# The Threat of Covert Channels in Network Time Synchronisation Protocols

Kevin Lamshöft*, Jonas Hielscher, Christian Krätzer and Jana Dittmann

*Otto-von-Guericke University Magdeburg, Germany*
*E-mail: kevin.lamshoeft@ovgu.de*
**Corresponding Author*

Received 15 December 2021; Accepted 19 December 2021;
Publication 16 March 2022

## Abstract

Synchronized clocks are vital for most communication scenarios in networks of Information Technology (IT) and Operational Technology (OT). The process of time synchronisation requires transmission of high-precision timestamps often originating from external sources. In this paper, we analyze how time synchronization protocols impose a threat by being leveraged as carrier for network covert channels.

This paper is an extended version version of our open-access paper [15] in which we performed an in-depth analysis of the Network Time Protocol (NTP) in regards to covert channels. In this extended version, we broaden the view and take a look and time synchronisation in a more general way as we provide two comprehensive threat scenarios regarding covert channels and discuss the applicability of such covert channels to another time synchronisation protocol, namely the Precision Time Protocol, PTP. While the Network Time Protocol (NTP) is the most prevalent protocol for synchronizing clocks in IT networks, the Precision Time Protocol (PTP) is mostly found in networks of Industrial Control Systems (ICS) due to higher demands regarding accuracy and resolution. To illustrate the threat of covert channels

*Journal of Cyber Security and Mobility, Vol. 11_2,* 165–204.
doi: 10.13052/jcsm2245-1439.1123

in such protocols we describe two threat scenarios, one for the Network Time Protocol and one for the Precision Time Protocol. For NTP we perform a systematic in-depth analysis of covert channels. Our analysis results in the identification of 49 covert channels, by applying a covert channel pattern-based taxonomy. The summary and comparison based on nine selected key attributes show that NTP proofs itself as a plausible carrier for covert channels. The analysis results are evaluated in regards to common behavior of NTP implementations in six major operating systems. Two channels are selected and implemented to be evaluated in network test-beds. By hiding encrypted high entropy data in a high entropy field of NTP we show in our first assessment that practically undetectable channels can be implemented in NTP, motivating the required further research. In our evaluation, we analyze 40,000 NTP server responses from public NTP server providers and discuss potential countermeasures. Finally, we discuss the relevance, applicability and resulting threat of these findings for the Precision Time Protocol.

**Keywords:** Network covert channels, information hiding, covert channels, time synchronisation, network time protocol, precision time protocol.

## 1 Introduction

In networks, covert channels are a technique aiming to plausibly hide information in legitimate and normal (overt) network traffic, in order to stay undetected. Beside the possibility of hiding the identity or communication as security protection goals, multiple malicious purposes exist, such as leakage of confidential data [53] or establishment of Command-and-Control (C&C) channels for Malware [5, 54]. The number of Malware using covert channels grew in recent years [26, 42]. While most of such covert channels do not use sophisticated hiding methods yet, expectation is they soon will, due to more advanced measures of intrusion detection in corporate networks [49]. In this paper, we focus on time synchronization and analyze how such synchronization protocols could be leveraged as carrier for network covert channels and discuss the resulting threat to IT and OT environments. The process of time synchronisation requires transmission of high-precision timestamps often originating from external sources, which makes time synchronisation vital for most networks yet leave them potentially vulnerable to covert channels. This paper is an extended version of our open-access paper [15] in which we performed an in-depth analysis of the Network Time Protocol (NTP) in regards to covert channels. This paper provides a broader view and takes the

results into perspective describing two threat scenarios. In addition to the in-depth analysis of NTP, we discuss whether the results for NTP are applicable to another protocol, the Precision Time Protocol (PTP) which is often found in environments of Operational Technology (OT), e.g. in the case of Cyber Physical Systems and discuss the resulting threat, for example in the case of critical infrastructure environments. The research community tries to keep up with the speed and hundreds of potential channels and corresponding countermeasures (mainly the detection) were studied [47]. However, most research focuses on description and testing of one, or few channels. In this paper, we perform a systematic analysis of a network protocol, motivated by the pattern-based taxonomy of Wendzel et al. from 2015 [50]. It seems promising for deriving required adaption of network architectures and countermeasures: The complexity and required resources often render advanced (e.g. Machine Learning based) detection methods for manifold covert channels in larger corporate networks impossible. In comparison, a systematic analysis and evaluation of such different channels can lead to superordinate countermeasures. With the introduction of a pattern-based taxonomy in 2015 [50] a classification or common description of covert channels was introduced that offers the foundation for a comparable evaluation of network protocols in regards to covert channels. In an updated version from 2018 [28] the taxonomy classifies 20 different types of network covert channels. By reversing this deductive approach and applying those patterns inductively to specifications of network protocols we expect to identify potential timing and storage covert channels in a more systematic way than before, where many covert channels were identified by gut-feeling and in-depth knowledge of networking experts.

The Network Time Protocol (NTP) is the default time synchronization protocol for all major desktop and mobile operating systems (OS), including Windows 10, macOS, Linux Ubuntu, iOS, Android and is also widely used in the domain of Cyber-Physical Systems (CPS) [9]. Therefore, it is present in most networks. Public NTP servers are typically organized in pools and clients are requesting time from multiple different sources. This perspective on the infrastructure alone makes NTP an interesting carrier for covert channels in real-world networks. As we show in our work, the high heterogeneity of protocol fields in NTP packets makes it a good carrier for covert channels. The systematic analysis of NTP in our work leads to the following contributions:

1. Our **systematic analysis** of covert channels in NTP led to the discovery of 49 plausible channels (hiding by modification), which are compared

by selecting and using nine key attributes as comparison criteria, such as capacity, suspicousness and reversibility. Further, we analyze the two existing NTP channels introduced by [2] and [46].
2. The theoretical results are compared with the **plausibility in real-world OS NTP implementations**, by testing operating systems (Windows 10, macOS, Ubuntu, iOS, Android). The findings show that some implementations are not NTP specification compliant, leading to more heterogeneous NTP traffic, which makes the detection of covert channel harder.
3. Our exemplary **implementations of two channels** (1) Undetectable channel in user-data value modulation and (2) Deep-Layer value modulation show that the high entropy in NTP packets can be used for the implementation of practically undetectable channels as well as that hidden information can be sent over multiple layers of the stratum architecture and that the data redundancy in these packets allows the implementation of reversible channels.
4. Discussion on potential **countermeasures or mitigation by elimination and distortion** on network level, e.g. by normalization and certain network designs.

Furthermore, in this extended version we provide following additional contributions:

- Description of two comprehensive **threat scenarios** leveraging NTP and PTP as covert channel for stealthy data infiltration into target networks.
- Discussion on the applicability of the findings regarding covert channels in NTP to the **Precision Time Protocol** and the resulting threats.

The remainder of the work is structured as follows: In Section **2.1** we reflect on previous work that evaluated manifold covert channels and derive a research gap. In Section **2.2** we describe NTP, PTP and important concepts behind covert channels. Section 3 is dedicated to the in-depth analysis of NTP: In Section **3.1** we introduce our research approach for deriving NTP channels and summarize our selected and used nine key attributes as comparison criteria. In Section **3.2** we describe all discovered channels and compare them by the selected nine key attributes. In Section **3.4** we explain the channels that we implemented as well as the results from their testing. In Section **4** we suggest a set of active countermeasures to suppress covert channels in NTP. In Section **5** we discuss the relevance of our findings for NTP and potential threats to validity. Sections 3 to 5 are identical to our previous publication [15]. In Section **6** we describe two threat scenarios

framing the results of the NTP analysis and giving a hindsight on potential covert channels in PTP. Section **7** is dedicated to the discussion on potential covert channels in PTP. Section **8** concludes this paper and encourages further work.

## 2 Related Work and Technical Background

This section gives an insight on related work and describes the technical background on NTP, PTP and network covert channels.

### 2.1 Related Work

The discovery and evaluation of covert channels in network protocols and corresponding countermeasures is often focused on one, or few specific channels. Different approaches were summarized by Zander et al. [53]. Larger, systematic approaches are rare: Murdoch et al. [37], took a look at seven TCP fields and the packet ordering, to describe the potential for covert channel embedding. Lucena et al. [25], did the same for IPv6. Mileva et al. [32], compared 13 IPv4 storage, 14 IPv4 timing, 19 IPv6 storage, 7 ICMP storage, 12 TCP storage, 3 UDP, 17 HTTP, 7 DNS, 11 RTP and 3 SSH covert channels from several previous publications. They introduced six abstract attributes to compare all those channels: method (how does the embedding work/at witch position in the protocol?), packet raw bit rate, type (storage/timing), advantages, disadvantages and defense. Lamshöft et al. [23] used the pattern-based taxonomy from Mazurczyk et al. [28] to categorize different channels they discovered in Modbus/TCP. Mileva et al. [33] introduced a variety of covert channels in the MQTT 5 standard by observing the protocol specifications and deriving c overtchannels.

NTP as a carrier for covert channels was also analyzed: Ameri et al. [2], evaluated a single covert channel in NTP, exploiting a small part of a timestamp field. Tsapakis [46], implemented a covert channel in NTP extension fields. In our work we show that both channels can be found and further described with our analysis approach. Also an extension of NTP, the *NTP control-queries* were exploited to implement a covert channel that could be used asynchronously by sender and receiver [40]. However, these queries are a comprehensive sub-protocol on their own [35] and outside of the scope of our analysis.

Since the number of Malware exploiting covert channels grows and is expected to grow further, it is necessary to investigate potential carriers for

covert channels and elaborate countermeasures. The previous research on few very specific channels motivates our own work to perform an in-depth analysis as a fundamental basis of a further risk assessment. In this work, we follow a more broad approach to systematically identify and evaluate covert channels. We use a two-stage approach in which we use the pattern-based taxonomy of Mazurczyk et al. [28] to find a wide variety of covert channels for a given network protocol. In this approach, we use protocol specifications to find channels and validate their plausibility against default implementations of the protocol. We apply the procedure to NTP, by evaluating every packet field against every out of the 20 patterns of the taxonomy and validate the plausibility against six default OS NTP implementations. Our results are further evaluated by the testing of two covert channels in a virtualized network test-bed.

## 2.2 Technical Background

In this section we provide background information regarding the Network Time Protocol, Precision Time Protocol, Network Covert Channels as well as attributes for describing and comparing such.

