AFRL-AFOSR-UK-TR-2022-0034

Covert Communication Detection (CoCoDe)

Cabaj, Krzysztof
POLITECHNIKA WARSZAWSKA
1 PLAC POLITECHNIKI
WARSZAWA, MAZOWIECKIE, 00-661
POL

03/19/2022
Final Technical Report

Air Force Research Laboratory
Air Force Office of Scientific Research
European Office of Aerospace Research and Development
Unit 4515 Box 14, APO AE 09421

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
20220319	Final	START DATE 20170901	END DATE 20210228

4. TITLE AND SUBTITLE
Covert Communication Detection (CoCoDe)

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
	FA9550-17-1-0254	61102F
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Krzysztof Cabaj

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
POLITECHNIKA WARSZAWSKA 1 PLAC POLITECHNIKI WARSZAWA, MAZOWIECKIE 00-661 POL	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
EOARD UNIT 4515 APO AE 09421-4515	AFRL/AFOSR IOE	AFRL-AFOSR-UK-TR-2022-0034

12. DISTRIBUTION/AVAILABILITY STATEMENT
A Distribution Unlimited: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This project studied covert communication channels and especially focused on means for detecting distributed covert networks. Covert communication channels (also known as network steganography) allows a hidden sender and hidden receiver to exchange secret data. These covert communication channels can be used to conduct command and control of malicious servers, exfiltrate confidential data, or download further malicious code without the user being made aware. Thus, the topic of covert channel detection is a very important one to any large organization with sensitive data and particularly the Department of Defense (i.e., Intellectual Property, Patents, For Official Use Only Data, and military Personally Identifiable Information).

Hundreds of techniques can be used to create covert channels – some of the most common techniques are to place data into unused fields of network protocol headers, change the size of network packets, manipulate inter-packet timing/order, or alter header elements (e.g., HTTP plaintext header lines). As adversaries grown in capability, more and more complex forms of covert channels will appear becoming increasingly difficult to detect and increasing in bandwidth. This includes, for example, steganographic botnets where all communication between bots is realized using some form of data hiding. The most concerning type of information hiding for botnets involves the study of Distributed Network Covert Channels (DNCCs).

Over the course of 3.5 years, the 5-member research team (the PI, Co-PI, a PhD student, and two Masters students) utilized theoretical and experimental approaches to conduct covert channel research focusing on DNCCs. Their effort can be described in three phases:

1. In the first phase of the project, the team focused on theoretical studies concerning creation of stealth channels using information hiding patterns and methods for detection. The team worked on identifying components of the TCP/IP protocols that are most susceptible to data hiding, analyzing the key aspects of the hiding patterns, and understanding the current state of the domain taxonomy. The team's main contribution during this phase was extending the information hiding patterns concept including modification of the pattern analysis process and extending the current taxonomy with new patterns.

2. In the second phase, the team focused on experimental studies to analyze the theory of DNCCs by developing a classification scheme, prioritizing performance, and characterizing performance features. The research team developed and implemented several different types of DNCCs to investigate these various DNCC properties. For example, changing the number of utilized flows, transmitters, receivers, the number of simultaneously utilized steganographic methods, and different networking environments from typical IP-based networks to IoT-based ones.

3. In the last phase, the researchers focused on studying the detectability of various DNCCs. The team proposed, used, and analyzed several approaches to study how efficiently DNCCs can be detected when facing a determined adversary using advanced covert communication means. The team explored various detection analysis techniques including: data mining, itemset trees; and machine Learning techniques. The results show that the best detection accuracy is achieved when the multiple approaches are combined. Lastly, the team generated and posted a unique steganography dataset which is freely available.

Overall, the research team produced four conference papers and two journal articles. The conference papers were published in the annual Availability, Reliability and Security (ARES) conference during the years 2018, 2019, 2020, and 2021 which is sponsored by the Association for Computing Machinery (ACM). The ARES conference, and ACM in particular, has a high academic pedigree. The two journal papers were both published in reasonable journals in 2020 and 2021.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		
U	U	U	SAR	75

19a. NAME OF RESPONSIBLE PERSON	**19b. PHONE NUMBER** *(Include area code)*
LOGAN MAILLOUX	314 235 6163

WARSAW UNIVERSITY OF TECHNOLOGY

FINAL REPORT

Project Name:
Covert Communication Detection (CoCoDe)

Project Number:
FA9550-17-1-0254

Principal Investigator:
Krzysztof Cabaj, DSc., PhD.

Period of Performance:
September 2017 - February 2021

May 11, 2021

CONTENTS

1 SUMMARY

The presented Final Performance Report summarizes all research conducted during the CoCoDe project. The research associated with the project started in September 2017, and initially was planned for 36 months with the ending date 31 August 2020. However, the COVID-19 outbreak has a serious impact on the access to the university infrastructure and thus on the final research phase conducted during the second and third quarter of 2020. Therefore, the CoCoDe project, which has been initially planned for three years, received a 'No Cost Extension' for an additional six months. Accordingly to the updated version of the Grant Agreement from 3rd August 2020, the new project end date is 28th February 2021.

The research team of the CoCoDe project consisted of five members, i.e.:

- Principal Investigator: Krzysztof Cabaj, DSc., PhD,
- co-Principal Investigator: Wojciech Mazurczyk DSc., PhD,
- PhD student Piotr Nowakowski, MSc,
- two Msc students: Piotr Żórawski and Maciej Purski.

There has been a change in Program Officer during the first year of the project. Former Program Officer Lt Col Ryan W. Thomas was handed over to Lt Col Logan Mailloux.

During our work, we utilize a theoretical and experimental approach to conduct research concerning network covert channels (network steganography). In the first phase of the project, we focus on theoretical studies concerning the forming of stealthy channels using steganographic patterns and methods for its detection. Later, we focused on experimental studies concerning the performance of covert channels and detection algorithms. Due to this fact, at the beginning of the project we finalized the purchase of the dedicated server. The server was delivered and installed in the beginning of 2019. Using it, we could deploy a very complicated topology which contains various sending and receiving nodes. This allows testing various aspects of DNCC (Distributed Network Covert Channels) in an experimental manner.

From the beginning of the project, we have tried to publish the most notable and recent results. During the whole project, we have published six scientific papers – four in international conferences and two in scientific journals. All details concerning the published papers are presented in Section 4.

The additional result of the conducted research was the generation of the steganography dataset, which contains network traces, JSON file which describes all packet changes, and an interactive image showing in which DNCC connections changes associated with cover transmission occurred. Sample dataset is freely available at our security team's webpage using the following URL: *http://cssg.zoak.ii.pw.edu.pl/iot_dncc_data_set.html*.

2 INTRODUCTION

Network covert channels define a way in which the covert sender and the covert receiver exchange some secret data, i.e., both communicators must agree on a signaling technique

in advance and create a so-called covert channel over which the hidden data is transmitted. The term covert channel was initially introduced in operating system research by Lampson as a channel not intended for communication, later the term was defined as policy-breaking communication channel by the U.S. Department of Defense.

Currently, hundreds of techniques can be applied to create a covert channel; some of the most common techniques are to place secret data into unused fields of network protocol headers or to change the size of network packets. Moreover, it is common to manipulate interpacket timing, i.e., the time between network packets or their order. Moreover, the order of header elements (e.g., HTTP plaintext header lines) can be altered or artificial errors can be introduced into a transmission.

The topic of network covert channels has been increasingly attracting attention of the security community in the last decade. This is mainly caused by the raising interest of various types of attackers in such techniques (e.g., for criminal purposes, cyberespionage, cyberwarfare, etc.) to cloak their malicious activities. This includes, for instance, hiding communication with the controlling server (C&C), exfiltration of confidential, sensitive data from the infected hosts, or downloading further modules of malicious code in a covert manner As already mentioned, the trend of utilization of various types of data hiding techniques is accelerating and we can expect more sophisticated data hiding techniques to be found in a main stream malware (not only in Advanced Persistent Threats) in the near future. Below we briefly review some of the most notable examples of information hiding techniques utilization in real-life threats.

Advancements in security systems over the past 15 years have forced malware developers to investigate new possibilities to make their "products" stealthier. Although it is difficult to determine the origin of information hiding techniques, the first massive usage of these techniques can be traced back to 2006, when Operation Shady RAT led to attacks against numerous institutions worldwide and inflicted damage for months. Years later, security experts agreed that the main program responsible for this attack was the phishing virus Trojan.Downbot. This malware created a back door and then downloaded files appearing as real HTML pages or JPEG images. These files were encoded with commands that would allow remote servers to gain access to local files on the infected host computer. Other notable examples of information hiding–capable malware include Regin and Linux.Fokirtor, which use network traffic to covertly leak data, and Alureon, Duqu, Lurk, and Trojan.Zbot, which use digital images as hidden data carriers. Even when rudimentary, new threats exploiting some form of information hiding continue to be discovered, as seen in Soundcomber and AirHopper, which modify the status of shared hardware/software resources to exfiltrate confidential data, and in Feederbot, W32.Morto, and Smuggler, which manipulate the network traffic for this purpose.

It must be also noted that originally, information-hiding techniques were implemented only in advanced persistent threats (APTs) like Duqu, Regin, or Hammertoss – the most sophisticated types of malware created with the support of nationwide sponsors. However, information-hiding techniques are slowly becoming the de facto standard for "ordinary" malware. For example, various types of popular threats like ransomware (TeslaCrypt, Cerber, and SyncCrypt) or exploit kits (Stegano/Astrum, DNSChanger, and Sundown) also incorporate some form of information hiding.

Furthermore, we can expect also that other, more complex forms of network covert channels

will appear, which will be used by cybercriminals to provide improved undetectability or steganographic bandwidth. This includes, for example, steganographic botnets where all communication between bots is realized using some form of data hiding and especially using network covert channels. In particular, the most suitable type of information hiding that can be utilized for this purpose involves Distributed Network Covert Channels (DNCCs).

During the research that we have performed within the CoCoDe project, we have analyzed DNCCs from a different perspective and in various scenarios. First of all, we analyzed the theory of DNCCs by developing their classification and by characterizing their main performance features. We have also developed and implemented several different flavors of DNCCs to investigate various DNCC properties. They have taken advantage of different steganographic methods, multiple senders, and multiple connections from each sender. We also assumed different networking environments from typical IP-based networks to IoT-based ones. It must be also noted that during the conducted research we used three steganographic methods implemented during the previous part of the project: TTL Modulation, HTTP Header Reorder and TCP Options Reorder. Finally, we have taken a significant amount of time to focus on the detectability of the DNCCs. We have analyzed and proposed several data mining-based approaches to determine how efficient DNCCs can really be when faced with a demanding adversary aiming to spot covert communication.

All aspects that we investigated have been described in detail in the following subsections of this report.

3 Methods, Assumptions and Procedures

From the beginning of the project, we have utilized two approaches for our research - theoretical and experimental. The theoretical studies were conducted mainly at the beginning of the project. During this period, we mainly focused on the information hiding patterns approach to identify these components of the network protocols which are most susceptible to data hiding. Within the CoCoDe project, we analyzed the key aspects of the hiding patterns and the current state of the taxonomy in the domain. Our main contribution to extending the information hiding patterns concept includes modification of the pattern analysis process and extending the current taxonomy with new patterns. In particular, in the improved form we take into account more details on the hiding method's inner workings, thus potentially this can contribute to a better understanding of the nature of network covert channels. We also introduced and described a pattern-based classification of distributed network covert channels (DNCC). Moreover, we conducted theoretical research associated with applying data mining approaches, more specifically the discovery of frequent patterns, for the detection of covert channels. Both aspects of the theoretical research are later investigated during the experimental part of the project.

Due to the extensive usage of the experimental approach during the conducted research, a dedicated server was purchased. The server has been used to configure the experimental testbed which allows performing experiments concerning the investigation of the DNCC channel performance as well as of various covert channel detection techniques. The first step in our research concerned the implementation of various basic steganography methods,

which were later utilized for the construction of more stealthy DNCC channels. The main reason for the implementation of our own network hiding techniques was associated with the tuning flexibility of the steganographic transmitter. In the freely available tools, depending on the author's concept, a few and often various steganographic parameters can be changed, for example, how many bits are transmitted in one PDU modification, how often steganographic changes are introduced, etc. This causes various problems during experiments, when one would like to compare different methods or their implementations. Moreover, available steganographic tools often integrate within a single tool the overt traffic and the covert message which is introduced by the packet modification. In such a case, when the two tools use completely different overt traffic, their reliable comparison is practically impossible. In effect, we decided that our research tool should not generate overt traffic but should rather add steganographic data to the indicated traffic flows. To implement this approach, we utilized the features introduced by the SDN paradigm, thus it was implemented as an SDN application integrated within the POX SDN controller. During this part of our research, four steganographic basic methods were implemented: 'TTL Modulation', 'HTTP Header Reorder', 'TCP Options Reorder' and 'Inter-packet Timing Modulation'.

In the second part of our research, we used these methods to form various types of DNCC, for example, changing the number of utilized flows, transmitters, receivers, and the number of simultaneously utilized steganographic methods. Moreover, during our research we investigated various types of overt traffic. During the first phase of the conducted research, we utilized web traffic – which is currently one of the most popular types of traffic in the Internet. Due to this popularity, we assume that steganographic methods can be frequently utilized to form covert communication channels. However, during our research conducted during the last year, we utilized for this purpose IoT (Internet of Things) traffic. As we observe a significant rise in the number of such devices, which is foreseen to continue in the future, this brings new security challenges as well as new steganographic opportunities. In our opinion, the network traffic generated by this kind of devices will most likely be the next choice for the attackers/steganographers. Performed experiments proved that even periodically transmitted, very short messages could be utilized to perform covert transmission. During performed research tasks, we form the DNCC channel using simulated network traffic of the thermometer of which measurements are periodically sent to the central server.

The last part of our research concerns the detection of DNCC channels. At first, we simply utilized data mining frequent item-sets discovery using appropriately preprocessed network traces. Later, to improve the detection accuracy, we constructed a custom tree, containing the results of multiple frequent item-sets detections using various values of the minimal support parameter. Obtained results proved that the analysis of leaves and nodes which appear in such a tree can increase the covert channel detection accuracy. The last investigated approach compared and combined previously described methods with Machine Learning techniques. Obtained results prove that the highest detection accuracy is achieved in the case where we combine both above-mentioned developed approaches, i.e., the item-set tree with Machine Learning-based one.

4 Results and Discussion

During the project, the most important results from our experiments are described with details in the scientific papers. During the project period, we published six scientific papers, four in international conferences and two in scientific journals.

Below is the list of all published and presented articles:

1. Cabaj Krzysztof, Mazurczyk Wojciech, Nowakowski Piotr, Piotr Żórawski: Towards Distributed Network Covert Channels Detection Using Data Mining-based Approach, in: ARES 2018 Proceedings of the 13th International Conference on Availability, Reliability and Security / Doerr Christian, Schrittwieser Sebastian, Weippl Edgar (red.), 2018, ISBN 978-1-4503-6448-5, pp. 1-10, DOI:10.1145/3230833.3233264

2. Mazurczyk Wojciech, Wendzel Steffen, Cabaj Krzysztof: Towards Deriving Insights into Data Hiding Methods Using Pattern-based Approach, in: ARES 2018 Proceedings of the 13th International Conference on Availability, Reliability and Security / Doerr Christian, Schrittwieser Sebastian, Weippl Edgar (red.), 2018, ISBN 978-1-4503-6448-5, pp. 1-10, DOI:10.1145/3230833.3233261

3. Cabaj Krzysztof, Mazurczyk Wojciech, Nowakowski Piotr, Piotr Żórawski: Fine-tuning of Distributed Network Covert Channels Parameters and Their Impact on Undetectability, in: Proceedings of the 14th International Conference on Availability, Reliability and Security - Ares 2019, ICPS, 2019, ISBN 978-1-4503-7164-3, pp. 1-8, DOI:10.1145/3339252.3341489

4. Nowakowski Piotr, Żórawski Piotr, Cabaj Krzysztof, Mazurczyk Wojciech: Network covert channels detection using data mining and hierarchical organisation of frequent sets: an initial study, in: ARES '20: Proceedings of the 15th International Conference on Availability, Reliability and Security, 2020, ISBN 978-1-4503-8833-7, pp. 1-10, DOI:10.1145/3407023.3409217

5. Cabaj Krzysztof, Żórawski Piotr, Nowakowski Piotr, Mazurczyk Wojciech: Efficient distributed network covert channels for Internet of things environments, in: Journal of Cybersecurity, vol. 6, nr 1, 2020, pp. 1-18, DOI:10.1093/cybsec/tyaa018

6. Nowakowski Piotr, Żórawski Piotr, Cabaj Krzysztof, Mazurczyk Wojciech: Detecting Network Covert Channels using Machine Learning, Data Mining and Hierarchical Organisation of Frequent Sets, in: Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 2021, vol. 12, no. 1, pp.20-43. DOI:10.22667/JOWUA.2021.03.31.020

The last two publications published in the journals are extended versions of the previously presented and published conference papers. The article [5] is an extended version of [3] and article [6] is an extended version of [4]. Furthermore, in this section the content of four papers (excluding those two which are further extended) are presented.

4.1 PAPER I

Cabaj Krzysztof, Mazurczyk Wojciech, Nowakowski Piotr, Piotr Żórawski: Towards Distributed Network Covert Channels Detection Using Data Mining-based Approach, in: ARES 2018 Proceedings of the 13th International Conference on Availability, Reliability and Security / Doerr Christian, Schrittwieser Sebastian, Weippl Edgar (red.), 2018, ISBN 978-1-4503-6448-5, pp. 1-10, DOI:10.1145/3230833.3233264

Towards Distributed Network Covert Channels Detection Using Data Mining-based Approach

Krzysztof Cabaj
Warsaw University of Technology
Warsaw, Poland
kcabaj@ii.pw.edu.pl

Wojciech Mazurczyk
Warsaw University of Technology
Warsaw, Poland
wmazurczyk@tele.pw.edu.pl

Piotr Nowakowski
Warsaw University of Technology
Warsaw, Poland
p.nowakowski@tele.pw.edu.pl

Piotr Żórawski
Warsaw University of Technology
Warsaw, Poland
p.zorawski@tele.pw.edu.pl

ABSTRACT

Currently, due to improvements in defensive systems network covert channels are increasingly drawing attention of cybercriminals and malware developers as they can provide stealthiness of the malicious communication and thus to bypass existing security solutions. On the other hand, the utilized data hiding methods are getting increasingly sophisticated as the attackers, in order to stay under the radar, distribute the covert data among many connections, protocols, etc. That is why, the detection of such threats becomes a pressing issue. In this paper we make an initial step in this direction by presenting a data mining-based detection of such advanced threats which relies on pattern discovery technique. The obtained, initial experimental results indicate that such solution has potential and should be further investigated.

CCS CONCEPTS

•**Security and privacy** → **Network security;** *Distributed systems security; Information flow control;* Pseudonymity, anonymity and untraceability; •**Social and professional topics** → *Computer crime;*

KEYWORDS

covert channels, data hiding, information hiding, data mining

ACM Reference format:
Krzysztof Cabaj, Wojciech Mazurczyk, Piotr Nowakowski, and Piotr Żórawski. 2018. Towards Distributed Network Covert Channels Detection Using Data Mining-based Approach. In *Proceedings of International Conference on Availability, Reliability and Security, Hamburg, Germany, August 27–30, 2018 (ARES 2018),* 10 pages.
DOI: 10.1145/3230833.3233264

1 INTRODUCTION

During the last few years information-hiding-capable malware is reported to be on the rise [7]. The main reason for this is that the current defensive systems are being continuously improved and thus the cybercriminals desire techniques that will provide them stealthiness to allow them to stay under the radar for as long as possible. In result, such modern malware possesses a serious challenge for the security community [8], [9].

Moreover, currently the data hiding methods used in malware are quite simple and naive, yet many more complex and sophisticated techniques have been proposed in the research community during the last decade for which there are no straightforward mitigation solutions. What is worse, there is a plethora of opportunities to create a network covert channel using various network steganography techniques and there is no one-size-fits-all detection solution to counter them. On the contrary, practically for each new data hiding method a dedicated mitigation solution must be devised, and adjusted in case the original technique has changed its inner workings.

Considering above a more flexible and scalable detection solutions need to be designed and developed. That is why, in this paper we make the first step toward a data mining-based detection. We especially would like to evaluate whether pattern discovery can be useful for this purpose. Moreover, we especially focus on the case of *distributed covert channels* i.e. network covert channels that spread the secret data among many flows/protocols/hosts or use multiple data hiding methods within the same flow or within PDUs in order to provide hidden data exchange.

Therefore the main contributions of this paper are to:

- assess feasibility of applying data mining pattern discovery technique for the detection of network covert channels,
- present experimental detection results focused especially on discovering distributed network covert channels.

The paper is structured as follows. Section 2 presents related work on data mining threats detection. In Section 3 fundamental on techniques allowing realization of distributed network covert channels are presented. Next, Section 4 is devoted to data mining methods, pre- and post-processing techniques and introduction of the proposed data hiding detection approach which utilizes frequent item sets. Section 5 contains all details related to the conducted experiments. In the first part of this section description of the experimental test-bed as well as of the proof-of-concept implementation of the detection system is presented. Later in this section

ARES 2018, Hamburg, Germany

DOI: 10.1145/3230833.3233264

all obtained experimental results and related discussions are presented. Finally, Section 6 concludes our work and present some future research directions.

2 RELATED WORK

In this section we first characterize the main types of countermeasures against network covert channels and then we follow with the description of how data-mining techniques have been used so far for cyber security purposes.

2.1 Countermeasures against network covert channels

To counter a network covert channel it is necessary to identify it, i.e., becoming aware that it exists [10]. Without knowing that a channel exists, even a naive data hiding techniques are covert. Alternatively network covert channels may be also discovered by more general anomaly detection solutions, however, there has been only limited research effort in this research area directed towards network hiding techniques.

After the network covert channel has been identified it can be eliminated completely, its bandwidth can be significantly limited, it can be detected and audited. It may be possible to eliminate the use of the resulting covert channel (typically with a help of traffic normalizer [12]). However, some hidden channels cannot be eliminated completely [11], but if their bandwidth is severely limited then this will make them useless in practice [13]. Alternatively, if the use of a covert channel is detected then it is possible to take actions against covert senders and receivers. In order to collect information on potential covert senders and receivers and their hidden communication characteristics it is possible to continuously audit the use of covert channels.

In this paper we focus on the detection of network covert channels as this allows to potentially pinpoint covert communication parties and infer their motivation and sometimes also exchanged secret data.

2.2 Data mining techniques for network covert channels detection

In the existing literature the most commonly used data mining approaches for anomaly detection utilize classification methods. It must be noted that typically the classification is a two-step process. First, the detection algorithm is trained using well-known and labeled data and then new samples are classified to the corresponding classes. It must be noted that current works in order to spot steganographic traffic focus in majority only on this approach when using data mining [1], [2]. However such an approach has two serious disadvantages.

When the data that needs to be subjected to classification changes, for example, an attacker switches to another steganographic method which was not previously present in the training data then the detection process will most likely fail or it will achieve significantly lower results. The other disadvantage is associated with the preparation of the training data where all samples used for this purpose must be appropriately labeled which is often a very tedious work. But it must be noted that the classification approach is not the only data mining technique available. In the paper [2] authors apart from describing classification techniques which cover ca. 75% of all used approaches also mention regression (12%) and clustering (9%). However, in the remaining 4% other data mining methods including pattern discovery have been mentioned which up till now received significantly less attention from the security community. That is why in this paper we want to utilize this approach and test its feasibility for network covert channel detection.

To authors' best knowledge this is the first attempt to apply pattern discovery technique for network covert channels' detection especially with regard to the more sophisticated i.e. distributed ones.

3 DISTRIBUTED NETWORK COVERT CHANNELS BASICS

Currently the majority of the data hiding methods utilizes only a single communication session as a hidden data carrier i.e. secret data is embedded into PDUs belonging to the same flow/protocol with only one hiding pattern in order to embed secret data.

One of the simplest way to improve the data hiding method undetectability is utilization of *pattern hopping* which is defined in the information hiding context in [3]. This is one of the techniques to enable creation of the *distributed covert channel*. In general, the distributed covert channel is defined as a network covert channel that spreads the secret data among multiple flows/protocols/hosts or uses multiple patterns within the same flow or PDU for the hidden data exchange.

