Covert Timing Channel Attack on OPC UA-based Industrial Control Systems

Erkin Kirdan
Framatome
erkin.kirdan@framatome.com

Karl Waedt
Framatome
karl.waedt@framatome.com

***Abstract*—As industrial control systems (ICS) become increasingly complex and intertwined with information technology networks, they inadvertently create new vulnerabilities for cyber threats. Among these threats, covert timing channel attacks, typically associated with advanced persistent threats, pose a unique challenge. This paper delves into the implications of such attacks in an OPC UA setting, a widely adopted standard for Ethernet-based ICS communication. We explore a range of potential covert timing channel attacks within OPC UA and provide a theoretical framework. The practical contribution of our research involves implementing and demonstrating a sample attack exploiting the monitored item feature of OPC UA. In this scenario, changes in a specific variable carry a secret message transmitted through notification messages. The client can subsequently decode this secret message by observing the time intervals between variable change notifications. Our practical setup successfully sends a binary secret embedded within two distinctive notification intervals, symbolizing binary 1 and 0. As the covert timing channel attacks pose considerable risk to OPC UA-based ICS, we believe directing future research efforts toward detecting and mitigating such threats is essential. The study underscores the urgency to enhance the security of ICS as they become increasingly interconnected and, consequently, more susceptible to sophisticated cyber attacks.**

***Index Terms*—covert timing channel attack, OPC UA, industrial control systems, cybersecurity**

I. Introduction

Industrial Control Systems (ICS) represent a critical pillar in the operation of many cyber-physical systems, supporting our increasingly interconnected global infrastructure. These systems are central to managing various devices and systems, from straightforward mechanisms like traffic lights to sophisticated setups in nuclear power plants or advanced manufacturing facilities. The evolution of industrial Ethernet has effectively transformed ICS from isolated entities into interconnected systems that communicate with broader Information Technology (IT) networks. This change has prompted an unprecedented confluence of IT and operational technology [1].

The increased interconnectivity has certainly brought significant advantages in automation and operational efficiency. However, this interconnectedness has inadvertently exposed these systems to new frontiers of cyber threats. Malevolent actors have been known to manipulate vulnerabilities in ICS, disrupting critical services or, in more severe cases, inflicting physical harm [2]. The landscape of threats has become increasingly sophisticated with the emergence of advanced persistent threats that employ stealthy techniques to stay concealed within networks for extended durations.

One such elusive technique is the covert timing channel (CTC) attack, where the temporal sequencing of messages is manipulated to encode and secretly transmit information, as illustrated in Figure 1. The firewall monitors and allows the packets based on their content. In contrast, sensitive or confidential information is sent through the inter-arrival times of the packets. In this example, longer intervals between packet arrivals represent a binary '0', while shorter intervals signify a binary '1'.

While traditionally associated with IT network breaches, this method poses a significant risk to ICS, especially those based on the OPC Unified Architecture (OPC UA). OPC UA has gained widespread acceptance as the de facto standard for industrial Ethernet-based ICS communication, given its capabilities in interoperability, robust security features, and overall system resilience [3]. Yet, the possibility of CTC attacks in such a setting still needs to be explored.

The focus of this paper lies in filling this research gap. We delve into theoretical and practical analyses of the feasibility of conducting a CTC attack on an OPC UA-based ICS. The theoretical aspect of our research is based on defining a mathematical model of this kind of attack. The practical aspect of our study rests on exploiting the monitored item feature of OPC UA. We implement a method to encode a secret message into the time intervals between variable change notifications, thereby enabling hidden information transmission from the server to the client.

With this paper, we make the following contributions to advance the field of study:

1) identifying potential OPC UA features vulnerable to CTC attacks (Section III),
2) defining the theoretical framework for modeling such attacks (Section IV),
3) implementing and demonstrating a sample attack (Section V),
4) projecting promising directions for future work (Section VI).