## 2.3 Network Time Protocol (NTP)

NTP is an UDP based OSI-Layer 7 protocol, meant for time synchronization. The current version is 4 [36], which is compatible to version 3. Typically, NTP operates in a client-server mode, where clients are constantly polling time from (different) servers, typically three. Two other modes, broadcast and peer-to-peer are possible as well.[1] NTP is closely related to Simple NTP (SNTP) [34] and can not be distinguished by the content of the network packets. The main difference is that SNTP clients only use one server for synchronization. NTP servers are organized in the hierarchical stratum architecture. A server with direct access to a time source (e.g. atomic clock) is a reference server with stratum value 0. Other servers on layer 1 can request time from this server and so on, down to a maximum depth of 15. An NTP client is called synchronized in case it requested enough reliable time information and was able to adapt its local clock to those information.

Every NTP packet consists of at least 13 fields (48 bytes, see Figure 1). Whether a field is used or just filled with zeros, depends on the operation

[1] Since peer-mode packets do not differ from client- and server-mode packets, they are not considered separately in our work.

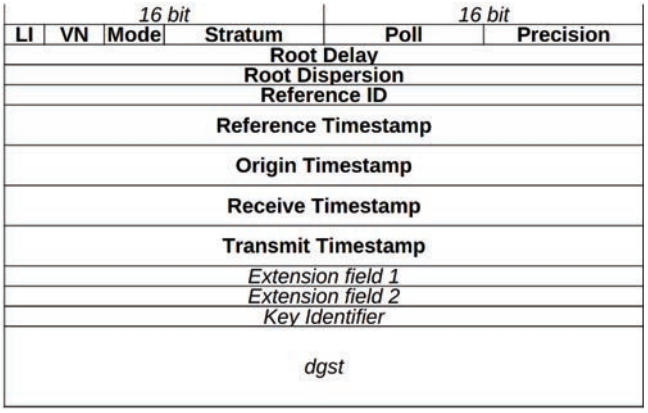



**Figure 1** The structure of an NTP packet. 13 fields are mandatory, the two extension fields and the MAC field (key identifier plus dgst) are optional.

mode. An extension with one or two extension fields or a Message Authentication Code (MAC) field is possible but unusual in practice. In the context of covert channels the most promising fields are the four 64 bit timestamps (reference, origin, receive, transmit) that are used for the roundtrip-delay and time-calculation. The upper 32 bits represent a second counter. The lower 32 bits of the timestamp fields resolve one second (down to the level of picoseconds). Those bits resolving higher than the system clock of a sender should be set to random values [36]. The purpose of NTP fields are the following:

*LI* (2 bits) indicates whether the current day has a leap second. *VN* (3 bits) is the version number. *Mode* (3 bits) is the operating mode. *Stratum* (8 bits) is the stratum value of the sending server and zero in client packets. *Poll* (8 bits) is the polling interval between to requests of the client. *Precision* (8 bits) is the precision of the system clock of the sender. *Root delay* (32 bits) is the estimated delay between the sender and the reference clock. *Root dispersion* (32 bits) is the estimated dispersion between the server and the reference clock. *Reference ID* (32 bits) typically holds the IP-address of the reference clock. In case the stratum value is zero it can hold a so called KISS-code [20] for debugging purposes. *Reference Timestamp* (64 bits) states when the system clock was last set. *Origin Timestamp* (64 bits) in a server response matches the transmit timestamp of the clients request. *Receive Timestamp* (64 bits) is the time when a response was received at the server. *Transmit timestamp* (64 bits) is the time a packet left the sender.

The inner structure of extension fields further divides into: *Field Type* (16 bits), *Length* (16 bits), *Value* (up to 32,766 bytes), *Padding* (variable). NTP does also know a sub-protocol, the so-called "control-queries" [35]. These are NTP packets in a special operation mode, containing debug information rather than time related information. This sub-protocol is not further considered in our evaluation. NTP is not encrypted nor cryptographically signed (past approaches like the Autokey-protocol [13] are broken [39], the newer Network Time Security (NTS) [11] was introduced recently in 2020 and has no practical relevance yet).

## 2.4 Precision Time Protocol (PTP)

The Precision Time Protocol (PTP) was first specified in IEEE 1588-2002 (also known as Version 1) [17], has been superseded in 2008 by IEEE 15588-2008 (also known as Version 2) [18] and has recently been updated with IEEE 1588-2019 [19]. PTP is an industry-standard protocol enabling clock synchronisation over Ethernet with sub-nanosecond granularity using special hardware. PTP is a master/slave protocol with different types of clocks. Relevant for this paper are *ordinary clocks* which are single port devices working either as master/slave and *boundary clocks*, which are two port devices which can work as master and slaves simultaneously. PTP is a hierarchical protocol with one system wide grandmaster clock, whose time is distributed downwards the hierarchy as illustrated in Figure 2. The master clocks distribute their time using `Sync` and `Follow_Up` messages. For slaves to accurately calculate the correct round trip delay, they send `Delay_Req` messages, which are answered by the master clock with `Delay_Resp` messages containing the precise timestamp when the message was received. Using those timestamps the slave then is able to calculate the correct time with sub-nanosecond precision. In Figure 3 the PTP header frame is illustrated as well as the body payload for `Sync` and `Delay_Req` messages which happen to be identical.

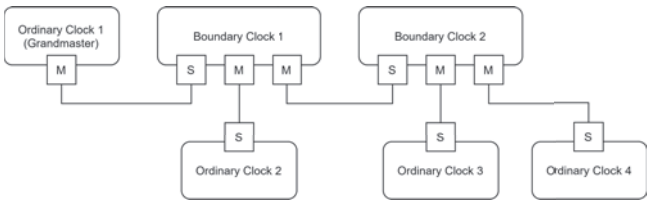



**Figure 2** Exemplary PTP architecture with one grandmaster clock distributing time downwards the hierarchy via boundary clocks. Figure based on [43].

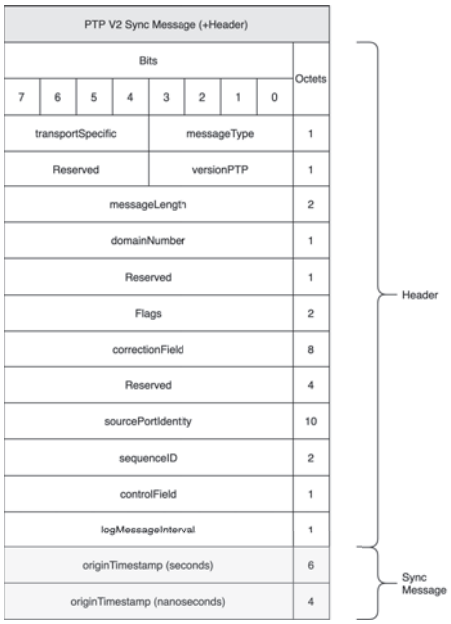



**Figure 3** PTP (V2) frame with header and `Sync` message sent by master clocks which also happens to be identical to the `Delay_Req` message sent by slave clocks. Figure based on [43].

As shown in Figure 3, the PTP header is 34 bytes long, consists of 14 fields and is the same for for all PTP messages. The `messageType` defines the type of message found in body payload, e.g. aforementioned `Sync` and `Delay` messages. The `messageLength` describes packet length of header and body combined. The `domainNumber` identifies the correlating logical grouping (domain) the clock belongs to. PTP specifies 12 different `flags` indicating various statuses (see [52] for reference). The `correctionField` contains a nanosecond correction value used for synchronization while also taking residence time and path delays into account. The `sourcePort Identity` specifies the originating port/clock. The `sequenceID` contains a sequence number for each message type. The `controlField` conforms to the PTP V1 and depends on the message type. The `logMessageInterval` depends on the type of message. The payload of PTP messages consists with only few exceptions mainly of either origin timestamps (e.g. in the case of `Sync` and `Delay_Req`) or receive timestamps. PTP timestamps are split into two parts, seconds are encoded using a 48 bit unsigned integer, nanoseconds 32 bit unsigned integer.

### 2.5 Covert Channels & Taxonomy

Network covert channels need legitimate network traffic as cover. They embed hidden information either by targeted modification of timing aspects of the traffic (timing channel) or by (over-)writing header and payload fields (storage channel) [29]. The aim of the covert communication partners is to keep their information exchange hidden, even in face of a so-called warden that can observe all ongoing communication.

The pattern-based taxonomy used in this paper was created by summarizing commonalities of dozens of channels in a wide variety of protocols. In this paper we use its version published in 2018 [28], in which eight timing- and twelve storage-patterns are distinguished. However, it is worth mentioning that in 2021 work on a revised taxonomy started in a joint effort [48].

Usually the distribution of so-called *steganographic keys* [51] takes place when covert channels are developed and evaluated. Since we are not taking an in-depth look at the embedding algorithms themselves, we do not further consider them as part of the evaluation.

### 2.6 Covert Channel Attributes

In the domain of information hiding it is common to differentiate between synthesis, modification and selection. In our case, we only consider *information hiding by modification*, the alteration of either timing aspects of network packets and flows (timing channels) or modifications of the contents of packets or flows (storage channels). It is common as well to define network traffic which is used as a cover for hidden information as *overt* communication whereas traffic containing hidden messages is called *covert*. As the aim of covert channels is generally to stay undetected (by the warden), the modified packets or flows need to be *plausible* to the warden. In [23] and [24] we described the plausibility as a combination of two kinds of compliance: protocol-compliance and warden-compliance. A network covert channel which employs information hiding by modification is considered protocol-compliant, if the modification to the cover object (packet or network flow) has no direct impact on the correct reception, acceptance and processing of that packet or flow. Warden-Compliance relates to the detectability and conspicuousness of a covert channel and defines three levels of compliance in regards to a warden: (1) the message is hidden in a way that a potential warden has no knowledge of the existence of a hidden message (inconspicuous), (2) the warden has a suspicion that there is a hidden message but can not access it and (3) the warden can identify and access but not reconstruct the hidden message.

Another common attribute of covert channels is the differentiation between *active* and *passive* information hiding. In an active scenario Alice and Bob are creating both the overt traffic as well as the covert traffic. In a passive scenario Alice and Bob are using overt traffic from other communication partners to hide their covert communication. In addition to that, in [23] we introduced the terms *semi-active* and *semi-passive* hiding which describe a combination of both active and passive hiding, where one of the overt communication partners is secretly communicating with device indirectly involved in the communication, e.g. network elements like firewalls and switches. This relates closely to the concept of *reversibility* in covert channels [27]. It describes whether a Man-in-the-Middle (MITM) can restore the overt information in an passive, or semi-passive scenario, before the information reaches the overt receiver (such MITM is called R-MITM).

