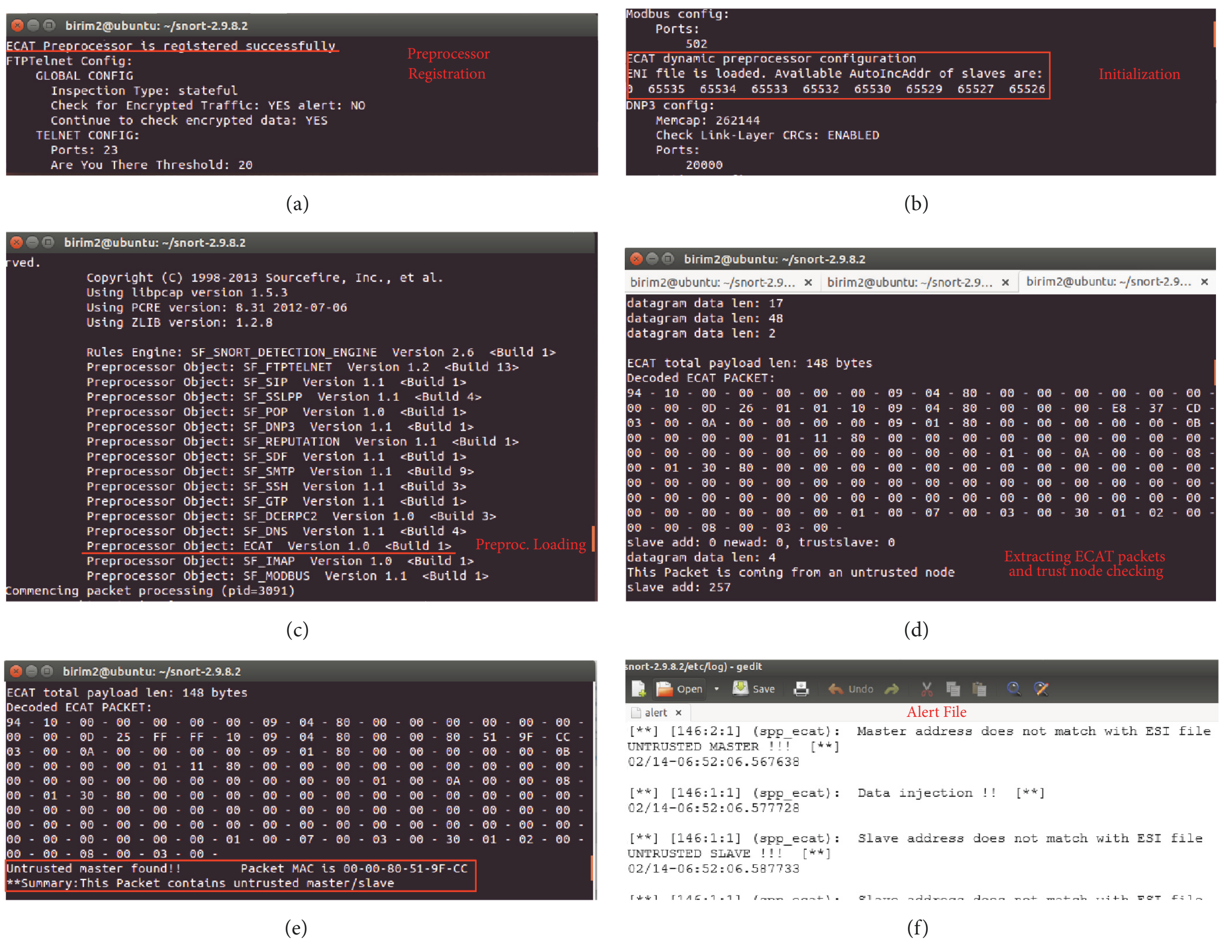
The trust-node approach is able to detect MAC spoofing, data injection, and slave-address access attack vectors, when they attempt to reach disapproved components by the created rule. Moreover, MAC spoofing attacks can be prevented by checking the master hardware addresses in the ENI files, while data injection can be determined by data field checking for each expected command. Slave-address access attacks can be prevented by checking the logical, autoincremented, or physical addresses of slaves. Moreover, since the ECAT preprocessor is a base IDS/IPS proposal for EtherCAT systems and allows creating new rules, replay or other types of attacks can also be eliminated by defining additional rules, such as time-based rules.

