Information Hiding in Industrial Control Systems: An OPC UA based Supply Chain Attack and its Detection

Mario Hildebrandt
Kevin Lamshöft
Jana Dittmann
mario.hildebrandt@iti.cs.uni-magdeburg.de
kevin.lamshoeft@ovgu.de
jana.dittmann@iti.cs.uni-magdeburg.de
Research Group Multimedia and Security,
Otto-von-Guericke-University
Magdeburg, Germany

Tom Neubert
Claus Vielhauer
tom.neubert@th-brandenburg.de
claus.vielhauer@th-brandenburg.de
Brandenburg University of Applied Sciences
Brandenburg an der Havel, Germany

ABSTRACT

Industrial Control Systems (ICS) help to automate various cyber-physical systems in our world. The controlled processes range from rather simple traffic lights and elevators to complex networks of ICS in car manufacturing or controlling nuclear power plants. With the advent of industrial Ethernet ICS are increasingly connected to networks of Information Technology (IT). Thus, novel attack vectors on ICS are possible. In IT networks information hiding and steganography is increasingly used in advanced persistent threats to conceal the infection of the systems allowing the attacker to retain control over the compromised networks. In parallel ICS are more and more a target for attacks as well. Here, simple automated attacks as well as targeted attacks of nation state actors with the intention of damaging components or infrastructures as a part of cyber crime have already been observed. Information hiding could bring such attacks to a new level by integrating backdoors and hidden/covert communication channels that allow for attacking specific processes whenever it is deemed necessary. This paper sheds light on potential attack vectors on Programmable Logic Controllers (PLCs) using OPC Unified Architecture (OPC UA) network protocol based communication. We implement an exemplary supply chain attack consisting of an OPC UA server (Bob, B) and a Siemens S7-1500 PLC as OPC UA client (Alice, A). The hidden storage channel is using source timestamps to embed encrypted control sequences allowing for setting digital outputs to arbitrary values. The attack is solely relying on the programming of the PLC and does not require firmware level access. Due to the potential harm to life caused by attacks on cyber-physical systems any presentation of novel attack vectors need to present suitable mitigation strategies. Thus, we investigate potential approaches for the detection of the hidden storage channel for a warden W as well as potential countermeasures in order to increase the warden-compliance. Our machine learning based detection approach using a One-Class-Classifier yields a detection performance of 89.5% with zero false positives within an experiment with 46,159 OPC UA read responses without a steganographic message and 7,588 OPC UA read responses with an embedded steganographic message.

CCS CONCEPTS

• **Security and privacy** → *Intrusion detection systems*; **Network security**; • **Theory of computation** → Cryptographic protocols; • **Applied computing** → *Command and control*; • **Social and professional topics** → *Automation*.

KEYWORDS

Information Hiding; Steganography; Process Automation; Industrial Control Systems; OPC UA

ACM Reference Format:
Mario Hildebrandt, Kevin Lamshöft, Jana Dittmann, Tom Neubert, and Claus Vielhauer. 2020. Information Hiding in Industrial Control Systems: An OPC UA based Supply Chain Attack and its Detection. In *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '20), June 22–24, 2020, Denver, CO, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3369412.3395068

1 INTRODUCTION

Industrial Control Systems (ICS) play a vital role in our current economy – manufacturing, energy production and distribution or traffic control usually employ automation techniques. Especially with the advent of Ethernet connectivity and TCP/IP based protocols the lines between such operational technology (OT) and information technology (IT) are increasingly blurred. Currently only a few manufactures dominate the market of automation systems in general and programmable logic computers (PLCs) in particular.

Due to the similarities of the connectivity between IT and OT similar threats and developments can be observed. However, specific trends from the IT domain can be observed with some delay in the OT domain. A particular trend in IT networks is an increasing utilization of information hiding techniques to avoid a detection of the attack on a particular system. This trend cannot be observed for OT networks, yet. However, within the last decade several directed attacks on OT systems have gained some attention, e.g. Stuxnet [3] (also known as a part of Operation Olympic Games), Ukraine power grid attack [9] or even the attack on special safety systems [2]. All those attacks show quite a high level of sophistication and are presumably performed by nation state actors.