## 3 Analysis of NTP

### 3.1 Analysis Approach

The general approach we follow in this paper can be divided in two major steps: 1. Systematically identifying potential covert channels based on modification using the taxonomy from [28] 2. Comprehensive comparison and analysis of these channels by using a set of commonly known properties: Protocol-compliance from [23], reversibility from [27], from information hiding the known properties of maximum and minimum capacity, as well as suspiciousness, three NTP specific criteria depth, direction and server-synchronization and one criterion which describes the impact of real-world implementations on the plausibility of the covert channel (contextual plausibility). We provide the first set of criteria in our work, but additional criteria for the comparison are possible and can be further elaborated.

#### 3.1.1 Systematic identification of covert channels

Our approach for systematically identifying potential covert channels by modification in NTP follows a two-stage procedure in combination with the previously described pattern-based taxonomy from [28].

**1. Pattern Application:** Here we take the hiding patterns from the pattern-based taxonomy and discuss if and how each pattern could be applied (in a theoretical manner) to each field found in such network packets and communication flows. This includes storage channels and timing channels considering the communication flows, which are defined in the specification.

The pattern-based taxonomy from Mazurczyk et al. [28] was derived by the authors in an deductive way by merging techniques from known covert channels into generalized categories or patterns. We reverse this approach and inductively apply those patterns to the specifics of the protocol.

**2. Real-world observations:** Further, we take a look at common implementations of NTP on a variety of OS. By analyzing the usual behavior of such OS implementations we are able to (1) identify potential covert channels which can not be found in the static analysis of the specification, e.g. such that are implementation-dependent and (2) check whether potential channels found in the static analysis of the specification are applicable in a plausible way in real-world scenarios and implementations. This can be achieved by setting up network environments in which the network traffic is recorded and then manually analyzed in regards to potential covert channels known from the patterns, e.g. timing properties, such as the order of network packets and fields.

### 3.1.2 Comparison criteria

As comparison criteria, nine key attributes are selected and proposed to describe every plausible covert channel identified in our work: **Protocol-compliance** [23] describes whether an overt receiver of an altered packet (containing hidden information) does accept and further process the packet or not. An extended model of **reversibility** [27] is used: A channel is *fully-reversible* in case the overt information can be fully restored by the R-MITM. It is *non-reversible* in case the overt information is lost without the chance of restoring it. It is *semi-reversible* in case the R-MITM can approximately restore the overt information in a way that the result does not significantly interfere with the processing on the receiver side. In NTP that could be that the fraction of a second value is restored with only a few bits difference. The **suspiciousness** is an (estimated) assessment of the likelihood that a warden will discover the covert channel, for instance used in [16]. We introduce the following scale: *High*: The channel can be detected with a simple syntax analysis of the packets. *Medium*: The channel can be detected with a semantic analysis of a single packet or the flow of packets. *Low*: The channel can be detected with statistical analysis of multiple packets. *Unsuspicious*: Even with advanced statistical evaluation the channel can not be detected. In respect to capacity see for instance [6]. The **maximum capacity** is the highest amount of hidden information that the covert channel can carry. In case the suspiciousness is the same, no matter how much information is

transferred, the **minimum capacity** is equal. In case it might grow with the used capacities, they are unequal. The capacity is specified as bits per packet (bits/pkt) for storage channels. For timing channels we do not elaborate the capacity, since it is heavily implementation depended.

The **depth** describes how far a hidden information can be passed alongside the Stratum architecture: *1* means that hidden information can only reach the direct neighbors (client and servers). $\approx$ (for approximately) means that hidden information might reach devices deeper in the architecture but not with the exact hidden information. Such could for instance happen in case the hidden information is embedded in the timestamp fields and information like a manipulated year reach deeper levels, but not the fractions of a second and therefore parts of the information are lost. $\infty$ (for infinity) guarantees that the hidden information can be send down the stratum hierarchy without information loss. The **direction** states whether the channel can be used in *client-*, *server-*, or *broadcast-mode*. The **server-synchronization**[2] states whether an overt client would remove a server using such channel from its list of valid sources. It is only relevant for channels in server packets. $ok$ means that the client would not remove the server. $e$ (for error) means that the client would remove the server at some point due to an error in the communication. $t$ (for timing) means that the client would remove the server for its bad timing information.

In order to describe the plausibility of a covert channel in a given context we use the term **contextual plausibility**. As described by Fridrich [12], the secret information (message) is hidden in innocuous data objects. To achieve innocuous characteristics, the cover selection process needs to choose a cover which is for the warden expected, such as usually used or commonly known and used data objects in the normal information (data) flow or behaviour. We call this also a plausible cover. In our evaluation, a network covert channel is considered contextual plausible, if the modified packet or flow is (1) syntactically and (2) semantically correct in the given context (e.g. protocol specification, vendor-specific implementation, network environment, configuration and point in time). This relates closely to the concept of protocol-compliance but in relation to a given context. The contextual plausibility then can be put into consideration when assessing the level of warden-compliance for a given covert channel in a specific context. In this evaluation regarding NTP, the contextual plausibility describes which influence the actual context has on the plausibility of the covert channel,

[2]The description from https://www.eecis.udel.edu/~mills/ntp/html/select.html is used

whereas the context here is described by one of six analyzed default OS implementations (see Table 1 for those results). A ∘ means that the context has no effect on the plausibility of the channel. A ↓ means that the channel can not be implemented as intended or that the resulting suspiciousness is higher in that context. A ↑ means that the implementation has a positive effect for the plausibility (and therefore is harder to detect).

### 3.2 Analysis Results

First, we evaluate NTP behavior in a selection of OS implementation. Secondly, we describe 49 discovered channels based on the methodology from Section 3.1.

### 3.3 Operating System Defaults

We evaluate the content of NTP client packets and the polling behavior of Windows 10 Enterprise Version 2004, Ubuntu 18.4 with timesyncd, Ubuntu 18.4 with chrony, macOS 10.15, Android 7 and iOS 13.7. They are installed in virtual machines (Windows, Ubuntu, macOS) or run on native devices (HTC Nexus Tablet, iPhone 11) behind a local test-gateway, where all NTP traffic is captured. The systems are installed with default configurations. With the evaluation of the captured network traffic it is possible to make statements about the plausibility of covert channels in real world environments.

The test results can be found in Table 1. They show that the clients differ significantly in their behavior. For macOS, iOS and Android it remains

**Table 1** The Table compares key properties of different NTP client implementations, r.p.u=Requests per update, p.c.p=Peer clock precision, p.p.i=Peer pooling interval, TC=true (changing value), R=fully random value, NULL=filled with zeros

| Property | Windows 10 | Ubuntu chrony | Ubuntu timesyncd | macOS | iOS | Android |
|---|---|---|---|---|---|---|
| Version | 3 | 4 | 4 | 4 | 4 | 3 |
| Update | 7 days | 54s-1204s | 32s-2048s | ? | ? | ? |
| r.p.u. | 1 | 3-20 | 1 | 3 | (3) | 1 |
| Ref. Time | TC | NULL | NULL | NULL | NULL | NULL |
| Tran. Time | TC | R | TC | TC | TC | TC |
| Ref. ID | TC | NULL | NULL | NULL | NULL | NULL |
| p.c.p | NULL | TC | NULL | NULL | NULL | NULL |
| p.p.i | TC | TC | NULL | TC | TC | NULL |

unclear how often updates are sent. A further analysis would either require the observation of the NTP traffic of those devices over months or longer or a dive into the code of the NTP implementations, both which are outside of scope in our research. From the observed behavior of Windows and Android we would suspect they rather use SNTP than NTP, since they synchronize with only one server.[3] We also found chrony using a scarcely documented security feature that randomizes all 64 bits in the transmit timestamp [7]. Overall, expectation is that anomaly detection in multi-device networks is comparatively harder due to such heterogeneous behavior.

### 3.3.1 Discovered channels

We discovered 49 plausible covert channels in NTP. In the following, we shortly describe each channel and compare the proposed key attributes in Table 2.

- **Message Timing**: MT1: Altering the polling interval between client requests.
- **Rate/Throughput**: RA1: Changing the number of client requests to one server, per synchronization cycle.
- **Artifical Loss**: AL1: corrupt the TEST1 checksum (content of the receive timestamp field) in a server response.
- **Message Ordering**: MO1: Altering the order in which a client is requesting time from its (three) servers.
- **Retransmission**: RT1: Denying a server response to a client requests, what will force the client to send another request after some while.
- **Sequence modulation**: SU1: Changing the order of both extension fields. Currently there is no application where the usage of both NTP fields is plausible. SU2: Altering the number of used extension fields. SU3: Altering the usage of the MAC field. Currently there is no practical relevance for the MAC fields. In case the server requires an authentication, the channel is not protocol-compliant. This channel reflects the findings from [46].
- **Random Value**: RV1: Using the complete root delay field. This is partly plausible since its value is expected to change either way in server responses. RV1a: Using only those parts that are outside of the precision of the system clock (indicated by the precision field). RV2 and RV2a: Doing the same with the Root Dispersion field.

[3] Even if it is differently documented by Microsoft and Google

**Table 2** Comparison of key attributes for covert channels. Direction in three columns: s(erver), c(client) and b(roadcast). Capacity in bits/pkt. T=Timing capacity. U=UDP packet size. Reversibility: reversible (✓), semi-reversible (s), non-reversible (✗). Suspiciousness: $h$: high, $m$: medium, $l$: low, $u$: unsuspicious. Server-synchronization: $e$: error, $ok$: no effect, $t$: timing. Depth: 1: one, $\infty$: infinity, $\approx$: approximately. Contextual plausibility: $\uparrow$: more plausible, $\circ$: no difference, $\downarrow$: less plausible. "-": Not applicable. "?": Unable to answer. **VM5**: Found in [2]. **SU3**, **UV6**: Found in [46]