Our endeavor, through this study, is to deepen the understanding of security vulnerabilities inherent in OPC UA-based ICS. We aim to provide a foundation to promote the development of effective countermeasures against such sophisticated threats, thereby fortifying these critical systems.

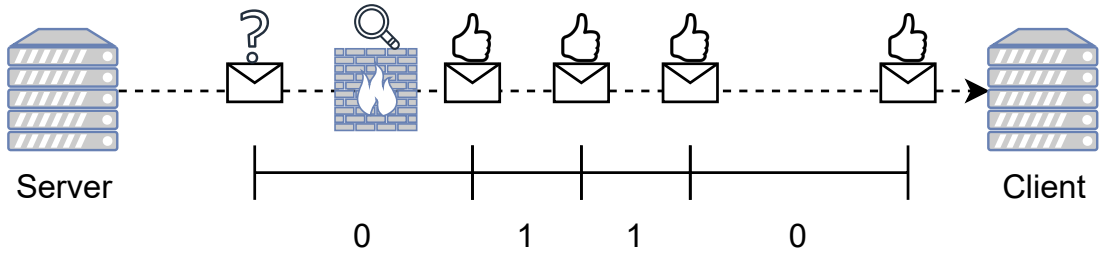

Fig. 1. Covert Timing Channel Attack.

II. Related Work

Cabuk et al. propose a network covert channel via timing and provide a foundation for covert communication concepts in networked systems [4]. It introduces a timing channel implementation, presents performance data, and discusses its detectability. This work is similar to our research regarding the CTC concept but differs in its application domain, as it is not focused on OPC UA but on IP-based networks.

Liu et al. advance the field by presenting a robust and undetectable CTC [5]. They introduce a method to modulate the message in the inter-packet delay, approximating regular traffic, and use spreading techniques for robustness.

Archibald and Ghosal present a model-secure CTC utilizing Fountain codes [6]. The authors enhance the robustness of CTC against jitter, a significant concern in networked communications. Their approach is novel and could be applied in an OPC UA environment.

Mazurczyk et al. offer a taxonomy for hiding methods in network communication [7]. They focus on creating a comprehensive categorization and understanding of various data-hiding methods. This work provides insights into the broader context of data hiding, including the timing channel attacks.

Two significant related works focus on ICS. Alcaraz et al. discuss covert channels in industrial networks and present two signaling strategies exploiting the Modbus/TCP protocol [8]. They demonstrate the potential risks associated with such attacks and propose countermeasures. Similarly, Hildebrandt et al. investigate potential attack vectors on Programmable Logic Controllers using the OPC UA protocol [9]. They implement a supply chain attack and discuss detection mechanisms. These two works are closely related to our research, albeit with different attack methodologies.

Finally, Lamshöft and Dittmann investigate how known patterns of information hiding can be applied to protocols found in ICS networks [10]. This work demonstrates practical applications of hidden channels in industrial communication protocols, which is relevant to our research context. However, the focus is on the Modbus/TCP protocol, not OPC UA.

Table I shows the position of our work among the related works. Our study is the first to explore potential CTC attacks in OPC UA, make the theoretical analysis, and implement a sample attack.

III. Covert Timing Channel Potentials in OPC UA

OPC UA is a machine-to-machine communication protocol for industrial automation [11], [12]. Its design strongly emphasizes security, including authentication, authorization, encryption, and data integrity. However, as with any system, it is not entirely impervious to all potential threats. CTC attacks are one type of possible threat, although it is essential to note that these are not specific to OPC UA and could be conducted on any networked system.

A CTC is an attack where information is extracted not from the message itself but from the timing and order of the messages. This type of attack is challenging to execute and even more challenging to detect, as it exploits seemingly normal behavior to transmit information.