IH&MMSec '20, June 22–24, 2020, Denver, CO, USA

ACM ISBN 978-1-4503-7050-9/20/06.
https://doi.org/10.1145/3369412.3395068

In many cases such components exchange information based on proprietary protocols. However, in order to allow for combining OT components from different manufacturers, more and more standard protocols such as Modbus/TCP [1] or OPC UA [10] are utilized in those networks. Standard protocols are usually well documented and contain no security mechanisms which follow the principle of security by obscurity. Thus, within the scope of a supply chain attack, with already compromised components being installed, an attacker could easily modify the communication in order to embed additional messages using information hiding techniques. In this paper we introduce an exemplary XOR-encrypted steganographic (stego) channel based on the OPC UA protocol to send hidden commands from an OPC UA server to a PLC. This resembles a typical ICS environment, where setpoints for PLCs as well as the general exchange of data is handled via specific servers. We use an exemplary setup with a Siemens S7-1500 PLC (Alice **A**) acting as an OPC UA client initiating the connection to OPC UA server (Bob **B**). Usually a firewall is placed between **A** and **B** in order to limit the communication flow. **A** reads specific inputs such as variables or setpoints from **B** using OPC UA read responses. Afterward, **B** sends the data using OPC UA read responses to **A**. A warden **W** (see [7]), e.g. a network intrusion detection system (NIDS), might observe the communication. Thus, hiding the command injection to the PLC **A** is a reasonable objective within an advanced persistent threat. We investigate the warden compliance by analyzing the probabilities of the digits used for embedding. As a result, we add an additional obfuscation to the stego message to minimize the detection risk for the covert channel.

In addition to the message encryption and embedding scheme, we introduce a detection approach and explore optimal windows sizes for the detector. The paper is structured as follows: A brief overview of hidden communication in ICS protocols and the OPC UA protocol is given in Section 2. Section 3 introduces our concept for a keyed stego channel based on source timestamps in OPC UA read responses. The detection of the stego communication is discussed in Section 4. Finally, the paper is concluded in Section 5.

2 STATE-OF-THE-ART

In this section we provide a brief overview of potential hidden communication channels in network communication in general and industrial control system protocols in particular. Afterward, we summarize the OPC UA protocol as an example of a manufacturer-independent TCP/IP-based protocol for automation systems.

2.1 Potential Hidden Communication in ICS Protocols

Steganography in network communication is more constrained in comparison to multi-media data as cover data. In the first place the amount of available data is significantly smaller specified by the maximum transmission unit (MTU). For ordinary Ethernet usually the MTU is 1500 bytes and in addition to that, 18 bytes for the headers. However, in ICS communication the network packets are usually much smaller than the MTU because only a few values and some meta-data are transmitted. In Table 1 potential hidden channels in network communication in general and in ICS communication for the example of Modbus/TCP are compared with each other. In addition to that a brief estimation of the capacity and the conspiciousness is performed by [8]. Especially the conspicuousness is an important factor towards a potential detection of the hidden communication. From the perspective of the attacker, ideally a warden observing the communication, i.e. an Intrusion Detection System, would be unable to differentiate between genuine network traffic and cover data with embedded hidden communication.

2.2 OPC UA

OPC UA [10] is a manufacturer independent communication protocol. OPC UA can be operated in three different modes. In the first mode a secure authentication is performed but the following messages are not protected regarding the integrity or authenticity. In the second mode additionally all messages are signed protecting the integrity and authenticity of the communicated information. In the third mode the packets are additionally encrypted to ensure the confidentiality of the information.

Within OPC UA data is organized within nodes, whereas each node has a specific data type. Besides the value of a node, additionally a source timestamp indication the particular data of a measurement can be transferred. Moreover, a value indicating the status of a node is transferred. The OPC UA header does also include sequence numbers and various flags.

OPC UA supports two different modes for data exchange: a traditional data access mode with read and write requests and a publish-subscribe mode where data is only transferred if a value has changed. In this paper we use the data access mode, in particular the read responses being sent from the OPC UA server to the PLC acting as an OPC UA client.

3 CONCEPT: OPC UA READ RESPONSES FOR INJECTING COMMANDS TO THE PLC

Before a hidden communication channel for command injection to a PLC can be designed, it is important to point out the peculiarities of PLCs in comparison to standard IT systems. A PLC is usually a cycle-driven embedded computer with limited resources and storage. During each cycle the PLC reads the state of the inputs and sets the defined outputs corresponding to its programming. In addition to that, data might be read from or sent to connected components. As the output ports control the actors of the process, a command injection likely targets on those outputs. For setting an output, there are basically two options during the programming of the PLC – an output can be set during each cycle or it could be set only if its state should be altered.

Requirements: from the perspective of a command injection to manipulate the process the latter option is more challenging because the full command needs to be embedded into each message in order to make sure that the command is executed in each cycle of the PLC. This also requires the usage of a storage channel because a timing channel would stretch the command over multiple cycles and additionally require some persistent storage within the PLC. However, this increases the amount of data which needs to be embedded and thus, increases the risk of the detection of the hidden channel. Nevertheless, we concentrate on this case because the less challenging case is covered by this approach as well.