| Channel Name | Min. Capacity | Max. Capacity | Reversible | Suspiciousness | Protocol-Compliant | Direction | | | Depth | Synchronization | Windows 10 | Ubuntu Chrony | Ubuntu timesyncd | macOS | iOS | Android |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Superordinate** | | | | | **NTP specific** | | | | | **Contextual OS Plausibility** | | | | | |
| MT1 | T | T | ✗ | $l$ | ✓ | c | ✗ | ✗ | 1 | – | $\downarrow$ | $\circ$ | $\circ$ | ? | ? | ? |
| RA1 | T | T | ✗ | $m$ | ✓ | c | ✗ | ✗ | 1 | – | $\downarrow$ | $\uparrow$ | $\downarrow$ | $\circ$ | $\circ$ | $\downarrow$ |
| AL1 | T | T | ✗ | $m$ | ✗ | ✗ | s | ✗ | 1 | $e$ | – | – | – | – | – | – |
| MO1 | T | T | ✗ | $m$ | ✗ | c | ✗ | ✗ | 1 | – | $\downarrow$ | $\uparrow$ | $\circ$ | $\circ$ | $\circ$ | $\downarrow$ |
| RT1 | T | T | ✗ | $m$ | ✗ | c | s | ✗ | 1 | $e$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ |
| SU1 | 1 | 1 | ✗ | $h$ | ✓ | c | s | b | 1 | $ok$ | $\downarrow$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\downarrow$ |
| SU2 | 1 | 1 | ✗ | $h$ | ✓ | c | s | b | 1 | $ok$ | $\downarrow$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\downarrow$ |
| **SU3** | 1 | 1 | ✗ | $h$ | ✗ | c | s | b | 1 | $ok$ | $\downarrow$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\downarrow$ |
| RV1 | 16 | 32 | s | $m$ | ✓ | ✗ | s | ✗ | $\approx$ | $t$ | – | – | – | – | – | – |
| RV1a | 1 | 16 | s | $l$ | ✓ | ✗ | s | ✗ | $\approx$ | $ok$ | – | – | – | – | – | – |
| RV2 | 16 | 32 | s | $m$ | ✓ | ✗ | s | ✗ | $\approx$ | $t$ | – | – | – | – | – | – |
| RV2a | 1 | 16 | s | $l$ | ✓ | ✗ | s | ✗ | $\approx$ | $ok$ | – | – | – | – | – | – |
| AR1 | 1 | 1 | ✓ | $h$ | ✓ | c | s | b | 1 | $ok$ | $\downarrow$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\downarrow$ |
| VM1 | 2 | 3 | s | $h$ | ✗ | c | s | b | 1 | $e$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ |
| VM1a | 1 | 1 | ✗ | $m$ | ✓ | c | s | b | 1 | $ok$ | $\downarrow$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\downarrow$ |
| VM2 | 1 | 1 | s | $m$ | ✓ | c | s | b | $\infty$ | $ok$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ |
| VM3 | 3 | 24 | ✗ | $h$ | ✗ | ✗ | s | ✗ | 1 | $e$ | – | – | – | – | – | – |
| VM3a | 24 | 24 | ✗ | $h$ | ✗ | ✗ | s | ✗ | 1 | $e$ | – | – | – | – | – | – |
| VM4 | 32 | 32 | ✗ | $h$ | ✓ | c | s | b | 1 | $ok$ | $\uparrow$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ | $\circ$ |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **VM5** | 1 | 8 | s | $m$ | ✓ | c | s | b | 1 | $ok$ | ○ | ↑ | ○ | ○ | ○ | ○ |
| VM6 | 1 | 8 | ✓ | $m$ | ✓ | c | s | b | 1 | $ok$ | ↑ | ↑ | ○ | ↑ | ↑ | ○ |
| VM7 | 1 | 7 | ✓ | $m$ | (✓) | ✗ | s | b | $\infty$ | $ok$ | – | – | – | – | – | – |
| VM8 | 1 | 4 | s | $m$ | ✗ | c | s | b | 1 | $ok$ | ↓ | ○ | ○ | ○ | ○ | ↓ |
| VM9 | 1 | 16 | ✗ | $m$ | ✓ | c | s | b | 1 | $ok$ | ↓ | ○ | ○ | ○ | ○ | ↓ |
| RU1 | 32 | 32 | ✓ | $m$ | ✓ | (c) | ✗ | b | 1 | – | ○ | ○ | ○ | ○ | ○ | ○ |
| RU2 | 32 | 32 | ✓ | $m$ | ✓ | (c) | ✗ | b | 1 | – | ○ | ○ | ○ | ○ | ○ | ○ |
| RU3 | 1 | 1 | ✓ | $h$ | ✗ | – | - | - | 1 | $ok$ | ○ | ○ | ○ | ○ | ○ | ○ |
| RU4 | 7 | 7 | ✓ | $h$ | ✓ | c | s | b | 1 | $e$ | ○ | ○ | ○ | ○ | ○ | ○ |
| RU5 | 32 | 32 | ✓ | $m$ | ✓ | c | ✗ | ✗ | 1 | – | ↓ | ○ | ○ | ○ | ○ | ○ |
| RU6 | 15 | 15 | s | $h$ | ✗ | c | s | b | 1 | $ok$ | ↓ | ○ | ○ | ○ | ○ | ↓ |
| PS1 | U/2 | U | ✓ | $h$ | ✗ | c | s | b | 1 | $ok$ | ↓ | ○ | ○ | ○ | ○ | ↓ |
| PS2 | U | U | ✓ | $h$ | ✓ | c | s | b | 1 | $ok$ | ↓ | ○ | ○ | ○ | ○ | ↓ |
| MR1 | 1 | 128 | ✓ | $m$ | ✓ | ✗ | s | ✗ | $\approx$ | $t$ | – | – | – | – | – | – |
| MR1a | 1 | 96 | ✓ | $m$ | ✓ | ✗ | ✗ | b | $\approx$ | – | – | – | – | – | – | – |
| MR1b | 1 | 64 | ✓ | $m$ | ✓ | c | ✗ | ✗ | $\approx$ | – | ○ | ○ | ○ | ○ | ○ | ○ |
| MR2 | 1 | 1 | ✓ | $l$ | ✓ | c | s | b | 1 | $ok$ | ↓ | ○ | ○ | ○ | ○ | ↓ |
| MR3 | 16 | 16 | ✓ | $h$ | ✗ | c | s | b | 1 | $ok$ | ↓ | ○ | ○ | ○ | ○ | ↓ |
| UV1 | 1 | 128 | s | $u$ | ✓ | c | s | b | 1 | $ok$ | ○ | ↑ | ○ | ○ | ○ | ○ |
| UV1a | 1 | 96 | s | $u$ | ✓ | ✗ | s | ✗ | 1 | $ok$ | – | – | – | – | – | – |
| UV1b | 1 | 64 | s | $u$ | ✓ | ✗ | ✗ | b | 1 | – | – | – | – | – | – | – |
| UV1c | 1 | 32 | s | $u$ | ✓ | c | ✗ | ✗ | 1 | – | ○ | ↑ | ○ | ○ | ○ | ○ |
| UV2 | 1 | 128 | ✗ | $l$ | ✓ | c | s | b | $\approx$ | $ok$ | ○ | ↑ | ○ | ○ | ○ | ○ |
| UV2a | 1 | 96 | ✗ | $l$ | ✓ | ✗ | s | ✗ | $\approx$ | $ok$ | – | – | – | – | – | – |
| UV2b | 1 | 64 | ✗ | $l$ | ✓ | ✗ | ✗ | b | $\approx$ | – | – | – | – | – | – | – |
| UV2c | 1 | 32 | ✗ | $l$ | ✓ | c | ✗ | ✗ | 1 | – | ○ | ↑ | ○ | ○ | ○ | ○ |
| UV3 | 64 | 64 | ✓ | $m$ | ✓ | c | ✗ | b | 1 | $t$ | ○ | ○ | ○ | ○ | ○ | ○ |
| UV4 | 1 | 64 | ✗ | $l$ | ✓ | c | ✗ | ✗ | 1 | – | ↑ | ○ | ○ | ○ | ○ | ○ |
| UV5 | U | U | ✗ | $u$ | ✗ | c | s | b | 1 | $ok$ | ○ | ○ | ○ | ○ | ○ | ○ |
| **UV6** | U | U | ✗ | $h$ | ✗ | c | s | b | 1 | $ok$ | ○ | ○ | ○ | ○ | ○ | ○ |

- **Add redundancy**: AR1: Using the second extension field with the same value as the first.
- **Value modulation**: VM1: Using the four upper bits in the version field that are currently unused (since only four NTP versions exists). VM1a: Switching between NTPv3 and NTPv4, which are compatible in IPv4 networks. VM2: Using the leap indicator that is typically zero most days in most years. VM3: Using one out of the 14 predefined KISS-codes in the reference id field and setting the stratum value to zero to indicate that a KISS-code is used. VM3a: Also using custom KISS-codes (starting with a X). VM4: Using the complete reference id field. VM5: Using the full precision field. While such channel is protocol compliant it does change the time calculation at the receiver side. This channel reflects the findings from [2]. VM6: Using the poll field. Its value does typically change only in small steps for synchronized clients so that the suspiciousness grows with the used capacity. VM7: Using the valid stratum values between 1 and 15. VM8: Modulating the extension field type between the 31 defined values. VM9: Using the MAC key identifier.
- **Reserved/Unused**: RU1: Using the root delay field that is unused in client and broadcast mode. RU2: Doing the same with the root dispersion field. RU3: Using the reserved values (0, 7) of the mode field. RU4: Using the stratum values 17-255 that are reserved. RU5: Using the reference id field, which is NULL in client mode. RU6: Using those extension field types that are not pre-defined.
- **Payload Field and Size Modulation**: PS1: Modulating the extension field size. PS2: Indicating a smaller size of the value and padding field of the extension field with the length field than the length of the UDP packet is (hiding the rear bits).
- **Modify Redundancy**: MR1: Years, months, days and often hours and minutes in the four 64 bit timestamp fields are equal in packets from synchronized clients/servers. Replacing some of the time information with hidden information and restoring the overt value from the context is possible. MR1a: Using only the transmit and reference timestamp in broadcast mode. MR1b: Using only the transmit timestamp in client packets. MR2: Varying the chosen MAC algorithm between the both defined types, AES and MD5 [1]. MR3: Filling the extension fields padding with hidden information instead of zeros.
- **User-data Value Modulation**: UV1: Using those parts of the second fraction parts in the 64 bits timestamp fields that are outside of the

precision and therefore normally randomized. UV1a: Using the reference, origin and receive timestamp but fill the transmit timestamp with the exact value of the requests transmit timestamp, in server mode. UV1b: Using the origin and receive timestamp, in broadcast mode. UV1c: Using only the transmit timestamp, in client mode. UV2(a/b/c): Equal to UV1 with the difference that all bits of the fraction part are used now. UV3: Using the complete receive timestamp that is unused in client and server mode. UV4: Using the complete reference timestamp that is unused in client mode. UV5: Using the complete dgst field. UV6: Using a complete extension field. This channel reflects the findings from [46].

We could not find patterns of type *Inter-packet Times*, since a complete flow of NTP packets always consists of maximum one NTP request that is answered with a response. Both, the extension fields and the MAC are considered to be non-relevant in real-world networks yet (they might become, when NTS will be adapted). All channels using such fields could in general be called highly suspicious. However, we assume that the usage of the fields is common and evaluate the suspiciousness accordingly.