To visualize the preprocessor logs and trust-node approach alerts, a new virtual machine is set up and Elasticsearch, Logstash, and Kibana (ELK) stack is developed. The monitoring system receives Snort logs from the EtherCAT preprocessor through Filebeat tool, and based on the syslogs and JSON-based database queries, it visualizes the intrusions to the users. It is observed that, under a real working condition, the prepared EtherCAT dashboard presents the same intrusions using the system logs (Figure 14).

6. Conclusion

In this paper, an EtherCAT device-level vulnerability analysis is performed. Based on the vulnerabilities, the new ECAT preprocessor is developed using Snort by introducing a trust-node communication approach. To the best of our knowledge, no vulnerability analysis research has specifically focused on EtherCAT communication principles. Therefore, prior to the preprocessor development, communication analysis was performed on factory, device, and IP-based communication levels. Later, the device-level vulnerability analysis was achieved by creating attack vectors related to the device-level protocol specifications. Attack vectors were implemented as MAC spoofing, data injection, and slave-address access attacks.

The results confirmed that EtherCAT does not provide a level of security between slave and master communication. Attackers can easily sniff EtherCAT traffic and send their crafted packets over the Ethernet cable. This also proves the ability to apply well-known TCP/IP attacks on EtherCAT systems, as it is carried over the Ethernet. We observed that slave and master devices create the route while the connection is initiating, and this connection expires immediately after the power of the PLC is down. Except for this initiation, no security exists for each flow between master and slave.


(a)

```
birim2@ubuntu: ~/snort-2.9.8.2
ECAT Preprocessor is registered successfully
FTPTelnet Config:
    GLOBAL CONFIG
      Inspection Type: stateful
      Check for Encrypted Traffic: YES alert: NO
      Continue to check encrypted data: YES
    TELNET CONFIG:
      Ports: 23
      Are You There Threshold: 20
```

(b)

```
Modbus config:
    Ports:
        502
ECAT dynamic preprocessor configuration
ENI file is loaded. Available AutoIncAddr of slaves are:
0  65535  65534  65533  65532  65530  65529  65527  65526
DNP3 config:
    Memcap: 262144
    Check Link-Layer CRCs: ENABLED
    Ports:
        20000
```

(c)

```
birim2@ubuntu: ~/snort-2.9.8.2
rved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.5.3
           Using PCRE version: 8.31 2012-07-06
           Using ZLIB version: 1.2.8

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.6  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: ECAT  Version 1.0  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
Commencing packet processing (pid=3091)
```

(d)

```
birim2@ubuntu: ~/snort-2.9.8.2
birim2@ubuntu: ~/snort-2.9...  ×  birim2@ubuntu: ~/snort-2.9...  ×  birim2@ubuntu: ~/snort-2.9...  ×
datagram data len: 17
datagram data len: 48
datagram data len: 2

ECAT total payload len: 148 bytes
Decoded ECAT PACKET:
94 - 10 - 00 - 00 - 00 - 00 - 00 - 09 - 04 - 80 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 0D - 26 - 01 - 01 - 10 - 09 - 04 - 80 - 00 - 00 - 00 - E8 - 37 - CD -
03 - 00 - 0A - 00 - 00 - 00 - 00 - 09 - 01 - 80 - 00 - 00 - 00 - 00 - 00 - 0B -
00 - 00 - 00 - 00 - 01 - 11 - 80 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 01 - 00 - 0A - 00 - 00 - 08 -
00 - 01 - 30 - 80 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 00 - 00 - 00 - 00 - 01 - 00 - 07 - 00 - 03 - 00 - 30 - 01 - 02 - 00 -
00 - 00 - 08 - 00 - 03 - 00 -
slave add: 0 newad: 0, trustslave: 0
datagram data len: 4
This Packet is coming from an untrusted node
slave add: 257
```

(e)

```
birim2@ubuntu: ~/snort-2.9.8.2
ECAT total payload len: 148 bytes
Decoded ECAT PACKET:
94 - 10 - 00 - 00 - 00 - 00 - 00 - 09 - 04 - 80 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 0D - 25 - FF - FF - 10 - 09 - 04 - 80 - 00 - 00 - 80 - 51 - 9F - CC -
03 - 00 - 0A - 00 - 00 - 00 - 00 - 09 - 01 - 80 - 00 - 00 - 00 - 00 - 00 - 0B -
00 - 00 - 00 - 00 - 01 - 11 - 80 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 01 - 00 - 0A - 00 - 00 - 08 -
00 - 01 - 30 - 80 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 - 00 -
00 - 00 - 00 - 00 - 00 - 00 - 01 - 00 - 07 - 00 - 03 - 00 - 30 - 01 - 02 - 00 -
00 - 00 - 08 - 00 - 03 - 00 -
Untrusted master found!!        Packet MAC is 00-00-80-51-9F-CC
**Summary:This Packet contains untrusted master/slave
```