In [3] the authors introduce a classification of network hiding techniques into so-called *patterns* with the aim to potentially develop countermeasures for these patterns. In this perspective, information hiding patterns are defined as an abstract description of how to solve a problem in a given context. Thus, information hiding patterns provide an abstract description of a hiding technique and are categorized in a hierarchy. What must be emphasized each hiding pattern is a unified and generic description of a particular hiding method.

Considering above, pattern hopping is an enhancement of an undistributed network covert channel where various patterns are used (sequentially) within a single connection forming one steganographic session (Figure 1). In this figure each symbol (a circle or a triangle) represents a certain change in the protocol that carries some secret data. In the remaining of this paper we call such an event as *information hiding pattern instance* (IHPI).

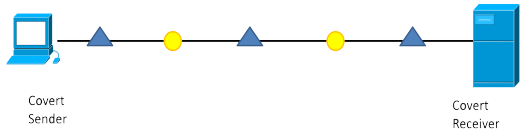


Figure 1: Pattern hopping – the simplest enhancement to the information hiding patterns i.e. various patterns used within one connection (defined in [3])

The other technique which can increase stealthiness of steganographic method is related to utilization of multiple overt data streams. In each connection used for steganographic purposes

a part of the secret data is injected, which reduces number of IHPIs per single connection. Figure 2 presents such situation (each circle represents one IHPI).

However, it must be noted that further enhancements to the scenario illustrated above can be performed by using several parallel connections in which more than one information hiding patterns are used (Figure 3). In result, in each connection fewer changes (caused by the steganographic method functioning) are introduced to the overt traffic which makes detection of such a covert transmission a challenging task.

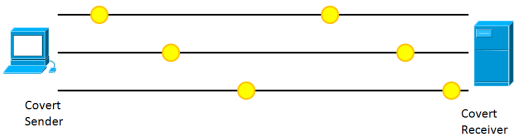


Figure 2: Multi-flow distribution for realizing a single (distributed) network covert channel.

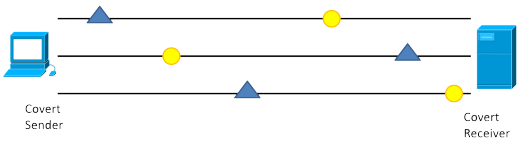


Figure 3: Multi-flow pattern hopping for realizing a single (distributed) network covert channel.

Finally, another improvement can be applied if the covert sender is able to utilize for data hiding purposes connections originated from various overt traffic senders and/or receivers. It must be noted that in this case secret data is injected into the overt transmissions of unaware overt senders i.e. hidden data is embedded in some intermediate device and not on the user's end device.

Obviously all improvements for the covert transmission presented above can be used at the same time. Figure 4 presents injection of secret data which is based on only one pattern and multiple connections. It must be noted that we investigate such an approach in the conducted experiments, which are described in details in the subsection 5.2. Such a data hiding method variant is a bit more challenging to implement but still feasible (as some form of man-in-the-middle attack is required for it to be successful). On the other hand a significant advantage of such an approach is that modifications associated with the information hiding patterns are distributed across many different streams that use various machines addresses. Thus the number of steganographic changes per single connection decreases.

As presented above various enhancements to even simple data hiding method are feasible. These improvements are quite easy to be implemented and can potentially significantly increase stealthiness of the covert transmission (e.g. by spreading secret data among many data flows). Even the usage of relatively simple and well-known method but distributed across various connections or even various senders/receivers can improve its stealthiness and reduce

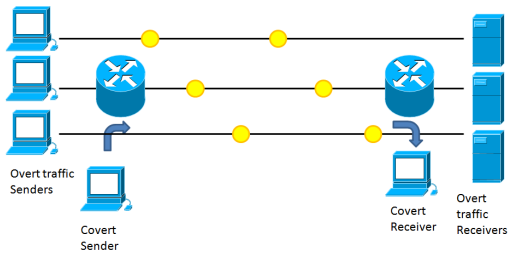


Figure 4: Injection of secret bits into the data streams of various overt traffic senders.

the chance of detection. That is why, in this paper we will investigate the feasibility of the detection of such distributed network covert channels realizations.

4 DATA MINING TECHNIQUES AND THE PROPOSED DETECTION SYSTEM

As it was mentioned in the previous section in the distributed network covert channel realization numerous information hiding patterns instances depending on the utilized data hiding method can occur within multiple data flows. Moreover, these connections can be sourced and/or destined to various machines that process overt traffic. A single anomaly, which can be detected as information hiding pattern instance, may be also introduced by anomalous network behavior (leading to false positives). However, if we are able to discover more and more such events then it is possible that some steganographic transmission occurs. In this paper we propose that for detection of the distributed network covert channels data mining techniques can be utilized.

As mentioned the most well-known and most often used type of data mining methods is associated with classification. In the classification problem user has some initial, training (labeled) data assigned to the appropriate classes. If we consider the problem of steganographic traffic detection then it is important to posses some network traffic traces with and without covert traffic and containing information that this trace contains or does not contain secret data. Using some data classification algorithms it is possible to decide if a new trace which represents previously unknown transmission contains some covert messages. The serious disadvantage of such an approach concerns preparation of the labeled, initial training data. It must be noted that typically a process of preparing such data or even marking them is very time-consuming and tedious. In the case that preparation of such labeled data is impossible other data mining methods can be used, for example *pattern discovery*. In this class of data mining techniques, without any a priori information about characteristics of the analyzed data some patterns can be automatically discovered. The *data mining pattern* is defined as a not trivial dependency in the analyzed data that has been previously unknown. The well-known data mining patterns types that were used so far in the literature include for example, frequent sets [4], frequent sequences [5] or frequent episodes [6]. In this paper we assume that the steganographic traffic is detected, if the particular number of patterns are discovered in the analyzed data set. This could be considered as classification method due to fact that this

decision is based using very simple test – the number of detected patterns. However, it must be noted that the most important part of the introduced method concerns detection of patterns and not a problem of classification, which in conducted experiments is reduced to the trivial comparison of the number of patterns with a certain threshold.

However, the data mining patterns discovery is only one step in the broader process of network steganography detection. The other, even more important steps are:

- *pre-processing* which transforms initial data to the form appropriate for data mining algorithms, and
- *post-processing* which using acquired data mining patterns indicates in which transmissions or between which machines utilization of data hiding techniques was potentially discovered.

Later in this paper we present obtained experimental results concerning usage of *frequent item sets* for the detection of covert traffic. What should be emphasized is that the particular pattern can be mined using one from various proposed algorithms. During conducted experiments for purpose of *frequent item sets* mining we utilize an *a-priori* algorithm. All algorithms used for the discovery of such patterns treat all data as item sets, however our initial data has different form as it is stored in the network traffic traces.

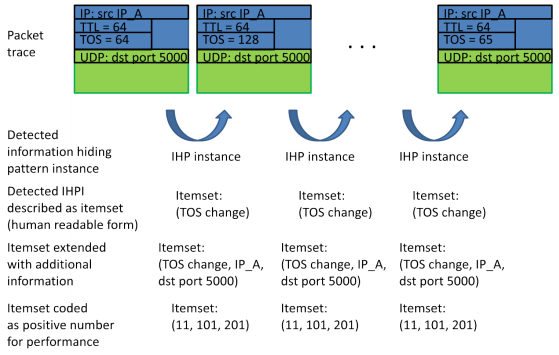


Figure 5: Exemplary packets' trace for the ToS-based covert channel and the steps of pre-processing from the detection of IHPI to the coded item sets.

That it why, the first step of the detection process is associated with pre-processing, which transforms our initial data to the appropriate form. Figure 5 presents a sample network transmission containing covert traffic encoded into modulation of the Type of Service (ToS) field from the IPv4 header (top of the figure). Each such change i.e. information hiding pattern instance is coded as an item set. For further processing initial item set, which contains only the name of used data mining techniques, is extended with information concerning network connection in which it appears (presented in the penultimate line). The content presented there is in a human-readable format, however due to performance reasons data mining algorithms work on the data optimized for pattern discovery. Such an optimized form encodes all items into the item sets as positive integers which can be efficiently compared using modern processor, in contrast to ineffective comparisons of the strings. The last line of Figure 5 shows data in the final form used for further data mining patterns discovery. In the presented example ToS change is coded as 11, IP address as 101 and port 5000 as 201. The data prepared in such a manner are used as an input for data mining technique i.e. frequent item sets mining. As a result of this process the detected frequent items sets are provided. The frequent item set is a subset, which appears in the analyzed data more than specific, predefined number of times. In the data mining field this parameter is called *minimal support* and it is often denoted as *minSup*. Moreover, it should be noted that the output data is in the form similar as input data – all features are encoded as positive integers and before they are passed to further human analysis they should be subjected to post-processing. If we perform the discovery of the frequent item sets in the data presented in the Figure 5, using *minSup* parameter set to 2 we receive only one frequent item set i.e. (11, 101, 201). After post-processing this item set can be presented in the human-readable form such as (ToS change, IP_A, dst port 5000). The major advantage of the data mining patterns is that for any network security specialist this information is quite clear i.e. that information hiding pattern instances concerning ToS field are detected in the stream that is originated from the machine with IP_A address to the destination port 5000. Another advantage of the data mining patterns is their generalization property. As we prove in next sections, the same process can be used for the detection of a simple steganographic method i.e. which uses only one communication stream, as well as some advanced i.e. distributed covert channel techniques described in the Section 3.

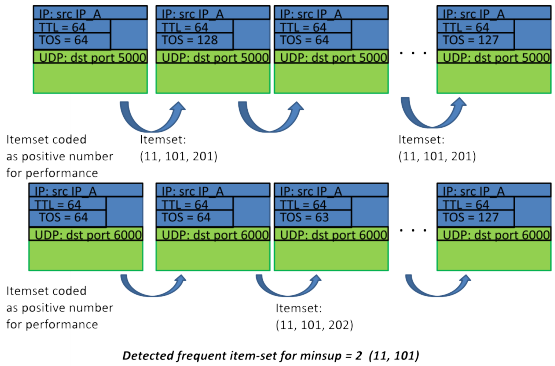


Figure 6: Distributed covert channel realization using two independent streams (multi-stream transmission), coded item sets associated with IHPIs and detected frequent item set.

The method illustrated in Figure 6 presents the data hiding method which conceals messages in the changes of the ToS field and distributes secret data into two communication sessions. Due to the fact that both streams utilize the same steganographic method and are sourced from the same IP address the first two elements of the input item sets have the same values: 11 and 101. However, because two separated streams are used then the last item in the item sets which codes destination port is different. Discovery of the frequent item sets in such input data with minimal support set to 2 results in only one frequent item set: (11, 101). In contrast to the previous

example, the current frequent item set does not have an item corresponding to the TCP port, which uniquely identifies particular communication session. Lack of this information can be treated as a sign that the detected patterns describe suspicious activities related to numerous connections. This is the next benefit of the data mining-based patterns approach. A single method using the same kind of input data can detect steganographic transmission both in one particular communication session and in various sessions sourced from a single IP address. However it must be noted that this uncomplicated data hiding method can be potentially detected also using simple counters of information hiding pattern instances. However, it must be noted that in such a case the counters must be dedicated to each connection and each IP address. If we only detect one steganographic transmission, for example, using only IPv4 TTL field, like in the experiments presented in this paper, we could simply measure the frequency of the TTL field changes. However, if the steganographer introduces another, new data hiding technique this will result in the need to introduce a new counter. Using frequent item sets we have one detection method which is able to discover any change in the observed fields. Moreover, it can detect that particular changes are introduced between given IP addresses or even in one particular TCP connection.

Introduced in this section data mining method is therefore more scalable – the same procedure can be used for the detection of various simple and distributed steganographic methods. More detailed description of the performed experiments is presented in the next section.

5 EXPERIMENTAL EVALUATION

In the two previous sections the distributed network covert channel techniques and methods for their detection were introduced. Here we want to experimentally prove that the presented detection solution is feasible. All described experiments were conducted using the dedicated experimental test-bed which is detailed below.

Moreover, due to problems with availability of fully functional and configurable steganographic tools which should enable easy configuration and tuning we decided to implement our own SDN-based covert senders and covert receivers. Additionally, experimental software which allows discovery of the frequent item sets associated with the covert traffic in an off-line manner has been prepared.

5.1 Experimental test-bed and methodology

Figure 7 presents our dedicated experimental test-bed which utilizes SDN application for insertion of covert traffic used for experimental evaluation. Overt traffic is generated by dedicated scripts which uses PythonJS. The same method is used for preparation of background, benign traffic destined to Alexa Top 5000 Global Sites[1]. Within the test-bed we have created two LANs that are connected to the Internet via NAT gateways (sdn_nat01 and sdn_nat02). Packets in both networks are routed by SDN switches (sdn_sw01 and sdn_sw02). In the first network sdn_01 and sdn_02 machines are connected which serve as the clients as they both have a web browser installed. In the second network sdn_03 machine which hosts a simple webpage is located. Visiting this webpage from the

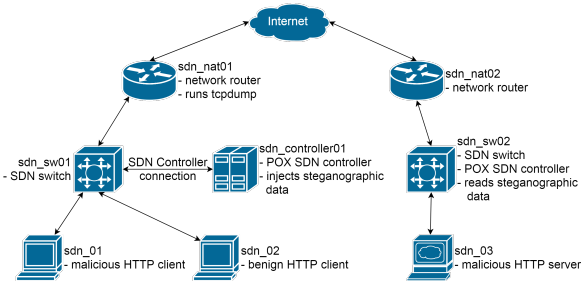


Figure 7: Experimental test-bed.

client machines (sdn_01 and sdn_02) generates HTTP traffic within which covert data is transfered between LANs.

In the considered scenario machine sdn_01 acts as a rogue device, which utilizes PhantomJS headless web browser[2] to periodically visit a website hosted on the sdn_03 machine. This generates the network traffic between the two LANs. When the traffic is passing through the sdn_sw01 machine it is sent to the sdn_controller01, where the steganographic data is injected into selected packets. On the other hand when the traffic passes through the second LAN it is inspected by the SDN controller installed on the sdn_sw02 and the secret data is extracted.

Additionally, the same test-bed is used for generating benign traffic. The sdn_02 machine utilizes a PhantomJS headless browser to periodically visit the most popular websites listed on Alexa Top 5000 Global Sites.

The details on the both datasets (i.e. benign and steganographically modified) generation are described further in subsection 5.5.

5.2 Utilized steganographic method

For our study, we have implemented a steganographic method which relies on the modulation of the IPv4 Time-To-Live (TTL) field values. This network covert channel works as follows.
When a new connection suitable for the covert data transfer (i.e. between hosts in our SDN networks) is established the first few packets are left unmodified. This allows the covert receiver to measure the expected value in the TTL field which depends on the number of intermediate nodes that the packets need to traverse before reaching their destination. This value can be seen as a "hidden data carrier" — subtracting it from the actual TTL in all consecutive packets within the connection reveals the differences where the covert data is embedded. In our data hiding technique a particular value of the difference that is mapped to a bit or a group of bits which forms a *symbol*. We modify the TTL field value within the connection by adding or subtracting a given symbol from its current value. For example, with the expected TTL=64, a change of +5 (i.e. TTL=69) denotes a binary 1 whereas a modification of -5 (i.e. TTL=59) denotes binary `0`. If the packet carries no secret data then the TTL value is left unchanged. By providing 2^n symbols our method allows us insert to n bits of secret data per packet — for example a list of symbols: `-5, -10, +5, +10` would map to the following TTL values: `59, 54, 69, 74` and secret data binary values: `00, 01, 10, 11`. We transmit the secret data by dividing

[1] https://www.alexa.com/topsites

[2] http://phantomjs.org

it into chunks of n bits and by mapping their binary values into symbols. This allows us to potentially embed up to 8 bits of secret data per packet (thus filling the entire TTL field) but in practice a too small TTL value may cause the packet to be dropped by the routers before reaching its destination. Additionally, significant and frequent changes of the TTL field over a broad range of values might raise suspicion and render the method easily detectable. By observing the characteristics of the benign network traffic, we concluded that no more than 4 bits of data should be transmitted in a single packet and the symbols should not exceed a difference of ±10 over an expected TTL value. However, in order to achieve increased stealthiness we used only 1 bit of secret data per packet during the experiments conducted in this paper. The benefit of our SDN-based approach for implementing covert communication is that we can spread the secret data between multiple TCP connections for the increased stealthiness without significantly degrading covert data transmission rate.

5.3 Packet sniffer and analyzer

One of the key components of the detection software is the packet sniffer and analyzer which is installed on the sdn_nat01 machine. Because sdn_nat01 machine acts as a network gateway, it has the capability to intercept the traffic crossing the LAN boundaries. We have decided to utilize it as the network traffic sniffer in order to more accurately simulate the third party monitoring network activity. To capture the traffic, we have utilized *tcpdump* tool and segmented the captured traffic into multiple .pcap files.

In order to analyze the collected traffic we have developed a packet analyzer software that compares pairs of consecutive packets belonging to each flow (*e.g.* sent within TCP connection generated by the browser) and we note the difference between them to a JSON file. We compare the header values of Layer 3 (IPv4) and Layer 4 (TCP, UDP, ICMP) protocols. This step acts as a pre-processing filter – packets that do not contain any difference in the fields' values are omitted and are not logged into the JSON file. We wrote this program in C++ using pcapp++ packet sniffer and parser library[3] and Niel Lohmann's JSON library[4].

5.4 IHPI detection software

The above mentioned JSON file which contains header values' differences is later used in an offline covert traffic detection process which utilizes *pyfim* – a frequent item sets discovery library for Python[5].

Our software reads the JSON file, analyzes each modified packet's entry and extracts its basic characteristics such as IPv4 source and destination addresses and/or TCP ports. In addition to the basic description, it detects a predefined set of IHPI events, such as a TTL change. If an IHPI event is not present, then this entry is discarded. Otherwise, a new entry is added to the list of "suspicious" events for further processing in *pyfim* library.

Because the library efficiently operates on the integer values, we have developed a convenient way to map value ranges inside of a 64-bit integer to represent IHPI entries (see Table 1). For example, IHP instance type is represented by a number ranging from `0x0000` to `0xFFFF`, IPv4 source address by a number from `0x10000` to `0x10000FFFF`, IPv4 destination address by a number from `0x100010000` to `0x20000FFFF`, etc.

Table 1: Numerical values mapping within a 64 bit integer.

Type	Integer range	Size	Name
ICMP	0x400070600 - 0x4000805FF	2^{16}	Echo ID
	0x400070500 - 0x4000705FF	2^{8}	Code
	0x400070400 - 0x4000704FF	2^{8}	Type
UDP	0x400060400 - 0x4000703FF	2^{16}	Destination port
	0x400050400 - 0x4000603FF	2^{16}	Source port
TCP	0x400050300 - 0x4000503FF	2^{8}	Flags
	0x300050300 - 0x4000502FF	2^{32}	Acknowledgement
	0x200050300 - 0x3000502FF	2^{32}	Sequence
	0x200040300 - 0x2000502FF	2^{16}	Destination port
	0x200030300 - 0x2000402FF	2^{16}	Source port
IPv4	0x200020300 - 0x2000302FF	2^{16}	Flags
	0x200010300 - 0x2000202FF	2^{16}	ID
	0x200010200 - 0x2000102FF	2^{8}	DSCP/TOS
	0x200010100 - 0x2000101FF	2^{8}	Protocol
	0x200010000 - 0x2000100FF	2^{8}	Time To Live (TTL)
	0x100010000 - 0x20000FFFF	2^{32}	Destination address
	0x000010000 - 0x10000FFFF	2^{32}	Source address
IHP	0x000000000 - 0x00000FFFF	2^{16}	IHP Instance Type

We have also developed an object-oriented software which allows to easily convert back and forth between human-readable IHP entries and their numerical representation.

The list of suspicious entries in the numerical form is then processed by the *pyfim* library to extract patterns that can be an indication of suspicious activity that has taken place during the experiment. The results are then decoded back into a human-readable form.

5.5 Experimental dataset generation and considered scenarios

During the conducted experiments we wanted to investigate how the proposed distributed network covert channel detection system performs with the real-life data. As already mentioned for this purpose we utilized our experimental test-bed in order to generate benign background traffic as well as covert transmissions with various characteristics.

For both, background and steganographic carrier traffic we used HTTP protocol as currently it is the most popularly used in the Internet and in most cases in the current communication networks it can be send without any restrictions. This makes it a very good candidate for the hidden data carrier.

The benign background traffic has been prepared using dedicated scripts which utilize PhantomJS browser to automatically visit most popular websites from the Alexa database, *i.e.* the list of the most popular websites worldwide, with 5 seconds delay between each visit. We ran the script for ca. 1 hour, during which ca. 800 MB of

[3]https://github.com/seladb/PcapPlusPlus
[4]https://github.com/nlohmann/json
[5]http://www.borgelt.net/pyfim.html

data were captured. During this experiment, 1215 unique domains were visited and 4947 HTTP Requests were sent. After careful analysis of initial data we observed that rarely, however repetitively, in the benign traffic some changes in the TTL field even within a single TCP connection appear. The recorded background traffic contains ca. 542 000 packets and within this dataset we observed more than 300 packets with different TTL field value than in the preceding packet. This may be caused by some rerouting of traffic due to e.g. bottlenecks in the network. More details related to this dataset are presented in the next section.

The malicious traffic dataset containing covert communication was generated in the same test-bed using dedicated SDN application responsible for injecting secret data into the selected streams of overt traffic. The overt connections have been prepared using PhantomJS and directed to the preconfigured web server. Depending on the requirements of the experiments web server hosts web pages using keep-alive mechanism or without it. The keep-alive mechanism allows transmission of multiple resources from the Web server using a single connection, however, we noticed that for the performance reasons browsers sometimes use a small number of connections despite the activated keep-alive option.

In this paper we consider two experimental scenarios. In the first scenario we simulate a simple steganographic method which at any given moment utilizes a single data stream for covert communication purposes (keep-alive option on). In the second scenario the configuration of the web server forces a script to download each resource in a separate connection. This in effect allows the distribution of the secret data over multiple TCP connections (keep-alive option off). We analyzed both scenarios in order to be able to compare the results for the typical network covert channel case with the distributed network covert channel realization.

5.6 Experimental results

The detection system introduced in Section 4 is capable of discovering frequent item sets in the stream of information hiding pattern instances which are detected within raw network traffic. During the pre-processing phase each detected IHPI is encoded as an item set containing six attributes: source and destination IP addresses, source and destination ports, protocol, type of detected IHPI.

After the pre-processing phase the frequent item sets are mined using *pyfim* library and the obtained results are post-processed. At the end of the detection process discovered frequent item sets are presented in the human-readable form with an additional information concerning its *support* i.e. the number of its occurrence. Figure 8 presents an exemplary detected frequent sets in the human-readable form after the final post-processing. It must be noted that in this figure the proposed detection system illustrates each detected frequent item set in a separate line and 4 frequent item sets are presented in total (due to the paper layout each itemset is presented in the 5 lines).

To make further analysis of the detected frequent sets easier the pre- and post- processing phases preserve not only the value of the item but also its initial type. In result, in the presented excerpt each item value is preceded by its type. The labels used are so evident that practically any network engineer should be able to understand them i.e., for example, the first 5 lines describe

Figure 8: An exemplary detected frequent sets

```
'IHP Instance': TTL change
'IP Source address': 192.168.171.100
'IP Destination address': 192.168.143.176
'Protocol': 6   'TCP Source port': 48162
'TCP Destination port': 80   6
'IHP Instance': TTL change
'IP Source address': 192.168.171.100
'IP Destination address': 192.168.143.176
'Protocol': 6   'TCP Source port': 48130
'TCP Destination port': 80   6
'IHP Instance': TTL change
'IP Source address': 192.168.171.100
'IP Destination address': 192.168.143.176
'Protocol': 6   'TCP Source port': 48076
'TCP Destination port': 80   6
'IHP Instance': TTL change
'IP Source address': 192.168.171.100
'IP Destination address': 192.168.143.176
'Protocol': 6   'TCP Source port': 48104
'TCP Destination port': 80   6
```

some anomaly in the TTL field within the TCP connection directed from 192.168.171.100:48162 to 192.168.143.176:80. The last positive integer number at the end of the line describes the support of the given frequent item sets i.e. the number of all item sets in the analyzed data which contains this subset. We can simply assume that in this particular analyzed connection six anomalous packets appeared.