Let us consider the software of an OPC UA server that a 3rd party provides. The sub-supplier may include a tiny ”debugging” back door that allows him to receive considerable data stored in memory within a small loop and later executed. Even for source code reviewers, there may be no obvious way for data to be received via a network and stored for execution. However, a module considered part of timing-related optimization, e.g., reduction of time jitters can stealthily evaluate the hidden information described in this paper. To summarize, this use case would facilitate the receipt of malware encoded by stealth timing without an explicit ‘receive()’ function to be detected during source code reviews. It would store the data in a tiny loop for later execution. The absolute or relative execution time for embedded servers not supporting clock time services can also be encoded within the stealth timing-based hidden data stream. Thus, the provision of the malware and the effect of the malware could be decoupled entirely and, therefore, even more challenging or impossible to trace.

Although OPC UA is a secure protocol, the following features could be exploited for a CTC attack.

1) **Session Activity:** OPC UA uses sessions for communication between clients and servers. These sessions have a unique identifier and state, and they begin with a CreateSession service and end with a CloseSession service. An attacker could manipulate the timing of these sessions to send information covertly. For example, the time gap between opening and closing sessions could represent different bits of information.

TABLE I
COMPARISON OF OUR WORK WITH RELATED WORKS

	System	Focus	Attack Type	Implementation
Our work	OPC UA	Attack	Timing	Yes
[4]	IP-based	Attack & Detection	Timing	Yes
[5]	General	Attack	Timing	Yes
[6]	General	Attack	Timing	Yes
[7]	Network	Understanding	-	No
[8]	Modbus/TCP	Attack & Mitigation	-	Yes
[9]	OPC UA	Attack & Detection	Storage	Yes
[10]	Modbus/TCP	Attack & Detection	Both	Yes

2) **Subscription and Monitored Items:** OPC UA clients can create subscriptions to monitor specific variables on the server for changes. If the client is malicious, it could attempt to use the timing or frequency of subscriptions to send covert signals. For example, subscribing and unsubscribing from a variable in a specific pattern could be used to encode data.
3) **Data Sampling Rates:** OPC UA allows clients to specify the rate at which data for a particular monitored item is sampled. This rate can vary between subscriptions, and it is possible that an attacker could manipulate this rate to create a covert channel. By changing the sampling rate, the attacker could effectively modulate a signal within the regular operation of the system.
4) **Service Request Timing:** OPC UA allows clients to specify the rate at which data for a particular monitored item is sampled. This rate can vary between subscriptions, and it is possible that an attacker could exploit this rate to create a covert channel. By changing the sampling rate, the attacker could effectively modulate a signal within the regular operation of the system.
5) **Network Traffic:** Network traffic in an OPC UA system could be manipulated for covert communication. For example, an attacker could modulate the message transmission rate or the size of data packets to send covert signals. Timing fluctuations or delays in communication packets can also provide a medium for CTCs.
6) **Data Value Update:** In OPC UA, the server updates the client on changes to the monitored items. The timing or frequency of these updates could theoretically be manipulated to send covert signals. For example, the rate or pattern of these updates could represent encoded data.
7) **Notification Messages:** When a monitored item changes, the server sends a *NotificationMessage* to the client. The timing and frequency of these notifications could be exploited in a timing attack. For example, varying the delay between notifications or sending notifications in a specific sequence could encode information.
8) **Asynchronous Response to Service Requests:** In response to a service request from a client, a server generates a service response. Delays or patterns in these responses could be used to send signals covertly.
9) **KeepAlive Messages:** OPC UA servers send *KeepAlive* messages periodically to inform clients that the subscription is still active. The timing of these messages could be manipulated to provide a CTC.
10) **Redundant Server Failover:** OPC UA supports server redundancy, where a backup server can take over if the primary server fails. The timing and pattern of these failovers could potentially be manipulated for a covert timing attack.
11) **Status and Diagnostic Information:** The server often provides status and diagnostic information to clients. The timing of these updates or the specific information chosen for updates could be used in a timing attack.

Table II shows the overview of potential CTC attacks in OPC UA. They are grouped under client-to-server or server-to-client communication directions.

IV. THEORETICAL FRAMEWORK

CTCs encode information by utilizing the temporal variance of communication events. This section gives the theoretical framework for CTC attacks.

