2017 2nd International Conference on Computer, Network Security and Communication Engineering (CNSCE 2017)
ISBN: 978-1-60595-439-4

Intrusion Detection of the ICS Protocol EtherCAT

Andreas GRANAT, Hans HÖFKEN and Marko SCHUBA*

FH Aachen, University of Applied Sciences, Eupenerstr. 70, Aachen, Germany

*Corresponding author

Keywords: ICS, SCADA, Protocol, Security, Intrusion detection, EtherCAT, Snort.

Abstract. Control mechanisms like Industrial Controls Systems (ICS) and its subgroup SCADA (Supervisory Control and Data Acquisition) are a prerequisite to automate industrial processes. While protection of ICS on process management level is relatively straightforward—well known office IT security mechanisms can be used—protection on field bus level is harder to achieve as there are real-time and production requirements like 24x7 to consider. One option to improve security on field bus level is to introduce controls that help to detect and to react on attacks. This paper introduces an initial set of intrusion detection mechanisms for the field bus protocol EtherCAT. To this end existing Ethernet attack vectors including packet injection and man-in-the-middle attacks are tested in an EtherCAT environment, where they could interrupt the EtherCAT network and may even cause physical damage. Based on the signatures of such attacks, a preprocessor and new rule options are defined for the open source intrusion detection system Snort demonstrating the general feasibility of intrusion detection on field bus level.

Introduction

During the last years malware like Stuxnet or Duqu has demonstrated that ICS or SCADA systems can be directly attacked [11, 10]. The increasing interconnection of ICS (in Germany often referred to as “Industry 4.0”) aggravates the situation as many automation systems have an indirect - often even a direct - Internet connection [9, 15]. Via this connection these systems, which were believed to be secure, can be directly attacked. Attacks can target the computer systems or embedded devices of the automation system or the field bus. There are various types and implementations of all different kinds of field bus protocols, such as EtherCAT [7], Profinet [4], or DNP3 [6]. Most of these protocols lack security mechanisms like authentication or integrity checks. It would therefore be beneficial to spot attacks on field level to mitigate for the weak security.

This paper describes the possibilities to detect attacks aimed at the real-time Ethernet field bus protocol EtherCAT. In a first step potential attack vectors are identified. To detect those attacks the open source intrusion detection system (IDS) Snort [1] has been extended with a set of preprocessors for EtherCAT.

EtherCAT

EtherCAT [7] is a so called real-time Ethernet, which can be deployed as a field bus in automation technology. EtherCAT transports process data in an Ethernet frame to the individual components of the automation system. EtherCAT is used to connect the individual components such as ICS, actuators and sensors. In an automation system a reaction has to follow an action in real-time. If for example a sensor registers an entry, the system has to react to this action accordingly. For this reason EtherCAT and the network layer underneath respectively have to be capable to run in real-time. The network layer underneath is basically Ethernet, which does not conform to the real-time capability as standard. In order to achieve real-time, EtherCAT works with a clock cycle of 100 µs and a limited jitter. Furthermore, the transmission of a frame can only be initiated by the Ether- CAT master, which ensures that time boundaries are met. A single Ethernet frame can transport several EtherCAT datagrams. As a result the protocol overhead, which is usually generated through one-to-one

addressing, is omitted. In addition EtherCAT's design allows slaves to read and write frames on-the-fly, i.e., there is no store-and-forward delay in the slaves. As already mentioned, EtherCAT uses Ethernet as link layer protocol. EtherCAT Ethernet frame. The source and destination fields are filled with addresses of the sending and receiving devices, respectively. The EtherType for EtherCAT is 0x88A4. The EtherCAT data field is subdivided into an EtherCAT frame header and a certain number of EtherCAT datagrams. The frame header includes the length of the EtherCAT frame and the EtherCAT version (version 0x0001 in the context of this paper). An EtherCAT datagram consists of the datagram header, datagram data field and a working counter. From an incident detection perspective essential header fields include: CMD: EtherCAT command, which is executed on the slave. Address: 32 bit with logical addressing, remaining addressing devided into 16 bit offset and 16 bit participant address. The data field includes EtherCAT application data written or read by the addressed slave. The working counter is used to indicate that the datagram has been processed by a slave. Table 1 shows different commands used by EtherCAT.