Frequent item sets detection algorithms work on one input data set, however, it must be noted that network traffic in general can be provided in an endless manner. Due to this fact and for performance and usability of the detection system reasons we decided to discover frequent item sets in 5 minute-long windows. The analysis of the obtained results within a single window is then used by a simple classifier which utilizes the number of detected frequent sets to decide whether covert communication has been discovered in the inspected network traffic trace or not.

5.6.1 Detection results for the network covert channel which uses only one connection at any given moment. As already mentioned during the first phase of the conducted experiments we utilize the web server configured with activated keep-alive mechanism which causes all IHPIs to be transmitted within a single data stream for the whole duration of the covert data transfer. This is the simplest version of the covert transmission and in result less stealthy. During each experiment we transmitted 512 bytes of secret data and we modified the steganographic bandwidth i.e. the rate of how fast the covert data can be transferred. In our proof-of-concept implementation the hidden data rate (i.e. the method aggressiveness) can be modified by defining the number of unchanged packets after which a single IHPI (i.e. stegpacket with modified TTL value) is introduced – we call this rate *stegpacket generation rate* – SG_R. For

Table 2: Experimental results for the covert transmission with the secret data injected in the single connection (web server configured with the activated keep-alive option).

Trans.		Minimal support																	
		3			10			25			50			100			150		
time [s]	SG_R	FIS#	Avg.	Sup.	FIS#	Avg	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.
228	10	435	6		341	6		143	6		1	5	(8235)	1	5	(8235)	1	5	(8235)
338	15	456	6		286	6		121	6		1	5	(7329)	1	5	(7329)	1	5	(7329)
456	20	552	6		293	6		1	5	(5398)	1	5	(5398)	1	5	(5398)	1	5	(5398)
567	25	620	6		78	6		1	5	(4362)	1	5	(4362)	1	5	(4362)	1	5	(4362)
680	30	241	6		134	6		41	6		1	5	(3686)	1	5	(3686)	1	5	(3686)
1134	50	362	6		3	6		1	5	(2209)	1	5	(2209)	1	5	(2209)	1	5	(2209)
1706	75	227	6		1	6	(10)	1	5	(1495)	1	5	(1495)	1	5	(1495)	1	5	(1495)
2251	100	114	6		1	5	(1104)	1	5	(1104)	1	5	(1104)	1	5	(1104)	1	5	(1104)
3501	150	103	6		1	5	(741)	1	5	(741)	1	5	(741)	1	5	(741)	1	5	(741)
4532	200	24	6		1	5	(555)	1	5	(555)	1	5	(555)	1	5	(555)	1	5	(555)
11360	500	1	6	(3)	1	5	(219)	1	5	(219)	1	5	(219)	1	5	(219)	1	5	(219)
16984	750	1	5	(147)	1	5	(147)	1	5	(147)	1	5	(147)	1	5	(147)	0		
21747	1000	1	5	(111)	1	5	(111)	1	5	(111)	1	5	(111)	1	5	(111)	0		
33985	1500	1	5	(72)	1	5	(72)	1	5	(72)	1	5	(72)	0			0		
45313	2000	1	5	(55)	1	5	(55)	1	5	(55)	1	5	(55)	0			0		
56642	2500	1	5	(46)	1	5	(46)	1	5	(46)	0			0			0		
67970	3000	1	5	(38)	1	5	(38)	1	5	(38)	0			0			0		
79299	3500	1	5	(32)	1	5	(32)	1	5	(32)	0			0			0		
90627	4000	1	5	(27)	1	5	(27)	1	5	(27)	0			0			0		
113284	5000	1	5	(23)	1	5	(23)	0			0			0			0		
169927	7500	1	5	(14)	1	5	(14)	0			0			0			0		
226569	10000	1	5	(12)	1	5	(12)	0			0			0			0		
283212	12500	1	5	(8)	0			0			0			0			0		
339854	15000	1	5	(8)	0			0			0			0			0		

SG_R: Stegpacket Generation Rate FIS#: The number of frequent item sets Avg: The average number of items Sup.: Support

example, if the stegpacket generation rate equals 10 then it means that one packet is steganographically modified every 10 transmitted packets.

During the conducted experiments i.e. frequent item sets detection we investigate how many frequent sets are discovered depending on the stegpacket generation rate and utilized minimal support. The first parameter improves the steganographic method stealthiness i.e. the higher it is the lower the number of packets that is covertly modified. The second parameter helps to decide how many IHPIs must be discovered in the analyzed time window to detect at least one frequent item set. All results obtained for this experimental phase are presented in Table 2 and concerns results achieved for the first 5-minute time window. Our research shows that the obtained results are consistent throughout all time windows, so we focus only on the first one. Due to the very long duration of some experiments especially those with the very high SG_R the results are estimated based on the first 20 minutes of the covert transmission (in Table 2 the estimated results are marked in italic).

From the presented results we can observe that when we reduce the stegpacket generation rate then the duration of the covert transmission increases (which is obvious). For the smaller values of minimal support we are able to detect more than one frequent item sets. In this case the detected frequent item sets consist of 6 items which means that full description of the TCP stream even with source port is provided by the detection the system. However when we increase the stegpacket generation rate or we set minimal support to the higher value we can detect only a single frequent set. In these cases the detected frequent set has only 5 items thus this means that we do not have complete information about TCP connection anymore.

What should be emphasized is that even for very high SG_R values e.g. 500 we are still able to detect at least one frequent item set regardless of the minimal support used (in the Table 2 if only one frequent item set is detected its support is presented in the brackets). Such SG_R value corresponds to the steganographic bandwidth of around 0.3 bps. Moreover, when SG_R is higher than 500 only for the smaller minimal support values frequent item sets are detected.

To conclude we are always capable of detecting covert data transfer, however, sometimes the minimal support needed for this purpose is very low. It must be also noted that when the size of the analyzed data set is huge then this can have negative impact on the overall system performance.

5.6.2 Detection results for the distributed multi-flow network covert channel. As already mentioned in the second phase of the

Table 3: Experimental results for the covert transmission with the secret data injected in many concurrent HTTP streams (web server configured without the keep-alive mechanism).

		Minimal support																	
Trans.		3			10			25			50			100			150		
time [s]	SG_R	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.
280	10	50	6		1	5	(7442)	1	5	(7442)	1	5	(7442)	1	5	(7442)	1	5	(7442)
416	15	21	6		1	5	(5317)	1	5	(5317)	1	5	(5317)	1	5	(5317)	1	5	(5317)
554	20	13	6		1	5	(4051)	1	5	(4051)	1	5	(4051)	1	5	(4051)	1	5	(4051)
727	25	5	6		1	5	(3233)	1	5	(3233)	1	5	(3233)	1	5	(3233)	1	5	(3233)
842	30	3	6		1	5	(2697)	1	5	(2697)	1	5	(2697)	1	5	(2697)	1	5	(2697)
1407	50	1	5	(1632)	1	5	(1632)	1	5	(1632)	1	5	(1632)	1	5	(1632)	1	5	(1632)
2080	75	1	5	(1081)	1	5	(1081)	1	5	(1081)	1	5	(1081)	1	5	(1081)	1	5	(1081)
2780	100	1	5	(802)	1	5	(802)	1	5	(802)	1	5	(802)	1	5	(802)	1	5	(802)
4186	150	1	5	(535)	1	5	(535)	1	5	(535)	1	5	(535)	1	5	(535)	1	5	(535)
5601	200	1	5	(391)	1	5	(391)	1	5	(391)	1	5	(391)	1	5	(391)	1	5	(391)
14002	500	1	5	(166)	1	5	(166)	1	5	(166)	1	5	(166)	1	5	(166)	1	5	(166)
21012	750	1	5	(145)	1	5	(145)	1	5	(145)	1	5	(145)	1	5	(145)	0		
28015	1000	1	5	(80)	1	5	(80)	1	5	(80)	1	5	(80)	0			0		
42023	1500	1	5	(54)	1	5	(54)	1	5	(54)	1	5	(54)	0			0		
56030	2000	1	5	(41)	1	5	(41)	1	5	(41)	0			0			0		
70038	2500	1	5	(30)	1	5	(30)	1	5	(30)	0			0			0		
84046	3000	1	5	(27)	1	5	(27)	1	5	(27)	0			0			0		
98053	3500	1	5	(22)	1	5	(22)	0			0			0			0		
112061	4000	1	5	(21)	1	5	(21)	0			0			0			0		
140076	5000	1	5	(14)	1	5	(14)	0			0			0			0		
210115	7500	1	5	(12)	1	5	(12)	0			0			0			0		
280153	10000	1	5	(7)	0			0			0			0			0		
350192	12500	1	5	(6)	0			0			0			0			0		
420230	15000	1	5	(5)	0			0			0			0			0		

SG_R: Stegpacket Generation Rate FIS#: The number of frequent item sets Avg: The average number of items Sup.: Support

Table 4: Experimental results for the background traffic, frequent items sets discovered in the traffic destined to Alexa Top 5000 Global websites.

	Minimal support																	
Probed	3			10			25			50			100			150		
window #	FIS#	Avg	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.	FIS#	Avg.	Sup.
1	10	4.9		2	4.5		1	4	(65)	1	4	(65)	0			0		
2	3	5		1	4	(22)	0			0			0			0		
3	5	5		1	4	(43)	1	4	(43)	0			0			0		
4	5	5		1	4	(30)	1	4	(30)	0			0			0		
5	9	5		3	4.667		2	4		1	4	(54)	0			0		
6	2	5		1	4	(19)	0			0			0			0		
7	3	4		1	4	(10)	0			0			0			0		
8	4	5		1	4	(32)	1	4	(32)	0			0			0		

FIS#: The number of frequent item sets Avg: The average number of items Sup.: Support

conducted experiments the web server is configured without keep-alive mechanism. In effect, we obtain distribution of the secret data across many overt HTTP connections (cf. Figure 2).

Like in the previous case during each experiment 512 bytes of secret data are transmitted using various stegpacket generation rates. All obtained experimental results are presented in the Table 3. Similarly to the previous phase of experiments when the covert transmission becomes more stealthy (i.e. with higher SG_R values) the duration of the hidden data transfer increases. However, in contrast to the previous experiments in most cases we can detect

only one frequent item set. It must be noted that in all cases where we detect one frequent item set it consist of only 5 items. This is caused by the fact that there are many connections in which IHPIs are introduced which prevents detection of more specific frequent item set.

Additionally, this proves that such distribution of steganographic activity across many traffic flows leads to its increased stealthiness. However, like in the previous phase of experiments we are always able to determine the appropriate value of minimal support which leads to detection of the covert data transfer.

5.6.3 Estimation of the potential false positives in the real-life Internet traffic. In both previously described experimental phases we were able to tune the proposed data mining-based detection system thus in all cases the covert transmission has been discovered. However, as mentioned earlier, when analyzing gathered benign traffic we identified some anomalous changes in the TTL fields and to our best knowledge they are not associated with the covert data exchange.

That is why, below we present results that, if included, will allow to reduce possible false positives for our detection method. The data set of the background traffic is divided into 5-minute windows (the same as when the steganographic traffic was previously analyzed). Based on the time needed to visit Alexa Top 5000 Global Websites we obtained 8 such windows. In each window we were able to detect frequent sets associated with the anomalous TTL activity which were treated by the detection system as IHPIs.

Obtained results for the benign traffic are presented in Table 4. What should be emphasized, in contrast to the previous phases of the described experiments in this data set most of the detected frequent item sets have only 4 items (thus there is no information on the source and destination ports). Further analysis revealed that this is caused by the streams destined to the 80 and 443 ports.

As it can be seen in Table 4 when we use a minimal support value less or equal to 50, at least in two windows some frequent item sets are discovered. Using a simple classifier which treats detection of the frequent item set as a sign of covert transmission an alert associated with such activity will be generated. Therefore, to reduce false positives rate of our detection system we should use the minimal support set to at least 100.

However, as showed during two phases of the experiments with covert traffic presented in this paper if we use minimal support not smaller than 100 then we are able to detect simple data hiding method when it utilizes SG_R not greater than 1000 (i.e. 1 packet out of 1000 consecutive packets carries secret data) and the distributed network covert channel when it utilize SG_R not greater than 750 (i.e. 1 packet out of 750 consecutive packets carries secret data). Thus, it may be concluded that the presented detection results are promising as above mentioned SG_R values corresponds to the steganographic bandwidth of 0.19 and 0.24 bps, respectively. This proves that even very slow and potentially less intrusive hidden transmission can still be successfully detected.

6 CONCLUSION AND FUTURE WORK

In this paper we have evaluated whether data mining techniques can be used for the purposes of the distributed network covert channels detection. We have considered the typical state-of-the-art network covert channel which is based on the IPv4 TTL field values modulation and additionally the secret data is distributed among several data flows. The initial results that we obtained are promising and thus the presented approach should be further investigated. Despite the fact, that the conduced experiments concern only data hiding method which relies on modification of the IPv4 TTL field, we are confident that it can be useful for other covert techniques as well. In order to incorporate other steganographic methods we would only have to provide appropriate pre-processing procedure which introduces additional item sets for them. In situation when only a single steganographic method is used we should discover analogous frequent item sets, with the only change in item which describes particular information hiding pattern. In the case when many IHPs are used, we can discover frequent item sets without an item which indicates utilized covert technique.

Our future work will be focused on performing a more thorough investigation of the data mining-based detection by evaluating it using more types of the distributed network covert channels realization e.g. pattern hopping and/or pattern combination. Also for the TTL-based covert channel utilized in this paper a more extensive evaluation will be performed by e.g. considering different window sizes. Moreover, more in-depth investigation of the introduced data mining-based detection system's performance and limitations would be pursued.

7 ACKNOWLEDGMENTS

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0254. The supported project is named CoCoDe - Covert Communication Detection. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

REFERENCES

[1] A. Salih, X. Ma, E. Peytchev, Detection and Classification of Covert Channels in IPv6 Using Enhanced Machine Learning, In Proc. of Int. Conf. on Computer Technology and Inf. Systems (ICCTIS' 2015), UAE, August 2015
[2] F. G. Mohammadi, M. S. Abadeh, A Survey of Data Mining Techniques for Steganalysis, Recent Advances in Steganography, Hedieh Sajedi (Eds.), ISBN 978-953-51-0840-5, 2012
[3] S. Wendzel, S. Zander, B. Fechner, and C. Herdin. A pattern-based survey and categorization of network covert channel techniques. ACM CSUR, 47(3), 2015.
[4] R. Agrawal, R. Srikant, Fast algorithm for mining association rules, In J.B. Bocca, M. Jarke, and C. Zaniolo, editors. Proceedings of VLDB, pp 487-499, (1994)
[5] M. Klemettinen, A Knowledge Discovery Methods for Telecommunication Network Alarm Database, PhD. Thesis, University of Helsinki, (1999)
[6] R. Agrawal, R. Srikant, Mining Sequential Patterns, In Proceedings of 1995 Int. Conf. Data Engineering (ICDE'95), 3–14, Taipei, Taiwan, (1995)
[7] K. Cabaj, L. Caviglione, W. Mazurczyk, S. Wendzel, A. Woodward, S. Zander, The New Threats of Information Hiding: the Road Ahead, IEEE IT Prof., 2018
[8] W. Mazurczyk, S. Wendzel, Information Hiding: Challenges for Forensic Experts, Communications of the ACM, Vol. 61 No. 1, pp. 86-94, January 2018
[9] W. Mazurczyk, L. Caviglione, Information Hiding as a Challenge for Malware Detection, IEEE Security and Privacy Magazine, Iss. 2, 2015, pp. 89-93
[10] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, K. Szczypiorski, Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures, IEEE Press-Wiley, April 2016
[11] I. S. Moskowitz and M. H. Kang, Covert Channels - Here to Stay?, In 9th Annual Conference on Computer Assurance, pages 235–244, 1994.
[12] G. Fisk, M. Fisk, C. Papadopoulos, J. Neil, Eliminating Steganography in Internet Traffic with Active Wardens, In Proc. of Int. Wksp. on Inf. Hiding, Oct. 2002.
[13] M. Kang, I. Moskowitz, A Pump for Rapid, Reliable, Secure Communication, In Proc. of ACM Conf. on Comp. and Comm. Sec. (CCS), pp. 119–129, 1993.

4.2 PAPER II

Mazurczyk Wojciech, Wendzel Steffen, Cabaj Krzysztof: Towards Deriving Insights into Data Hiding Methods Using Pattern-based Approach, in: ARES 2018 Proceedings of the 13th International Conference on Availability, Reliability and Security / Doerr Christian, Schrittwieser Sebastian, Weippl Edgar (red.), 2018, ISBN 978-1-4503-6448-5, pp. 1-10, DOI:10.1145/3230833.3233261

Towards Deriving Insights into Data Hiding Methods Using Pattern-based Approach

Wojciech Mazurczyk
Warsaw University of Technology
Warsaw, Poland
wmazurczyk@tele.pw.edu.pl

Steffen Wendzel
Worms University of Applied Science/
Fraunhofer FKIE
Worms/Bonn, Germany
wendzel@hs-worms.de

Krzysztof Cabaj
Warsaw University of Technology
Warsaw, Poland
kcabaj@ii.pw.edu.pl

ABSTRACT

In network information hiding, *hiding patterns* are used to describe hiding methods and their taxonomy. In this paper, we analyze the current state of hiding patterns and we further improve their taxonomy. In order to more thoroughly characterize and understand data hiding methods applied to communication networks we propose to distinguish between sender-side and receiver-side patterns. Additionally, we show how information hiding patterns can be utilized to conveniently describe the realization of the distributed network covert channels.

CCS CONCEPTS

• **Security and privacy** → **Network security**; *Distributed systems security*; *Information flow control*; Pseudonymity, anonymity and untraceability; • **Social and professional topics** → *Computer crime*;

KEYWORDS

information hiding patterns, network steganography, covert channels; network security; taxonomies; information hiding

ACM Reference Format:
Wojciech Mazurczyk, Steffen Wendzel, and Krzysztof Cabaj. 2018. Towards Deriving Insights into Data Hiding Methods Using Pattern-based Approach. In *ARES 2018: International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3230833.3233261

1 INTRODUCTION

Network covert channels belong to the research domain of *network information hiding* [15]. *Network covert channels* are stealthy, unforeseen communication channels in computer networks. These channels are increasingly used by cybercriminals, e.g. to allow a covert transfer of malware data. However, they can be also used for legitimate purposes, such as communicating illicit information under Internet censorship.

ARES 2018, August 27–30, 2018, Hamburg, Germany

ACM ISBN 978-1-4503-6448-5/18/08.
https://doi.org/10.1145/3230833.3233261

Hiding patterns are descriptions of hiding methods for network covert channels. Because of their abstract nature, each hiding pattern serves an umbrella for numerous hiding methods. For instance, hiding data in the least significant bit (LSB) of the Hop Limit field in IPv6 can be represented by the same pattern as modifying the LSB of the Time to Live field in IPv4. In addition to describing hiding methods, patterns can also form taxonomies and have predefined, searchable and comparable attributes, making them an advantageous tool over existing taxonomy approaches.

Hiding patterns have originally been proposed by Wendzel et al. in [22]. The authors also presented a novel taxonomy of hiding patterns in their article. Later, the taxonomy and patterns were updated and extended by Mazurczyk et al. in [15]. There are also publications that discuss whether a new hiding method can represent a new or an existing pattern [20] and there is moreover work that describes the way in which hiding methods should be described (in the context of patterns) [19].

In this work, we analyze the key aspects of the hiding patterns and the current state of the taxonomy in the domain. However, the main contributions of this paper are that we show how this concept can be further extended by modifying the pattern-analysis process and extending the current taxonomy with new patterns. By taking into account more details on the hiding method's inner workings we hope that the resulting pattern categorization will contribute to a better understanding of the nature of network covert channels. Moreover, we also introduce and describe a pattern-based classification of *distributed* network covert channels.

The rest of this paper is structured as follows. Section 2 introduces fundamentals and related work on hiding patterns. We discuss limitations of the current patterns approach in Section 3. Section 4 introduces our improved taxonomy, a process for pattern-analysis as well as new patterns dedicated to the payload field and our pattern-based categorization of distributed network covert channels. Finally, Section 5 concludes our work and provides an outlook on future research directions.

2 FUNDAMENTALS

To aid the understanding of information hiding methods, an analysis of the existing network covert channels and corresponding protocols should be performed. Patterns provide an abstract and hierarchical view on these methods and their utilization in combination with network protocols.

As a starting point, we utilize the work by Wendzel et al. [22] on network information hiding patterns. In this work, the authors introduce a classification of network hiding techniques into so-called

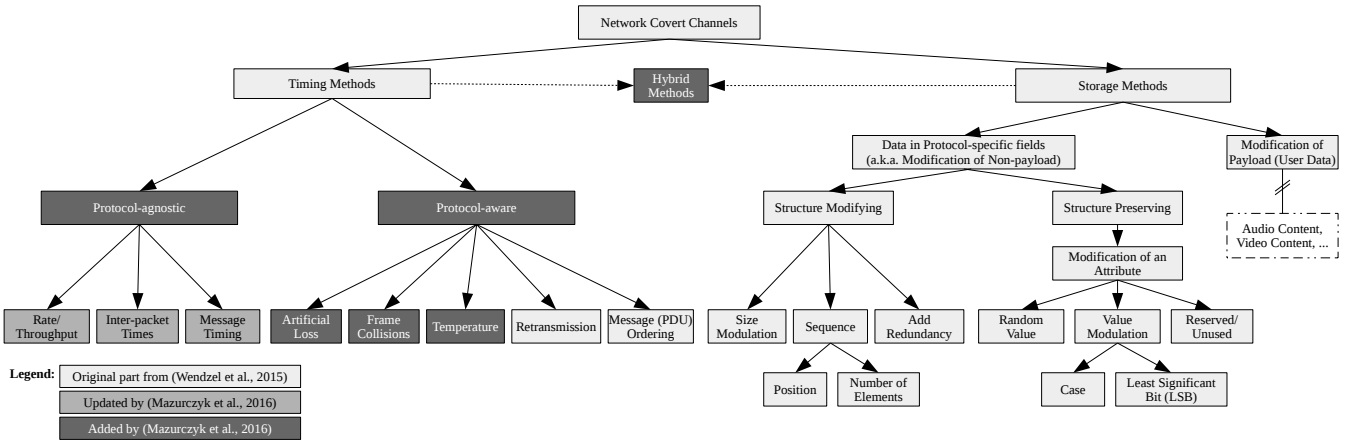


Figure 1: Information hiding patterns and their hierarchy introduced in [22] and updated in [15].

hiding patterns with the aim to potentially develop countermeasures for these patterns. In this perspective, information hiding patterns are defined as abstract descriptions of how to solve a problem (data hiding) in a given context (communication protocols). As patterns can be derived from other patterns, they can form hierarchies. Each hiding pattern is a unified and generic description of a particular family of hiding methods. Patterns must be described in a pre-defined format and require certain additional properties, such as at least three known occurrences of a pattern – cf. [22] for details. In [22] and [19], Wendzel et al. evaluated more than 130 existing network covert channel techniques from past decades and extracted abstract patterns from these techniques. It turned out that authors were able to represent all techniques by (only) 11 patterns, which were arranged in a hierarchical catalog described using *Pattern Language Markup Language* (PLML). While later work in [15] modified and extended their patterns, the core part of the hierarchy and several patterns remained (colored in white and light-gray in Fig. 1). Later modifications and extensions by [15] are colored in darker gray in Fig. 1. The latest description of all patterns shown in this figure is presented briefly in Table 1.

As it can be seen in Table 1, a hiding pattern's description is written in an abstract manner so that one pattern can be used to describe multiple hiding techniques at the same time. For instance, "modulate the least significant bits of a protocol field" is a very brief description of many published hiding methods which utilize the least significant bits of fields in arbitrary network protocols.