A. System Model

Consider a sender who encodes a binary secret within the temporal gaps, or inter-packet gaps, between the sequence of packets. Let us denote the inter-packet gap corresponding to the bit '0' as T_0 and for bit '1' as T_1. The receiver's task is to decode this binary secret by analyzing the inter-packet gaps of the received packets.

The symbol η denotes the jitter within the channel. This jitter causes random time shifts that each packet may individually experience during transmission. We assume that this jitter is bounded, formally expressed as $-\delta \leq \eta \leq \delta$.

B. Decoding

Despite the presence of jitter, it is necessary for the receiver to accurately differentiate between the inter-packet gaps for bit '0' and bit '1' to ensure successful decoding. Let the inter-packet gap signaling bit 'i' be represented as $t_i + 2\eta$.

The receiver interprets a received bit as '0' if the inter-packet gap is closer to T_0 and interprets it as '1' if it is closer to T_1. Therefore, the decision boundaries for the receiver can be expressed as:

$$\begin{cases} \text{Decode as '0'} & \text{if } |T_0 - (t_i + 2\eta)| < |T_1 - (t_i + 2\eta)|, \\ \text{Decode as '1'} & \text{if } |T_0 - (t_i + 2\eta)| > |T_1 - (t_i + 2\eta)|. \end{cases}$$

TABLE II
COVERT TIMING CHANNEL POTENTIALS IN OPC UA

Communication Direction	OPC UA Feature	Potential Covert Timing Attack
Client-to-Server	Session Activity	Manipulate timing of session creation, usage, closure
	Subscription and Monitored Items	Manipulate timing or frequency of subscriptions
	Data Sampling Rates	Change sampling rates to encode data
	Service Request Timing	Manipulate timing or sequence of service requests
	Network Traffic	Modulate rate of message transmission or size of data packets
Server-to-Client	Data Value Update	Manipulate timing or frequency of updates
	Notification Messages	Vary timing or sequence of notifications
	Asynchronous Response	Vary delays or patterns in responses
	KeepAlive Messages	Manipulate timing of KeepAlive messages
	Redundant Server Failover	Manipulate timing or pattern of failovers
	Status and Diagnostic Information	Vary timing of updates or information chosen for updates

C. Error Probability

Errors in decoding can occur when the jitter causes the inter-packet gap for one bit to cross the decision boundary and be closer to the inter-packet gap for the other bit. Let's denote the error probability as P_e, representing the probability that jitter culminates in an incorrect bit decoding. We can compute this probability by assuming that the jitter η is uniformly distributed over the interval $[-\delta, \delta]$.

Consider the instance when the transmitted bit is '0', i.e., $t_i = T_0$. An error arises when the summation of the inter-packet gap and the two jitters, denoted as $T_0 + 2\eta$, falls within the decision boundary for bit '1', closer to T_1. This occurs when $|T_0 - (T_0 + 2\eta)| > |T_1 - (T_0 + 2\eta)|$. To derive the error probability P_{e0} (bit '0' wrongly decoded as '1'), consider the case when $T_0 + 2\eta$ is interpreted as bit '1'. Given the bounding on η, the maximum value of η can be δ. Therefore, the probability of this event is the ratio of the range of η that satisfies the condition to the total range of η, i.e.,

$$P_{e0} = \max\left(0, \frac{4\delta - |T_1 - T_0|}{8\delta}\right).$$

Due to the symmetry of the problem, the error probability for the bit '1' would be identical to that for '0', i.e., $P_{e1} = P_{e0}$. Consequently, if the bits '0' and '1' are equally likely, the total error probability will be $P_e = P_{e0}$. The error probability, P_e, plays a significant role as it influences both the success criteria of the attack and the achievable bit rate.