Table 1. Example EtherCAT Commands.

Command	Description	Command	Description
APRD	Auto-Increment read	BRD	Broadcast read
APWR	Auto-Increment write	BWR	Broadcast write
APRW	Auto-Increment read and write	BRW	Broadcast read and write
FPRD	Fixed read	LRD	Logical read
FPWR	Fixed write	LWR	Logical write
FPRW	Fixed read and write	LRW	Logical read and write

Figure 1 depicts a Wireshark screenshot with the respective fields highlighted.

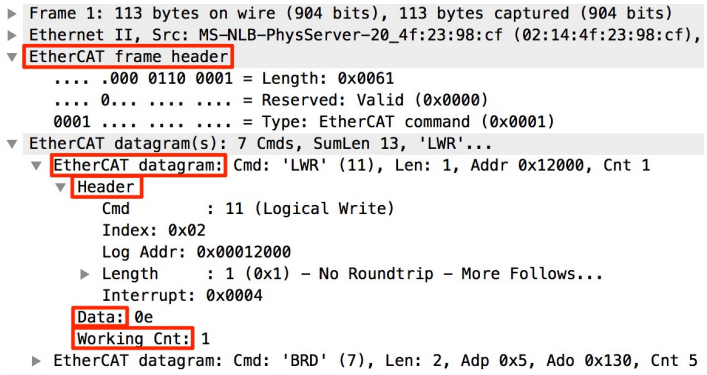

Figure 1. Wireshark screenshot of an EtherCAT frame.

Attacks on EtherCAT

As a matter of principle EtherCAT is vulnerable to all known attacks against Ethernet. Attacks like MAC address spoofing [8] might not directly cause damage but could disturb EtherCAT's real-time capability or even the frictionless operation of the system and its industrial process. More direct attacks, such as packet injection [3] or man-in-the-middle attacks [3], which have the potential to directly manipulate the automation system, could cause much more damage and could have a direct safety impact. Performing those attacks, e.g., data of a temperature sensor could be manipulated, which could lead to the destruction of component parts by overheating, or, in another example, a valve could remain closed despite too high pressure which could cause pipes or tanks to burst. Ethernet attacks require access to the EtherCAT network on layer two, e.g., by connecting to a respective switch. Executing the attacks is quite simple, particularly, as attack/pentesting tools are freely available in the Internet, e.g., the Kali Linux distribution [13].

MAC-Address-Spoofing A MAC address spoofing attack is performed by trying to counterfeit the MAC address of an authorized member of the network. In the case of EtherCAT the attacker has to

adopt the identity of the EtherCAT master and use its MAC address. This makes it possible for the attacker to distribute EtherCAT frames in the name of the ICS. The MAC address can be altered by Windows or Linux system tools [8].

Replay-Attack A replay attack sends a previously recorded (valid) frame once more into an EtherCAT network. The recorded frame can come from the same or of a different sending system. Such attacks can force system slaves to execute duplicate or older commands which might set it into an unwanted state. The fact that there is no proper authentication of frames makes this kind of attack possible. A replay attack can be easily implemented by the program tcpreplay [8, 2].

Packet-Injection A packet injection attack sends manipulated EtherCAT frames into an EtherCAT network. As there is no authentication and no integrity check, an attacker might either replay a slightly manipulated frame or generate a completely new frame. These frames are able cause directed damage to an automation system. Depending on the state of knowledge of the system, this damage has the power from deranging the real-time capability of the system to even causing physical damage. Potential impacts of EtherCAT packet injection attacks include: (1) Manipulation of I/O operation. (2) Manipulation of the Fieldbus Memory Management Unit (FMMU) configuration. (3) Manipulation of measured data.

In the context of this work, a program to perform a packet injection attack called ”ecat injection” was developed, which can be invoked with various parameters as shown below.

```
ecat_injection eth0 00:01:05:23:01:2e 01:01:05:01:00:00 \
   lwr 0x01000000 0xff 0 10000
```

Figure 2 shows the I/O block related to the above program call. The (arbitrarily) manipulated 0xff payload value sets all outputs of the EL2008 component.