The above-mentioned classification is carrier-oriented and a "carrier" is defined as one or more overt traffic flows that pass between the covert sender and the covert receiver, consisting of protocol data units (PDUs, e.g. frames or packets). Typically, the carrier can be multi-dimensional, i.e. it offers many opportunities "places" or "events" for hiding data (called sub-carriers). As in other network covert channel categorizations the two main groups of methods are (Fig. 1):

- *storage methods*: a class of network steganography methods that modify the "places" (sub-carriers) in a carrier to create a storage covert channel. These techniques hide information by modifying e.g. protocol fields, such as unused bits of a header.
- *timing methods*: a class of network steganography methods that modify the timing of "events" of a carrier to create a covert channel. These techniques hide information, e.g. in the timing of protocol messages or packets.

Some important changes have been introduced in [15] when compared with original categorization from [22]. These include:

- defining 14 patterns (8 timing patterns and 6 storage patterns), compared to 11 patterns (4 timing and 7 storage) proposed originally. Note that the increased number of hiding patterns is mainly caused due to adding new layer of classification in [15] for timing patterns which have been divided into "protocol agnostic" or "protocol aware" groups.
- the pattern 'PDU Corruption/Loss Pattern' has been removed from the storage patterns and instead the 'Artificial Loss' pattern which full name is 'Artificial Message/Packet Loss' and the 'Frame Collision' pattern have been added to the list of timing patterns.
- A few patterns have been slightly modified/renamed.

The paper [22] introduced also several other concepts which explain suitably some network covert channels' phenomena, i.e. pattern variation, pattern combination, and pattern hopping.

First, *pattern variation* is a transformation-like approach for covert channels. The utilized network protocol is defined as the pattern's context. Therefore, a pattern's application can change from one network protocol to another – without redesigning the most important aspects and inner workings of the hiding technique itself. Next, *pattern combination* allows the use of multiple patterns at the same time (within the same carrier, e.g. by modifying many sub-carriers at once). This is typically performed to increase available steganographic bandwidth – thus in short it is a parallel utilization of multiple network covert channels simultaneously. Finally, *pattern hopping* varies the use of patterns over time – usually it is applied in order to increase stealthiness. This can be briefly summarized as a sequential utilization of various network covert channels in time using different (sub-)carriers.

Table 1: Information hiding patterns as introduced in [22] and updated in [15].

Pattern Name	Pattern Description
Rate/Throughput	The covert channel sender alters the data rate of traffic from itself or a third party to the covert channel receiver.
Inter-packet Times	The covert channel alters timing intervals between network PDUs (interarrival times) to encode hidden data.
Message Timing	Hidden data is encoded in the timing of message sequences, e.g. acknowledging every *n*'th received packet or sending commands *m* times.
Artificial Loss	The covert channel signals hidden information via artificial loss of transmitted messages (PDUs).
Frame Collisions	The sender causes artificial frame collisions to signal hidden information.
Temperature	The sender influences a third-party node's CPU temperature, e.g. using burst traffic. This influences the node's clock skew. The clock skew can then be interpreted by the covert receiver by interacting with the node.
Retransmission	A covert channel retransmits previously sent or received PDUs.
Message Ordering	The covert channel encodes data using a synthetic PDU order for a given number of PDUs flowing between covert sender and receiver.
Size Modulation	The covert channel uses the size of a header element or a PDU to encode a hidden message.
Sequence Modulation	The covert channel alters the sequence of header/PDU elements to encode hidden information. This pattern divides further into: P2.a. Position and P2.b. Number of Elements patterns.
Add Redundancy	The covert channel creates new space within a given header element or within a PDU in which to hide data.
Random Value	The covert channel embeds hidden data in a header element containing a (pseudo-)random value.
Value Modulation	The covert channel selects one of the *n* values that a header element can contain to encode a hidden message. This pattern divides further into: P6.a. Case Pattern and P6.b. Least Significant Bit (LSB) patterns.
Reserved/Unused	The covert channel encodes hidden data into a reserved or unused header/PDU element.

It must be also noted that in the reminder of this paper we will rely on the unified description for network information hiding methods introduced in [19]. This paper has been the first attempt to standardize the description of network covert channels which is suitable, e.g. to assess their novelty and impact of the method on the state-of-the-art. In [19], the proposed description of data hiding methods is split into three categories: *(i)* general information about the hiding method; *(ii)* description of the hiding process, and *(iii)* potential or tested countermeasures. The first two categories comprise sub-categories and each (sub-)category can be mandatory or optional (Fig. 2).

The category "hiding method general information" consists of a link to existing network hiding patterns. It also includes a discussion of the application scenario and requirements of the carrier. From the perspective of this paper the most important category, i.e. "hiding method process", is split into four parts: the sender-side and the receiver-side description of the hiding method, the details of the covert communication channel, and the description of an associated covert channel control protocol (if applicable). The third category discusses both, potential and evaluated countermeasures, including those that detect, limit or prevent the particular hiding method's use. In the following we will reference to the fragments of this unified description when it comes to the pattern categorization.

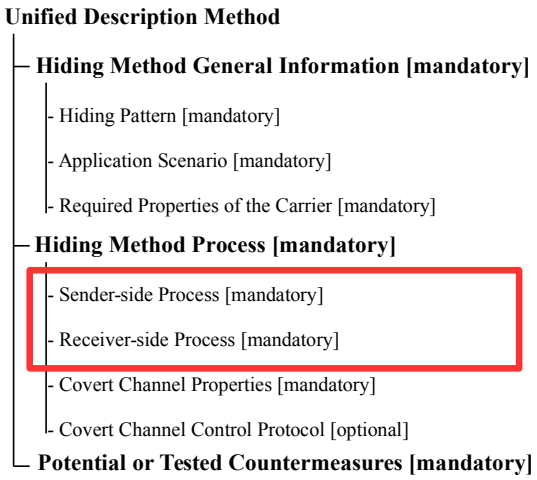


Figure 2: The unified description structure for data hiding methods as introduced in [19].

3 ANALYSIS OF THE EXISTING TAXONOMY

Our analysis has shown that the current information hiding patterns approach can be further extended to include the following aspects:

- *Incorporate More Details on Data Hiding Methods:* The key criterion of the current pattern taxonomy for deciding which pattern an analyzed method represents is to determine how the secret data is encoded. Thus, due to this it is omitting some details on how the data hiding method works (from the sender-side and receiver-side process – this will be shown further in the next sections). This "flattens" the description of the inner workings of the data hiding methods and thus may prevent that *all* details of a hiding method are considered. A more thorough patterns grouping is desired to more accurately categorize existing network steganography methods.
- *Support Hybrid Patterns:* For some cases it is difficult to assess whether the analyzed method is storage, timing or hybrid – a clearer distinction and unambiguous formula to deduce this is desirable.

- *Multi-Packet and -Flow Characteristics Support:* The current categorization makes no clear distinction between hiding methods that are focusing on a single packet or multiple packets. Also, there is no clear distinction between single- and multi-flow methods. For example, consider a covert channel that modulates IPv4 ToS values in such a way that the sequence of ToS values from the consecutive packets is interpreted as a single secret data bit – currently such a method does not match any hidden data pattern. Moreover, some hiding methods such as [10] utilize multiple flows. It is thus beneficial to make the original pattern descriptions more generic, i.e. less dependent on specific units such as PDUs or packets.
- *Coverage of Sophisticated Hiding Methods:* It is not exactly clear whether recent, more advanced network steganography concepts like inter-protocol steganography [9], protocol switching covert channels [21], multilevel steganography [5], adaptive covert communication [23], etc. can be accurately expressed using current pattern categorization. Pattern combination, pattern hopping and pattern variation are means to represent them, but not to the full extent.
- *Influence on Payload:* In the original design decision of the pattern-based approach, arbitrary content, e.g. digital files, were considered as part of digital media steganography instead of network information hiding. However, in some cases, such as in VoIP steganography, where there are data hiding methods that affect the payload field, it can be helpful to have a taxonomy that covers also the transmitted payload. In principle the patterns should be analogous as they too adhere to the storage group.
- *Distinction Between Secret Data Embedding and Transfer:* It is also worth to emphasize that from the pattern-based countermeasures perspective it is more important to know which pattern represents the covert technique *within* the communication channel. It must be noted that in particular the information hiding patterns used at the sender-side process to embed secret data may not exactly represent themselves in the same while traversing within the hidden data carrier through the communication network.
- *Embrace PDU Corruption Pattern:* As mentioned, in [22] 11 (4 timing and 7 storage) patterns have been defined while in [15] there are 14 (8 timing and 6 storage) patterns. However, the pattern 'PDU Corruption/Loss' has been removed from the storage patterns group by [15]. In fact, it is our belief that it is beneficial that the 'Artificial Message/Packet Loss' pattern has been added into timing patterns but still the 'PDU Corruption' pattern should be considered in storage scenarios.

Based on the above-mentioned points, we describe how we envision enhancements to the current information hiding patterns concept in the next section.

4 EXTENSION AND MODIFICATION OF THE PATTERNS APPROACH

In this section, we present the proposed modification for the original information hiding patterns concept which can help in deriving further insights into understanding the nature of various types of network covert channel techniques. More specifically, in subsection 4.1 we propose how the original pattern approach can be extended in order to include the sender-side and receiver-side processes which influences both pattern creation process and covert techniques categorization. Next, in subsection 4.2 we propose new patterns applicable to the payload field. Finally, in subsection 4.3 we discuss the *distributed* network covert channels and how the information hiding patterns concept can be used to conveniently describe them.

4.1 A New Process to Analyze the Details of Pattern-Application

Considering the arguments from Section 3, we propose an approach based on [20], which describes how to determine the novelty of a new hiding technique and whether a hiding technique actually represents a *new* pattern, or not. Instead, our goal is to gain additional insights into the inner-workings of the data hiding method, i.e. we do not *replace* the original approach.

In the current categorization, authors of a new data hiding technique first describe their technique, e.g. informally or using [19]. Then, based on how the secret data is *embedded* one pattern is selected that represents the hiding method. Therefore, authors first decide whether the hiding method is storage or timing, then, whether it is protocol-aware/agnostic (timing channel) or header structure preserving/modifying (storage channel). If a hiding method does *not* fit into the current pattern representation, it is considered a *new* pattern which can be added to the taxonomy. The related decision-making process can be found in [20].

We propose a similar but modified version of this approach. However, as mentioned, our approach targets a different goal, namely to derive *more insights* related to the information hiding method itself. It must be noted that we do not focus only on how the secret data for a certain data hiding method is embedded (which is only a part of the sender-side process) but instead we want to detail both the complete sender- and receiver-side processes and represent them with patterns (and for this purpose, we "borrow" the already existing patterns.

In our proposed approach, the known hiding patterns of existing publications and websites, e.g. [15, 22] or https://ih-patterns.blogspot.com, which are tagged as storage or timing patterns, are taken into account. Then for the hiding method that needs to be described using the network covert channel patterns approach, the corresponding patterns for both, sender- and receiver-side processes are selected. Finally, based on the result and depending on what types of patterns have been assigned to the method, the *method itself* is concluded as a storage, a timing or a hybrid method – this selection process is explained in the details below.

The described improved approach which aims to derive more insights from the data hiding methods using pattern approach allows to repaint the categorization from Fig. 1. However it must be noted that in the modified approach we categorize *network covert channel patterns* and not data hiding *methods*. Thus, we start the derived classification from the network covert channel patterns which are then divided into timing and storage ones (Fig. 3). Afterwards, each of the methods that needs to be evaluated is assigned with at least

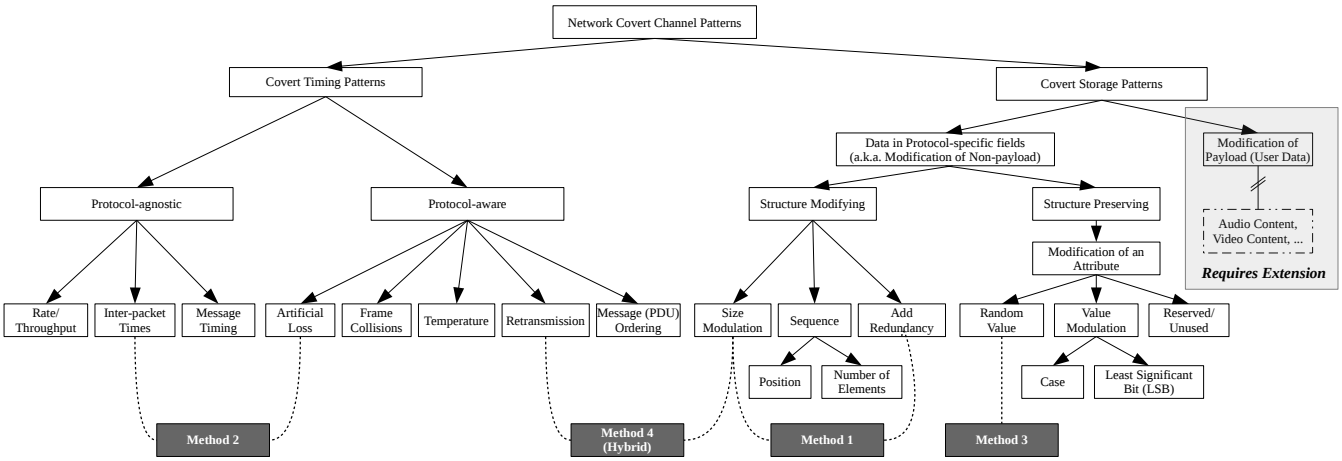


Figure 3: Improved aspects of the existing pattern-based taxonomy.

one or more patterns to its sender- and receiver-side processes separately (for each side at least one or alternatively more patterns must be selected).

It must be also noted that using this approach it may be possible to evaluate in greater detail which patterns are most often used jointly only at the sender-side process (as more than one pattern can be assigned) or only at the receiver-side process, or finally which patterns typically coexist at the sender-side and the receiver-side processes. This can be achieved by performing a thorough analysis of network covert channels defined in the literature (however, due to space limitation it will be not part of this paper). In result of such an analysis this can lead to the identification of potential relationships between defined patterns, i.e. whether for some of them it is "easier" to coexist with other patterns within the data hiding method (as in the case of the extended approach the sender and the receiver processes can be investigated separately or jointly).

But more importantly, it is also possible to investigate whether besides of joint patterns utilization (at the sender-side, receiver-side or both sides), other pattern mixes are also possible. For example, consider Method 4 in the Figure 3. It is characterized by the patterns *Retransmission* and *Size Modulation*, which makes it a hybrid method. However, the question arises whether is would be possible to construct a data hiding method that apart from these two patterns utilizes e.g. *Message (PDU) Ordering* pattern and how this will impact its properties.

In result, new, previously unknown network information hiding methods or improved versions of existing ones can be designed and developed and relationships between the existing patterns can be investigated and determined. It must be noted that using the existing pattern classification it was possible to assign only a single pattern for a certain hiding method which corresponds best with the secret data embedding process. However, in the extended approach (which is different when compared to the original concept) it is possible to:

- assign more patterns to the sender-side process if it is required in order to express to a full extent how the sender-side of the hiding method operates,

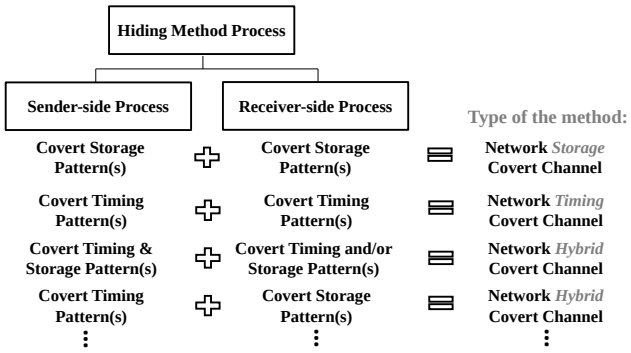


Figure 4: Improved process to decide on the network covert channel type based on the assigned patterns.

- include also the receiver-side process and its corresponding patterns.

Such an approach may not only help to better understand the nature of the network covert channels and their creation process, but it can also provide new insights into how to construct more efficient and effective detection solutions. This can be achieved by designing and developing detection methods, so they precisely will be looking for the specific artifacts related to the representation of the certain patterns in the communication channel (and/or e.g. the presence of their coexistence).

Finally, each method based on the selected patterns for the sender- and for the receiver-side processes is assigned to one element of the group {*storage*, *timing*, *hybrid*}. This is done as illustrated in Fig. 4. In principle, if both the sender- and the receiver-side processes are characterized with homogenic (only storage or only timing) patterns then the method is concluded as storage or timing. If there is heterogeneity across patterns that the method uses, i.e. storage and timing methods are mixed within the sender- and/or the receiver-side processes then it is concluded as a hybrid technique.

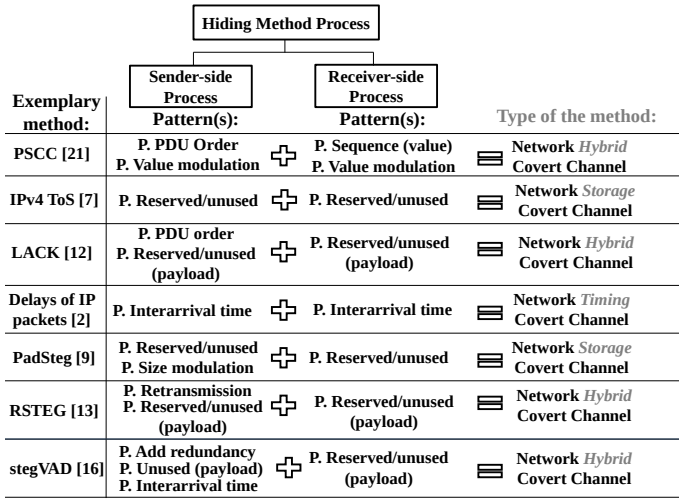


Figure 5: Classification of the exemplary network covert channels based on the assigned patterns.

To present how the proposed extended patterns' classification approach is functioning for some of the existing network steganography techniques, we have chosen seven different state-of-the-art network covert channels to demonstrate how they fit into our categorization (Fig. 5). For example, for a simple network covert channel which in order to conceal data utilizes Type of Service field from the IPv4 header [7], the sender- as well as receiver-side processes use the same pattern, i.e. *Reserved/Unused*, thus as both processes are assigned with the storage pattern then the method is concluded as storage. For the work related to modifying delays between the consecutive packets within the data stream [2] for both sender- and receiver-side processes the pattern *Inter-arrival time* is an obvious choice thus this technique is deemed as timing method. However, when we consider a more complex method like LACK (*Lost Audio Packets Steganography*) [12] then the situation is a bit different. As LACK operates by using intentionally delayed voice packets and replacing the original payload of these packets with secret data thus at the sender-side process two patterns must be selected – one storage (*Reserved/Unused*) and one timing (*PDU Order*), whereas when considering the receiver-side process the chosen pattern is only storage one (*Reserved/Unused*) – as at the covert receiver every incoming packet's payload, regardless of its order, is probed for the existence of the hash which will indicate presence of secret data. Therefore, the method is concluded to be hybrid. It is worth emphasizing that if we consider the original pattern approach (which as mentioned relied only on assigning pattern(s) based on how/where secret data is embedded) then LACK method would be only characterized by the storage *Reserved/Unused* pattern. This proves that the extended pattern approach proposed in this paper allows to characterize the data hiding methods in greater detail by including more information on inner workings of the information hiding technique.

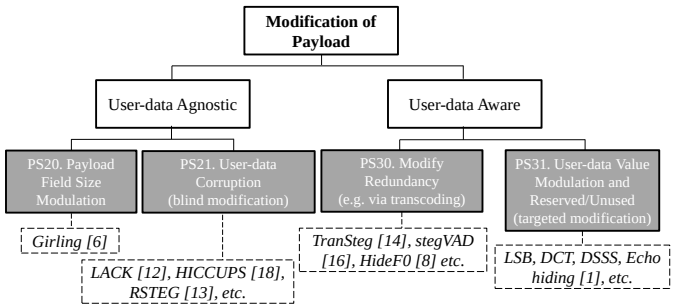


Figure 6: Classification of the network covert storage channels for the payload field and the corresponding patterns.

4.2 Introduction of Additional Patterns

As already mentioned, the current pattern-based categorization of [15, 22] makes a distinction between patterns applied to user-data (within the payload field) and protocol specific data (control information: headers, padding, etc.). In principle, all these patterns adhere to the storage group, i.e. modification of the certain "locations" of the carrier. However, in the original publications on hiding patterns, this distinction was made based on the idea of Fisk et al. [3] to separate structured (machine-readable) content from non-structured (human-readable) content, such as images. This means that in several cases similar rules apply to modify these fields (because structured data follows rules, e.g. protocol headers are built similarly to formal grammar) and to the data that they store. Obviously the most significant difference lays in the dissimilarities between the control information carried within protocol headers/padding and user-data transferred within the payload field. Thus, to fill this gap and by considering current research efforts in this area, we propose to extend the current taxonomy as shown in Fig. 6.

Network covert channels that modify the payload field and its content have been divided based on whether the characteristic of user-data is taken into account into: *(i)* user-data agnostic and *(ii)* user-data aware. In each of the two groups two patterns have been identified, which we describe in the same way as the patterns were originally described in [22] using a subset of the *Pattern Language Markup Language's* (PLML) attributes:

PS20. Payload Field Size Modulation
Illustration: This pattern uses a size of the payload field of a flow's PDUs/messages to encode the hidden message. This pattern is a variant (child) of the pattern *P1. Size Modulation* of [22] which has been already defined for the modification of the non-payload branch of storage methods (confirm Fig. 1).
References: PS1. Size Modulation
Context: Network Covert Channel Patterns → Covert Storage Channel Patterns → Modification of Payload → User-data Agnostic
Evidence:
1. Modulate the size of the data block field in Ethernet frames [6].
2. Any other method that modulates the size of the payload field in any network protocol.

PS21. User-data Corruption
Illustration: This pattern is related to the cases when steganographic methods do not take into account what kind of user-data is carried within a payload field and/or what its characteristic is (blind modification). It can be applied to single PDUs or to multiple PDUs (a flow). This typically happens if parts of (or the whole) user-data is replaced with secret bits and thus the user-data is corrupted/lost. This pattern is similar to the pattern *PDU Corruption* defined in the original pattern categorization of [22].
Context: Network Covert Channel Patterns → Covert Storage Channel Patterns → Modification of Payload → User-data Agnostic
Evidence:
1. Replace the user-generated data in the payload field with secret data in intentionally lost voice packets of the IP telephony call [12].
2. Replace the user-generated data in the payload field with secret data in retransmitted TCP segments [13].
3. Replace the user-generated data in the payload field with secret data in intentionally corrupted IEEE 802.11 frames [18].

PS30. Modify Redundancy
Illustration: This pattern is used when it is possible to exploit the redundancy of the user-data by means of transforming them in such a way that a free space for secret data is obtained (e.g. by means of transcoding). This pattern is a bit similar to the pattern *Add Redundancy* defined in [22] but can also decrease redundancy and is applied to payload instead of meta-data.
Context: Network Covert Channel Patterns → Covert Storage Channel Patterns → Modification of Payload → User-data Aware
Evidence:
1. Compress existing user-data in order to make a space for secret data [14].
2. Transform the VAD-enabled IP telephony voice stream into non-VAD one and fill the gaps using artificially generated RTP packets containing secret data [16].
3. Approximate the F0 parameter of the Speex codec which carries information about the pitch of the speech signal and use the "saved" space for secret data [8].

PS31. User-data Value Modulation and Reserved/Unused
Illustration: Characteristic features of user-data can be utilized to store secret information. This includes applying methods like LSB modification to speech samples or digital images carried within the payload field. Compared with previous patterns this is a targeted modification. This pattern is analogous to the combination of the patterns *Value Modulation* and *Reserved/Unused*, but applied to payload.
Context: Network Covert Channel Patterns → Covert Storage Channel Patterns → Modification of Payload → User-data Aware
Evidence:
1. Encode a stream of information by spreading the encoded data across as much of the frequency spectrum as feasible (e.g. DSSS) [1].
2. Embeds secret data into a carrier audio signal by introducing an echo (a.k.a. echo hiding) [1].
3. Replacing the least significant bit of e.g. each voice sample with secret data (LSB) [1].

As it is visible above, the identified patterns have mostly a number of examples in the state-of-the-art publications (Fig. 6). Every newly defined pattern corresponds to the patterns that have been already defined in the *non-payload* branch of the original classification.