Table 1: Comparison of network steganography and the applicability of the network information hiding storage and timing patterns (*simplified to T:Timing, S:Storage*) of [11] to Modbus/TCP [8]. The *Master*, *Slave* and *network* (element) column indicate protocol-compliance of the pattern (✓) to the specific type of device or not (✗). *Active*, *Passive* and *Hybrid* indicate the applicability of the information hiding approach. The *capacity* is a rough estimation based on theoretical considerations in a best-case-scenario. The *conspicuousness* (tendency of warden-compliance) is estimated on a binary scale (high/low). Storage channel utilized in this paper for OPC UA read responses highlighted by a gray background shading.

Patterns by [11]		Modbus/TCP [8]							
ID	Pattern	Master	Slave	Network	Active	Passive	Hybrid	Capacity	Conspicuousness
T1	Inter-Packet Times	✓	✓	✓	✓	✓	✓	1 bit/packet	low
T2	Message Timing	✓	✗	✗	✓	✗	semi-active	n bit/flow	high
T3	Rate/Throughput	✓	✗	✗	✓	✗	semi-active	n bit/flow	high
T4	Artificial Loss	✓	✓	✓	✓	✗	✓	1 bit/packet	high
T5	Message Ordering	✓	✓	✓	✓	✓	✓	1 bit/flow	low
T6	Retransmission	✓	✓	✓	✓	✓	✓	1 bit/time unit	high
T7	Frame Collisions	✗	✗	✗	✗	✗	✗	✗	✗
T8	Temperature	✓	✓	✓	✓	✓	✓	to be tested	to be tested
S1	Size Modulation	✓	✓	✓	✗	✓	✓	8 bit/packet	low
S2	Sequence Modulation	✓	✗	✓	✓	✓	✓	720 options/packet	high
S3	Add Redundancy	✓	✗	✓	✓	✓	✗	1 bit/packet	low
S4	Random Value	✗	✗	✗	✗	✗	✗	✗	✗
S5	Value Modulation	✗	✗	✗	✗	✗	✗	✗	✗
S6	Reserved/Unused	✓	✓	✓	✓	✓	✓	8 bit/packet	low
S7	Payload File Size Modulation	✓	✗	✓	✓	✓	✓	8 bit/packet	low
S8	User-data Corruption	✓	✓	✓	✓	✓	✓	16 bit/packet	high
S9	Modify Redundancy	✗	✗	✗	✗	✗	✗	✗	✗
S10	Value Modulation & Reserved/Unused	✗	✓	✓	✓	✓	✓	7 bit/packet	low

Objective: in our example the objective is the manipulation of arbitrary digital outputs of the PLC for influencing the process. The example we are using is the simulation of a breaker control for a power plant. The PLC reads the rotor speed of the generator, the grid frequency and the status of a setpoint for the breaker position periodically from an OPC UA server. If the setpoint of the breaker position is set to closed and the generator speed would generate electricity at a frequency matching the grid frequency, the breaker is closed by setting the first digital output to true and the second digital output to false. Otherwise, the second output will be set to true and the first one to false. An attacker with the intention of interrupting the power grid could try to manipulate the breaker state. Such an attack could be prepared by a supply chain attack in order to install a compromised component in the power plant.

Limitations: a supply chain attack describes an attack where the supplier of the equipment is delivering compromised components. In the ICS domain such a compromised component could either contain additional hardware implementing a hidden function or modified software. The software on PLCs consist of their firmware and the process-specific programming. In the case of Siemens PLCs, both software components have different access levels to the communication [6].

Concept: in our approach we establish the hidden communication based on the process-specific programming language. As a result, we can only access the value returned from the OPC UA server, the source time stamp and the OPC UA status code. In general hiding additional code in the process-specific programming is rather easy because the PLC could contain know-how-protected function blocks which contents are not readable[1].

Command Injection: the manipulation of arbitrary digital outputs of the PLC can be implemented using the function *POKE_BOOL*. The function expects a memory area, a byte offset, a bit offset and the target state as input parameters. In the S7-1500[2] PLC the memory area for the first digital output module is '*B#16#82*' which is kept constant in our experiment and thus, is not included in the stego message. The digital output is logically separated into four blocks of eight digital outputs each, whereas the byte offset contains the block ID and the bit offset the specific output port within this block. As a result a message for controlling arbitrary output ports is at least 6 bits long (2 bits for the byte offset, 3 bits for the bit offset and 1 bit for the target state). In addition to that, additional bits are necessary for plausibility and integrity checking of the stego message in order to avoid any unintended execution of the hidden functionality.