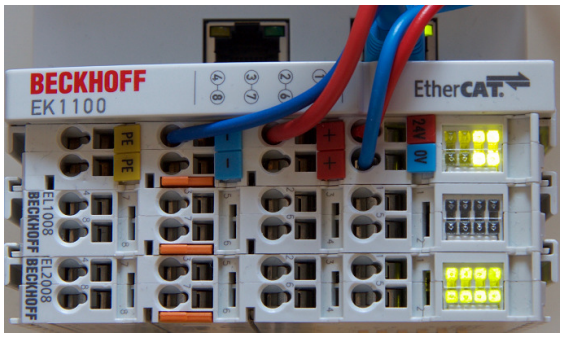

Figure 2. Wireshark screenshot of an EtherCAT frame.

Man-in-the-middle A man-in-the-middle attack on EtherCAT adds interception of original frames to the packet injection possibility. As EtherCAT datagrams are directly encapsulated in Ethernet frames, well known man-in-the-middle attacks like ARP spoofing do not work. Instead, the attacker requires access to the bus in some way in order to get access to the transmitted frames. The access may occur either physically (cable) or through an already existing participant of the EtherCAT communication. To perform such attack, a program ”ecat mitm” has been developed. In the example below the program call replaces the data 0x01 with 0x02 for an LWR EtherCAT datagram

```
ecat_mitm "00:01:05:23:01:2e" "lwr;data,0x01->0x02"
```

EtherCAT Intrusion Detection with Snort

Intrusion detection systems detect unauthorized actions inside a computer network and – if they support such a feature – react on them (intrusion detection and prevention systems, IDPS) [12]. A number of IDS/IDPS systems exist on the market [1, 5, 14], however, none of these solutions supports intrusion detection in EtherCAT.

Intrusion detection with Snort Snort is a so-called Network-IDS, which scans the network based on signatures or protocols for attacks or intrusion. In doing so, it can be used as a detection or

prevention system and be operated in three different ways. Depending on the position in the network, Snort captures network packages, processes and analyses them and generates an alarm signal where required. By default Snort already masters various protocols, but not EtherCAT.

Snort and EtherCAT When handling EtherCAT frames, Snort stops its analysis after decoding the Ethernet frame. This is due to the fact that Snort is very limited to process frame contents other than IP and related protocols. However, Snort offers the possibility to expand the functional range through so-called preprocessors. In the context of this paper the functional range of Snort was expanded to cover the EtherCAT protocol as well. After decoding the Ethernet frame, an EtherCAT preprocessor takes over. Its task is to decode and standardize an EtherCAT frame or datagram, respectively, and to detect potential attacks.

Attack detection using an EtherCAT preprocessor For the decoding of the Ethernet frame payload, the Snort rule base has been extended with EtherCAT specific rule options: (1) ecat: Detection related to the datagram header, (2) ecat_data: Analysis of the datagram payload field, (3) ecat_fmmu: Detection of Fieldbus Memory Management Unit (FMMU) manipulations, and (4) ecat_count: Frame count as a threshold for triggering alerts.

In addition, the existing Snort preprocessor has been expanded to provide the ability to define a global frame count and to perform checks on the MAC address of the EtherCAT master, as this can be used detect unauthorized frames inside the EtherCAT network.

Detecting arbitrary intrusions is difficult and was (not yet) part of the project. Therefore, the simplifying assumption was made that the valid behavior of the automation system is known to the developer. Or in other words, the developer knows what attack patterns to look for in datagram headers, datagram payload fields or related to frame counts. In that case, the rule options mentioned above can be used as follows.