Finally, the complete picture of the extended information hiding patterns classification is illustrated in Fig. 7 and the corresponding descriptions of all defined patterns which include also potential multi-packet/multi-flow characteristics of some data hiding methods are enclosed in Tab. 2.

4.3 Distributed Covert Channel Realization

In [22], authors defined three concepts which can be used to explain suitably some of the existing network covert channels' phenomena, i.e. pattern variation, pattern combination and pattern hopping.

The above-mentioned concepts are especially suitable and important when trying to depict, explain, and analyze the realization of *distributed* network covert channels. We define a *distributed covert channel* as a network covert channel that spreads secret data among multiple flows/protocols/hosts or uses multiple patterns within the same flow or PDU for the hidden data exchange. In contrast, the typical (undistributed) network covert channel is a storage or a timing channel that uses PDUs of a single flow/protocol with only one hiding pattern in order to embed secret data.

In Fig. 8 we have illustrated that these three pattern concepts practically exhaust possibilities for distributed network covert channel realization. While explaining these concepts we apply the terms of *spatial*, *temporal*, and *transform domains* which are "borrowed" from the digital media steganography research area [17] and which helps to described and define them better.

The first group i.e. *pattern combination* is related to the distribution of secret data in a *spatial domain*. This means that many patterns are utilized in parallel for the same hidden data carrier e.g. by modifying many of its sub-carriers or using several carriers at once. This includes the case when the hybrid data hiding methods are used (cf. Fig. 1) as well as the case of simultaneous utilization of multiple network covert channels at once. Consider an example of HTTP traffic (e.g. web browsing) where three separate network covert channels are used simultaneously: one is used for the IPv4 protocol, the next for the TCP protocol, and finally the third is applied to HTTP. Pattern combination applies also to the case when, e.g. three separate connections are used for hidden data purposes and in each connection a separate network hiding pattern is utilized at the same time (e.g. IPv4-based in the first connection, TCP-based in the second, and HTTP-based in the last one). Typically such an approach is used in order to increase the overall steganographic bandwidth.

The second group of distributed covert channels realization is *pattern hopping* which allows to spread secret data in the temporal domain (time). In a nutshell it means that different patterns' utilization varies over time and thus they are applied sequentially for various (sub-)carriers. Usually, such an approach helps to improve the stealthiness of the covert data exchange as in order to detect it more "locations" must be monitored by the warden. An example of pattern hopping is the tool *PHCCT*. PHCCT implements a so-called *protocol hopping covert channel* that distributes data over different

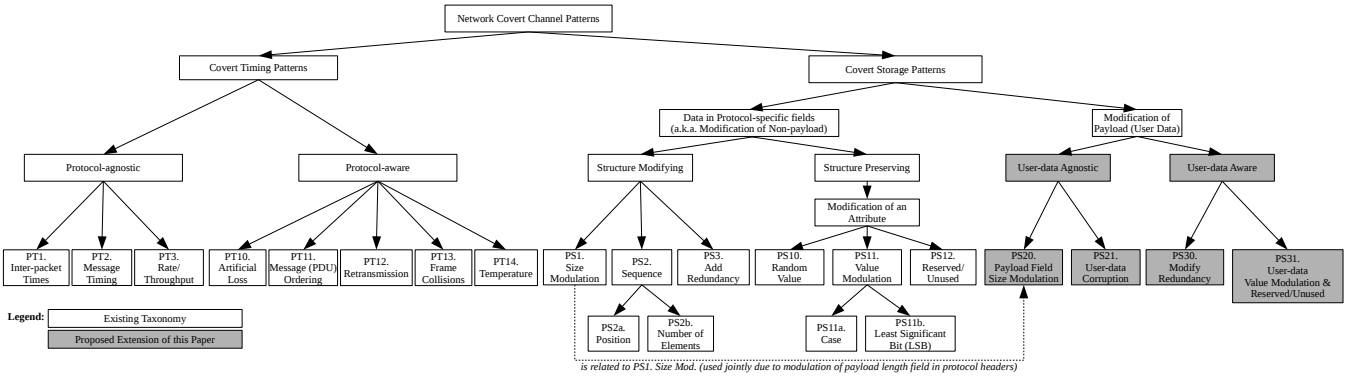


Figure 7: Classification of network covert channel patterns.

Table 2: Descriptions of hiding patterns in our improved and extended taxonomy.

Pattern Name	Pattern Description
PT1. Inter-packet Times	The covert channel alters timing intervals between network messages of a flow (interarrival times) to encode hidden data.
PT2. Message Timing	Hidden data is encoded in the timing of message sequences within a flow, e.g. acknowledging every n'th received message or sending commands m times.
PT3. Rate/Throughput	The covert channel sender alters the data rate of a flow from itself or a third party to the covert receiver.
PT10. Artificial Loss	The covert channel signals hidden information via artificial loss of a flow's transmitted messages, e.g. by frame-corruption or message drop.
PT11. Message Ordering	The covert channel encodes data using a synthetic message order in a flow.
PT12. Retransmission	A covert channel retransmits previously sent or received messages of a flow.
PT13. Frame Collisions	The sender causes artificial frame collisions to signal hidden information.
PT14. Temperature	The sender influences a third party node's hardware temperature using traffic of a flow. There must be a technique for the covert receive to measure the temperature (indirectly).
PS1. Size Modulation	The covert channel uses the size of flow metadata (e.g. PDU size or size of a header element) to encode hidden messages.
PS2. Sequence Modulation	The covert channel alters the sequence of flow metadata to encode hidden information. This pattern divides further into: P2.a. Position and P2.b. Number of Elements patterns.
PS3. Add Redundancy	The covert channel embeds redundant metadata (e.g. by adding an unused IP option) in which data is hidden into a flow. Note that in comparison to PS1, the data is hidden in the redundant data's presence, not in the size of an PDU or header element).
PS10. Random Value	The covert channel embeds hidden data into flow metadata that contains a (pseudo-)random value.
PS11. Value Modulation	The covert channel selects one of the n values that a flow's metadata element can contain to encode a hidden message. This pattern divides further into: P11.a. Case Pattern and P11.b. Least Significant Bit (LSB) patterns.
PS12. Reserved/Unused	The covert channel encodes hidden data into a flow's reserved or unused metadata elements.
PS20. Payload Field Size Modulation	The size of the payload in a flow is used to encode hidden information (this is a derivate of PS1 but for the payload since it involves the modification of a PDU's payload length field, i.e. PS1).
PS21. User-data Corruption	The covert channel performs a (blind) insertion of covert data into a flow's payload (similar PT10).
PS30. Modify Redundancy	The covert channel compresses a flow's payload and the resulting free space is used to hide data.
PS31. User-data Value Modulation and Reserved/Unused	The covert channel performs a modification of a flow's payload in a way that is not reflected by PS30 and that does not result in a significantly modified interpretation of the data, e.g. by modifying least significant bits of digital images or hiding data in unused/reserved payload bits.

network protocols [15]. To this end, PHCCT utilizes more than one pattern, namely *Add Redundancy* (embedded in HTTP) and *User-data Corruption* (embedded in FTP-Data).

Finally, the last group of techniques which allows to realize a distributed network covert channel is *pattern variation*. The original idea of pattern variation is that each of the defined patterns is considered in the certain context, i.e. the utilized hidden data carrier

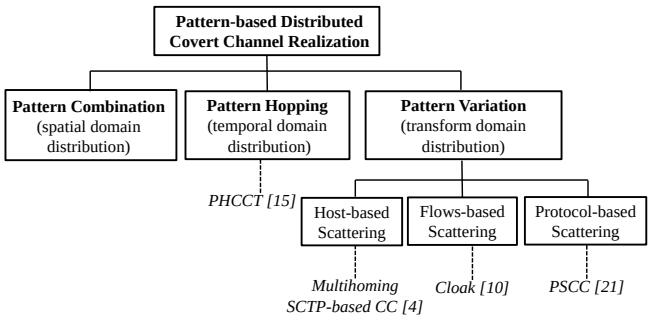


Figure 8: Classification of pattern-based distributed network covert channel realization.

(e.g. a network protocol). In our case, we extend this view and define pattern variation in different contexts. In particular, three contexts can be distinguished: *host-based scattering*, *flow-based scattering*, and *protocol-based scattering* which will be described in detail with examples below. In all cases of pattern variation, the *same* pattern is applied to *different* contexts, i.e. its essence does not change.

Host-based scattering requires the covert sender and/or the covert receiver to control more than one physical host or other networking devices. Parts of the secret data are hidden in the legitimate traffic sent from or directed towards different hosts using the same pattern. An example of this kind of distributed covert channel is the SCTP multi-homing-based method (i.e. the host's ability to be visible in the network through more than one IP address) [4]. In such a scenario, each IP address of the covert receiver can be used to represent a single bit of secret data (or a sequence of bits). Then, by modulating the way that packets are addressed and sent secret data can be transferred in a distributed manner.

Next, *Flow-based* scattering takes advantage of the capability to set up multiple flows between two hosts and using them to signal secret data bits in a distributed way while utilizing the same pattern. This can be realized, for example, by dividing secret data into fragments and using a certain information hiding pattern (or several) to send each fragment using one of the available flows. An idea of using many flows for a distributed covert channel is exemplified by the Cloak method [11], which is a timing data hiding technique that encodes secret data bits by uniquely distributing N packets over M TCP flows. Please note that while in the case of *pattern hopping* a utilization of multiple flows is possible as well, *flow-based scattering* serves under the umbrella of pattern variation, i.e. it is required to apply the *same pattern* to *different flows*, and pattern hopping must apply *different patterns*.

Finally, *Protocol-based* scattering applies a pattern to different communication protocols instead of hosts or flows. In contrast to flow-based scattering, it does not necessarily utilize flows of the same protocol but changes the actual protocol (which can generate multiple flows, too). This group is exemplified via *protocol switching covert channels* (PSCC) [21]. These channels assign hidden information to network protocols. For instance, one could link the HTTP protocol to the hidden value "0" and the DNS protocol to the hidden value "1". Then, by sending the packet sequence HTTP, DNS, DNS, HTTP, one would transfer the secret information "0110".

Obviously, there are other possibilities to create distributed network covert channels by developing mixed solutions so that it involves the parallel use of, e.g. pattern hopping and pattern variation or any other fusion of the concepts mentioned above.

5 CONCLUSIONS

We identified limitations of the existing pattern-based taxonomy, most importantly a lack of payload-based hiding patterns and a limited definition of distributed covert channels. For this reason, we extended the list of existing hiding patterns for network covert channels and their related taxonomy. We also extended the description of hybrid/distributed hiding methods and proposed an extension and improvement of the related concepts (especially pattern variation to handle multi-host, multi-flow and multi-protocol techniques).

We hope this work will help to derive new insights into existing and new data hiding techniques.

Future work will be devoted to analyzing relationships between patterns with respect to their joint occurrence in existing methods as well as we will investigate whether any new data hiding methods can be deuced based on the less obvious pattern mixes.

ACKNOWLEDGMENTS

Wojciech Mazurczyk and Krzysztof Cabaj are supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0254. The supported project is named CoCoDe (Covert Communication Detection).

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

REFERENCES

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu. 1996. Techniques for data hiding. *IBM Systems Journal* 35, 3.4 (1996), 313–336. https://doi.org/10.1147/sj.353.0313

[2] V. Berk, A. Giani, and G. Cybenko. 2005. *Detection of Covert Channel Encoding in Network Packet Delays.* Technical Report TR2005-536. Department of Computer Science, Dartmouth College. http://www.ists.dartmouth.edu/library/149.pdf http://www.ists.dartmouth.edu/library/149.pdf.

[3] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil. 2003. Eliminating steganography in Internet traffic with active wardens. In *Proc. Revised Papers from the 5th International Workshop on Information Hiding*. 18–35.

[4] Wojciech Fraczek, Wojciech Mazurczyk, and Krzysztof Szczypiorski. 2012. Hiding Information in a Stream Control Transmission Protocol. *Comput. Commun.* 35, 2 (Jan. 2012), 159–169. https://doi.org/10.1016/j.comcom.2011.08.009

[5] W. Fraczek, W. Mazurczyk, and K. Szczypiorski. 2012. Multilevel Steganography: Improving Hidden Communication in Networks. *Journal of Universal Computer Science* 18, 14 (jul 2012), 1967–1986.

[6] C. G. Girling. 1987. Covert Channels in LAN's. *IEEE Transactions on Software Engineering* 13, 2 (1987), 292–296.

[7] Theodore G. Handel and Maxwell T. Sandford. 1996. Hiding data in the OSI network model. In *Information Hiding*, Ross Anderson (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 23–38.

[8] Artur Janicki. 2016. Pitch-based Steganography for Speex Voice Codec. *Security and Communication Networks* 9, 15 (2016), 2923–2933. https://doi.org/10.1002/sec.1428

[9] B. Jankowski, W. Mazurczyk, and K. Szczypiorski. 2013. PadSteg: introducing inter-protocol steganography. *Telecommunication Systems* 52, 2 (01 Feb 2013), 1101–1111. https://doi.org/10.1007/s11235-011-9616-z

[10] X. Luo, E. W. W. Chan, and R. K. C. Chang. 2007. Cloak: A Ten-Fold Way for Reliable Covert Communications. In *Computer Security – ESORICS 2007*, Joachim Biskup and Javier López (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 283–298.

[11] Xiapu Luo, Edmond W. W. Chan, and Rocky K. C. Chang. 2007. Cloak: A Ten-Fold Way for Reliable Covert Communications. In *Computer Security – ESORICS*

2007, Joachim Biskup and Javier López (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 283–298.

[12] Wojciech Mazurczyk and Józef Lubacz. 2010. LACK—a VoIP steganographic method. *Telecommunication Systems* 45, 2 (01 Oct 2010), 153–163. https://doi.org/10.1007/s11235-009-9245-y

[13] W. Mazurczyk, M. Smolarczyk, and K. Szczypiorski. 2011. Retransmission steganography and its detection. *Soft Computing* 15, 3 (2011), 505–515. https://doi.org/10.1007/s00500-009-0530-1

[14] Wojciech Mazurczyk, PawełSzaga, and Krzysztof Szczypiorski. 2014. Using Transcoding for Hidden Communication in IP Telephony. *Multimedia Tools Appl.* 70, 3 (June 2014), 2139–2165. https://doi.org/10.1007/s11042-012-1224-8

[15] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, and K. Szczypiorski. 2016. *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. Wiley-IEEE.

[16] Sabine S. Schmidt, Wojciech Mazurczyk, Jörg Keller, and Luca Caviglione. 2017. A New Data-Hiding Approach for IP Telephony Applications with Silence Suppression. In *Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES '17)*. ACM, New York, NY, USA, Article 83, 6 pages. https://doi.org/10.1145/3098954.3106066

[17] Mansi S. Subhedar and Vijay H. Mankar. 2014. Current status and key issues in image steganography: A survey. *Computer Science Review* 13-14 (2014), 95 – 113. https://doi.org/10.1016/j.cosrev.2014.09.001

[18] Krzysztof Szczypiorski. 2012. A performance analysis of HICCUPS—a steganographic system for WLAN. *Telecommunication Systems* 49, 2 (01 Feb 2012), 255–259. https://doi.org/10.1007/s11235-010-9363-6

[19] S. Wendzel, W. Mazurczyk, and S. Zander. 2016. Unified Description for Network Information Hiding Methods. *Journal of Universal Computer Science* 22, 11 (nov 2016), 1456–1486.

[20] S. Wendzel and C. Palmer. 2015. Creativity in Mind: Evaluating and Maintaining Advances in Network Steganographic Research. *Journal of Universal Computer Science* 21, 12 (2015), 1684–1705.

[21] S. Wendzel and S. Zander. 2012. Detecting Protocol Switching Covert Channels. In *37th IEEE Conf. on Local Computer Networks*. 280–283.

[22] S. Wendzel, S. Zander, B. Fechner, and C. Herdin. 2015. Pattern-based Survey and Categorization of Network Covert Channel Techniques. *Computing Surveys (CSUR)* 47, 3 (2015).

[23] F. V. Yarochkin, S.-Y. Dai, C.-H. Lin, and Y. Huang. 2008. Towards Adaptive Covert Communication System. In *Proc. Pacific Rim International Symposium on Dependable Computing (PRDC)*. 153–159.

4.3 PAPER III

Cabaj Krzysztof, Żórawski Piotr, Nowakowski Piotr, Mazurczyk Wojciech: Efficient distributed network covert channels for Internet of things environments, in: Journal of Cybersecurity, vol. 6, nr 1, 2020, pp. 1-18, DOI:10.1093/cybsec/tyaa018

This is extended version of:

Cabaj Krzysztof, Mazurczyk Wojciech, Nowakowski Piotr, Piotr Żórawski: Fine-tuning of Distributed Network Covert Channels Parameters and Their Impact on Undetectability, in: Proceedings of the 14th International Conference on Availability, Reliability and Security - Ares 2019, ICPS, 2019, ISBN 978-1-4503-7164-3, pp. 1-8, DOI:10.1145/3339252.3341489

JOURNAL OF CYBERSECURITY

Journal of Cybersecurity, 2020, 1–18
doi: 10.1093/cybsec/tyaa018
Research Paper

Efficient distributed network covert channels for Internet of things environments†

Krzysztof Cabaj *, Piotr Żórawski , Piotr Nowakowski , Maciej Purski, and Wojciech Mazurczyk

Warsaw University of Technology, plac Politechniki 1, 00-661 Warsaw, Poland

*Correspondence address. Krzysztof Cabaj, Institute of Computer Science, Warsaw University of Technology, Nowowiejska 15/19, Warsaw 00-665, Poland. E-mail: kcabaj@ii.pw.edu.pl.

Received 26 April 2020; revised 29 July 2020; accepted 14 October 2020

Abstract

Each day more and more Internet of Things (IoT) devices are being connected to the Internet. In general, their applications are diverse but from the security perspective, it is evident that they are increasingly targeted by cybercriminals and used for nefarious purposes. Network covert channels form a subgroup of the information-hiding research area where secrets are sent over communication networks embedded within the network traffic. Such techniques can be used, among others, by malware developers to enable confidential data exfiltration or stealth communications. Recently, distributed network covert channels have raised the attention of security professionals as they allow the cloaking of secret transmission by spreading the covert bits among many different types of data-hiding techniques. However, although there are many works dealing with IoT security, little effort so far has been devoted in determining how effective the covert channels threat can be in the IoT henvironments. That is why, in this article, we present an extensive analysis on how distributed network covert channels that utilize network traffic from IoT devices can be used to perform efficient secret communication. More importantly, we do not focus on developing novel data-hiding techniques but, instead, considering the nature of IoT traffic, we investigate how to combine existing covert channels so the resulting data transfer is less visible. Moreover, as another contribution of our work, we prepare and share with the community the network traffic dataset that can be used to develop effective countermeasures against such threats.

Key words: covert channels; network security; Internet of Things; information hiding

Introduction

Nowadays, many everyday appliances are empowered with networking functionality in order to allow monitoring of devices and processes, remote management, etc. The phenomenon of the Internet of Things (IoT) is growing rapidly, and the number of connected devices is envisioned to increase from 13.4 billion in 2015 to 38.5 billion in 2020.[1] The main benefits of the IoT include automation of processes, monitoring, increased intuitiveness of the environment by offering a mixture of sensing, communication and computing services, as well as access to these services on demand [2]. Considering the above, currently the most popular IoT applications include smart home systems, wearables, smart grids, smart cities, connected cars and industrial IoT.

From the protocols perspective, the IoT has never reached a widely accepted, common standard despite previous efforts, e.g. from IEEE. Thus, the IoT is considered highly heterogeneous and this is enabled due to the variety of different technologies. However,

†This is the extended version of the paper entitled 'Fine-tuning of Distributed Network Covert Channels Parameters and Their Impact on Undetectability' [1] presented at the CUING Workshop held in conjunction with the 14th International Conference on Availability, Reliability and Security (Canterbury, UK, 2019).

1 https://www.juniperresearch.com/press/press-releases/iot-connected-devices-to-triple-to-38-bn-by-2020

it must be noted that the rapid development of this concept does not go hand in hand with the proper adoption of security measures, which might put users and their privacy in danger [3–6]. The lack of standardization mentioned above, as well as limited resources when it comes to the processing power and battery life, is (among others) to blame and thus the current traditional security solutions are not suitable for IoT systems. Moreover, due to the small volume of traffic produced by these kinds of devices when compared to typical users' equipments such as desktops, laptops, or mobile phones, it is commonly believed that this traffic may be not suitable for covert transmission. For example, during the experiments conducted, described in this article, we simulated the network traffic of a thermometer currently available on the market. It sends about 400 bytes and receives less than 3 kilobytes of data less often than once per second, which is a relatively small volume. However, if several such IoT devices are utilized to orchestrate a 'distributed' network covert channel, then the resulting throughput becomes reasonable.

During the last decade, various information-hiding techniques have been used by cybercriminals for nefarious purposes [7]. Among potential methods, the so-called 'network covert channels' [8–10] are increasingly used by attackers to cloak their malicious activities, e.g. to invisibly communicate with the Command and Control servers, exfiltrate confidential and sensitive data, or download further malware modules [11]. Note that this trend is expected to continue in the future and we will witness an increase in sophisticated covert channel techniques utilized with malicious intentions [7], which will be a real challenge for the digital forensics and security communities [12].

Moreover, it is foreseen that even more complex forms of the network covert channels will be utilized affecting novel services [13, 14], types of traffic [15–17], its characteristic features [18, 19], or user behaviour [20] in response to the increasingly efficient and effective information-hiding countermeasures. Therefore, the aim of the research in information hiding should be focused not only to improve the defence systems but also to analyse other potential 'constructions' of data-hiding techniques, especially those that provide improved undetectability and/or covert channel bandwidth. One such example is information-hiding-based botnets [21–23], which offer covert orchestration of the bots and confidential data exfiltration.

Another example of sophisticated data-hiding schemes is 'Distributed Network Covert Channels' (DNCCs) [24, 25]. DNCCs are defined as network covert channels that spread the secret data among many flows/protocols/hosts or use multiple data-hiding methods within the same flow or within Protocol Data Units (PDUs) in order to provide hidden data exchange. In contrast, the typical (undistributed) network covert channels are storage or timing channels, in order to embed secret data utilize PDUs of a single flow or protocol.

DNCCs are nowadays receiving increased attention among the security community as they provide the following benefits for the attacker: (i) they can further improve the overall stealth and bandwidth of the hidden communication as they can transmit smaller parts of secret data using various covert channels and (ii) they can be used to bypass existing defensive solutions, as due to the distributed nature, the modifications applied using a single data-hiding technique can be limited, thus staying under the radar.

Note, however, that so far in the literature no systematic analyses of the DNCCs both from the properties perspective and from the detection angle have been presented. Our previous works contributed to the initial investigation of the distributed network covert channels. In [24], we extended a network information-hiding patterns concept so it is able to incorporate DNCCs and, additionally, we introduced their novel taxonomy. Moreover, in [25], we have evaluated whether data-mining techniques can be suitable for the detection of distributed network covert channels. We have considered the typical state-of-the-art network covert channel, which is based on the IPv4 Time to live (TTL) field value modulation, and additionally, the secret data are distributed among several data flows. The initial results that we obtained were promising, and thus, we concluded that such an approach should be further investigated. Finally, in [1], we further investigated the DNCC concept by performing an initial analysis of the DNCCs capabilities, and in particular, we provided more insights on how DNCCs should be constructed and used in order to achieve increased undetectability.

This work is an extended version of the conference paper [1]. Note, that in this research, we significantly expanded the experimental evaluation. First of all, during the experiments we utilize simulated IoT traffic, in contrast to the real web activity used previously. This kind of traffic possesses completely different challenges as its characteristics impact significantly the way in which steganographic communication must be performed. Secondly, we use an enhanced DNCC scheme where not only three distinct steganographic methods are used, but also multiple overt traffic senders (to better model the IoT environment scenario). Thirdly, in this article, we introduce a new method for tuning the parameters of each network covert channel forming the DNCC. To sum up, the main differences and contributions of this article are to:

- explore the DNCC in the IoT environments—this impacts the characteristics of the analysed communication scenario as well as the nature of the network traffic that can be utilized for data-hiding purposes;
- we perform a thorough experimental evaluation of the DNCC created and show how such covert channels should be created in order to optimize their performance when it comes to their bandwidth and undetectability; and
- we prepare and share with the community the network traffic dataset that can be used to develop effective countermeasures against DNCCs, e.g. based on machine learning techniques.