3.1 Analysis of Potential Locations for Message Embedding

With programming-level access three different values originating from the OPC UA read response are readable: value of the requested variable, corresponding OPC UA status code and source timestamp of the variable. The OPC UA status code is usually zero indicating a 'Good' reading, any deviating value is considered as an anomaly, e.g. a broken sensor, which requires further investigation. Thus, the status code is rather unsuitable for embedding additional information. For small amounts of data, the hidden message could be embedded in the value. However, if at least 6 bits need to be embedded into the value, the value would probably be out of a specified range, which would trigger an investigation. As a result, the only viable embedding location is the timestamp which is provided for each variable, representing a user-data corruption storage channel (S8 in Table 1). Each source timestamp is a 8 byte/64 bit value and thus, contains in theory enough space for embedding stego messages.

[1] https://support.industry.siemens.com/cs/document/10025431/how-can-you-install-block-protection-for-self-created-blocks-?dti=0&lc=en-AE

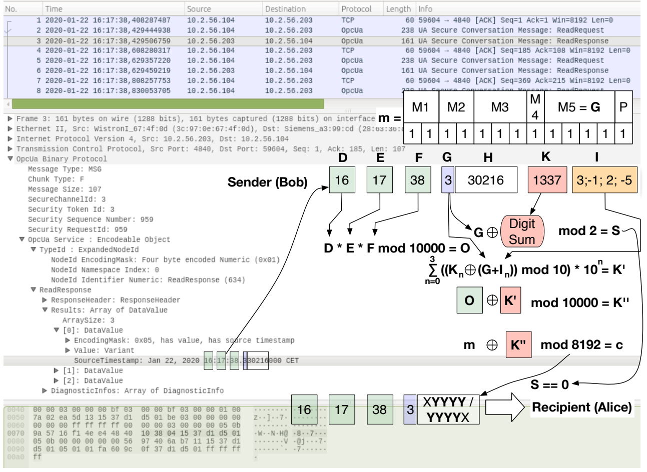
[2] https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-1500.html

3.2 Creation of a Steganographic Message for Command Injection to the PLC

Our proposed stego message m consists of six parts:

M1 : 2 bits preamble (fixed value 11_b),
M2 : 2 bits byte offset for the POKE_BOOL function on the PLC,
M3 : 3 bits bit offset for the POKE_BOOL function on the PLC,
M4 : 1 bit target state for the POKE_BOOL function on the PLC,
M5 : 4 bits 1/10s value of the source timestamp,
P : 1 parity bit.