D. Success Criteria

The covert channel attack is successful if the receiver can decode the binary secret correctly. Firstly, both inter-packet gaps T_0 and T_1 should be greater than the jitter of two sequential packets, formally expressed as:

$$T_0, T_1 > 2\delta$$

Secondly, we need to minimize the error probability to zero. So, the possible values, or in other words, the ranges of inter-packet gaps T_0 and T_1 should not overlap to be decoded correctly. As each inter-packet gap may experience two times the jitter, the difference should be greater than four times the jitter, formally expressed as:

$$|T_0 - T_1| > 4\delta$$

These conditions guarantee that even under a worst-case scenario, where the jitter results in maximum deviation, the receiver can still correctly distinguish between bit '0' and bit '1'.

E. Maximum Achievable Bit Rate

Assuming the secret is a random binary, the bit rate of the attack is determined by the frequency of the packets sent, which is inversely proportional to the average of the two inter-packet gaps. Formally, the bit rate R can be expressed as:

$$R = \frac{1}{\frac{T_0 + T_1}{2}}$$

To maximize this bit rate, we need to minimize the average of the two inter-packet gaps while still maintaining the success criteria. Considering the success criteria, the minimum value for T_0 and T_1 is 2δ, and the minimum difference between them is 4δ. Therefore, the minimum average of T_0 and T_1 is 4δ, which leads to the maximum achievable bit rate:

$$R_{max} = \frac{1}{4\delta}$$

This maximum bit rate is achieved when the inter-packet gaps are chosen such that $T_0 = 2\delta$ and $T_1 = 6\delta$ or vice versa. This ensures that the average of the inter-packet gaps is minimized while still satisfying the success criteria.

Let's consider an example to better understand the theoretical framework. We assume the jitter, denoted by δ, in the system to be 5 ms. Let us choose values for T_0 and T_1 that hold the success criteria. In this case, we choose $T_0 = 10$ ms and $T_1 = 30$ ms. These values meet the success criteria since T_0 and T_1 are greater than or equal to $2\delta = 10$ ms, and $|T_0 - T_1| = 20$ ms, which is equal to $4\delta = 20$ ms. The maximum achievable bit rate is calculated using the formula:

$$R_{max} = \frac{1}{4\delta} = 50 \text{ bits/s}.$$

Thus, in this example, with a correct selection of inter-packet gaps, the receiver can successfully decode the transmitted binary secret with a bit rate of 50 bps despite a jitter of 5 ms in the system.

F. Discussion

In summary, we have presented a theoretical framework based on defining and analyzing the mathematical model of CTCs. Through a study of the system model, we have characterized how the sender encodes information in the inter-packet gaps of a packet sequence and how the receiver decodes this information. This analysis was extended to incorporate factors such as channel jitter, revealing their significant influence on decoding accuracy.

We have derived expressions for essential parameters like the error probability and maximum achievable bit rate, considering various factors such as jitter distribution. These expressions provide insights into the performance and limitations of the CTC attacks.

Altogether, the framework laid out in this section offers a foundation for understanding and further researching CTCs. It illuminates the interactions between multiple variables in these systems and provides a roadmap for future exploration of this critical topic.

V. Implementation

This section details our implementation, demonstrating a CTC attack in OPC UA using the monitored item and subscription features.

A. Covert Timing Channel Attack Mechanism

Our project is implemented using the node-opcua library[1]. It is one of the most commonly used open-source OPC UA libraries designed to develop OPC UA servers and clients [13]. It is built on the Node.js platform, leveraging JavaScript's event-driven nature to create efficient, high-performance applications for industrial automation and IoT devices [14]. OPC UA, as a set of standards for industrial automation, offers interoperability and security for manufacturing and other process control systems. The node-opcua library makes these standards accessible to developers by providing many functionalities, including encrypted communications, a binary protocol for efficient data transfer, subscription services for real-time data monitoring, and more [15], [16].

Our server is written in JavaScript, and the client is in TypeScript. The client subscribes to a variable in the server, which enables the client to receive a notification message whenever the subscribed variable changes. The server sends a notification message to the client every time there is a change in the value. The server encodes a secret between the notification messages by intentionally changing this value. The client observes the intervals between the received notifications to decode the secret embedded by the server.