(f)

```
snort-2.9.8.2/etc/log) - gedit
alert ×
[**] [146:2:1] (spp_ecat):  Master address does not match with ESI file
UNTRUSTED MASTER !!!  [**]
02/14-06:52:06.567638

[**] [146:1:1] (spp_ecat):  Data injection !!  [**]
02/14-06:52:06.577728

[**] [146:1:1] (spp_ecat):  Slave address does not match with ESI file
UNTRUSTED SLAVE !!!  [**]
02/14-06:52:06.587733
```

Figure 13: ECAT preprocessor, results, and log.

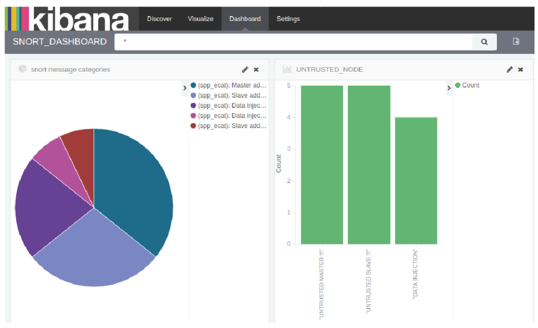


Figure 14: ELK dashboard.

This vulnerability leads to a MAC spoofing attack. Moreover, there is no mechanism to recognize the master for a slave or to prevent a brute-force network slave scan. If the slave-address map or address hierarchy of the EtherCAT network can be predicted, other memory addresses of the slave or other slave addresses can be accessed. In addition, there is no security check for data-size variations, which could even cause physical damage.

Regarding the analysis results, it is determined that the EtherCAT protocol requires a security mechanism to prevent attacks coming from the Ethernet infrastructure.

Since there is no preprocessor on Snort for detecting EtherCAT packets, our proposed ECAT preprocessor provides basic protection over the EtherCAT-applied critical systems. It works in passive sniffing mode so that we do not interrupt the real-time communication or send packets into the network. The ECAT preprocessor is loaded during Snort initialization and catches every EtherCAT packet Snort receives. Since Snort supports other industrial automation system protocols, such as Modbus or DNP3, it will also provide the requested real-time feature of the EtherCAT.

We have extended the Snort features to handle EtherCAT protocol packets by our *DecodeECAT* contribution into the Snort *decoder engine*. This way, contributions, such as rule addition or new prevention methods, can be easily adopted by defining related functions of the proposal into the ECAT preprocessor. One rule is created in the preprocessor rule file, which uses the trust-node communication technique.

This approach exports master hardware addresses, slave, logical, incremented, or physical addresses, and commands, order of commands, and data sizes in ENI files. By checking these data, it detects MAC spoofing, data injection, and slave-address access attacks to and from the addresses that are not defined in the ENI.xml network configuration files. In fact, the XML configuration files also exist in other ICS protocols, such as PROFINET or Sercos. Thus, the introduced method can be applied for other protocols and preprocessors in ICS or be globalized by giving an overall solution for critical systems.

Another point is that some of the real-time Ethernet protocols execute over in layer 2. However, Snort can handle layer 3, layer 4, or application layer protocols. Our improvement presents how preprocessors of protocols executing at layer 2 can be developed.

The ICS systems are dedicated systems. Thus, configurations are made by the engineering station in advance. In this configuration, existing hardware has individual ID information and master devices that control the system and compute all processes regarding these ID data. The ID data are saved as GSD, ESI, ENI, and other formats depending on the protocol. The given trust-node approach is applicable to other protocols/systems by examining the file formats and principles of operations of the systems.

In our research, since packet processing or overloading the system in runtime causes serious problems in ICS, the passive monitoring technique is investigated and applied on a real-time protocol using Snort. Performing penetration tests on running ICS is not recommended, and this obstructs the detection of vulnerabilities. Using the passive monitoring approach, threats and risks caused by the vulnerabilities can be reduced by defining rules.

7. Future Work

There are several directions for further research. First, we see this work as a foundation and encourage the vulnerability analysis and prevention proposals of the EtherCAT protocol to increase industrial automation system security. For instance, EtherCAT obviously needs a lightweight encryption-based security mechanism for secure communication like other industrial protocols. This could be a further study over the EtherCAT network. Second, we need to improve the preprocessor to prevent factory and IP-based communication attacks. Therefore, the EtherCAT protocol requires more vulnerability analyses at all communication levels. However, we believe that our analysis shows the fundamental vulnerabilities, and the proposal is a novel approach that provides an efficient means for critical systems.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the Scientific Research Projects Unit of the Sakarya University (Grant no. 2015-50-02-025).