Thus, in total each message m has a length of 13 bits with 6 bits containing the parameters for the command injection, 6 bits for plausibility checking of the message and 1 bit for integrity checking using a parity bit. Since $M2$ to $M4$ could have any arbitrary value in order to influence a specific output, it is necessary to integrate plausibility checking for the stego message. Otherwise, statistically in every second PLC cycle - even without a hidden message - a random action would be triggered. The correct parity bit P for the evaluation of the integrity of the message is going to be correct with a probability of 0.5. Thus, the additional fields for the evaluation of the authenticity of the message are necessary. As a result at least four digits of the source timestamp need to be used for the embedding of the message ($2^{13} = 8192$). With a cycle time of the PLC of 50ms and an OPC UA request every second cycle any of the timestamp digit after one-tenth of a second should not influence the process at all. However, with the nanosecond digits at zero within the OPC UA implementation Open62541[3], only five non-zero digits can be used for the embedding of the message without raising any attention as depicted in Figure 1. In order to minimize the risk of a randomly valid message m we suggest to use a fixed preamble $M1$ and a value which can be determined on the foundation of the timestamp $M5$. For $M5$ we use the value for the 1/10s field **G**. However, other values e.g. the modulus of digit sums of other parts of the timestamp or even the value of the queried variable could be used.



```
No.  Time                           Source       Destination  Protocol Length Info
   1 2020-01-22 16:17:38,408287487  10.2.56.104  10.2.56.203  TCP          60 59604 → 4840 [ACK] Seq=1 Ack=1 Win=8192 Len=0
   2 2020-01-22 16:17:38,429444938  10.2.56.104  10.2.56.203  OpcUa       238 UA Secure Conversation Message: ReadRequest
   3 2020-01-22 16:17:38,429506759  10.2.56.203  10.2.56.104  OpcUa       161 UA Secure Conversation Message: ReadResponse
   4 2020-01-22 16:17:38,608280317  10.2.56.104  10.2.56.203  TCP          60 59604 → 4840 [ACK] Seq=185 Ack=108 Win=8192 Len=0
   5 2020-01-22 16:17:38,629357220  10.2.56.104  10.2.56.203  OpcUa       238 UA Secure Conversation Message: ReadRequest
   6 2020-01-22 16:17:38,629459219  10.2.56.203  10.2.56.104  OpcUa       161 UA Secure Conversation Message: ReadResponse
   7 2020-01-22 16:17:38,808257753  10.2.56.104  10.2.56.203  TCP          60 59604 → 4840 [ACK] Seq=369 Ack=215 Win=8192 Len=0
   8 2020-01-22 16:17:38,830053705  10.2.56.104  10.2.56.203  OpcUa

▶ Frame 3: 161 bytes on wire (1288 bits), 161 bytes captured (1288 bits) on interface
▶ Ethernet II, Src: WistronI_67:4f:0d (3c:97:0e:67:4f:0d), Dst: Siemens_a3:99:cd (28:63:36:
▶ Internet Protocol Version 4, Src: 10.2.56.203, Dst: 10.2.56.104
▶ Transmission Control Protocol, Src Port: 4840, Dst Port: 59604, Seq: 1, Ack: 185, Len: 107
▼ OpcUa Binary Protocol
    Message Type: MSG
    Chunk Type: F
    Message Size: 107
    SecureChannelId: 3
    Security Token Id: 3
    Security Sequence Number: 959
    Security RequestId: 959
  ▼ OpcUa Service : Encodeable Object
    ▼ TypeId : ExpandedNodeId
        NodeId EncodingMask: Four byte encoded Numeric (0x01)
        NodeId Namespace Index: 0
        NodeId Identifier Numeric: ReadResponse (634)
    ▼ ReadResponse
      ▶ ResponseHeader: ResponseHeader
      ▼ Results: Array of DataValue
          ArraySize: 3
        ▼ [0]: DataValue
          ▶ EncodingMask: 0x05, has value, has source timestamp
          ▶ Value: Variant
            SourceTimestamp: Jan 22, 2020 16:17:38.330216000 CET
        ▶ [1]: DataValue
        ▶ [2]: DataValue
      ▶ DiagnosticInfos: Array of DiagnosticInfo

0040  00 00 03 00 00 00 bf 03  00 00 bf 03 00 00 01 00   ........
0050  7a 02 ea 5d 13 15 37 d1  d5 01 be 03 00 00 00 00   z··]··7·
0060  00 00 00 ff ff ff ff 00  00 00 03 00 00 00 05 0b   ········
0070  9a 57 16 f1 4e e4 48 40  10 38 04 15 37 d1 d5 01   ·W··N·H@ ·8··7···
0080  05 0b 00 00 00 00 00 56  97 40 6a b7 11 15 37 d1   ·······V ·@j···7·
0090  d5 01 05 01 01 fa 60 9c  0f 37 d1 d5 01 ff ff ff   ······`· ·7······
00a0  ff                                                 ·
```

Figure 1: Message encryption and embedding within the OPC UA source timestamp

In practical applications it is likely that only a limited number of digital output ports of the PLCs are supposed to be manipulated using the injected commands. Thus, the data fields $M1$ to $M4$ of m will likely have a low variance which results in a deviation of the distribution of digits within a source timestamp without a stego message. This would be also true for a message encrypted with a static key. In order to avoid static cipher messages c, the encryption key is derived from the pre-shared key K, the pre-shared initialization vector I and the digit for the 1/10s value of the timestamp **G**:

$$K' = \sum_{n=0}^{3}(((K_n \oplus (G + I_n)) \ mod \ 10) \cdot 10^n) \qquad (1)$$

In this equation K_n denotes the n-th digit of the pre-shared key K and I_n the n-th position of the initialization vector I, whereas $\oplus$ denotes a binary XOR operator. As the result the actually applied key is altered with every change of the digit **G** of the timestamp. In the case of an OPC UA request every 100ms two consecutive messages will not use the same key. The key K' is then used to encrypt the message m:

$$c = (m \oplus K') \ mod \ 8192 \qquad (2)$$

The modulus of 8192 is necessary to avoid values of c exceeding a length of four digits. In addition to that, the offset for the embedding position **S** is determined dynamically:

$$S = (G \oplus \sum_{n=0}^{3} K_n) \ mod \ 2 \qquad (3)$$

As a result, one of two potential embedding positions is determined for each message c. The digit not used for embedding of c remains unaltered (X in Figure 1), the other four digits (Y in Figure 1) are replaced with the digits of c.