The rest of this article is structured as follows. In the next section, we review the most relevant works related to the IoT security and data hiding in this environment. In 'Distributed network covert channels' section, a detailed analysis of the distributed network covert channels is presented. Then, the experimental methodology utilized, as well as the testbed, is presented in 'Experimental methodology and testbed' section followed by the results obtained in 'Experiments, results and discussion' section. Finally, 'Conclusions' section concludes our research and outlines potential future work.

Related work

In this section, we first review works on IoT security and then we focus on describing relevant research related to IoT-based covert channels.

IoT security

The security of IoT systems is an emergent and a rather well-studied topic, see, e.g. [3–6] for surveys. In the remainder of this subsection, we briefly summarize the main aspects of IoT security.

Many IoT devices have their network interfaces exposed to the public and due to their limited resources (like processing power or battery), they are lacking proper security measures. As a result, they can be easily exploited by attackers. This may potentially result not only in serious security (i.e. the attacker is able to access user data), but also safety threats for a user. Consider, for example, security issues related to medical devices. In 2016, Muddy Water's Research[2] revealed in their report that some implantable cardiac devices possess serious security flaws. When exploited, they allowed an attacker to access the devices and then deplete the battery or administer incorrect pacing or shocks.

As IoT devices are sometimes called the 'internet's least powerful hosts', it must be noted that they can be compromised and used to become a part of the botnet and then take part in Distributed Denial of Service attacks on other devices and services. One of the most recent and large-scale attacks of this kind was launched by the Mirai botnet, which reportedly infected up to 600 K devices to successfully take down many online services [26].

Finally, Hron [27] from Avast presented how smart homes that deploy popular IoT protocol—Message Queue Telemetry Transport (MQTT) can be exploited. An attacker can connect to an open and unprotected MQTT broker and receive all the messages of all the registered topics. This means that the attacker is able to monitor the status of window sensors, locks, heating/cooling systems, usage of light switches, etc.

Covert channels in the IoT

The problem of covert channels in the IoT as a security and privacy threat has been recently recognized, and it is starting to raise attention in the security community [28, 29], although this type of research is still not significantly explored. The most relevant works from the perspective of this article are described below.

Note that currently published papers mostly deploy data-hiding techniques in some IoT protocols, such as several storage covert channels in the Extensible Messaging and Presence Protocol [30], two-storage and one timing covert channels in the Building Automation and Control Networking Protocol [31], six-storage and two timing covert channels in the Constrained Application Protocol [32].

Moreover, Wendzel *et al.* [33] have shown that one can hide data in a cyber-physical system (e.g. smart building) by slightly modifying some of its components, such as sensors, controllers, actuators, as well as by storing secret data in unused registers.

Recently, in [34], a comprehensive analysis of the MQTT protocol has been performed from the information-hiding perspective. In more detail, authors characterized seven direct and six indirect covert channels applicable for MQTT-based IoT environments, and for the selected data-hiding methods, their experimental evaluation has been presented.

Next, in [35], authors introduced three different sensor-based covert channels that provide a trade-off between the achievable covert channel bandwidth and undetectability. They present covert channels that require read- and write-access for sensor registers as well as a covert channel that transfers data by just triggering sensor readings so that the malicious behaviour cannot be distinguished from typical, normal sensor usage.

In [36], various kinds of covert timing channels were analysed to investigate their feasibility in the IoT environments. These data-hiding techniques included packet-reordering-based, rate-switching-based, packet-loss-based, retransmission-based and scheduling-based covert timing channels.

Moskowitz *et al.* [37] proposed a method for covert communication that utilizes transmission timing to obscure symbols. The authors also showed that IoT side channels are susceptible to network covert channels and that it is possible to create a data-in-motion data-hiding technique without network protocol modifications.

Finally, in [38], Herzberg and Kfir introduced a provably-covert channel for Cyber Physical Systems, which relies on a corrupt actuator that is located in one zone and is able to send secrets to a sensor in a different zone, breaking the isolation. The same authors extended their work in [39] by exploring data-hiding possibilities for indirect covert communication between a sensor and actuator via a benign threshold-based controller. The covert information was encoded within the output noise of the sensor in an indistinguishable manner when compared to that of a benign sensor.

As already mentioned, so far in the literature, no comprehensive research related to the distributed network covert channels has been published. Our previous works presented in [24, 25] and [1] provided some initial insights into general classification of DNCCs, their capabilities and potential detection opportunities. However, in this article, we perform a thorough analysis of the distributed network covert channels in IoT environments and we explore how they should be constructed to be the most effective. Moreover, based on our prototype implementation, we created and would like to share with the scientific community a dataset that will help to further develop effective countermeasures.

Distributed network covert channels

In general, network covert channels utilize the network traffic as a hidden data carrier that is modified in order to conceal secret data. Typically, two main groups of methods are distinguished: (i) those that modulate the 'timing' behaviour of network traffic and (ii) those that modulate 'storage' values of the network traffic. For instance, timing channels can modify the timing between network packets to encode secret data, whereas storage channels can modify unused header bits of network packets (among several other methods). Such types of data-hiding techniques can then be utilized to form distributed network covert channels. As already hinted, DNCCs are covert channels that spread the secret data among many flows/protocols/hosts or use multiple data-hiding methods within the same flow or within PDUs in order to provide hidden data exchange.

In this section, we first present the classification of the DNCCs followed by their main communication scenarios and performance metrics.

Distributed network covert channel classification

In [40], the authors introduced a classification of network-hiding techniques into the so-called 'information-hiding patterns', i.e. abstract descriptions of how data can be concealed in network transmissions. Each pattern presents one core idea of how secret data can be represented through network traffic. Note that the originally proposed classification has been extended a couple of times, with the latest extension being the one of Mazurczyk *et al.* [24]. Note that in [40], the authors also defined three concepts which can be used to suitably explain some of the existing network covert channels'

2 http://d.muddywatersresearch.com/content/uploads/2016/08/MW_STJ_08252016_2.pdf

phenomena, i.e. pattern variation, pattern combination and pattern hopping. The above-mentioned concepts are especially suitable and important when trying to depict, explain and analyse the realization of 'distributed' network covert channels.

In [24] as well as in Fig. 1, these three-pattern concepts have been illustrated, which practically exhaust possibilities for distributed network covert channel realization. Below, we explain these concepts in detail by applying the terms: 'spatial', 'temporal' and 'transform domains' which are 'borrowed' from the digital media steganography research area [41] and which help to describe and define them better.

The first group, i.e. 'pattern combination', is related to the distribution of secret data in a 'spatial domain' [see Fig. 2(a)]. This means that many patterns are utilized in parallel for the same hidden data carrier, e.g. by modifying many of its subcarriers or using several carriers at once. This includes the case when hybrid data-hiding methods are used, as well as the case of simultaneous utilization of multiple network covert channels. Consider an example of HTTP traffic (e.g. web browsing) where three separate network covert channels are used simultaneously: one is used for the IPv4 protocol, the next for the TCP protocol and finally the third is applied to HTTP. Pattern combination also applies to the case when, e.g. three separate connections are used for hidden data purposes and in each connection, a separate network-hiding pattern is utilized at the same time (e.g. IPv4-based in the first connection, TCP-based in the second and HTTP-based in the last one). Typically, such an approach is used in order to increase the overall covert channel bandwidth.

The second group of distributed covert channels realization is 'pattern hopping' which allows the spreading of secret data in the temporal domain (time) [see Fig. 2(b)]. In a nutshell, it means that different patterns' utilization varies over time and thus, they are applied sequentially for various (sub)carriers. Usually, such an approach helps to improve the stealth of the covert data exchange as in order to detect it, more 'locations' must be monitored by the warden. An example of pattern hopping is realized within the *PHCCT* tool [7]. PHCCT tool implements the so-called 'protocol hopping covert channel' that distributes data over different network protocols. To this end, PHCCT utilizes more than one pattern, i.e. one uses the HTTP and the other FTP data traffic.

Finally, the last group of techniques which allows the realization of a distributed network covert channel is 'pattern variation' (see Fig. 3). The original idea of pattern variation is that each of the defined patterns is considered in a certain context, i.e. the utilized hidden-data carrier (e.g. a network protocol). In our case, we extend this view and define pattern variation in different contexts. In particular, three contexts can be distinguished: 'host-based scattering', 'flow-based scattering' and 'protocol-based scattering', which will be described in detail with examples below. In all cases of pattern variation, the 'same' pattern is applied to 'different' contexts, i.e. its essence does not change.

'Host-based' scattering requires the covert sender (CS) and/or the covert receiver (CR) to control more than one physical host or other networking devices [see Fig. 3(a)]. Parts of the secret data are hidden in the legitimate traffic sent from or directed towards different hosts using the same pattern. An example of this kind of distributed covert channel is the Stream Control Transmission Protocol (SCTP) multi-homing-based method (i.e. the host's ability to be visible in the network through more than one IP address) [42]. In such a scenario, each IP address of the CR can be used to represent a single bit of secret data (or a sequence of bits). Then, by modulating the way that packets are addressed and sent secret data can be transferred in a distributed manner.

'Flow-based' scattering takes advantage of the capability to set up multiple flows between two hosts and using them to signal secret data bits in a distributed way while utilizing the same pattern [see Fig. 3(b)]. This can be realized, for example, by dividing secret data into fragments and using a certain information-hiding pattern (or several) to send each fragment using one of the available flows. An idea of using many flows for a distributed covert channel is exemplified by the Cloak method [43], which is a timing data-hiding technique that encodes secret data bits by uniquely distributing N packets over M TCP flows.

Note that while in the case of 'pattern hopping', a utilization of multiple flows is possible as well, 'flow-based scattering' serves under the umbrella of pattern variation, i.e. it is required to apply the 'same pattern' to 'different flows', and pattern hopping must apply 'different patterns'.

Finally, 'protocol-based' scattering applies a pattern to different communication protocols instead of hosts or flows [see Fig. 3(c)]. In contrast to flow-based scattering, it does not necessarily utilize flows of the same protocol but changes the actual protocol (which can generate multiple flows, too). This group is exemplified via 'protocol switching covert channels' [44]. These channels assign hidden information to network protocols. For instance, one could link the HTTP protocol to the hidden value '0' and the DNS protocol to the hidden value '1'. Then, by sending the packet sequence HTTP, DNS, DNS, HTTP, one would transfer the secret information '0110'.

Obviously, there are other possibilities to create distributed network covert channels by developing mixed solutions so that they involve the parallel use of, e.g. pattern hopping and pattern variation or any other fusion of the concepts mentioned above.

DNCC communication scenarios

Distributed network covert channels can be utilized to achieve higher covert channel bandwidth or to send covert messages in a

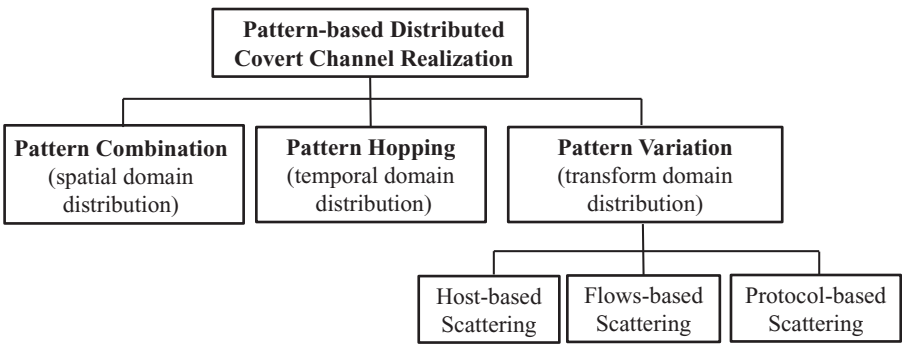


Figure 1: Classification of pattern-based distributed network covert channel realization [24].

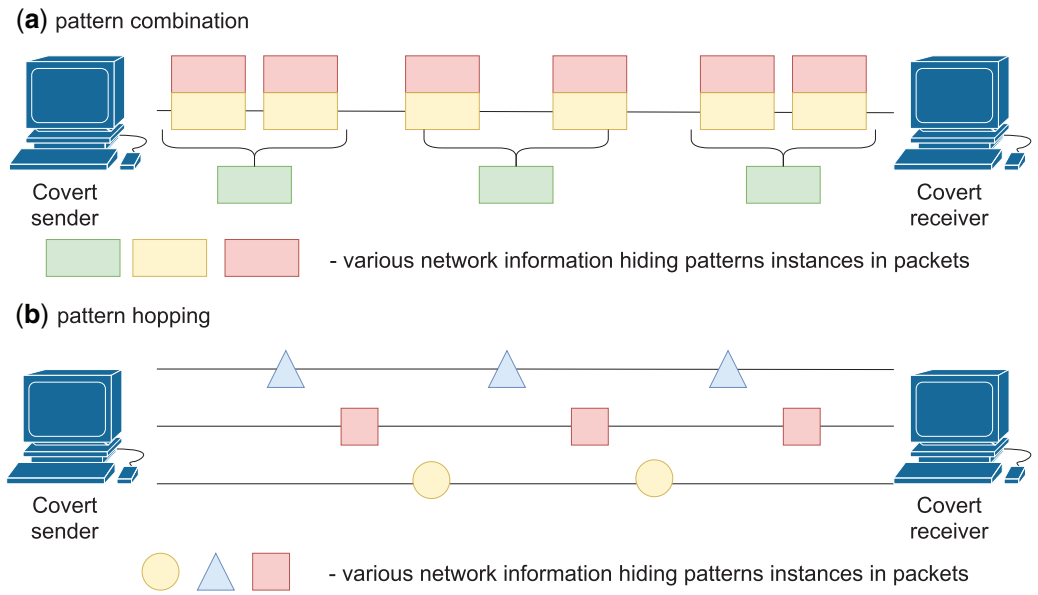


Figure 2: Illustration of the (**a**) pattern combination and (**b**) pattern hopping concepts.

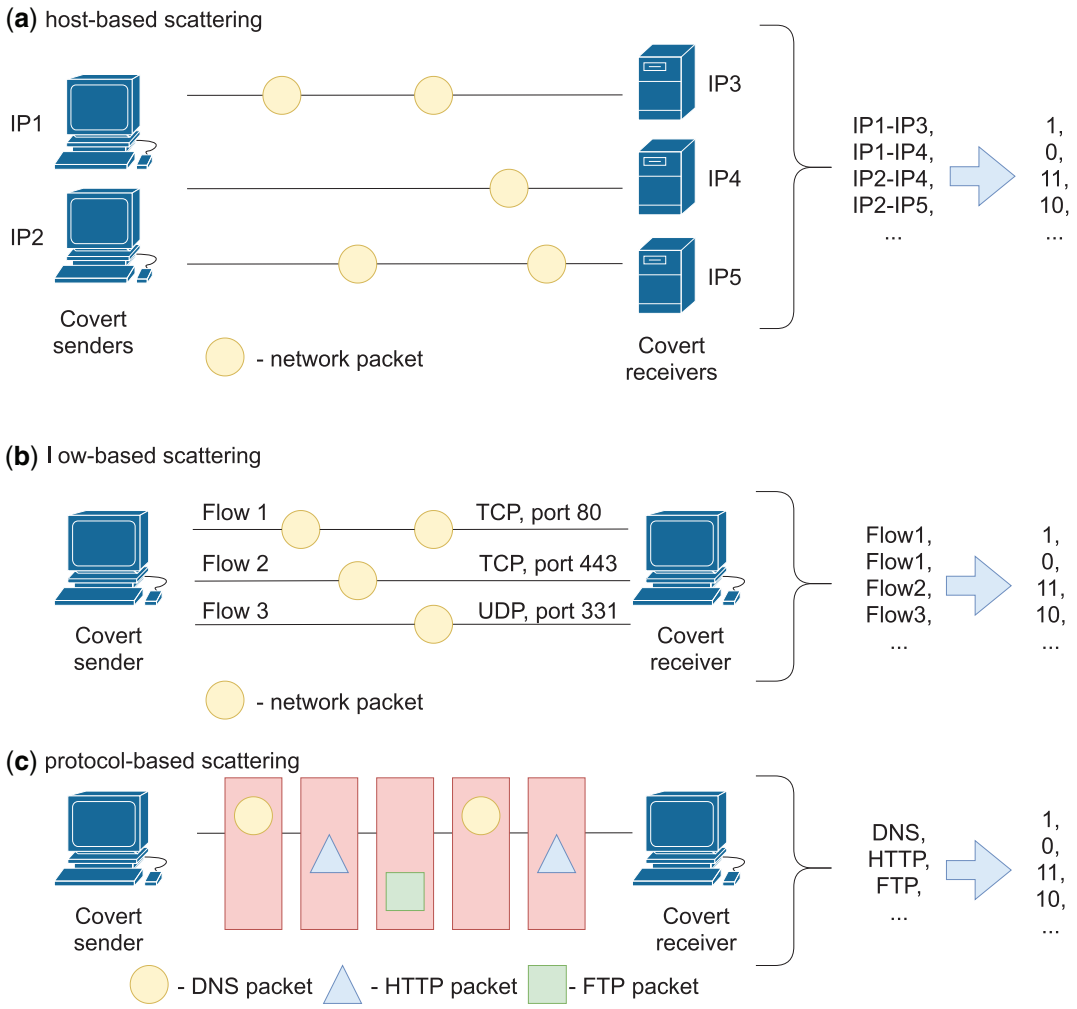


Figure 3: Illustration of the pattern variation techniques (**a**) host-based scattering, (**b**) flow-based scattering and (**c**) protocol-based scattering.

stealthier manner. In this case, the covert transmission of a single secret message not only utilizes various data-hiding methods and multiple connections, but also various senders' and/or recipients' machines. Note that in principle, the overt traffic sender and the overt receiver have no knowledge that their traffic is utilized for the hidden data exchange purposes.

Considering the above, we can define three distinct configurations for the DNCC referenced later in the text as follows: (i) one-to-many (1: N); (ii) many-to-one (M: 1); and (iii) many-to-many (M: N). In the first case, the traffic utilized for DNCC is coming from one sender (a client) and it is destined to a number of recipients (servers). In the second case, the traffic generated by various senders is destined towards one recipient (a server). Finally, in the third setup, the number of senders and receivers varies (but both numbers are higher than one). All three configurations are presented in Fig. 4. Due to properties of the IoT systems, during the experiments conducted and described in this article, we only analyse the many-to-one configuration scenario. In such a configuration, we utilize many IoT sensors which transmit their measurements to a single central server.

DNCC performance metrics

Generally, every network covert channel can be described by the following set of characteristics: its bandwidth, undetectability and robustness [7]. The term 'bandwidth' refers to the amount of secret data that can be sent per time unit when using a particular data-hiding technique. 'Undetectability' is defined as the inability to detect secret data embedded within a certain carrier. The most common approach to detect the presence of hidden data is to analyse the statistical properties of the captured traffic and compare them with values typical for that carrier. The final characteristic is 'robustness', which is defined as the amount of alteration (intentional or not) that the hidden data carrier can withstand without destroying the embedded secret data. Obviously, an ideal covert channel should be as robust and as difficult to detect as possible while offering the highest bandwidth. However, it must be noted that there is always a fundamental trade-off necessary among these three measures. Additionally, it is also useful to measure the data-hiding technique 'cost'. This is a characteristic that belongs to the sphere of carrier fidelity and has a direct impact on undetectability. It describes the degradation or distortion of the carrier caused by the application of the data-hiding method. For example, in the case of VoIP covert channels, this cost can be expressed as a measure of the conversation quality degradation induced by applying a particular technique for hiding information.

From this perspective, it is visible that the DNCC consists of utilization of many data-hiding techniques, which creates many network covert channels that can be used simultaneously (or interleaved as in case of pattern hopping). Therefore, for each of the covert channels forming DNCC, it is possible to use the above four performance metrics to characterize them. However, although they are important when describing DNCC, it must be noted that:

- if we denote overall covert channel bandwidth of DNCC as B_O, then the resulting data rate of each of the covert channels $(B_1, B_2, \ldots, B_i)$ forming DNCC contributes to B_O as follows: $B_O = B_1 + B_2 + \cdots + B_i$. This means that the higher the number of utilized NCCs forming DNCC, the higher its overall bandwidth;
- from the DNCC undetectability perspective, two things must be noted. On one hand, if a certain amount of secret data are to be transferred covertly, then dividing them into smaller portions and sending them using many separate covert channels may be beneficial from the undetectability perspective. In this case, the artefacts created due to network covert channels utilization may be less 'visible' when compared with the scenario where only a single data-hiding technique is used heavily. On the other hand, the DNCC is as detectable as the weakest data-hiding technique

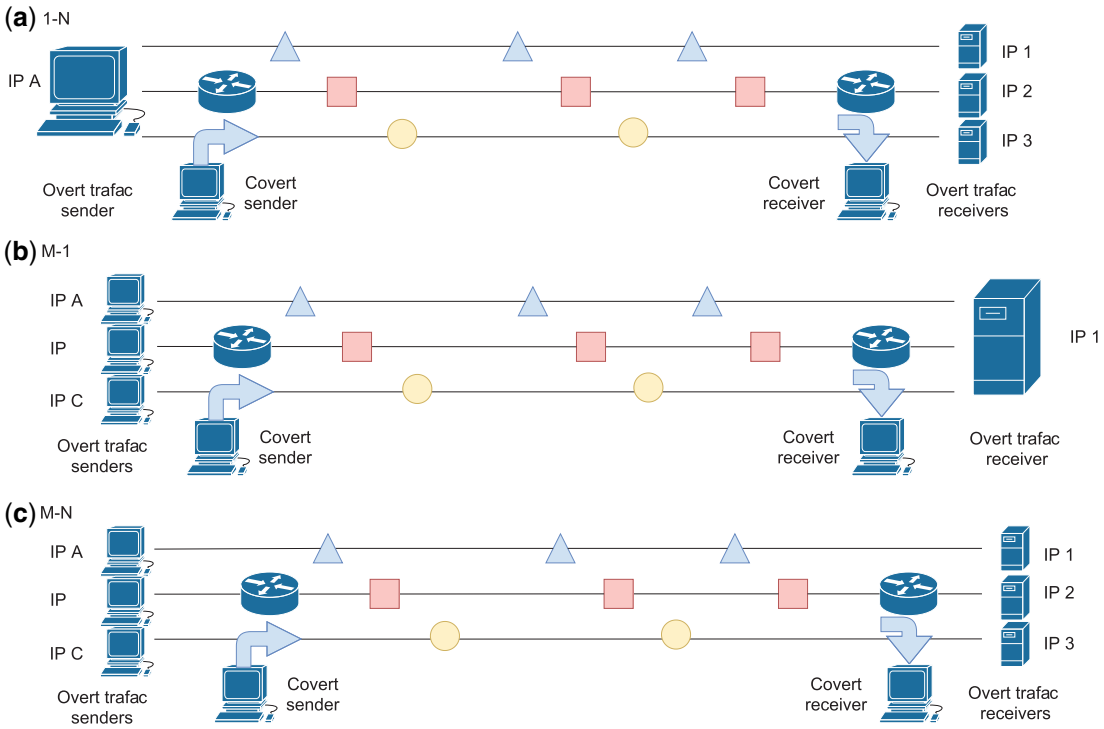


Figure 4: Comparison of the enhanced DNCC scenario: (**a**) one-to-many (1: N), (**b**) many-to-one (M : 1) and (**c**) many-to-many (M: N).

it is utilizing which means that if one of them is easy to detect, then the overall DNCC undetectability is jeopardized;
- robustness—the DNCC would be as robust as the NCC which has the weakest robustness. Thus, when the (intentional or not) modifications to the hidden data carrier make it impossible to extract secret data via even a single information-hiding method which is a part of the DNCC, then the robustness of the whole scheme (and thus part of the secret) is endangered, especially, if no error correction codes are used;
- in the case of data-hiding methods' cost, the number of affected subcarriers is obviously related to the number of the data-hiding techniques utilized to form DNCC. As already hinted when discussing undetectability of the DNCC, when we spread parts of secret data across many subcarriers, then their modification can be potentially less spottable when compared to the case when only a single network covert channel is used. Therefore, the overall cost of the DNCC is also amalgamate of all costs related to every data-hiding technique used to create DNCC. However, it is worth noting that as these costs are associated with various (often disjointed) subcarriers, then they cannot be just easily combined (summed) as they pertain to 'different dimensions' of the carrier.