References

[1] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Computers in Industry*, vol. 81, pp. 11–25, 2016.

[2] D. Kang, B. Kim, J. Na, and K. Jhang, "Whitelists based multiple filtering techniques in SCADA sensor networks," *Journal of Applied Mathematics*, vol. 2014, Article ID 597697, 7 pages, 2014.

[3] J. Xu and D. Feng, "Identification of ICS Security Risks toward the Analysis of Packet Interaction Characteristics Using State Sequence Matching Based on SF-FSM," *Security and Communication Networks*, vol. 2017, Article ID 2430835, 17 pages, 2017.

[4] E. J. Byres, M. Franz, and D. Miller, "The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems," in *Proceedings of the International Infrastructure Survivability Workshop (IISW'04)*, 2004.

[5] D. Security, "Dell security annual threat report 1," Tech. Rep., 2015.

[6] P. Kevin, "Slammer worm crashed Ohio nuke plant," *SecurityFocus*, 2016, http://www.securityfocus.com/news/6767.

[7] S. Tony, "Hacker Jailed for Revenge Sewage Attacks," 2001, http://www.theregister.co.uk/2001/10/31/hacker_jailed_for_revenge_sewage/.

[8] M. Abrams and J. Weiss, *Control System Cyber Security Case Study*, Bellingham, Washington, DC, USA, 2007.

[9] National Transportation Safety Board, *Safety study: supervisory control and data acquisition (SCADA) in liquid pipelines*, 2005.

[10] J. Weiss, "A review of selected actual control system cyber incidents," in *Proceedings of the in ICSJWG 2009 Fall Conference*, 2009.

[11] B. Miller and D. C. Rowe, "A survey of SCADA and critical infrastructure incidents," in *Proceedings of the 1st Annual Conference on Research in Information Technology, RIIT 2012*, pp. 51–56, ACM, October 2012.

[12] E_ISAC, *Analysis of the cyber attack on the Ukrainian power grid*, 2016.

[13] FireEye iSight Intelligence, *Overload critical lessons from 15 years of ICS vulnerabilities*, 2016.

[14] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 277–293, 2013.

[15] EtherCAT Technology Group, *Industrial Ethernet Technologies: Overview*, 2014.

[16] G. Prytz, "A performance analysis of EtherCAT and PROFINET IRT," in *Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2008*, pp. 408–415, September 2008.

[17] KingStar, *White paper: 5 Real-Time, Ethernet-Based Fieldbuses Compared Which Standard Stands Apart?*, 2016.

[18] Beckhoff, *Moving up to Industrial Ethernet: The EtherCAT protocol*, 2008.

[19] W. T. Shaw, *Cybersecurity for SCADA Systems*, PennWell Corp., Tulsa, Okla, USA, 2006.

[20] A. Kleinmann and A. Wool, "A statechart-based anomaly detection model for multi-threaded SCADA systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 9578, pp. 132–144, 2016.

[21] A. Basaran, *Siber Savas Cephesinden Notlar*, 2012, http://securitist.blogspot.com.tr/2012/04/guvenlik-101.html.

[22] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Internet-facing PLCs as a network backdoor," in *Proceedings of the 3rd IEEE International Conference on Communications and Network Security, CNS 2015*, pp. 524–532, September 2015.

[23] J. Wang, L. C. K. Hui, S. M. Yiu, G. Zhou, and R. Zhang, "F-DDIA: A framework for detecting data injection attacks in nonlinear cyber-physical systems," *Security and Communication Networks*, vol. 2017, Article ID 9602357, 12 pages, 2017.

[24] D. Beresford, "Exploiting Siemens Simatic S7 PLCs," in *Proceedings of the Black Hat USA*, pp. 1–26, 2011.

[25] Siemens CERT, *Security Vulnerabilities in Siemens SIMATIC S7-1200 CPU*, 2011.

[26] Siemens CERT, *Web Vulnerability in SIMATIC S7-1200 CPU*, 2015.

[27] Sans Institute-Andrew Hildick-Smit, *Security for Critical Infrastructure SCADA Systems*, 2005.

[28] A. Timorin, "SCADA Deep Inside: Protocols and Security Mechanisms," in *Proceedings of the Balkan Computer Congress*, 2014.