In order to minimize the risk of an accidentally valid stego message, the execution of commands contained in the timestamps needs to be activated in the first place. For that, we suggest to use an unused digital output of the PLC as a binary switch. Since the non-assigned outputs are not altered by the PLCs programming, it is possible to use such an output as an almost invisible storage for a binary variable. The sole indicator of the activated steganography mode is an illuminated LED corresponding to the output port. Thus, in the attack at first this particular output needs to be activated, afterwards other outputs could be manipulated using the stego channel.

3.3 Analysis of the Warden Compliance of the Steganographic Channel

In order to achieve warden compliance, i.e. minimizing the risk of detection of the stego channel, the statistic behavior of the cover channel and the impact of the embedding of stego messages need to be analyzed. In contrast to steganography in media files, where an attacker can choose particularly suitable cover data, in network steganography the selection of potential cover channels is more limited and constrained by network protocols. For our example of the OPC UA source timestamps the variance of the probability of occurrence for each digit in **G** and **H** (**H** = {**H1**, **H2**, **H3**, **H4**, **H5**}) from Figure 1 is used as an indicator. We assume that the digits of the timestamp are uniformly distributed. However, due

[3] https://open62541.org/

to the cycle time of the process simulation for creating the values and corresponding source timestamps this is initially not the case for all digits as depicted in Figure 2 (NTS - natural timestamps from the simulation). In real automation scenarios intrinsic latency will probably result in a more homogeneous distribution of the digits for the decimal places. Ideally the variances are close to zero, indicating a probability of 0.1 for each digit of a decimal place. From Figure 2 we can learn that the decimal places of **H** are not uniformly distributed in *NTS*. Thus, in order to minimize the risk of detection of the stego channel within *ETS* the decimal places of **H** are replaced with uniformly distributed random digits. However, at least for **H4** and **H5** this approach does not yield the anticipated effect, yet. However, in a supply-chain attack scenario it is a realistic possibility that the cover channel is modified in such a manner in order to favor the embedding of stego messages. The embedding of the stego messages increases the variance of the digit occurrence probabilities to values around 0.01 in *STS* which differs by a magnitude of 10 in comparison to *NTS*. Thus, we suggest to use an additional obfuscation of the stego message as depicted in Figure 1. With Equation 2 a ciphertext c is repeated with an interval of one second if its content for $M2 - M4$ remains unaltered. In essence one specific message m results in ten potential ciphertexts c. In order to increase the variance of the cipher texts we suggest to use an additional obfuscation step in *O-STS* by introducing a higher variance to the XOR encryption key. In particular the obfuscation key **O** is determined by the product of the hour, minute and second values modulo 1000: $O = D \cdot E \cdot F \ mod \ 10000$. Then a key **K"** is calculated by $K'' = O \oplus K' \ mod \ 1000$. Finally the variance enhanced ciphertext c is determined by calculating $c = (m \oplus K'') \ mod \ 8192$. This additional obfuscation decreases the variance by a magnitude of 10 to 100 in comparison to *STS* as depicted in Figure 2.

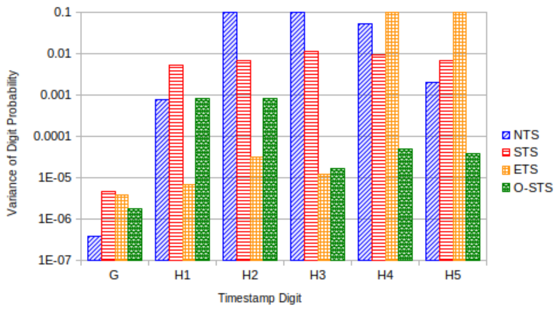


Figure 2: Comparison of the variance digit probabilities for the source timestamp (*NTS*: natural timestamps from the simulation, *STS*: timestamps with stego message, *ETS*: enhanced timestamps from the simulation, *O-STS*: timestamps with obfuscated stego message

Even though the probabilities of occurrences of the digits are not identical for *ETS* and *O-STS*, detecting this improved stego channel is more challenging in comparison to *NTS* and *STS*. In the latter, any deviation of the values for **H2** and **H3** is an indicator of an embedded message yielding a 100% chance of detection.

4 DETECTION APPROACH AND EVALUATION

The proposed hidden communication channel is fully protocol compliant and warden compliant. Thus, an off-the-shelf intrusion detection system (IDS) would be unable to detect the altered messages because the timestamps would be plausible after all. In this chapter we introduce the concept of our detection approach and we present the results of our evaluation. The detection is performed for the non-encrypted, non-signed operating mode of OPC UA. However, the results are valid for the signed operating mode as well. If the hidden communication mode is used in conjunction with the OPC UA encryption, the warden needs access to the encryption keys. Otherwise, the source timestamps cannot be analyzed by a warden as an external observer. In our setting the warden **W** is placed between **A** and **B**. In line with Kerckhoffs principle, the algorithm is known to **W**. The fixed key $\mathbf{K} = 1337$, initialization vector $\mathbf{I} = [3; -1; 2; -5]$ and message m with $\mathbf{M2} = 0$, $\mathbf{M3} = 0$ and $\mathbf{M4} = 0$ are unknown to **W**. The OPC UA read responses with and without the steganograpic message are extracted from network captures.