B. Results

In our implementation, we have selected a 1-second interval to represent bit 1 and a 2-second interval to represent bit 0. This relaxed encoding choice is made for demonstrative purposes and can be optimized according to the jitter and the calculations shown in the theoretical framework section. With this encoding, the server correctly transmits a message to the client at a 0.66 bps bit rate. The bit rate can, of course, increase depending on the network conditions.

Performance metrics are not significantly related to our context. The attack aims to transmit a secret correctly, regardless of the bit rate. Such attacks are mainly used for transmitting small but sensitive or confidential data, such as *passwd*, instead of a multimedia stream.

CTC attacks, like all cyberattacks, require specific conditions to be effective. The presence of a network component, such as an intrusion detection system, that strictly controls or monitors the consistency of packet interarrival times can prevent these attacks. Thus, the effectiveness of the attack discussed in this paper is predicated on either the absence or the poor design of traffic regulation mechanisms. Such network security gaps allow attackers to manipulate the timing between packet arrivals without triggering any alarms [17].

C. ASCII Mapping and Delimiter Implementation

We successfully run the attack while transmitting the secret as a bitstring. Furthermore, we have also implemented ASCII mapping to make the output of the attack in a human-readable form. This mapping converts a given message to its corresponding ASCII bitstring before transmission.

The client may start receiving notifications at any time. Thus, it cannot determine when to start decoding the received bits. To counter this issue, we used a unique delimiter, "111111110", to indicate the beginning of the message. This delimiter is chosen due to its inability to occur arbitrarily in a stream of ASCII characters, making it suitable for our use case.

The server is configured to transmit the encoded message in a loop. When the client encounters the delimiter during the decoding process, it converts the received bitstring back into ASCII characters, revealing the secret message.

D. Reproducible Research

We have created a public repository on GitHub[2], which hosts all the scripts and instructions required to demonstrate the attack. The repository contains two scripts:

- *server.js*, an OPC UA server that encodes a secret message into notification delays, and
- *client.ts*, an OPC UA client that connects to the server, monitors the notifications, and decodes the delays into the original secret message.

Interested readers are encouraged to clone this repository, follow the setup and execution instructions in the README

[1]https://github.com/node-opcua/node-opcua

[2]https://github.com/erkinkirdan/CovertTimingChannelAttack_OPCUA

file, and explore the demonstration first-hand. This hands-on experience provides valuable insights into executing the practicalities and nuances of the attack.

VI. Conclusion

In conclusion, this research paper explores the CTC attacks on ICS, specifically within the OPC UA framework. As ICS become more intertwined with IT networks, they are increasingly exposed to sophisticated cyber threats, including CTC attacks. By analyzing this potential threat theoretically and practically, we identified vulnerabilities within the OPC UA features, particularly the monitored item feature. We demonstrated the feasibility of these attacks by successfully transmitting a secret embedded within the notification intervals. The significance of our findings is amplified by the increasing adoption of OPC UA for industrial Ethernet-based ICS communication. This study emphasizes the urgent need to further research in this area, specifically focusing on developing effective detection and mitigation strategies to safeguard the increasingly interconnected ICS, which are critical to our global infrastructure. This work provides a foundation for understanding and fortifying these systems against CTC attacks.

Several potential future directions emerge from the research on covert channel attacks. The concept of artificial steganographic network data generation, as proposed by [18], could offer valuable training and test data for evaluating detection mechanisms, thus aiding in combating covert channel attacks. This aspect could be incorporated into future studies. [19] and [20] present machine learning-based methods to detect covert channel attacks on ICS. Adapting such techniques in the context of OPC UA-based ICS could provide an effective means to identify and mitigate CTC attacks. Furthermore, there could be an exploration into the usage of TCP payload entropy as engineered features, as indicated by [19], or a Convolutional Neural Network based detection approach, as mentioned in [20]. These techniques could be influential in enhancing the accuracy of detecting CTC attacks and thus improving the security of OPC UA-based ICS.