### 3.4 Selected Implementations & Test Results

To show the practical application, in the following subsections we describe two channels that we selected, implemented and evaluated in network testbeds: Firstly, a practically undetectable channel UV1c – User-data Value Modulation using only the transmit timestamp in client mode and secondly deep-layer-channel VM7 – Value modulation using the valid stratum values between 1 and 15 among all discovered channels these two were chosen for implementation in order to validate the attacker model drawn in section 6 that requires a channel with low suspiciousness as well as a channel that can be used to send hidden data into isolated networks. The code of our implementations as well as the network captures can be found online.[4]

#### 3.4.1 Practically Undetectable Channel

We implement a covert channel based on UV1c and embed the hidden information in the last 16 bits of the fraction part of the transmit timestamp. The payload is encrypted with AES in ECB mode, using a dynamic key. The assumption is that the encryption creates an equally high entropy like the randomized bits in the timestamp. Following [4], a covert channel is

[4] https://github.com/ntpdrop/ieeesp2021

considered undetectable in case the entropy distributions of overt traffic and traffic containing the hidden information are equal. Since the content of overt transmit timestamps is implementation depended, the verification of our assumption is performed by comparing a sufficient number of overt packets and those containing hidden information: We request $n$=10,000 NTP packets from pool.ntp.org, time.google.com and a server that we setup locally, in two rounds. We are doing the same with our server that implements the covert channel. For our covert channel two different types of tests are performed: One where the covert sender (NTP server) generates as much packets as possible in a short amount of time and another where an artificial delay from 0 to 2 seconds between the client requests is added. Those test runs should help to evaluate whether the results are based on the time span of their sending or not. We then compare the entropy norm of the transmit timestamp for different $n$ out of these samples.

Entropy ($H$) is defined by $H = -\sum_{z \in Z} q_z \cdot \log_2 q_z$ with $q_z$ as the probability that an element $z$ of a given alphabet $Z$ occurs [41]. The alphabet used in our research is the one containing all digits: $Z = \{0, 1, 2, ...., 9\}$, with a size of $j = 10$. The *norm* of the entropy ($H_w$) transforms the entropy into an interval of $[0, 1]$, with 1 being the highest entropy value. It is defined by $H_w = \frac{H}{log_2 j}$. To calculate the entropy of the 32 bits in the fraction part of the transmit timestamp we use the following formula: For every one out of the digit positions in the fraction part ($d = 9$), the entropy is calculated individually. Then the average over all nine entropy values is drawn. This entropy ($H_c$) therefore can be calculated as $H_c = \frac{\sum_{i=0}^{d} H_{d_i}}{d}$ with $H_{d_i}$ as the entropy of the digit at position $i$.

The results show that our assumption holds true for our setup and that the entropy is equally high for overt traffic and traffic containing the covert channel. With other words: In all test cases the distribution of digits per position tends towards perfect entropy (i.e. 1). Both this is only true for sufficient large $n$ =10,000. For small $n \leq$ 1,000 neither the distribution nor the entropy are stable. As a result reliable detection the presence of this covert channel seems unlikely for the given test setup.

### 3.4.2 Deep-Layer-Channel

Another implemented channel is based on the manipulation of the stratum value in server packet fields (based on VM7). The assumption is that downstream servers (those with higher stratum numbers) will adapt dynamically

to changing stratum numbers and calculate their own stratum number accordingly. We want to test whether it is sufficient to manipulate only one out of three upstream servers, to send hidden information embedded in the stratum field (the alteration of its value) through multiple stratum layers. We therefore setup a network test-bed with three layers (see Figure 4). A server in the second layer requests time from three upstream servers. Out of those servers only one is malicious and changing its stratum value. The other two stay constant. Such a network setup would be common in any organizational network where a local time server is the only valid time source for local clients but this server by itself requests time from a public pool. In our test-bed, all NTP overt servers and clients use chrony with default configurations. During the tests we alter two parameters to optimize the robustness against packet loss of the channel: The polling interval of the client and how often the malicious server sends the same stratum value before it encodes the next bit. The test results show the following:

1. It is indeed possible to send hidden information through different network layers, even if only one of the upstream NTP servers is malicious: The requesting chrony client will adapt its own stratum number to the highest it received. Therefore, the hidden information can be passed on as long as it is encoded with higher stratum numbers than those of the overt servers.
2. A chrony client will change its own stratum value immediately whenever it receives a new value from an upstream server (as long as the client is synchronized).
3. The alteration of stratum values is not handled as error by chrony. The time calculation is therefore not affected.
4. The channel can only be used robustly if the stratum values encoding the hidden information are not changed after every request of the intermediate server. Since the timing of its request is not synchronized with those of the malicious client, information might be lost.
5. The bandwidth of the channel is low and depends on the polling interval of the intermediate server (less then 1 bit/min in case of a synchronized chrony client).

Since this channel is easy to detect with simple Network Intrusion Detection System (NIDS) rules and the bandwidth is low, it can not be considered a general threat. However, it might be interesting in those scenarios where an attacker needs to send small command queries from the Internet to a device, deeply embedded in an industrial network.

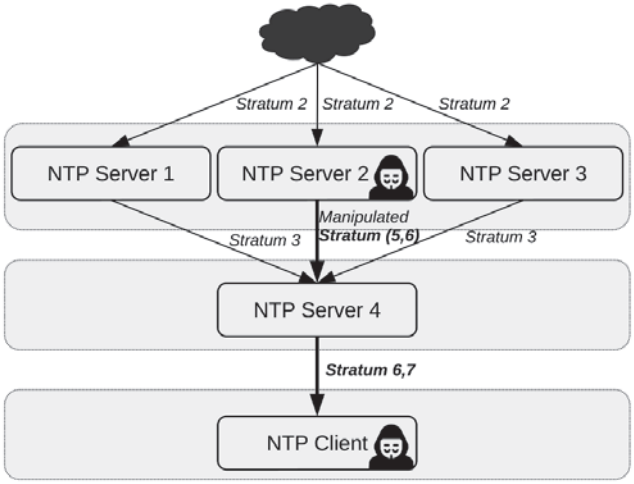



**Figure 4** The covert channel two uses the stratum values in NTP server packets to send hidden information from a malicious NTP server (2) through an non-involved server (4) to a client. Server 4 is requesting time from three upstream servers (1,2,3), but only server 2 is malicious.

## 4 Countermeasures

The primary step towards countermeasures against covert channels usually is detection [31]. In the literature a wide variety of anomaly detection, statistical analysis and classifier training approaches exist [53]. Due to the high number of potential channels and high entropy in NTP server packets, detection is often erroneous and not a promising approach to counter covert channels in NTP. Therefore, we propose two active measures: Normalization and a certain network setup, which can eliminate, distort or modify covert channels, especially those working on server packets or using reserved values.

### 4.1 Normalization

Normalization is a countermeasure where an active network element, e.g. a Firewall or Intrusion Prevention System (IPS) like Snort,[5] changes the content of network packets to a normalized value [14]. This is especially relevant for unencrypted protocols like NTP, since a normalization of all fields of the network is possible.

[5] https://www.snort.org/

The potential for normalization in NTP client packets was already discussed in [10]: "The stratum, Root Delay, Root Dispersion, Reference ID, Reference timestamp, Origin timestamp, and Receive timestamp SHOULD be set to zero". Combined with the results from our evaluation of chrony's default behavior it can be concluded that NTP client packets can be normalized. Only the version, the mode and the transmit timestamp fields are required in NTP client requests and all three can be set to static values (the transmit timestamp is set to a random value in chrony but it could be set to any other value instead). A traffic normalizer can therefore be used to suppress covert channels in at least one direction.

In server (and broadcast) packets the potential for normalization is smaller: The content of the the root dispersion, root delay, reference timestamp and origin timestamp are required for correct time calculation. All other fields can be normalized as well. In these timestamp fields at least the last bits can be normalized without a relevant effect on the time calculation. With this approach the high entropy can be reduced or removed.

NTP packets should be normalized to a size of 48 bytes. As extension fields and the MAC field (see Figure 1) seem to have no practical relevance they should not be allowed,further reducing the attack surface. The same holds true for NTP control queries.

### 4.2 Network Setup

Several authors and guidelines suggest to set-up local NTP servers in organization networks and block all direct NTP requests from any other clients [38, 45]. This can also be recommended to prevent covert channels. In difference to other protocols where the set-up of intermediate servers is suitable (e.g. DNS), an NTP client request to a local server has no effect on the communication behavior of the local server: It will answer the clients request directly and requests its own time from upstream-servers independently. For comparison, in DNS a server would need to directly access information from public DNS servers, in case the corresponding address is not cached. Therefore, the setup of a local NTP server can suppress all channels in direction from local clients to public servers. As shown in our practical evaluation, would this approach still not block incoming covert channels. Therefore, it is additionally relevant to use only trusted upstream NTP servers [3]. The usage of public pools with thousands of different servers and IP addresses is not recommended from the perspective of covert channel suppression. Like previously shown in [21] and [22], is the securing of DNS

important in this context. The following exemplary, simple firewall rules in local networks can reduce the attack surface for local covert channels and can be seen as a countermeasure of elimination: (1) Block NTP packets using extension fields and MAC. (2) Block NTP packets with control queries. (3) Block NTP broadcast packets. (4) Redirect all client requests to the local NTP server. (5) Block all NTP server packets that do not originate from the local NTP server. (6) Block all NTP packets containing KISS-codes.

## 5 Discussion

While the capacity of some covert channels in NTP can be considered high (e.g. UV1 and UV2 with 128 bits out of the full 384 bits per NTP packet), the practical bandwidth is limited. This is due to the fact that synchronized NTP clients only poll time every few minutes (in case of Ubuntu), or even only once per week (in case of Windows). Increasing the bandwidth by just increasing the number of NTP packets would lead to a trivial detectable anomaly.

### 5.1 Bandwidth & Plausibility

The plausible bandwidth is limited. Even if the maximum capacity of 128 bits/pkt and the lowest polling interval of 32 seconds is used, the 240 bits/min are not enough to exfiltrate large amounts of (potentially confidential) data. Therefore, NTP can not be considered a general purpose carrier for all use cases, since a higher bandwith can only be achieved by sending more NTP packets which would easily trigger alerts in NIDS and other network monitoring systems. However, especially for the drawn attacker model NTP is a suitable and plausible carrier. The high entropy of the evaluated channel might also not hold for smaller network samples and should evaluated in real-world network traffic inside organizational IT- and OT-networks.