As seen above, the 'classic' performance metrics for typical network covert channels are able to cover some aspects of DNCCs (even if they need to be slightly redefined, i.e. to, e.g. include overall covert channel bandwidth or cost rather than focusing on the single data-hiding technique). However, it is our opinion that more metrics must be defined in order to have a complete picture of DNCCs.

That is why we now provide the definitions and description of the additional performance metrics and mechanisms that allow the characterization of the DNCC in a more accurate manner. Thus, apart from the above-mentioned performance metrics, the additional ones are required in order to fully characterize DNCCs:

- 'synchronization' between covert channels within the set of data-hiding techniques that form the DNCC. This is required in order to ensure that each chosen network covert channel functions in a way which allows for the reliable secret data communication so the fragments of hidden data are not corrupted or lost and they can be extracted in the correct format and ordered to, in the end, form a complete secret message at the CR and
- 'secret data-scattering strategy'—the size of the fragments of secret data at the CS side should be adjusted in order to fit network covert channels characteristics especially from the bandwidth perspective. If this is not taken into account, then this may lead to the situation when the hidden data are received out of order by the CR, with many fragments not fitting each other and as a result, this can inflict significant delays (caused mostly by different data rates of the covert channels utilized).

Moreover, in order to ensure that the above-mentioned mechanisms are functioning correctly, it may be necessary to enable some kind of more formal exchange of control information between the CS and the CR. This may be achieved by enhancing DNCC with μprotocols [45], which are typically used with the cost of lowering usable available covert channel bandwidth to enable fundamental features such as reliability, dynamic routing, proxy capabilities, simultaneous connections, or session management for network covert channel features. For DNCC, the most important features that μprotocols can provide are reliability, establishing, controlling and management over simultaneous covert channel connections. It must be also noted that it is crucial to construct μprotocol in such a way that it does not make the DNCC more prone to detection [46].

The metrics mentioned above are then incorporated within the DNCC creation process and they form the following requirements:

> **Requirement 1:** When the DNCC is formed, the DNCC type should be first established between the CS and the CR and the network covert channel techniques it would involve. Without this, the CS can embed secret data using some arbitrary data-hiding techniques, but then it would be difficult for the CR to detect and extract the hidden data.
>
> **Requirement 2:** In the next step, the CS and the CR must also exchange information about the specific details related to the overt transmission that they will be using to enable network covert channels. This, depending on the DNCC type, may include information on flows, hosts, protocols, etc., that are essential in the DNCC-forming process.
>
> **Requirement 3:** Another vital functionality that the CS and the CR need to agree on is a synchronization for the various network covert channels utilized as it allows them to establish the 'sensitivity' of the hidden communication parties so that the CS is able to correctly interpret the hidden data bits transmitted in each network covert channel.
>
> **Requirement 4:** Finally, it is essential that the efficient secret data fragmentation/defragmentation mechanism is used, as without it, it may be difficult to restore the parts of secret data received into the complete secret message.

Experimental methodology and testbed

During the experiments, we analysed whether the utilization of DNCC techniques used in the IoT environment could result in a reasonable covert channel data rate. To verify this hypothesis, we developed and deployed selected network covert channels in the Software-Defined Networking (SDN) environment and experimented with various DNCC configurations to find the best performing ones. In the remainder of this section, we detail the testbed used and implemented network covert channels, as well as the network traffic dataset created.

All experiments presented in the details in the remainder of this section were performed using simulated IoT traffic. For simulation purposes, we used custom Python scripts, but the traffic characteristics are based on a real product available on the market (more details are presented in Subsection 'Created dataset'). However, the scenario evaluated during our experimental evaluation could easily be used in a real environment. Our initial research indicated that the home-based IoT scenario could be the first choice for attackers. It must be noted that it is often the case that home devices are configured without proper security and sometimes such devices function for years with default configuration (factory default). The user simply buys a device, takes it from the box and powers it up. In effect, due to lack of monitoring of its status its infection could remain undetected for a long time. Moreover, an analysis performed by security experts revealed that many vulnerabilities in home routers exist. This makes it easier for the attacker to exploit such a device. To sum up, for the DNCC presented in this article, the most important issue is that the home device in most cases is the only a single point via which all traffic goes to/from the Internet. Its infection gives access to all the network traffic transmitted from sensors to the central server and enables usage of DNCC. Moreover, we cannot forget how valuable data can be captured from IoT devices installed in the home environment, for example, those related to user presence, private information. The lack of proper security measures and potential

sensitive data suggests better trade-off for the attacker than, e.g. attacking smart grids, smart cities or the industrial IoT. The latter systems are deployed by organizations; thus, in contrast to the private home-based IoT, they typically possess better security configuration and mechanisms and in most cases 24/7 monitoring by a dedicated security operations centre (SOC).

Threat model

In this article, we assume the Man-in-the-Middle covert communication scenario which is illustrated in Fig. 5. The covert traffic sender and the covert traffic receiver utilize existing connections coming from benign devices within the communication network in order to enable secret data transfer. For the IoT case, it means that IoT devices (e.g. sensors) within the IoT LAN periodically transmit their data that are then sent to the arbitrary server. This sever is responsible for collecting these measurements and it is accessible via the Internet.

Note, that in this scenario it is assumed that the CS and the CR are both able to capture the whole traffic exchanged between the IoT LAN and the overt traffic receiver. In this case, covert communication parties are able to modify the passing traffic using agreed data-hiding techniques which form a DNCC.

Testbed

Our experiment scenario consists of two LANs that are separated from each other (Fig. 6). In the production installation, these networks will be connected via a WAN or the Internet—the first one consists of IoT devices and the second one contains a central server. Due to fact that our research mainly focuses on DNCC and not on detailed analysis of the particular data-hiding method used for its creation, all experiments are performed in the virtual environment. Moreover, for the sake of simplicity of the topology utilized, default routers of both LANs are connected directly by a virtual link. In such a scenario, the first network contains data that need to be secretly transmitted to the second network by using data-hiding techniques. In the testbed, covert transmission is inserted by a custom SDN application running at the SDN controller. During our experiments, we vary the number of clients, which represent IoT devices, sending data to one central server.

Gateway machines are connected to the Internet, which allows communication between LANs. The IoT sensors periodically transmit measurements to the server machine, which creates a flow of packets that passes through both SDN switches where the covert messages are injected (at the first SDN switch) and extracted (at the second SDN switch).

In our testbed, we use the following software:

- all virtual machines are running Centos 7.5.1804 (Core);
- open vSwitch v 2.10.1 on the SDN Switches;
- RYU Python SDN controller and
- apache HTTPD v 2.4.6 on the HTTP Server machine.

Devices sdn_cl1, sdn_cl2, sdn_cl3, sdn_cl4, sdn_cl5 act as smart temperature sensor nodes. A Python script is used to push sensor readings via HTTP POST messages to the central HTTP server running on sdn_srv01. Machine sdn_sw01 runs Open vSwitch network switch which is managed by the sdn_controller01 machine running RYU Controller. Machine sdn_gw01 acts as a router for this network. A similar configuration is used on the receiving side of the testbed, with sdn_sw02 running another Open vSwitch controlled by RYU running on sdn_controller02 machine. Machine sdn_gw02 acts as a router for the receiver network.

Covert channels utilized for DNCC creation

During experiments conducted in this article, we utilize a DNCC that combines various network covert channels. At first, we were determined to experiment with the mix of storage and timing techniques. However, the initial research performed showed that timing methods require intensive buffering which needs more packets than the simulated IoT system is able to produce. That is why we decided that during the experimental evaluation only storage methods will be used. Moreover, we have developed proprietary implementations of the data-hiding methods for two main reasons. First of all, the functionality provided in available, existing implementations was limited as in most cases they did not allow fine-grained tuning of the data-hiding technique's parameters, e.g. the number of the inserted covert bits or the frequency of overt traffic modification. Another reason is the overt traffic used for steganographic purposes which is typically closely associated with a given tool and cannot be easily modified. Therefore, using our proprietary implementations we are able to tune covert channels' parameters exactly to our needs. The following three subsections present functioning and implementation details of the state-of-the-art information-hiding techniques utilized, which we denote in the remainder of this article as: 'TTL Modulation', 'HTTP Header Reorder' and 'TCP Options Reorder'. General details of all implemented methods are described in the existing literature—TTL modulation technique was introduced in [47], reorder of HTTP headers was described in [48, 49] and there are many papers that proposed hiding information by TCP options manipulations, for the summary of such methods see [9].

To be more specific, in the first method, we conceal information by modulating the TTL field values within the IPv4 header, for the second the order of the headers within the HTTP message is

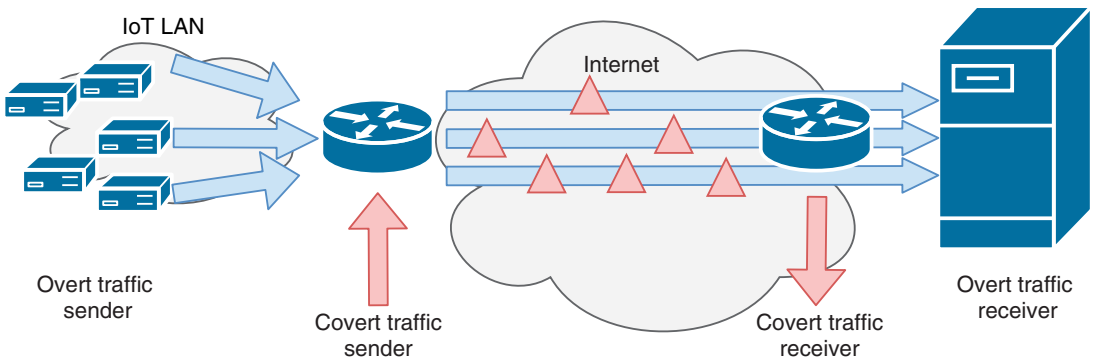


Figure 5: Assumed threat model, i.e. DNCC in the IoT environment.

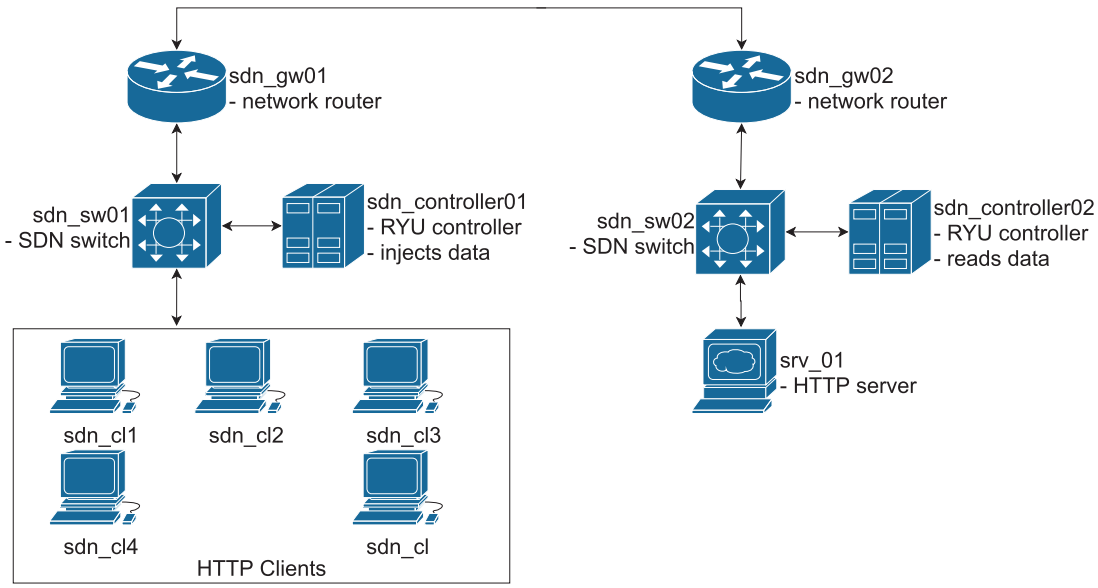


Figure 6: Utilized experimental testbed.

influenced, whereas for the third covert channel we embed secret data into the Options field of the TCP header. The motivation behind selecting these data-hiding techniques is as follows: (i) based on the analysed IoT traffic we concluded that often such devices rely on HTTP protocol and (ii) we decided also to choose network covert channels belonging to different layers of the TCP/IP stack to demonstrate information-hiding potential at various levels.

TTL modulation

The first network covert channel utilizes which modify TTL values, functions as follows. When a new connection suitable for the covert data transfer is established, the first few packets are left unmodified. This allows the CR to measure the expected value of the TTL field which depends on the number of intermediate nodes that the packets need to traverse before reaching their destination. This value can be seen as a 'hidden data carrier'—subtracting it from the actual TTL in all consecutive packets within the connection reveals the differences where the covert data is embedded. In our data-hiding technique, a particular value of the TTL difference is mapped to a bit or a group of bits and thus it forms a *symbol*. We modify the TTL field value within the connection by adding or subtracting a given symbol from its current value. For example, with the expected TTL = 64, a change of +5 (i.e. TTL = 69) denotes a binary 1 whereas a modification of –5 (i.e. TTL = 59) denotes binary 0. If the packet carries no secret data then the TTL value is left unchanged. By providing 2^n symbols our method allows the insertion of n bits of secret data per packet. In the implemented module a list of symbols: +1, +2, +3, +4 would be mapped to the following TTL values: 65, 66, 67, 68 and the secret data binary values: 00, 01, 10, 11. Note that we transmit the secret data by dividing it into chunks of n bits and by mapping their binary values into symbols. This allows us to potentially embed up to 8 bits of secret data per packet (thus filling the entire TTL field). However, in practice a modification leading to a too-small TTL value may cause the packet to be dropped by routers before reaching its destination. Additionally, significant and frequent changes of the TTL field over a broad range of values may raise suspicion and render the method easily detectable. In the approach that we took in this article, i.e. by utilizing SDN concept for implementing covert communication (for details see Subsection Testbed), we are able to spread the secret data between multiple available TCP connections for increased stealth without significantly degrading covert data transmission rate.

HTTP header reorder

The second data-hiding technique utilizes overt HTTP protocol messages as a hidden data carrier. Covert communication is implemented by tampering with the HTTP protocol messages as follows. We capture the HTTP requests and modify them on the fly by modifying the order of the HTTP headers. By changing the sequence of headers in the HTTP request and not the data contained within headers, the request itself stays compliant with the HTTP standard. The current HTTP/1.1 standard does not specify any requirements for ordering the headers within requests and responses. In practice, this means that the headers are sent in the order in which they are stored in the client and server memory—usually a hashmap. During our research, we have estimated the number of changes to the order of the headers occurring within a normal HTTP traffic by browsing 'Alexa's Top 5000' websites.[3] It turned out, that in the typical user traffic the order of headers changed quite frequently, i.e. in about 18% of all cases. Common HTTP request parsers store headers in a hashmap; thus, they ignore the discrepancies in the header order and cannot distinguish the difference between a normal HTTP request and a modified one.

Our data-hiding algorithm assumes that repeating the HTTP requests (sent by the same browser and on the same domain and URL) preserves the same set and the order of the HTTP headers. With such an assumption, we establish a baseline by recording the original order of the headers sent on a given URL and compare the subsequent requests with the original one. If there is any difference in the order, then we can easily compare the expected and actual order within the packet. The data are encoded by the header index within the original HTTP request that is relocated into the first (0)

3 https://www.alexa.com/topsites

position. The index is treated as a symbol which represents a group of bits. The number of headers present in the request determines how many different symbols are possible, which in turn influences how many bits a single symbol can represent. The symbol list can be configured by the user (i.e. the user can work with fewer symbols than the maximum available). The number of symbols must be a power of 2, i.e. to send 1 bit of covert data you need to specify 2 symbols, for 2 bits you need 4 symbols, for 3 bits—8 symbols, etc. For example, we can send 2 bits of secret data per request by specifying 4 symbols: 1, 2, 3, 4, which map to 00, 01, 10, 11 bits. If we consider, for example, the original HTTP request as follows:
and if we want to encode symbol '2', the header with index 2 (Accept-Encoding) should be relocated to the beginning, resulting in a request:

The server software treats the modified request as identical to the original one. Additionally, this request still looks legitimate for a human manually inspecting the network traffic. If there is no change in the HTTP headers order, it simply means that there is no data encoded within this message.

```
POST / HTTP/1.1
(0) Host: srv1-on.sdn
(1) User-Agent: python-requests/2.22.0
(2) Accept-Encoding: gzip, deflate
(3) Accept: */*
(4) Connection: keep-alive
(5) Content-Type: application/xml
(6) Content-Length: 406
```

TCP options reorder

For this network covert channel, the secret communication is implemented by tampering with the TCP segments that carry transmission between the IoT sensor and central server. We capture this type of network traffic and we modify the TCP header of these packets on the fly by changing the order of TCP options.

```
POST / HTTP/1.1
(0) Accept-Encoding: gzip, deflate
(1) Host: srv1-on.sdn
(2) User-Agent: python-requests/2.22.0
(3) Accept: */*
(4) Connection: keep-alive
(5) Content-Type: application/xml
(6) Content-Length: 406
```

Because the order of options (apart from option 0, i.e. END OF OPTIONS, which terminates the list and has to be at the end) contained in the non-mandatory TCP options field is not strictly defined, it is possible to send hidden data by rearranging the entries present in the packet without breaking or altering the content it carries in any way.

Due to the existence of many different implementations of the network stack on machines connected to the Internet, we suspect that options rearrangement occurs with some frequency even in benign user traffic. If this assumption is correct, this would mean that our data transmission is harder to spot and thus more viable. During our research, we have assessed the number of changes to the order occurring within normal HTTP traffic by browsing the 'Alexa's Top 5000' websites. It turned out that changes to the order of options do, in fact, occur quite frequently (i.e. 2.5% from 2 billion packets tracked) within user normal traffic, especially with the order of option no. 8 (TIMESTAMP) and no. 1 (NO OPERATION, i.e. a null option used for padding). That is why, we have chosen the order of these two fields as a hidden-data carrier for the secret data.

Our algorithm completely ignores packets that do not contain both TCP option no. 8 and TCP option no. 1. If both are present, the algorithm checks if these options follow the initial 'default' order which was used in the first packet in the flow observed by the SDN controller. If not, these options are swapped. It should be noted that only the order is checked and not the adjacency. This is because there could be other TCP options present in the list between the two that are used within our data-hiding method. It is also possible that the options list contains more than one option no. 1 entry. That is why the algorithm works only on the first option no. 1 entry found in the list. The order of these two options is preserved until a bit transfer is requested. If this happens, the order is swapped to indicate that the bit transfer is about to occur. If the transferred bit carries 0 value, the next packet also has these options swapped (i.e. the same order as in the 'reference' packet), which then becomes the 'default' order until another bit transfer occurs. If the transferred bit is 1, the next packet has the TCP options in the same order as the 'default' one and the 'default' order remains the same.

For example, for the sequence of packets like those presented below, three secret bits would be transmitted:

1. order (1,8) – nothing
2. order (1,8) – nothing
3. order (8,1) – next packet contains secret data
4. order (8,1) – encoded bit: 0
5. order (8,1) – nothing (the default order has changed)
6. order (8,1) – nothing
7. order (1,8) – next packet contains secret data
8. order (8,1) – encoded bit: 1
9. order (8,1) – nothing (the default order has not changed)
10. order (1,8) – next packet contains secret data
11. order (1,8) – encoded bit: 0
12. order (1,8) – nothing (the default order has changed again)
13. order (1,8) – nothing

For the network covert channels presented in this section and in the remainder of the article, we present our experimental DNCC evaluation and the results obtained.

Created dataset

During the research, we simulated the DNCC which uses IoT sensors traffic as overt traffic. Simulated traffic was based on an available thermometer with ethernet connectivity developed by Papouch company.[4] The product uses HTTP protocol to send sensor readings

4 https://cdn.en.papouch.com/data/user-content/old_eshop/files/TME_C_EU/tme_en.pdf

Figure 7: Automatically generated plot presenting particular packet modifications in connections used by DNCC (included in the dataset).

in the XML format. The frequency of the temperature measurements and their transmission to the central server is configured in the simulation parameters. Moreover, during experiments we use various sensors which used traffic for the DNCC purposes. Additionally, it must be noted that each experiment conducted lasted five minutes.

All traffic generated during each experiment is captured and archived in the.pcap format for later offline analysis. Moreover, to simplify further analysis concerning DNCC parameter tuning, for each test run we automatically generate a dedicated plot. Each plot graphically presents for each sender, receiver and any connection time instants when a particular packet is modified by insertion of covert information. Such an exemplary plot is presented in Fig. 7. As discussed earlier during the experiments conducted, we utilized three covert channels for which packet modifications are presented in the plot as red, green and blue ellipses. The top part of the plot combines information concerning changes introduced by all methods for a given source and destination. As can be easily observed, Fig. 7 illustrates the situation when five IoT sensors send data to the one central server. In the bottom part of the plot, detailed information concerning changes applied to each connection used by the data-hiding methods forming DNCC is visualized. In the excerpt presented, we can see the detailed information concerning changes introduced by the covert communication for the transmissions between a machine with source IP address 192.168.1.3 and a machine with IP address 192.168.2.3. In the first connection during the DNCC transmission two packets are modified, in the second two, in the third three and so on. The figure presented shows only a part of the entire plot which covers all five sending machines.

It must be also emphasized that an additional contribution of this work is that we share the created dataset with the scientific community. We will make available practically all data gathered during the experiments. The most important part of the data provided is the packet traces with covert transmissions in the pcap format. These files contain all packets used during a particular test-run and they can be used by the security community to develop further detection methods. Moreover, to ease experiments concerning only classification of the pre-processed data we also provide all statistics concerning suspicious packet behaviour in the JSON format. Moreover, for each test-run automatically generated visualization of the DNCC transmission in SVG format (with interactive features allowing finding modified packets in provided pcap traces) is also enclosed. At the URL: http://cssg.zoak.ii.pw.edu.pl/iot_dncc_data_set.html sample DNCC configurations are freely available. All the other data gathered will be provided to interested researchers upon request.

Experiments, results and discussion

In general, the DNCC can be applied 2-fold. In the first case, when maximizing covert data rate adding new steganographic methods contributes to the overall DNCC steganographic data rate. In the second case, when higher undetectability is desired the overall covert bandwidth is intentionally not increased and the usage of various steganographic methods allows improvement of DNCC channel stealt. Note that the research presented in this section concerns tuning the DNCC channel in such a way that the steganographic methods used modify a similar number of packets. As the research described in the [1] proved, without such a tuning a single steganographic method constituting the DNCC which modifies noticeably more packets can compromise the entire covert transmission. However, as presented previously in [1] in order to improve undetectability, all methods forming DNCC have to be tuned conveniently. Note, that in the paper mentioned above, the approach utilized to form DNCC relied on generating all possible combinations of configurations and then identified which parameter sets introduced a similar number of changes to overt network traffic. For this purpose, a well-known statistics parameter, i.e. the coefficient of variation (CV) metric was used.

The CV is defined as a ratio between the standard deviation and a corresponding mean value. This metric is typically used to express the extent of variability in relation to the mean and is also known as relative standard deviation. The higher the coefficient of variation, the greater the level of dispersion around the mean. During the experiments, we calculate CV using the number of modified packets by each steganographic method which forms the analysed DNCC.

In our case, the lower the CV for the certain configuration the more balanced the approach is. So, for example, for the three covert techniques utilized the number of modified packets is 115, 148 and 145, then the standard deviation is equal to 14.9, the mean is 136, and thus, the resulting CV is equivalent to 11%. The rationale for using CV is that it allows, independently of the order of magnitude of the resulting modified packets number for any specific method, to compare different parameter sets. Therefore, in this article to present the best DNCC configurations, only these parameter sets that result in the smallest CV value were analysed.

However, it must be noted that the approach used in [1] was not optimal. First, one had to execute all possible parameter sets in the testbed. Then, only a small fraction of results which met the CV metric threshold were presented to the user. That is why in this article, we introduce a different approach, which provides better results with fewer computations.