[29] NIST-National Vulnerability Database, *CVE-2014-2252 Detail*, 2014.

[30] D. K. Sadhasivan and K. Balasubramanian, "A fusion of multiagent functionalities for effective intrusion detection system," *Security and Communication Networks*, vol. 2017, Article ID 6216078, 15 pages, 2017.

[31] I. Ismail, S. Mohd Nor, and M. N. Marsono, "Stateless Malware Packet Detection by Incorporating Naive Bayes with Known Malware Signatures," *Applied Computational Intelligence and Soft Computing*, vol. 2014, Article ID 197961, 8 pages, 2014.

[32] S. Ntalampiras, Y. Soupionis, and G. Giannopoulos, "A fault diagnosis system for interdependent critical infrastructures based on HMMs," *Reliability Engineering & System Safety*, vol. 138, pp. 73–81, 2015.

[33] N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63–75, 2013.

[34] A. Kleinmann and A. Wool, "Accurate Modeling of the Siemens S7 SCADA Protocol for Intrusion Detection and Digital Forensics," *Journal of Digital Forensics, Security and Law*, vol. 9, no. 2, pp. 37–50, 2014.

[35] A. Cook, A. Nicholson, H. Janicke, L. Maglaras, and R. Smith, "Attribution of Cyber Attacks on Industrial Control Systems," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 3, no. 7, p. 151158, 2016.

[36] K. F. Alotaibi, M. M. Hamidi, M. Talebi, J. Xu, and A. Homaifar, "Using spy node to identify cyber-attack in power systems as a novel approach," in *Proceedings of the IEEE International Conference on Electro/Information Technology, EIT 2015*, pp. 581–586, May 2015.

[37] R. S. Ramachandruni and P. Poornachandran, "Detecting the network attack vectors on SCADA systems," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015*, pp. 707–712, August 2015.

[38] J. Sheng, S. Chung, L. Hansel et al., "JAUS to EtherCAT bridge: Toward real-time and deterministic joint architecture for unmanned systems," *Journal of Control Science and Engineering*, vol. 2014, Article ID 631487, 20 pages, 2014.

[39] E. Knapp, *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*, Syngress, Waltham, MA, USA, 2011.

[40] W. Lei and Q. Junyan, "The real-time networked data gathering systems based on EtherCAT," in *Proceedings of the 2009 International Conference on Environmental Science and Information Application Technology, ESIAT 2009*, vol. 3, pp. 513–515, July 2009.

[41] T. Feng, Q. Li, G. Ren et al., "The implementation of distributed high-speed high-accuracy data acquisition system based on EtherCAT," in *Proceedings of the 2013 IEEE 8th Conference on Industrial Electronics and Applications, ICIEA 2013*, pp. 1649–1653, June 2013.

[42] J. Qi, L. Wang, H. Jia, and B. Yang, "Design and performance evaluation of networked data acquisition systems based on EtherCAT," in *Proceedings of the 2010 2nd IEEE International Conference on Information Management and Engineering, ICIME 2010*, pp. 467–469, April 2010.

[43] M. Knezic, B. Dokic, and Z. Ivanovic, "Increasing EtherCAT performance using frame size optimization algorithm," in *Proceedings of the 2011 IEEE 16th Conference on Emerging Technologies and Factory Automation, ETFA 2011*, pp. 1–4, September 2011.

[44] M. Knezic and Z. Ivanovic, "Evaluation of Ethernet over EtherCAT Protocol Efficiency," in *Infoteh-Jahorina*, 2013.

[45] G. Kalman and D. Orfanus, "Measuring latencies over industrial Ethernet switches," in *Proceedings of the 2013 21st Telecommunications Forum Telfor, TELFOR 2013*, pp. 365–368, November 2013.

[46] BeStorm, *Dynamic Testing (Fuzzing) on the Ethercat Protocol by beSTORM*, 2015, http://www.beyondsecurity.com/dynamic_fuzzing_testing_ethercat_protocol.

[47] A. Granat, H. Höfken, and M. Schuba, "Intrusion Detection of the ICS Protocol EtherCAT," in *Proceedings of the 2nd International Conference on Computer, Network Security and Communication Engineering*, pp. 113–117, 2017.

[48] EtherCAT, *EtherCAT-the Ethernet fieldbus*, 2012.

[49] T. Aven, "A unified framework for risk and vulnerability analysis covering both safety and security," *Reliability Engineering & System Safety*, vol. 92, no. 6, pp. 745–754, 2007.