### 5.2 Limitations of the Pattern-based Taxonomy

While a large number of potential covert channels are identified by applying the pattern-based taxonomy, there is no guarantee that the list is in any means complete. Another fact is, that the pattern-based taxonomy is only based on channel types already evaluated in the past. It can not fully support identification of new types of covert channels. Also, the evaluation of covert channels in application layer protocols is relatively new [30]. However, it is not clear whether other patterns would classify covert channels on this

layer more suitable. The original pattern-based taxonomy [50] was extended twice [28, 30], showing the potential for further improvements and additions.

## 6 Threat Scenarios

In this section we describe two threat scenarios to illustrate how time synchronisation processes of NTP and PTP could be leveraged for covert communication. In case of NTP, we use the previously discussed practically undetectable covert channel (UV1c) from Section 3.4.1 and put into the context of a more complex threat scenario. For better comparison we use a similar scenario for PTP in which the timestamps are modulated to encode hidden information. This covert channel is covered in more detail in Section 7. Threat scenarios are often described using common descriptive frameworks like the Lockheed Martin Cyber Kill Chain®[6] and MITRE ATT&CK®.[7] However, information hiding or covert channels are not mentioned in either of these frameworks. However, covert channels seem most likely to be used in the Command & Control (C2) phase, found in both frameworks. Therefore, for the description of our threat scenarios we assume that all prior steps were already performed by the adversary. In case of the Lockheed Martin Cyber Kill Chain® these are the steps of *Reconnaissance, Weaponization, Delivery, Exploitation and Installation*. For our threat scenarios this means, that we assume that the adversary already achieved to compromise a system within the targeted network perimeter and now wants to communicate with that system, for example in order to send further instructions, other malware components or to exfiltrate information. For the sake of simplicity, here we focus of the infiltration part of C2 (however, as discussed in the previous sections exfiltrating covert channels in NTP are possible, too.) In case of MITRE ATT&CK®, the adversary performed *Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Priviledge Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement and Collection*. However, some of these steps are not mandatory or might be performed in different order. What makes covert channels in NTP and PTP so interesting for adversaries in the Command & Control phase is the dependency on *external* reference clocks. In case of NTP this either done via network, e.g. by contacting servers on the internet (NTP Pool) or using specific receivers and antennas for using reference time distributing radio transmissions from

[6] https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html
[7] https://attack.mitre.org/

Global Navigation Satellite Systems (GNSS) or official local radio stations. This external dependency can be seen as Achilles heel in the time synchronisation process when it comes to covert channels. The basic idea here is to modify the reference time, embed a hidden message and wait till the modified timestamps get distributed towards the compromised system in the target network. In the following, we describe how this basic threat scenario would play out in the cases of NTP and PTP.

### 6.1 Covert Channel Threat Scenario: NTP

In this section we describe an exemplary attack scenario for the Network Time Protocol, NTP, in which an adversary establishes a covert command-and-control (C2) channel to send instructions or further malware components to a compromised system within a target network perimeter. As illustrated in Figure 5, the selected NTP scenario assumes an architecture with one local Stratum-2 server, which synchronizes to an external clock and is distributing this reference time downwards the hierarchy in the local network upon request (server-client architecture). For such a central corporate time server multiple options are available for external clock synchronisation: *Option A* is to use public available NTP server pools to synchronize time. *Option B* is to use the time signal distributed over Global Navigation Satellite Systems (GNSS), like GPS,[8] GLONASS[9] and Galileo.[10] Finally, *Option C* is to use reference times from radio transmissions, for example DCF77[11] in Germany or WWV[12] in the United States. Though multiple options are available, each of these provides adversaries enough attack surface to alter the time signals. In our case, in order to encode hidden messages into the timestamps to covertly infiltrate instructions or further malware components into the target network. In case of *Option A*, the first step for the adversary is to compromise the used NTP server pool, e.g. by adding own malicious NTP servers to a public pool or by taking control over them directly. In the context of the aforementioned attack frameworks this scenario would require proper

[8] https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps

[9] https://www.glonass-iac.ru/en/about_glonass/

[10] https://www.gsc-europa.eu/system-service-status/constellation-information

[11] https://www.ptb.de/cms/ptb/fachabteilungen/abt4/fb-44/ag-442/verbreitung-der-gesetzlichen-zeit/dcf77.html

[12] https://www.nist.gov/pml/time-and-frequency-division/time-distribution/radio-station-wwv

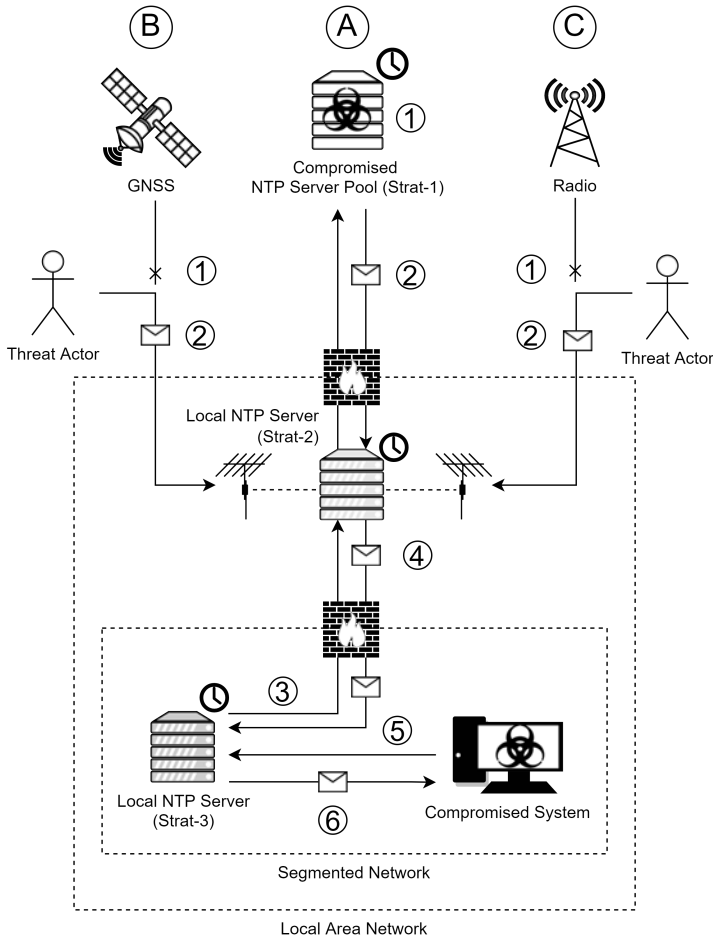



**Figure 5** Threat Scenario leveraging the Network Time Protocol as carrier for data infiltrating covert channels.

reconnaissance to be able to tell which NTP pool is used by the corporate time server. Moreover, this scenario requires a full scale attack on the selected NTP server pool. This circumstances render this scenario rather difficult and would require high amounts of resources. Though, as such capabilities are often found in the context of Advanced Persistent Threats and are not unusual in real world incidents (as such a scenario is quite similar to the scale of known supply-chain attacks in the past) the described threat scenario seems plausible in that case, too. For example, the solar storm supply chain

attack[13] is quite similar in terms of capabilities and scale and illustrates the imminence of such threats. However in the case of the two other scenarios (*Option B and C*) using radio transmissions the attack vectors require even less resources and capabilities. In case of *Option B* and *Option C*, the first step for the adversary is to physically position herself in the near of the antenna of the receiving device. For example, a parking lot in front of the target building would be fully sufficient to jam the radio signal originating from global navigation satellite systems (*Option B*) or radio stations (*Option A*). As such radio systems use common frequencies cheap jamming devices are publicly available and can be implemented using low-cost software defined radios (SDR).[14] By jamming the original signal, the adversary achieves that the original reference time is not reaching the corporate time server. Instead, the adversary encodes a hidden message into the timestamp (see channel UV1c in 3.4.1 for reference) and emits the modified signal. By jamming the original signal and transmitting the modified signal in the near of the antenna, the adversary can make sure that only the modified signal is received [8]. Now, the central corporate time server has received the modified reference time. This step is critical as it marks the intrusion of illicit information into the network perimeter. As shown in 5, the next steps (3,4,5,6) cover the distribution towards the targeted system within this network. Additionally, to highlight the threats potential we assume that the compromised system is located within a firewall segmented network. This segmented network uses a separate Stratum-3 time server, so that the clients in the segmented network only communicate within the segmented network. However, the Stratum-3 server synchronizes with the corporate Stratum-2 time server located in the local area network. In Step 3 the Stratum-3 requests the time from the Stratum-2 server and receives in Step 4 the reference timestamp with the embedded hidden message. In Step 5 the compromised system within the segmented network requests the time from the Stratum-3 server located in the segmented network as well. In Step 6 the Stratum-3 server responds the synchronization request with the embedded message. The compromised system can now retrieve the hidden information.

### 6.2 Covert Channel Threat Scenario: PTP

As NTP and PTP share many similarities, so do the corresponding threat scenarios. An important finding, as it illustrates how time synchronisation

[13] https://unit42.paloaltonetworks.com/solarstorm-supply-chain-attack-timeline/