4.1 Concept of Detection Approach

The detection of hidden communication is based on the ability to identify anomalies in recorded network traffic. In this work, we design a pattern recognition based anomaly detection approach with handcrafted features and a one-class-classifier. Based on an initial analysis, the records of the hidden communication channel are warden compliant for a package sequence size $PS_s > 200$. Due to the warden compliance of the hidden communication channel, we analyze smaller package sequences ($PS_s <= 200$) for our detection approach based on the assumption that the warden compliance decreases for smaller hidden communication channel package sequences. Due to this, the PS_s is a parameter for our detection approach. We determine the optimized PS_s in our evaluation process by exploratory analyzing different sequence sizes by comparing detection accuracy and false positive rates (FPR).

For our detection approach we analyze the six digits of **G** and **H**. Therefore, we determine the probability of occurrence of digits for the decimal places of **G** and **H** of a $PS_s = n$ where $\{15 >= n < 200\}$. This means, we get 10 percentage values for each of the six digits for a package sequence with a PS_s of n. Based on the assumption that authentic network traffic (*no-stego*) follows a uniform distribution, we extract the standard deviation of the 10 percentage values for all six digits. This creates our handcrafted six dimensional feature space which trains our classifier for our detector. We assume that the standard deviation for the six digits increases significantly for stego data in smaller package sequence sizes caused by the stego embedding process.

Our classifier is trained with WEKA 3.9 [4] and its *OneClassClassifier* [5]. We train the classifier only with the features of authentic (*no-stego*) network traffic to determine the 'target' class of the *OneClassClassifier*. The trained classifier is now able to detect samples with anomalies (compared to training data) and to flag them as 'outliers'. We use the default parametrization of the classifier. Additionally, we set the target-rejection-rate to 0.001 to keep the false positive rate as low as possible, because we are aware, that the detector would be shut off in a real world application, when it would trigger few false alarms. We evaluate our detection approach in the next subsection.

4.2 Evaluation of Detection Approach

For the evaluation of our approach we use one training and two test datasets with OPC UA read response packets (RP) approximately every 100ms within the raw network capture files:

- $DS_{training}$ (RP: 76,000; class: 'target'),
- $DS_{test_{tgt}}$ (RP: 46,159; class: 'target') and
- $DS_{test_{stego}}$ (RP: 7,588; class: 'outlier').

Due to this, the detector is trained with $DS_{training}$ and tested with stego network traffic $DS_{test_{stego}}$ to determine the detection accuracy and with known-good traffic $DS_{test_{target}}$ to determine the corresponding false positive rate (FPR). As mentioned in Section 4.1, we use different parameterizations of PS_s to determine the optimized package sequence size (consecutive OPC UA read responses) of a sample for our detection approach. We evaluate PS_s stepwise, by comparing detection accuracy and FPR. We determine the accuracy of our detector simply with $ACC = \frac{a}{b} \cdot 100$, where a is the number of correct classified samples in a dataset and b is the total number of samples in a dataset. The results of our detection approach for different PS_s are visualized in Table 2.

Table 2: Accuracy of detection approach trained with $DS_{training}$ for independent test datasets $DS_{test_{stego}}$ and $DS_{test_{tgt}}$ for different package sequence sizes PS_s.

	Accuracy (ACC) for			**Accuracy (ACC) for**	
PS_s	$DS_{test_{stego}}$	$DS_{test_{tgt}}$	PS_s	$DS_{test_{stego}}$	$DS_{test_{tgt}}$
15	64.0	100.0	21	43.0	100.0
16	33.2	99.8	25	26.0	99.6
17	82.8	99.6	30	31.1	99.8
18	89.6	99.8	40	34.1	99.6
19	**89.5**	**100.0**	50	36.9	99.0
20	68.1	100.0	100	29.2	99.3

Table 2 shows that the detector delivers the best results with a package sequence size $PS_s = 19$. Here, the detectors detects 89.5% of network traffic which includes steganography – $ACC = 100$ for $DS_{test_{tgt}}$ means an FPR of 0 and no triggered false alarms by the detector. If PS_s increases the detection accuracy for stego data decreases significantly due to warden compliance. The detection results for $PS_s = 18$ are slightly better (+0.1%) for $DS_{test_{stego}}$ but we trigger 0.2% false alarms in $DS_{test_{tgt}}$. The accuracy drops down significantly for $PS_s < 18$.