References

[1] S. Paiho, J. Kiljander, R. Sarala, H. Siikavirta, O. Kilkki, A. Bajpai, M. Duchon, M.-O. Pahl, L. Wüstrich, C. Lübben *et al.*, "Towards cross-commodity energy-sharing communities–a review of the market, regulatory, and technical situation," *Renewable and Sustainable Energy Reviews*, vol. 151, p. 111568, 2021.

[2] K. Waedt, I. Ben Zid, J. Schindler, and E. Kirdan, "Cybersecurity education programmes & laboratories brainstorming," in *Cybersecurity for Critical Infrastructure Protection via Reflection of Industrial Control Systems*. IOS Press, 2022, pp. 100–107.

[3] N. Mühlbauer, E. Kirdan, M.-O. Pahl, and G. Carle, "Open-source opc ua security and scalability," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2020, pp. 262–269.

[4] S. Cabuk, C. E. Brodley, and C. Shields, "Ip covert timing channels: design and detection," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 178–187.

[5] Y. Liu, D. Ghosal, F. Armknecht, A.-R. Sadeghi, S. Schulz, and S. Katzenbeisser, "Hide and seek in time—robust covert timing channels," in *Computer Security–ESORICS 2009: 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings 14*. Springer, 2009, pp. 120–135.

[6] R. Archibald and D. Ghosal, "A covert timing channel based on fountain codes," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2012, pp. 970–977.

[7] W. Mazurczyk, S. Wendzel, and K. Cabaj, "Towards deriving insights into data hiding methods using pattern-based approach," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–10.

[8] C. Alcaraz, G. Bernieri, F. Pascucci, J. Lopez, and R. Setola, "Covert channels-based stealth attacks in industry 4.0," *IEEE Systems Journal*, vol. 13, no. 4, pp. 3980–3988, 2019.

[9] M. Hildebrandt, K. Lamshöft, J. Dittmann, T. Neubert, and C. Vielhauer, "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*, 2020, pp. 115–120.

[10] K. Lamshöft and J. Dittmann, "Assessment of hidden channel attacks: Targetting modbus/tcp," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11 100–11 107, 2020.

[11] E. Kirdan, F. Rezabek, N. Mülbauer, G. Carle, and M.-O. Pahl, "Real-time performance of opc ua," *arXiv preprint arXiv:2310.17052*, 2023.

[12] I. E. Commission, "Opc unified architecture," International Electrotechnical Commission, Tech. Rep., 2020.

[13] N. Mühlbauer, E. Kirdan, M.-O. Pahl, and K. Waedt, "Feature-based comparison of open source opc-ua implementations," *INFORMATIK 2020*, 2020.

[14] J. Schindler, S. Belaidi, E. Kirdan, and K. Waedt, "Securing javascript runtime of opc ua deployments," *INFORMATIK 2022*, 2022.

[15] E. Kirdan, J. Schindler, and K. Waedt, "Optimizing opc ua deployments on node. js through advanced logging techniques," 2023.

[16] J. Schindler, E. Kirdan, and K. Waedt, "Secure opc ua server configuration for smart charging stations," *INFORMATIK 2021*, 2021.

[17] R. Yatagha, K. Waedt, J. Schindler, and E. Kirdan, "Security challenges and best practices for resilient iiot networks: Network segmentation," 2023.

[18] T. Neubert, C. Vielhauer, and C. Kraetzer, "Artificial steganographic network data generation concept and evaluation of detection approaches to secure industrial control systems against steganographic attacks," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–9.

[19] H. Li and D. Chasaki, "Network-based machine learning detection of covert channel attacks on cyber-physical systems," in *2022 IEEE 20th International Conference on Industrial Informatics (INDIN)*. IEEE, 2022, pp. 195–201.

[20] T. Neubert, A. J. Caballero Morcillo, and C. Vielhauer, "Improving performance of machine learning based detection of network steganography in industrial control systems," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–8.