A Systematic Analysis of Covert Channels in the Network Time Protocol

Jonas Hielscher
Kevin Lamshöft
Christian Krätzer
Jana Dittmann
Research Group Multimedia and Security, Otto-von-Guericke-University
Magdeburg, Germany
firstname.lastname@iti.cs.uni-magdeburg.de

ABSTRACT

Covert channels in network protocols are a technique aiming to hide the very existence of secret communication in computer networks. In this work we present a systematic in-depth analysis of covert channels by modification for the Network Time Protocol (NTP). 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 is 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. We discuss the general approach of the research community that detection of covert channels is the more promising countermeasure, compared to active suppression of covert channels. Therefore, normalization approaches and a secure network environment are introduced.

CCS CONCEPTS

• **Security and privacy** → *Web protocol security*.

KEYWORDS

Network Covert Channels, Steganography, Network Time Protocol, Network Security, Network Protocol Security

ACM Reference Format:
Jonas Hielscher, Kevin Lamshöft, Christian Krätzer, and Jana Dittmann. 2021. A Systematic Analysis of Covert Channels in the Network Time Protocol. In *The 16th International Conference on Availability, Reliability and Security (ARES 2021), August 17–20, 2021, Vienna, Austria.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3465481.3470075

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 [45] or establishment of Command-and-Control (C&C) channels for Malware [5, 46]. The number of Malware using covert channels grew in recent years [22, 38]. 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 [42]. The research community tries to keep up with the speed and hundreds of potential channels and corresponding countermeasures (mainly the detection) were studied [41]. 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 [43]. To the best of our knowledge, such analysis of the potential of protocols as carrier for a wide variety of covert channels was not performed yet (the closest might be [19]). 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 [43] 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 its latest version from 2018 [24] 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) [8]. 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 [40].
(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.

The remainder of the work is structured as follows: In Section **2** we reflect on previous work that evaluated manifold covert channels and derive a research gap. In Section **3** we describe NTP and important concepts behind covert channels. In Section **4** we introduce our research approach for deriving NTP channels and summarize our selected and used nine key attributes as comparison criteria. In Section **5** we describe all discovered channels and compare them by the selected nine key attributes. In Section **6** we explain the channels that we implemented as well as the results from their testing. In Section **7** we suggest a set of active countermeasures to suppress covert channels in NTP. In Section **8** we discuss the relevance of our findings and potential threats to validity. In Section **9** we conclude and encourage further work.

2 RELATED WORK & MOTIVATION

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. [45]. Larger, systematic approaches are rare: Murdoch et al. [33], took a look at seven TCP fields and the packet ordering, to describe the potential for covert channel embedding. Lucena et al. [21], did the same for IPv6. Mileva et al. [28], 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. [19] used the pattern-based taxonomy from Mazurczyk et al. [24] to categorize different channels they discovered in Modbus/TCP. Mileva et al. [29], introduced a variety of covert channels in the new MQTT 5 standard by observing the protocol specifications and deriving the channels.

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 [40], 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 [36]. However these queries are a complete sub-protocol of NTP [31] 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-state approach in which we use the pattern-based taxonomy of Mazurczyk et al. [24] 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.

3 TECHNICAL BACKGROUND

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

3.1 NTP

NTP is an UDP based OSI-Layer 7 protocol, meant for time synchronization. The current version is 4 [32], 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) [30] 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

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

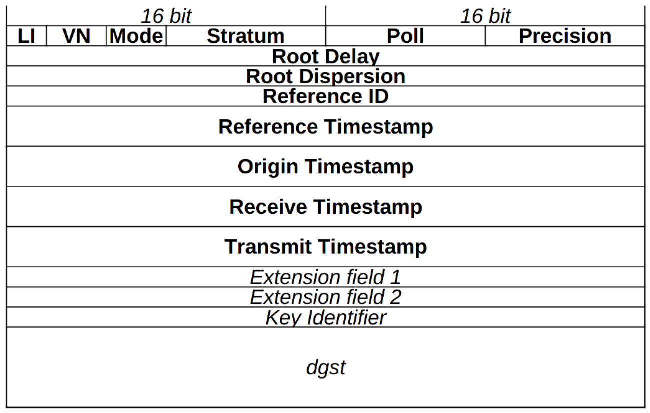

Figure 1: The structure of an NTP packet. 13 fields are mandatory, the two extension fields and the MAC field (key identifier plus dgst) are optional.