5 SUMMARY AND FUTURE WORK

In this paper we introduce a novel keyed stego channel for command injection to a PLC using OPC UA as cover channel. The exemplary attack resembles a supply chain attack where the attacker provides a compromised OPC UA server and a compromised programming of the PLC in order to implement a hidden command injection. A message with a length of 13 bits is embedded into the source timestamp of the variables within specific OPC UA read responses from the server, all other communication is not affected. As a result arbitrary digital outputs of the PLC can be manipulated by the attacker in order to influence the controlled process. Such an approach could be used to cause black outs in power grids by tampering with the breaker control. In theory other malicious intents, such as traffic light manipulation, are possible using this approach as well.Although the experiments are performed using a S7-1500 PLC, the covert channel should be portable to other OPC UA capable devices under the assumption that the source timestamps are accessible by their software. We propose a detection approach based on a one-class classifier yielding a performance of up to 89.5% detection rate within capture windows of 1.9 seconds containing 19 OPC UA read responses. The detection performance deteriorates significantly when the window size for the detector is increased. Thus, we can assume that the detector needs to be configured according to the frequency of the OPC UA read requests and responses in order to maintain a sufficient detection performance. However, this value can usually be determined by the system operator.

In future work an increased warden compliance should be further evaluated by using more sophisticated methods to achieve digit distributions closer to the stego messages. Additionally, other protocols or even complex channels with protocol converters should be investigated as well.

ACKNOWLEDGMENTS

The work in this paper has been funded by the German Federal Ministry for Economic Affairs and Energy (BMWi, Stealth-Szenarien, Grant No. 1501589A and 1501589C) within the scope of the German Reactor-Safety-Research-Program.

REFERENCES

[1] 2005. *INTRODUCTION TO MODBUS TCP/IP.* Technical Report. ACROMAG INCORPORATED. https://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf

[2] 2018. *TRISIS Malware - Analysis of Safety System Targeted Malware.* Technical Report. Dragos Inc. https://dragos.com/wp-content/uploads/TRISIS-01.pdf

[3] Nicolas Falliere, Liam O Murchu, and Eric Chien. 2011. *W32.Stuxnet Dossier.* Technical Report. Symantec Corporation. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

[4] M. Hall. 2009. The WEKA data mining software: An update. *In SIGKDD Explorations* (2009).

[5] K. Hempstalk, E. Frank, and I. Witten. 2008. One-Class Classification by Combining Density and Class Probability Estimation. *In: Proceedings of the 12th European Conference on Principles and Practice of Knowledge Discovery in Databases and 19th European Conference on Machine Learning, ECMLPKDD2008, Berlin (2008) 505-519* (2008).

[6] Mario Hildebrandt, Robert Altschaffel, Mathias Lange, Martin Szemkus, Tom Neubert, Claus Vielhauer, Yongdian Ding, and Jana Dittmann. 2020. Threat Analysis of Steganographic and Covert Communication in Nuclear I&;C Systems. In *International Conference on Nuclear Security: Sustaining and Strengthening Efforts*. International Atomic Energy Agency, Vienna, Austria.

[7] Andrew D. Ker. 2016. *Information Hiding (complete).* Department of Computer Science, Oxford University. http://www.cs.ox.ac.uk/andrew.ker/docs/informationhiding-lecture-notes-ht2016.pdf

[8] Kevin Lamshöft and Jana Dittmann. 2020. Assessment of Hidden Channel Attacks: Targetting Modbus/TCP. In *to appear in 21st IFAC World Congress, Elsevier ScienceDirect IFAC-PapersOnLine*. Berlin, Germany.

[9] Robert M. Lee, Michael J. Assante, and Tim Conway. 2016. *Analysis of the Cyber Attack on the Ukrainian Power Grid.* Technical Report. SANS Institute. https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf

[10] Stefan-Helmut Leitner and Wolfgang Mahnke. 2006. OPC UA - Service-oriented Architecture for Industrial Applications. *Softwaretechnik-Trends* 26 (2006).

[11] Wojciech Mazurczyk, Steffen Wendzel, and Krzysztof Cabaj. 2018. Towards Deriving Insights into Data Hiding Methods Using Pattern-Based Approach. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (Hamburg, Germany) *(ARES 2018)*. Association for Computing Machinery, New York, NY, USA, Article 10, 10 pages. https://doi.org/10.1145/3230833.3233261