[14] https://greatscottgadgets.com/hackrf/

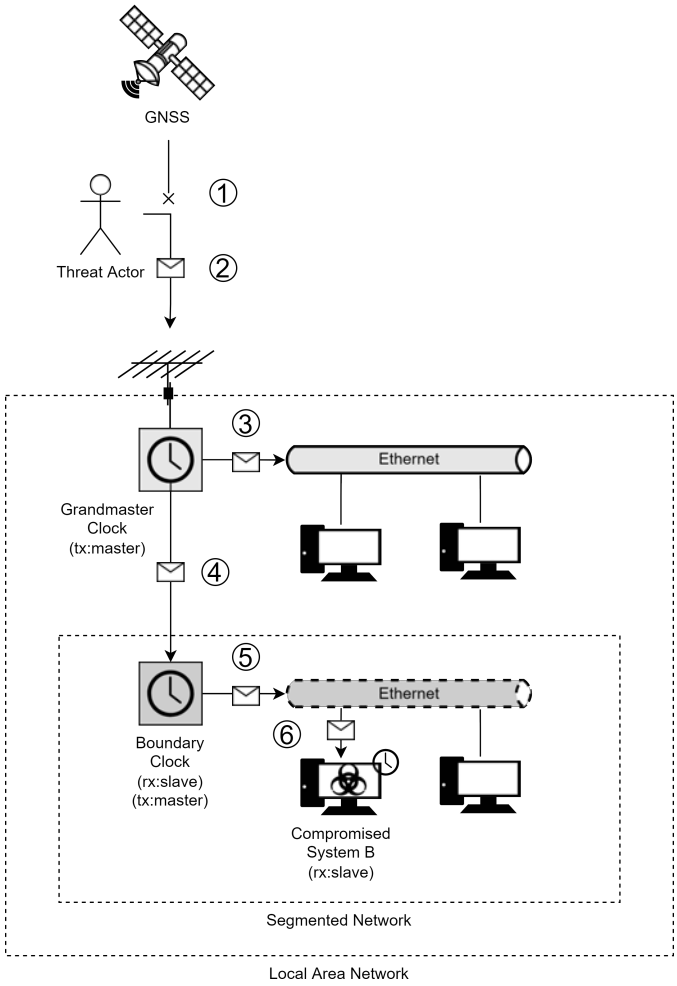



**Figure 6** Threat Scenario leveraging the Precision Time Protocol as carrier for data infiltrating covert channels.

imposes a threat to environments of Operational Technology, e.g. in the industry sector. As shown in Figure 6, the general concept remains the same with two exceptions: First, the major difference is that due to the required accuracy of PTP, grandmaster clocks usually get their reference time only from GNSS satellites as they provide the best accuracy. There are WAN based scenarios, however, the resulting latency renders high-precision synchronisation nearly impossible. Therefore, in most scenarios GNSS is used. This of

course, especially in comparison to the prior described NTP scenario limits the available options to and adversary and therefore reduces the attack surface in parts. However, the approach of manipulating the radio transmissions originiating from the satellites remains the same. In Step 1, the original signals are jammed using a Software-Defined-Radio. In the Step 2 the adversary uses the original signal to modify it and embedd a hidden message. This manipulated signal is then transmitted using another SDR (see [8] for reference). As this signal has a higher signal strength (the adversary emits the signal in the near of the antenna, jams the original signal, which is more than sufficient to achieve a higher signal strength than the original signal from the satellites.) The other difference lays in the architecture of PTP as described in Section 2.2 and illustrated in 2, as master clocks distribute their time as broadcast on the network. This happens in Step 3 as the grandmaster clock distributes the manipulated time signal it received via GNSS on its local area (sub-)network. Therefore, anyone on the same subnet could receive the hidden message. This is another interesting fact of PTP, as this means, that multiple infected systems could be communicated with at the same time. However, as we did in the NTP scenario, here we assume the compromised system to be located in a segmented network (see Figure 6). In order to distribute the time to this segmented network a boundary clock is used, that acts as slave towards the grandmaster clock and receives the reference time from there. This is illustrated in Step 4 as the boundary clock, which can be see as a broker between the normal LAN and segmented LAN, receives the manipulated time signal from the grandmaster clock as a slave clock. In Step 5 the boundary clock than acts as a master clock and distributes its time to the segmented network LAN. Finally, From there the compromised system can receive the broadcasted (hidden) message in Step 6. In summary, it can be noticed that even though the architecture and communication procedures differ in detail the high-level threat scenario remains similar between NTP and PTP.

## 7 The Threat of Covert Channels in PTP

While NTP and PTP have many similarities on a functional level, the actual protocol fields and communication flows differ significantly. Therefore, here we focus on the main similarity and most promising characteristic for covert channels, the transmission of high-resolution timestamps. As discussed before, the proposed NTP covert channel makes use of the fact that random bits are inserted into the timestamps as the hardware does not match the resolution of the network timestamps. However, the major difference of

NTP and PTP is better accuracy/precision due to specialized hardware using high accurate clocks. Still, accuracy of PTP is estimated to be around 200 nanoseconds (in best case scenarios). As such fluctuations follow a gaussian distribution [44], an adversary can assume that this leaves at least 7 bits of fluctuations in the 32bit nanosecond timestamp to be used for embedding hidden information. When keeping the modifications such small and within usual fluctuations and inaccuracies, it is expected to be hard to detect as shown previously for NTP. Furthermore, in the case of PTP staying within the usual distribution of inaccuracies is is at utmost importance as not properly synchronized clocks might have an impact on the Operational Technology and the controlled processes. The resulting bandwidth depends on the intervals between the messages used for embedding: A common interval for `Sync` messages is one second, which would lead to a bandwidth of 7 bit per second or 48kb per minute, respectively. To put that in perspective, a transfer of one 128 bit cipher block would require 19 seconds. On first sight, this might seem to be a low bandwidth, however, in the given threat scenario, it would be sufficient to load further malware components into the target network within one day. This seems viable for many scenarios, as common attacks, especially in the OT domain, are performed over weeks, months or even years. Regarding other covert channels, for example storage channels in the header fields it can be said that when analyzing the header fields (see Section 2.4) many opportunities come to mind (e.g. by using the reserved fields). However, when analyzing publicly available network captures of PTP traffic (e.g. [52]), it quickly becomes clear, that such modifications would either break the communication (e.g. when altering the `sourcePortIdentidity`) or would be very obvious and easily detectable as these fields are either static or not used at all. Timing channels seem to be an option though, e.g. the modulation of message timing (see MT1 in the NTP analysis for reference) in case of master clocks `Sync` messages. However, such channels are mostly protocol-agnostic, yet implementation depended. Furthermore, they usually incorporate even less bandwidth, e.g. in the case of the aforementioned `Sync` messages the bandwidth would be estimated to be around 1bit/second. Timing channels can also interfere with the time synchronisation process, as they might introduce additional (non-deterministic) delays in the communication between clocks. In summary, PTP can be seen as a promising candidate for covert channels, similar to NTP. Especially higher resolution timestamps make PTP interesting for injecting covert messages. On the other hand, due to the use of high precision clocks and the resulting higher accuracy of PTP covert channels might have a influence on the time synchronisation

process when not carefully implemented by an adversary. In most cases, covert channels in PTP require prior reconnaissance to fit the covert channel to the actual implementation. In order to stay undetected an adversary has to monitor PTP traffic and analyze which PTP fields could be used for storage channels and how much modification of the timestamps are viable without interfering with the time synchronisation process. Therefore, such covert channels seem most likely to be carried out in threat scenarios with high amounts of resources, e.g. in the case of Advanced Persistent Threats (APT).

## 8 Conclusion

In this work, we shed light on the potential threat of covert channels in network time synchronisation protocols at the example of NTP and PTP. At the example of two threat scenarios we could illustrate the attack surface revealed by requirements of highly accurate timestamps and external clock synchronisation. We applied a pattern-based taxonomy on NTP to discover plausible covert channels by modification. 49 channels were discovered and compared with key attributes. By taking six NTP default implementations into account, we could also show the contextual plausibility in real-world scenarios. Two channels were implemented in network test-beds. The evaluation shows that practical undetectable channels are possible in NTP, due to its high payload entropy. The data redundancy in NTP server packets also makes reversible covert channels plausible. By exploiting the stratum number in NTP packets the implementation of robust covert channels through multiple network layers seems plausible. We consider a small subset of the covert channels in NTP hard or impossible to detect. However, with certain network setups and traffic normalization most channels can be suppressed. While NTP traffic can be found in most IT- and CPS- networks, the number of client requests per hour is limited and by that the resulting covert channel bandwidth. Therefore, the potential of NTP as a carrier for covert channels lies in low-attention, low-bandwidth C&C channels, especially in industrial networks. This is especially true for the Precision Time Protocol, which shares many similarities with NTP. Our systematic approach and its two-stage procedure allows for an in-depth analysis of NTP. While not all potential covert channels are necessarily found by this method, it provides the examiner a systematic procedure to identify potential covert channels. Its generic approach allows for using it for other network protocols with only minor adjustments. As the threat of covert channels is seemingly growing [26, 42], this approach might also render useful during the development and specification process of

network protocols. This might especially be helpful in cases were general security considerations are at odds with considerations to suppress covert channels. Further work on covert channels in NTP should take the distribution and usage of steganographic keys into account, when algorithms, based on the discovered channels, are implemented and evaluated. Moreover, as highlighted in the discussion on the applicability of the findings regarding NTP to the specifics of PTP, a systematic analysis for PTP should be performed as well. Furthermore, practical tests for PTP are required, to investigate in which way covert channels might affect the clock synchronisation process.

## Acknowledgements

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

## References

[1] A. Malhotra and S. Goldberg. Message Authentication Code for the Network Time Protocol. *Internet Engineering Task Force*, June 2019. https://www.rfc-editor.org/rfc/rfc8573.html.

[2] Aidin Ameri and Daryl Johnson. Covert Channel over network time protocol. In *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy*, ICCSP '17, page 62–65, New York, NY, USA, 2017. Association for Computing Machinery.

[3] M. Bishop. A security analysis of the NTP protocol version 2. In *[1990] Proceedings of the Sixth Annual Computer Security Applications Conference*, pages 20–29, 1990.

[4] Christian Cachin. An Information-Theoretic Model for Steganography. *Inf. Comput.*, 192(1):41–56, July 2004.

[5] L. Caviglione, M. Gaggero, J. Lalande, W. Mazurczyk, and M. Urbański. Seeing the unseen: Revealing mobile malware hidden communications via energy consumption and artificial intelligence. *IEEE Transactions on Information Forensics and Security*, 11(4):799–810, 2016.

[6] Rajarathnam Chandramouli and Nasir Memon. Steganography capacity: A steganalysis perspective. *Proc SPIE*, pages 173–177, 06 2003.