(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 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 [32]. 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 [16] 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"[31]. 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 [12] are broken [35], the newer Network Time Security (NTS) [10] was introduced recently in 2020 and has no practical relevance yet).

3.2 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) [25]. 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 its latest version [24], eight timing- and twelve storage-patterns are distinguished.

Usually the distribution of so-called *steganographic keys* [44] 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.

3.3 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 [19] and [20] Lamshöft et. al 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 [19] Lamshöft et al.

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 comunication, e.g. network elements like firewalls and switches. Based on this is the new concept of *reversibility* in covert channels [23]. 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).

4 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 [24] 2. Comprehensive comparison and analysis of these channels by using a set of commonly known properties: Protocol-compliance from [19], reversibility from [23], 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.

4.1 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 [24].

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. [24] 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.

4.2 Comparison Criteria

As comparison criteria, nine key attributes are selected and proposed to describe every plausible covert channel identified in our work: **Protocol-compliance** [19] 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** [23] 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 [14]. 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. ∞ (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 [11], 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

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

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

4.3 Attacker Model

As we will discuss in section 8, NTP packets can be found in most IT- and OT-networks, however only in small numbers. Therefore, the bandwidth of any covert channel within NTP is limited. NTP is only relevant for those attackers that either want to deeply hide their command & control or data exfiltration channel or attackers how need a reliable carrier that is available in most networks. We assume that both is especially relevant in industrial networks for Advanced Persistent Threat attackers. The attacker utilizes NTP to send commands over manipulated stratum servers into an isolated industrial network to reach an compromised industrial control system controller. This stratum server could either be part of the public stratum pool or a server within the organizational network. The controller could be compromised via an supply chain attack or an insider, e.g. maintenance personal (such attacks are plausible and further elaborated e.g. in [15]).

5 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 4.

5.1 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 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.

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

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.

5.2 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 [40].

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

- **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.
- **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 [40].

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.

6 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 test-beds: 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 4.3 that requires a channel with low suspicousness 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]

6.1 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 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

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

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	-	↓	∘	∘	?	?	?
RA1	T	T	✗	*m*	✓	c	✗	✗	1	-	↓	↑	↓	∘	∘	↓
AL1	T	T	✗	*m*	✗	✗	s	✗	1	*e*	-	-	-	-	-	-
MO1	T	T	✗	*m*	✗	c	✗	✗	1	-	↓	↑	∘	∘	∘	↓
RT1	T	T	✗	*m*	✗	c	s	✗	1	*e*	∘	∘	∘	∘	∘	∘
SU1	1	1	✗	*h*	✓	c	s	b	1	*ok*	↓	∘	∘	∘	∘	↓
SU2	1	1	✗	*h*	✓	c	s	b	1	*ok*	↓	∘	∘	∘	∘	↓
SU3	1	1	✗	*h*	✗	c	s	b	1	*ok*	↓	∘	∘	∘	∘	↓
RV1	16	32	s	*m*	✓	✗	s	✗	≈	*t*	-	-	-	-	-	-
RV1a	1	16	s	*l*	✓	✗	s	✗	≈	*ok*	-	-	-	-	-	-
RV2	16	32	s	*m*	✓	✗	s	✗	≈	*t*	-	-	-	-	-	-
RV2a	1	16	s	*l*	✓	✗	s	✗	≈	*ok*	-	-	-	-	-	-
AR1	1	1	✓	*h*	✓	c	s	b	1	*ok*	↓	∘	∘	∘	∘	↓
VM1	2	3	s	*h*	✗	c	s	b	1	*e*	∘	∘	∘	∘	∘	∘
VM1a	1	1	✗	*m*	✓	c	s	b	1	*ok*	↓	∘	∘	∘	∘	↓
VM2	1	1	s	*m*	✓	c	s	b	∞	*ok*	∘	∘	∘	∘	∘	∘
VM3	3	24	✗	*h*	✗	✗	s	✗	1	*e*	-	-	-	-	-	-
VM3a	24	24	✗	*h*	✗	✗	s	✗	1	*e*	-	-	-	-	-	-
VM4	32	32	✗	*h*	✓	c	s	b	1	*ok*	↑	∘	∘	∘	∘	∘
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	∞	*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	✗	≈	*t*	-	-	-	-	-	-
MR1a	1	96	✓	*m*	✓	✗	✗	b	≈	-	-	-	-	-	-	-
MR1b	1	64	✓	*m*	✓	c	✗	✗	≈	-	∘	∘	∘	∘	∘	∘
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	≈	*ok*	∘	↑	∘	∘	∘	∘
UV2a	1	96	✗	*l*	✓	✗	s	✗	≈	*ok*	-	-	-	-	-	-
UV2b	1	64	✗	*l*	✓	✗	✗	b	≈	-	-	-	-	-	-	-
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*	∘	∘	∘	∘	∘	∘

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, ∞: infinity, ≈: approximately. Contextual plausibility: ↑: more plausible, ∘: no difference, ↓: less plausible. "-": Not applicable. "?": Unable to answer. VM5: Found in [2]. SU3, UV6: Found in [40].