Rule Option ecat To check the EtherCAT-datagram header, the rule option ecat can be used. The following arguments are available: (1) EtherCAT-command, (2) Address-field: log or adp and ado, and (3) Working-Counter. Example rules:

```
alert (msg:"EtherCAT Output 0xFF!"; \
   ecat:lwr,log=0x01000000,wc=1; gid:256; sid:1000000;))
```

Detects all EtherCA-datagrams, which include a logical write command (lwr), working-counter is equal to one and the logical address is 0x01000000.

```
alert (msg:"EtherCAT Output 0xFF!"; \
   ecat:ecat:aprd,adp=0x0005,ado=0x0130; gid:256; sid:1000000;))
```

Detects all EtherCAT datagrams, which include an auto-increment-read command (aprd), address (adp) is equal to 0x0005 and the address offset (ado) is 0x0130.

Rule Option ecat_data The datagram payload can be checked by the rule option ecat_data. Arguments can be given in hex or binary format: 0x (hex), bx (bin) Also the argument can be configured to be equal, greater or less than the value which the frame contains. Example rules:

```
alert (msg:"ECAT Data Rule!"; ecat_data:0x0e; gid:256; sid:7;)
```

Triggers an alert if the datagram payload is equal to 0x0e.

```
alert (msg:"ECAT Data Rule!"; ecat_data:<0x09; gid:256; sid:10;)
```

Triggers an alert if the datagram payload is less than 0x09.

Rule Option ecat_fmmu To check the FMMU configuration the ecat_fmmu option can be used. This rule-option expects all relevant FMMU fields as argument: (1) Log Start: lstart, (2) Log Length: llen, (3) Log StartBit: lsbit, (4) Log EndBit: lebit, (5) Phys Start: pstart, and (6) Phys StartBit: psbit. Example rule:

```
alert alert (msg:"ECAT FMMU Match!"; ecat_fmmu:lstart=0x01000800, \
   llen=0x0001,lsbit=0x00,lebit=0x07, pstart=0x1000,psbit=0x00; \
   gid:256;sid:8;))
```

Summary

When it comes to the development of ICS field bus protocols, the implementation of security features used to play a minor role. Once an attacker manages to gain access to the field bus, they can transmit, manipulate and block EtherCAT frames at will. On the basis of missing security features like authentication or integrity, it is very simple to attack an automation system on the field bus level. Well known attacks like packet injection or man-in-the-middle can be used to interrupt an EtherCAT network, which could lead to major damage. Detection of such attacks could be achieved using a field bus specific intrusion detection and prevention system. The EtherCAT Snort preprocessor is a first step into that direction, as it allows for analyzing frames on EtherCAT level, thus forming a basis to develop more comprehensive rule sets that allow more complicated attacks to be detected.

References

[1] Cisco and/or its affiliates, Snort. 2016, https://snort.org

[2] Fred Klassen AppNeta, Tcpreplay - Pcap editing and replaying utilities, 2016, http://tcpreplay.appneta.com

[3] Sherri Davidoff and Jonathan Ham, Network Forensics - Tracking Hackers through Cyberspace, 1st ed. Prentice Hall, 2012.

[4] PROFIBUS user organization, PROFINET - Machine Building Animation, 2016, http://www.profibus.com/technology/profinet/

[5] Open Information Security Foundation, Suricata, 2016, https://oisf.net/suricata/

[6] DNP Users Group. Distributed Network Protocol, 2016, http://www.dnp.org/Default.aspx

[7] EtherCAT Technology Group, EtherCAT - The Ethernet Fieldbus, 2016, https://www.ethercat.org/ default.htm

[8] Martin Kappes. Network and Data Security (in German), 2nd ed. Springer Vieweg, 2013.

[9] Eric D. Knapp and Joel Thomas Langill, Industrial Network Security, 2nd ed. Syngress, 2015.

[10] Joel Langill, Duqu Reference Material, 2016, https://scadahacker.com/resources/duqu.html

[11] Joel Langill, Stuxnet Reference Material, 2016, https://scadahacker.com/resources/stuxnet.html

[12] Karen Scarfone and Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS), Tech. rep. NIST National Institute of Standards and Technology, 2016, http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=50951

[13] Offensive Security, The Kali Linux Distribution, 2016, https://www.kali.org

[14] Quadrant Information Security. The Sagan Log Analysis Engine, https://quadrantsec.com/sagan_log_analysis_engine/

[15] Shodan, Shodan search engine, 2016, https://www.shodan.io