[7] chrony. Comparison of NTP implementation. August 2018. https://chrony.tuxfamily.org/comparison.html.
[8] Takuji Ebinuma. GPS-SDR-SIM, December 2021. https://github.com/osqzss/gps-sdr-sim.
[9] P. Ferrari, P. Bellagente, A. Depari, A. Flammini, M. Pasetti, S. Rinaldi, and E. Sisinni. Evaluation of the impact on industrial applications of ntp used by iot devices. In *2020 IEEE International Workshop on Metrology for Industry 4.0 IoT*, pages 223–228, 2020.
[10] D. Franke and A. Malhotra. NTP Client Data Minimization. *Internet Engineering Task Force*, 2019. https://tools.ietf.org/id/draft-ietf-ntp-data-minimization-04.html.
[11] D. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sunblad. Network Time Security. *Internet Engineering Task Force*, March 2020. https://tools.ietf.org/html/draft-ietf-ntp-using-nts-for-ntp-28.
[12] Jessica Fridrich. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, USA, 1st edition, 2009.
[13] B. Haberman, D. Mills, and U. Delaware. Network Time Protocol Version 4: Autokey Specification. *Internet Engineering Task Force*, June 2010. https://tools.ietf.org/html/rfc5906.
[14] Mark Handley, Vern Paxson, and Christian Kreibich. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *Proceedings of the 10th Conference on USENIX Security Symposium – Volume 10*, SSYM'01, USA, 2001. USENIX Association.
[15] Jonas Hielscher, Kevin Lamshöft, Christian Krätzer, and Jana Dittmann. A systematic analysis of covert channels in the network time protocol. In *The 16th International Conference on Availability, Reliability and Security*, pages 1–11, 2021.
[16] Mario Hildebrandt, Robert Altschaffel, Kevin Lamshöft, Matthias Lange, Martin Szemkus, Tom Neubert, Claus Vielhauer, Yongjian Ding, and Jana Dittmann. Threat analysis of steganographic and covert communication in nuclear I&C systems. In *International Conference on Nuclear Security 2020*, Februray 2020.
[17] IEEE. IEEE 1588-2002 – IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. https://standards.ieee.org/standard/1588-2002.html.
[18] IEEE. IEEE 1588-2008 – IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. https://standards.ieee.org/standard/1588-2008.html.

[19] IEEE. IEEE 1588-2019 – IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. https://standards.ieee.org/standard/1588-2019.html.

[20] Internet Assigned Numbers Authority. NTP Kiss-o'-Death Codes. *Network Time Protocol (NTP) Parameters*, March 2010. https://www.iana.org/assignments/ntp-parameters/ntp-parameters.xhtml.

[21] P. Jeitner, H. Shulman, and M. Waidner. Pitfalls of Provably Secure Systems in Internet the Case of Chronos-NTP. In *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, pages 49–50, 2020.

[22] P. Jeitner, H. Shulman, and M. Waidner. The Impact of DNS Insecurity on Time. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 266–277, 2020.

[23] Kevin Lamshöft and Jana Dittmann. Assessment of hidden channel attacks: Targetting modbus/tcp. *IFAC-PapersOnLine*, 53(2):11100–11107, 2020. 21th IFAC World Congress.

[24] Kevin Lamshöft, Tom Neubert, Mathias Lange, Robert Altschaffel, Mario Hildebrandt, Yongjian Ding, claus Vielhauer, and Jana Dittmann. Novel challenges for anomaly detection in i&c networks: Strategic preparation for the advent of information hiding based attacks. *Atw. Atomwirtschaft*, 65:504–508, 10 2020.

[25] Norka B. Lucena, Grzegorz Lewandowski, and Steve J. Chapin. Covert channels in ipv6. In George Danezis and David Martin, editors, *Privacy Enhancing Technologies*, pages 147–166, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[26] Wojciech Mazurczyk and Luca Caviglione. Information Hiding as a Challenge for Malware Detection. *IEEE Security & Privacy*, 13(2):89–93, Mar 2015.

[27] Wojciech Mazurczyk, Przemysław Szary, Steffen Wendzel, and Luca Caviglione. Towards Reversible Storage Network Covert Channels. In *Towards Reversible Storage Network Covert Channels*, 08 2019.

[28] Wojciech Mazurczyk, Steffen Wendzel, and Krzysztof Cabaj. Towards Deriving Insights into Data Hiding Methods Using Pattern-based Approach. In *Towards Deriving Insights into Data Hiding Methods Using Pattern-based Approach*, 08 2018.

[29] Wojciech Mazurczyk, Steffen Wendzel, Sebastian Zander, Amir Houmansadr, and Krzysztof Szczypiorski. *Background Concepts, Definitions, and Classification in Information Hiding in Communication*

*Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*, chapter 2, pages 39–58. John Wiley & Sons, Ltd, 2016.

[30] Wojciech Mazurczyk, Steffen Wendzel, Sebastian Zander, Amir Houmansadr, and Krzysztof Szczypiorski. *Information Hiding in Communication Networks: Fundamentals, Mechanisms, and Applications*. Wiley-IEEE Press, 03 2016.

[31] Wojciech Mazurczyk, Steffen Wendzel, Sebastian Zander, Amir Houmansadr, and Krzysztof Szczypiorski. *Network Steganography Countermeasures*, chapter 8, pages 207–242. John Wiley & Sons, Ltd, 2016.

[32] Aleksandra Mileva and Boris Panajotov. Covert channels in tcp/ip protocol stack – extended version-. *Central European Journal of Computer Science*, 4:45–66, 06 2014.

[33] Aleksandra Mileva, Aleksandar Velinov, Laura Hartmann, Steffen Wendzel, and Wojciech Mazurczyk. Comprehensive analysis of mqtt 5.0 susceptibility to network covert channels. *Computers & Security*, 104:102207, 2021.

[34] D. Mills. *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI*, January 2006. https://tools.ietf.org/html/rfc4330.

[35] D. Mills. Control Messages Protocol for Use with Network Time Protocol Version 4, October 2011. https://tools.ietf.org/id/draft-odonoghue-ntpv4-control-00.html.

[36] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network Time Protocol Version 4: Protocol and Algorithms Specification. *Internet Engineering Task Force*, 06 2010. https://tools.ietf.org/html/rfc5905.

[37] Steven J. Murdoch and Stephen Lewis. Embedding covert channels into tcp/ip. In Mauro Barni, Jordi Herrera-Joancomartí, Stefan Katzenbeisser, and Fernando Pérez-González, editors, *Information Hiding*, pages 247–261, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[38] NTP FAQ. How should I provide NTP services for a huge network? September 2019. https://www.ntp.org/ntpfaq/NTP-s-config-adv.htm.

[39] Stephen Röttger. Analyse des NTP-Autokey-Verfahrens (German). *TU Braunschweig – Institut für Theoretische Informatik*, September 2011.

[40] Tobias Schmidbauer and Steffen Wendzel. Covert Storage Caches using the NTP Protocol. *ARES 2020: Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020.

[41] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[42] SIMARGL. Stegware – the latest trend in cybercrime. February 2019. https://simargl.eu/blog/technical/stegware-the-latest-trend-in-cybercrime.

[43] Calnex Solutions. Implementing ieee 1588v2 for use in the mobile backhaul, 2009.

[44] Endace Technology. Ieee 1588 ptp clock synchronization over a wan backbone, 2016. https://www.endace.com/ptp-timing-whitepaper.

[45] Timur Snoke. Best Practices for NTP Services. April 2017. https://insights.sei.cmu.edu/sei_blog/2017/04/best-practices-for-ntp-services.html.

[46] Nikolaos Tsapakis. Alternative communication channel over NTP. *Virus Bulletin*, April 2019. https://www.virusbulletin.com/virusbulletin/2019/04/alternative-communication-channel-over-ntp/.

[47] Steffen Wendzel. Get Me Cited, Scotty! Analysis of Citations in Covert Channel/Steganography Research. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ARES 2018, New York, NY, USA, 2018. Association for Computing Machinery.

[48] Steffen Wendzel, Luca Caviglione, Wojciech Mazurczyk, Aleksandra Mileva, Jana Dittmann, Christian Krätzer, Kevin Lamshöft, Claus Vielhauer, Laura Hartmann, Jörg Keller, and Tom Neubert. A Revised Taxonomy of Steganography Embedding Patterns. In *The 16th International Conference on Availability, Reliability and Security*, ARES 2021, pages 1–12, New York, NY, USA, August 2021. Association for Computing Machinery.

[49] Steffen Wendzel, Wojciech Mazurczyk, Luca Caviglione, and Michael Meier. Hidden and Uncontrolled – On the Emergence of Network Steganographic Threats. In Helmut Reimer, Norbert Pohlmann, and Wolfgang Schneider, editors, *ISSE 2014 Securing Electronic Business Processes*, pages 123–133, Wiesbaden, 2014. Springer Fachmedien Wiesbaden.

[50] Steffen Wendzel, Sebastian Zander, Bernhard Fechner, and Christian Herdin. Pattern-Based Survey and Categorization of Network Covert Channel Techniques. *ACM Computing Surveys*, 47:50:1–26, 04 2015.

[51] Andreas Westfeld and Andreas Pfitzmann. Attacks on steganographic systems. In Andreas Pfitzmann, editor, *Information Hiding*, pages 61–76, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

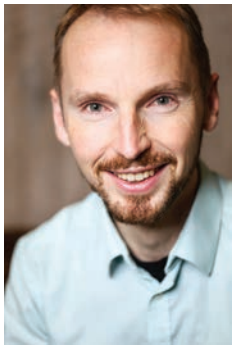
[52] Wireshark. Protocols/ptp – The Wireshark Wiki. https://wiki.wireshark.org/Protocols/ptp, 2021.

[53] Sebastian Zander, Grenville Armitage, and Philip Branch. Covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys and Tutorials*, 9:44–57, 09 2007.
[54] Elzbieta Zielińska, Wojciech Mazurczyk, and Krzysztof Szczypiorski. Trends in steganography. *Commun. ACM*, 57(3):86–95, March 2014.

## Biographies

**Kevin Lamshöft** is a research assistant and PhD student at the Otto-von-Guericke University Magdeburg Germany. His research topics are located in the field of Cyber Security (especially IT/OT Security regarding Cyber Physicals Systems, Industry 4.0 and the Internet of Things) and (Network-) Steganography/Information Hiding/Covert Channels/Side Channels.

**Jonas Hielscher** is a research assistant and PhD student at the Horst Görtz Institute for IT Security in Bochum Germany, where he is part of the interdisciplinary phd program *SecHuman*. His research area is Human-Centred Security with a special focus on productivity-friendly IT Security solutions for organizations. He received his M.Sc. from Otto-von-Guericke University Magdeburg where he also was a research assistant at the Advanced Multimedia and Security Lab.



**Christian Krätzer** studied Computer Science at the Otto-von-Guericke University Magdeburg, Germany and joined the Advanced Multimedia and Security Lab (AMSL) in 2004, where he is still working as a post-doc researcher. He has a PhD in Computer Science and his research focuses on issues of performance and trust in applied pattern recognition and information fusion.

**Jana Dittmann** studied Computer Science and Economy at the Technical University in Darmstadt. She has been a Professor in the field of multimedia and security at the University of Otto-von-Guericke University Magdeburg since September 2002. Jana Dittmann is the leader of the Advanced Multimedia and Security Lab (AMSL) at Otto-von-Guericke University Magdeburg (OvGU) specialised in multimedia specific aspects of security from the technical and computer science dimension as well as from the user perception, user interaction and legal dimension. AMSL is partner in national and international research projects and has a wide variety of well recognized publications in the fields.