[37]. 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.

6.2 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 2). 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).

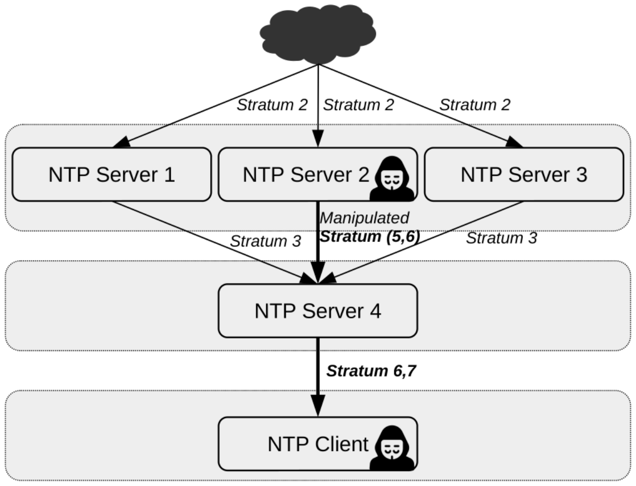

Figure 2: The covert channel two uses the stratum values in NTP server packets to send hidden information from a malicious NTP server (2) through an non-involved server (4) to a client. Server 4 is requesting time from three upstream servers (1,2,3), but only server 2 is malicious.

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.

7 COUNTERMEASURES

The primary step towards countermeasures against covert channels usually is detection [27]. In the literature a wide variety of anomaly detection, statistical analysis and classifier training approaches exist [45]. 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.

7.1 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 [13]. 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 [9]: “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.

7.2 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 [34, 39]. 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 [17] and [18], 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.

8 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.

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.

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 [26]. However, it is not clear whether other patterns would classify covert channels on this layer more suitable. The original pattern-based taxonomy [43] was extended twice [24, 26], showing the potential for further improvements and additions.

9 CONCLUSION

In this work, 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.

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 [22, 38], 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.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback. Parts of the work in this paper has 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. The work was partly build on the concept and results from the master thesis of Jonas Hielscher.

REFERENCES

[1] A. Malhotra and S. Goldberg. 2019. 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. 2017. Covert Channel over Network Time Protocol. In *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy* (Wuhan, China) *(ICCSP '17)*. Association for Computing Machinery, New York, NY, USA, 62–65.

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

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

[5] L. Caviglione, M. Gaggero, J. Lalande, W. Mazurczyk, and M. Urbański. 2016. Seeing the Unseen: Revealing Mobile Malware Hidden Communications via Energy Consumption and Artificial Intelligence. *IEEE Transactions on Information Forensics and Security* 11, 4 (2016), 799–810.

[6] Rajarathnam Chandramouli and Nasir Memon. 2003. Steganography Capacity: A Steganalysis Perspective. *Proc SPIE* (06 2003), 173–177.

[7] chrony. 2018. Comparison of NTP implementation. (August 2018). https://chrony.tuxfamily.org/comparison.html.

[8] P. Ferrari, P. Bellagente, A. Depari, A. Flammini, M. Pasetti, S. Rinaldi, and E. Sisinni. 2020. 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*. 223–228.

[9] D. Franke and A. Malhotra. 2019. NTP Client Data Minimization. *Internet Engineering Task Force* (2019). https://tools.ietf.org/id/draft-ietf-ntp-data-minimization-04.html.

[10] D. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sunblad. 2020. Network Time Security. *Internet Engineering Task Force* (March 2020). https://tools.ietf.org/html/draft-ietf-ntp-using-nts-for-ntp-28.

[11] Jessica Fridrich. 2009. *Steganography in Digital Media: Principles, Algorithms, and Applications* (1st ed.). Cambridge University Press, USA.

[12] B. Haberman, D. Mills, and U. Delaware. 2010. Network Time Protocol Version 4: Autokey Specification. *Internet Engineering Task Force* (June 2010). https://tools.ietf.org/html/rfc5906.

[13] Mark Handley, Vern Paxson, and Christian Kreibich. 2001. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10* (Washington, D.C.) *(SSYM'01)*. USENIX Association, USA, Article 9, 18 pages.

[14] Mario Hildebrandt, Robert Altschaffel, Kevin Lamshöft, Matthias Lange, Martin Szemkus, Tom Neubert, Claus Vielhauer, Yongjian Ding, and Jana Dittmann. 2020. Threat analysis of steganographic and covert communication in nuclear I&C systems. In *International Conference on Nuclear Security 2020*.

[15] Mario Hildebrandt, Kevin Lamshöft, Jana Dittmann, Tom Neubert, and Claus Vielhauer. 2020. Information Hiding in Industrial Control Systems: An OPC UA Based Supply Chain Attack and Its Detection. In *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security* (Denver, CO, USA) *(IHandMMSec '20)*. Association for Computing Machinery, New York, NY, USA, 115–120.

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

[17] P. Jeitner, H. Shulman, and M. Waidner. 2020. 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)*. 49–50.

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

[19] Kevin Lamshöft and Jana Dittmann. 2020. Assessment of Hidden Channel Attacks: Targetting Modbus/TCP. *IFAC-PapersOnLine* 53, 2 (2020), 11100–11107. 21th IFAC World Congress.

[20] Kevin Lamshöft, Tom Neubert, Mathias Lange, Robert Altschaffel, Mario Hildebrandt, Yongjian Ding, claus Vielhauer, and Jana Dittmann. 2020. Novel Challenges for Anomaly Detection in I&C Networks: Strategic Preparation for the Advent of Information Hiding based Attacks. *Atw. Atomwirtschaft* 65 (10 2020), 504–508.

[21] Norka B. Lucena, Grzegorz Lewandowski, and Steve J. Chapin. 2006. Covert Channels in IPv6. In *Privacy Enhancing Technologies*, George Danezis and David Martin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 147–166.

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

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

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

[25] Wojciech Mazurczyk, Steffen Wendzel, Sebastian Zander, Amir Houmansadr, and Krzysztof Szczypiorski. 2016. *Background Concepts, Definitions, and Classification in Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. John Wiley & Sons, Ltd, Chapter 2, 39–58.

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

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

[28] Aleksandra Mileva and Boris Panajotov. 2014. Covert channels in TCP/IP protocol stack - Extended version-. *Central European Journal of Computer Science* 4 (06 2014), 45–66.

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

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

[31] D. Mills. 2011. Control Messages Protocol for Use with Network Time Protocol Version 4. *Internet Engineering Task Force* (October 2011). https://tools.ietf.org/id/draft-odonoghue-ntpv4-control-00.html.

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

[33] Steven J. Murdoch and Stephen Lewis. 2005. Embedding Covert Channels into TCP/IP. In *Information Hiding*, Mauro Barni, Jordi Herrera-Joancomartí, Stefan Katzenbeisser, and Fernando Pérez-González (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 247–261.

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

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

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

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

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

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

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

[41] Steffen Wendzel. 2018. Get Me Cited, Scotty! Analysis of Citations in Covert Channel/Steganography Research. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (Hamburg, Germany) *(ARES 2018)*. Association for Computing Machinery, New York, NY, USA, Article 13, 8 pages.

[42] Steffen Wendzel, Wojciech Mazurczyk, Luca Caviglione, and Michael Meier. 2014. Hidden and Uncontrolled – On the Emergence of Network Steganographic

Threats. In *ISSE 2014 Securing Electronic Business Processes*, Helmut Reimer, Norbert Pohlmann, and Wolfgang Schneider (Eds.). Springer Fachmedien Wiesbaden, Wiesbaden, 123–133.

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

[44] Andreas Westfeld and Andreas Pfitzmann. 2000. Attacks on Steganographic Systems. In *Information Hiding*, Andreas Pfitzmann (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 61–76.

[45] Sebastian Zander, Grenville Armitage, and Philip Branch. 2007. Covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys and Tutorials* 9 (09 2007), 44–57.

[46] Elundefinedbieta Zielińska, Wojciech Mazurczyk, and Krzysztof Szczypiorski. 2014. Trends in Steganography. *Commun. ACM* 57, 3 (March 2014), 86–95.