The details of the proposed procedure are as follows. In the first step, we execute each steganographic method used for DNCC creation separately with various parameter sets. As already mentioned, each such experiment lasts for five minutes. During the experiments conducted, we modify various parameters like (among others): the number of clients, the delay between IoT device reports, the number of secret bits included and the interval in which covert bits are inserted.

For the three steganographic methods described in subsection 'Covert channels utilized for DNCC creation', this means that during experiments we evaluate 1344 parameter sets for TTL Modulation, 672 for HTTP Header Reorder and 336 for TCP Options Reorder methods. The different numbers of experiments indicated above are associated with the number of secret bits inserted during the embedding process using a given steganographic method, which are, respectively, 4, 2 and 1. Tables 1, 2 and 3 include the fraction of the obtained results. In particular, they present the highest achieved steganographic throughput for each method separately, top 25 results for the TTL Modulation method and top 10 for the remaining two. We present more results for the TTL Modulation method because the insertion of four covert bits allows us to achieve greater steganographic bandwidth and slower transmission is used further in the text for comparison purposes. Note, that in these tables the first two columns show IoT environment parameters, i.e. the number of clients and the delay between consecutive measurements. Then, the next three columns represent parameters describing formed DNCC. Further columns contain detailed results concerning the covert channel data rate achieved and statistics of steganographically modified packets. As it is visible, the utilized methods have various steganographic bandwidths up to 225 bps for the TTL Modulation and up to 30 for the remaining two data-hiding techniques.

During the second step of the introduced DNCC-forming procedure, it is determined which combinations of a single method's parameter sets introduce a comparable number of packet changes. Therefore, the initial data obtained in the first step of this procedure is utilized. In this article, similar to [1] CV metric is used to filter only interesting parameter sets; however, we do not perform each experiment, but we only point out which parameters' sets are of interest. In this step, CV metric defines the maximum percentage difference between the numbers of modified packets in all methods used by the DNCC. For further evaluation, we use only parameters sets for which CV values are below the threshold defined by the user. Tables 4 and 5 present results for 15 highest data rates for the DNCC formed from all three methods for two different CV thresholds, i.e. 10 and 1%, respectively. The results are average values from ten executions of each experiment having the same parameter set. Note, that Tables 4 and 5 contain practically the same columns, as in the case of previously presented results. The only exception is the last column that contains CV values actually achieved, calculated based on the number of modified packets.

As already mentioned, we utilize the covert channels forming the DNCC with the aim to increase the overall stealth. Note that for such a scenario where the three covert channels are utilized, the highest DNCC throughput achieved is around 20 bps and in total, this requires applying modifications to fewer than 2850 packets. For this configuration, each data-hiding method does not alternate more than 1100 packets. In contrast, when we compare the number of modifications required when each steganographic method is used separately (for a comparable covert data rate), then we would need to modify from 1600 to 3400 packets for the TTL Modulation, more than 10,000 for the TCP Options Reorder and more than 3000 for the HTTP Header Reorder. Therefore, it is visible that finding the proper DNCC configuration provides benefit with more balanced packets modifications across the data-hiding techniques used.

Note also that CV threshold utilized has a significant impact on the results achieved, mainly on the resultant steganographic data rate. For the 10% CV threshold, the covert bandwidth achieved is around two-thirds of the slowest single method and exceeds 21 bps. What should be emphasized is that in this situation the CV values observed for the real data gathered during the experiments never exceeds the 10% level. The results for CV with the 1% threshold are poor. In this case, for all measured cases the real CV values are worse than predicted. In the worst-case scenario, it reaches 7%. Moreover, the steganographic bandwidth achieved is very low, and only for the fastest parameter sets exceeds 1 bps.

We further investigated the impact of the CV threshold on the two most important variables—the number of parameter sets that fulfils these conditions and the predicted steganographic data rate. Figure 8 illustrates how the number of parameter sets depends on the CV. For the values larger than 10%, this relationship seems to be linear. However, for the lower CV values the curve exhibits a more exponential shape. This behaviour explains why the usage of the 1% threshold has such an impact on the achieved covert data rate.

Furthermore, Fig. 9 presents experimental results for the predicted and steganographic actually achieved bandwidth. Note that the actually achieved values are calculated as an average from 10 repetitions of an experiment with a given parameter set. As it is visible, there are numerous saturation regions in which modifications of the CV parameter do not cause any changes in the resultant covert data rate. This behaviour is associated with a finite number of experiments in the first step of the proposed DNCC creation procedure which utilizes separate steganographic methods. Due to this fact, even if we slightly modify the CV threshold, then for this case there are no matching parameter sets. It is also worth noting that conducting more experiments in the first step should improve this situation and in effect, this would allow a user to better tune predicted steganographic throughput for their needs. Note that this still makes the current DNCC creation procedure superior when compared to the one presented in [1].

From Fig. 9, it is also visible that for a low CV threshold the predicted and actually achieved steganographic bandwidth is almost the same. For the moderate CV values (CV threshold higher than 15% and lower than 45%), the values actually achieved are slightly lower than predicted. Analysis of the parameter sets used during these experiments revealed that this situation was caused by one of

Table 1: The results obtained for a single method—highest throughput for the TTL Modulation technique (note: in the 'Symbol frequency' column '–' denotes that the method introduces steganographic changes to every packet)

DNCC parameters					Traffic statistics (at the CS)					Packets modified		
Clients	Delay	Used method	Bits per symbol	Symbol frequency	Steg. band. (bps)	TCP connections	Overall packet count	Modified packet count	Secret bits transmitted	HTTP Header Reorder	TTL Mod.	TCP Options Reorder
15	1	TTL	4	–	225.59	15	17009	16919	67676	0	16919	0
15	1	TTL	3	–	173.16	15	17406	17316	51948	0	17316	0
10	1	TTL	4	–	146.71	10	11063	11003	44012	0	11003	0
15	1	TTL	2	–	113.95	15	17183	17093	34186	0	17093	0
10	1	TTL	3	–	109.87	10	11047	10987	32961	0	10987	0
5	1	TTL	4	–	73.96	5	5577	5547	22188	0	5547	0
10	1	TTL	2	–	73.62	10	11103	11043	22086	0	11043	0
15	1	TTL	1	–	57.62	15	17377	17287	17287	0	17287	0
5	1	TTL	3	–	54.58	5	5488	5458	16374	0	5458	0
15	5	TTL	4	–	45.49	15	3502	3412	13648	0	3412	0
15	1	TTL	4	5	45.05	15	16987	3379	13516	0	3379	0
10	1	TTL	1	–	38.18	10	11515	11455	11455	0	11455	0
5	1	TTL	2	–	35.67	5	5380	5350	10700	0	5350	0
15	1	TTL	3	5	33.54	15	16860	3354	10062	0	3354	0
15	5	TTL	3	–	32.98	15	3388	3298	9894	0	3298	0
10	1	TTL	4	5	30.00	10	11314	2250	9000	0	2250	0
10	5	TTL	4	–	29.24	10	2253	2193	8772	0	2193	0
15	5	TTL	2	–	22.71	15	3496	3406	6812	0	3406	0
10	1	TTL	3	5	22.49	10	11309	2249	6747	0	2249	0
15	1	TTL	4	10	22.15	15	16707	1661	6644	0	1661	0
15	1	TTL	2	5	21.89	15	16509	3283	6566	0	3283	0
10	5	TTL	3	–	21.87	10	2247	2187	6561	0	2187	0
5	1	TTL	1	–	18.20	5	5490	5460	5460	0	5460	0
15	1	TTL	3	10	16.63	15	16729	1663	4989	0	1663	0

Table 2: The results obtained for a single method—highest throughput for the TCP Options Reorder technique (note: in the 'Symbol frequency' column '–' denotes that the method introduces steganographic changes to every packet)

DNCC parameters					Traffic statistics (at the CS)					Packets modified		
Clients	Delay	Used method	Bits per symbol	Symbol frequency	Steg. band. (bps)	TCP Connections	Overall packet count	Modified packet count	Secret bits transmitted	HTTP Header Reorder	TTL Mod.	TCP Options Reorder
15	1	TCPOptions	1	–	28.12	15	16910	16880	8437	0	0	16880
10	1	TCPOptions	1	–	18.10	10	10882	10862	5430	0	0	10862
5	1	TCPOptions	1	–	9.16	5	5513	5503	2749	0	0	5503
15	1	TCPOptions	1	5	8.04	15	16898	4826	2412	0	0	4826
15	5	TCPOptions	1	–	5.71	15	3464	3434	1714	0	0	3434
10	1	TCPOptions	1	5	5.38	10	11218	3230	1615	0	0	3230
15	1	TCPOptions	1	10	4.41	15	16693	2646	1323	0	0	2646
10	5	TCPOptions	1	–	3.73	10	2264	2244	1120	0	0	2244
15	1	TCPOptions	1	15	2.97	15	16516	1783	891	0	0	1783
10	1	TCPOptions	1	10	2.94	10	11155	1765	882	0	0	1765

Table 3: The results obtained for a single method—highest throughput for the HTTP Header Reorder technique (note: in the 'Symbol frequency' column '–' denotes that the method introduces steganographic changes to every packet)

DNCC parameters					Traffic statistics (at the CS)					Packets modified		
Clients	Delay	Used method	Bits per symbol	Symbol Frequency	Steg. band. (bps)	TCP connections	Overall packet count	Modified packet count	Secret bits transmitted	HTTP Header Reorder	TTL Mod.	TCP Options Reorder
15	1	HTTPReorder	2	–	30.08	15	35325	4512	9024	4512	0	0
10	1	HTTPReorder	2	–	20.05	10	23319	3007	6014	3007	0	0
15	1	HTTPReorder	1	–	15.04	15	34889	4511	4511	4511	0	0
10	1	HTTPReorder	1	–	10.03	10	23367	3009	3009	3009	0	0
5	1	HTTPReorder	2	–	10.01	5	11577	1502	3004	1502	0	0
15	1	HTTPReorder	2	5	6.01	15	35099	902	1804	902	0	0
15	5	HTTPReorder	2	–	6.00	15	7023	900	1800	900	0	0
5	1	HTTPReorder	1	–	5.00	5	11486	1501	1501	1501	0	0
10	1	HTTPReorder	2	5	4.01	10	23203	601	1202	601	0	0
10	5	HTTPReorder	2	–	4.00	10	4725	600	1200	600	0	0

Table 4: The highest throughput results obtained for DNCC utilizing three steganographic methods: HTTP Header Reorder, TTL Modulation and TCP Options Reorder (CV <10%)

DNCC parameters					Traffic statistics (at the CS)					Packets modified			
Clients	Delay	Used methods	Bits per symbol	Symbol frequency	Steg. band. (bps)	TCP connections	Overall packet count	Modified packet count	Secret bits transmitted	HTTP Header Reorder	TTL Mod.	TCP Options Reorder	CV (%)
15	1	All	2,4,1	5,15,30	21.48	15.0	22585.8	2841.9	6444.4	868.2	1063.2	910.5	8.9
15	1	All	1,4,1	5,15,30	18.19	15.0	22172.5	2795.6	5459.1	869.6	1036.2	889.8	8.0
15	1	All	2,3,1	5,15,25	17.92	15.0	22346.8	2951.0	5375.5	863.7	1041.9	1045.4	8.6
15	1	All	2,4,1	5,20,30	17.89	15.0	22506.2	2569.3	5368.0	870.5	793.7	905.1	5.6
15	1	All	2,3,1	5,15,30	17.74	15.0	22334.7	2809.1	5323.2	867.9	1046.8	894.4	8.4
15	1	All	2,4,1	5,20,40	17.61	15.0	22399.6	2394.0	5284.2	874.8	792.9	726.3	7.6
15	1	All	2,4,1	5,20,35	17.49	15.0	22124.8	2441.3	5248.9	871.7	777.4	792.2	5.1
15	1	All	2,3,1	5,20,30	15.12	15.0	22297.9	2546.2	4537.2	870.2	783.6	892.4	5.6
15	1	All	2,3,1	5,20,35	15.07	15.0	22406.4	2471.7	4520.9	872.0	790.9	808.8	4.2
15	1	All	1,3,1	5,15,30	14.92	15.0	22431.4	2819.9	4475.5	867.9	1052.8	899.2	8.6
15	1	All	2,3,1	5,20,40	14.92	15.0	22271.3	2390.8	4474.9	875.8	786.4	728.6	7.6
15	1	All	1,3,1	5,15,25	14.91	15.0	22177.5	2922.8	4472.3	864.0	1031.6	1027.2	8.0
15	1	All	1,4,1	5,20,30	14.79	15.0	22254.8	2542.7	4439.8	869.7	781.1	891.9	5.7
15	1	All	1,4,1	5,20,35	14.71	15.0	22273.4	2460.4	4412.5	873.9	784.4	802.1	4.7
15	1	All	1,4,1	5,20,40	14.56	15.0	22193.3	2381.1	4367.3	875.0	782.7	723.4	7.9

Table 5: The highest throughput results obtained for DNCC utilizing three steganographic methods: HTTP Header Reorder, TTL Modulation and TCP Options Reorder (CV <1%)

DNCC parameters					Traffic statistics (at the CS)					Packets modified			
Clients	Delay	Used methods	Bits per symbol	Symbol frequency	Steg. band. (bps)	TCP connections	Overall packet count	Modified packet count	Secret bits transmitted	HTTP Header Reorder	TTL Mod.	TCP Options Reorder	CV (%)
15	1	All	2,4,1	70,250,500	1.42	15.0	22889.9	196.3	427.1	63.9	66.6	65.8	1.8
15	1	All	1,4,1	70,250,500	1.19	15.0	22615.1	193.8	358.5	64.0	65.6	64.2	1.5
15	1	All	2,2,1	70,250,500	0.97	15.0	22716.5	194.7	292.2	64.0	65.9	64.8	1.4
15	1	All	1,2,1	70,250,500	0.81	15.0	24100.8	206.6	242.0	68.0	69.8	68.8	1.5
5	5	All	2,4,1	10,35,70	0.66	5.0	1558.7	91.3	199.4	29.0	31.5	30.8	4.0
5	5	All	1,4,1	10,35,70	0.56	5.0	1542.7	90.2	167.4	29.0	30.8	30.4	2.7
1	1	All	2,4,1	15,50,100	0.43	1.0	1452.1	60.5	129.8	19.0	20.3	21.2	4.6
10	15	All	2,4,1	10,35,70	0.43	10.0	1092.5	59.0	130.4	19.0	20.7	19.3	4.6
5	5	All	2,3,1	15,50,100	0.38	5.0	1561.4	63.1	114.8	19.0	21.9	22.2	7.0
1	1	All	1,4,1	15,50,100	0.38	1.0	1474.2	61.7	113.5	19.0	20.9	21.8	5.8
10	15	All	1,4,1	10,35,70	0.37	10.0	1100.5	60.5	112.9	19.0	20.9	20.6	4.3
1	1	All	2,3,1	15,50,100	0.37	1.0	1456.4	61.4	111.1	19.1	20.7	21.6	5.1
15	30	All	2,4,1	10,35,70	0.33	15.0	887.5	44.1	97.6	14.0	15.6	14.5	5.6
5	5	All	1,3,1	15,50,100	0.32	5.0	1545.5	62.3	94.7	19.1	21.6	21.6	5.9
1	1	All	1,3,1	15,50,100	0.31	1.0	1453.3	60.6	90.9	19.2	20.4	21.0	3.8

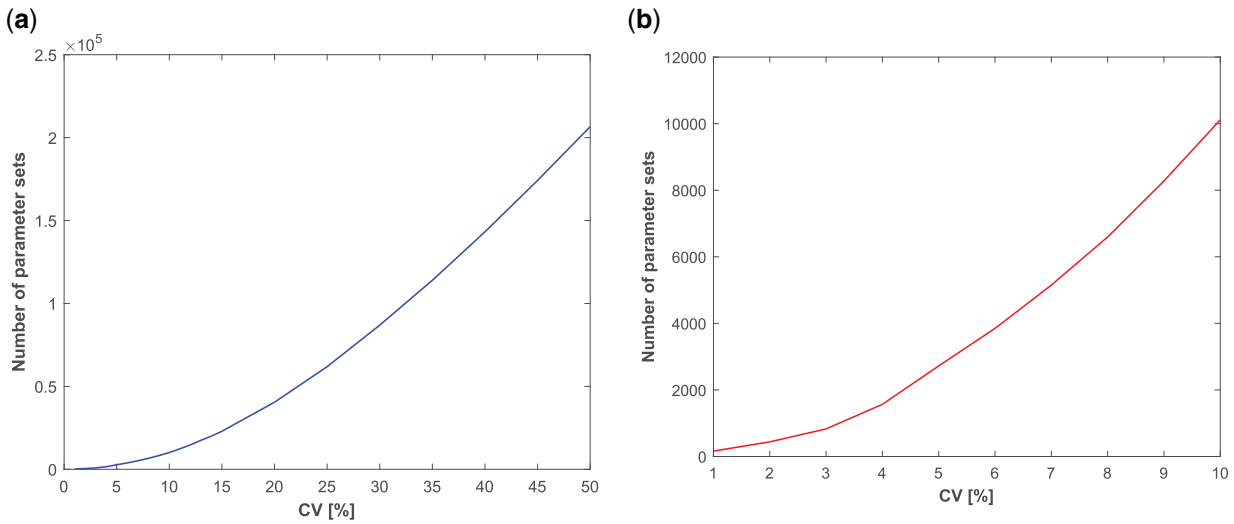


Figure 8: The number of parameter sets which fulfil CV threshold: for CV from 1% to 50% (left); for CV from 1% to 10% (right).

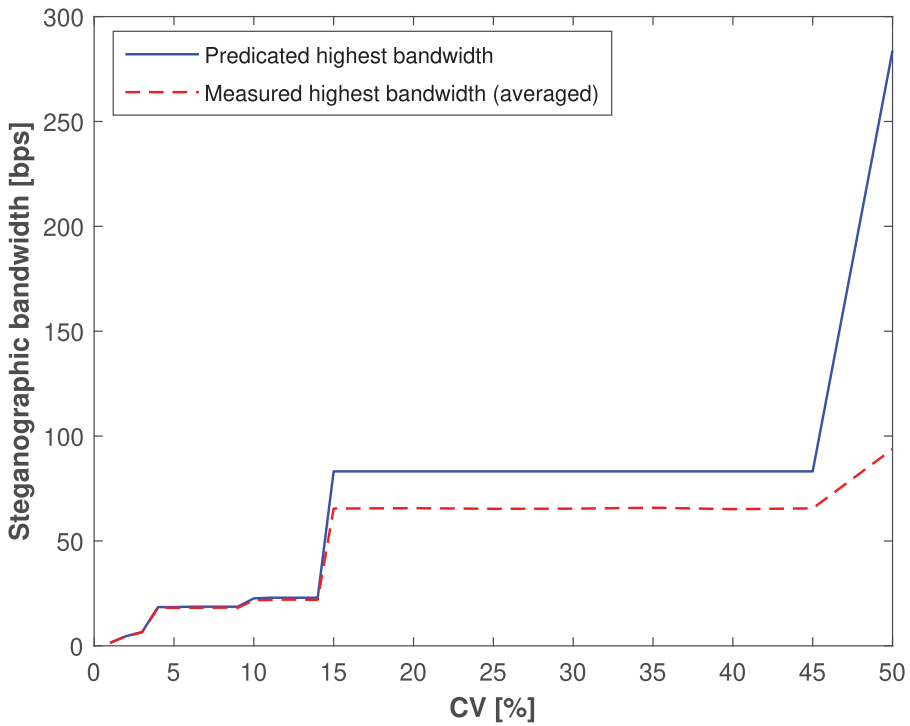


Figure 9: Predicted and measured covert data rate depending on the CV value.

the three steganographic methods used, which was supposed to embed secret data in every incoming packet. In some cases, more than one method competes to introduce modifications to the same packet. Current implementation of our DNCC software does not allow such a situation. In consequence, one method is not able to embed covert data, which in effect reduces the overall steganographic bandwidth. It is also worth noting that for a CV threshold >45%, all three methods should insert data in each consecutive packet. This situation results in complete collisions, which in turn dramatically reduces bandwidth achieved. Analysis of this data leads to one more conclusion. If we would like to properly predict DNCC steganographic bandwidth, we should reduce the possibilities of collisions. We could ensure such behaviour by using parameter sets, which do not require modifications of each packet.

Conclusions

In this article, we have investigated the nature of distributed network covert channels in IoT environments. In particular, we have analysed how DNCCs should be created and configured for this scenario in order to achieve the highest data rate while ensuring that the secret data are fairly distributed across all available data-hiding techniques to limit the chance of detection. The results obtained show that although the volume of traffic generated by the IoT devices is not high, the DNCC can be constructed to maximize its performance while not jeopardizing overall undetectability of the scheme.

As our future work we will focus on performing a thorough investigation of the potential methods for DNCC detection while also

observing how they need to be adapted depending on the selected networking environment. The main goal of our future research would be to develop a countermeasure that would be able to discover network traffic modifications in various layers of the TCP/IP stack and correlate these events for successful DNCC detection.

Supplementary data

Supplementary data are available at *CYBERS Journal* online.

Acknowledgments

Any opinions, finding and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Air Force.

Funding

This work was supported by Air Force Office of Scientific Research under 'CoCoDe—Covert Communication Detection' [grant number FA9550-17-1-0254].

Conflict of interest statement. None declared.

References

1. Cabaj K, Mazurczyk W, Nowakowski P. *et al.* Fine-tuning of distributed network covert channels parameters and their impact on undetectability. In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*, *ACM, New York, NY*, 2019, 65:1–**65**:8.
2. Mosenia A, Jha NK. A comprehensive study of security of Internet-of-Things. *IEEE Transact Emerging Top Comput* 2017;**5**:586–602.
3. Kumar JS, Patel DR. A survey on internet of things: security and privacy issues. *Int J Comput Appl* 2014;**90**.20–26.
4. Lin J, Yu W, Zhang N. *et al.* A survey on Internet of Things: architecture, enabling technologies, security and privacy, & applications. *IEEE Int Things J* 2017;**4**:1125–42.
5. Alaba FA, Othman M, Hashem IAT. *et al.* Internet of Things security: a survey. *J Network Comput Appl* 2017;**88**:10–28.
6. M.Sadeeq MA, Zeebaree SRM, Qashi R. *et al.* Internet of things security: a survey. In: *2018 International Conference on Advanced Science and Engineering (ICOASE)*, pp. 162–6, 2018. Duhok.
7. Mazurczyk W, Wendzel S, Zander S. *et al. Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, & Countermeasures*. Hoboken: Wiley-IEEE, 2016.
8. Zander S, Armitage G, Branch P. A survey of covert channels and countermeasures in computer network protocols. *Commun Surveys and Tutorials* 2007;**9**:44–57.
9. Mileva A, Panajotov B. Covert channels in TCP/IP protocol stack - extended version. *Central Eur J Comput Sci* 2014;**4**:45–66.
10. Cabuk S, Brodley CE, Shields C. IP covert timing channels: design and detection. In: *Proceedings of 11th ACM Conference on Computer and Communications Security* (ACM), pp. 178–87, 2004.
11. Cabaj K, Caviglione L, Mazurczyk W. *et al.* The new threats of information hiding: the road ahead. *IT Prof* 2018;**20**:31–9.
12. Mazurczyk W, Wendzel S. Information hiding: challenges for forensic experts. *Commun ACM* 2017;**61**:86–94. http://doi.acm.org/10.1145/3158416.
13. Caviglione L, Podolski M, Mazurczyk W, *et al.* Covert channels in personal cloud storage services: the case of dropbox. *IEEE Transact Ind Inf* 2017;**13**:1921–31.
14. Tahir R, Taha Khan M, Gong X, Ahmed A. *et al.* Sneak-peek: high speed covert channels in data center networks. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, 2016.
15. Blumbergs B, Pihelgas M, Kont M. *et al.* Creating and detecting ipv6 transition mechanism-based information exfiltration covert channels. In: Brumley BB, Röning J (eds), *Secure IT Systems*. Cham: Springer International Publishing, 2016, 85–100.
16. Nussbaum L, Neyron P, Richard O. On robust covert channels inside dns. In: Gritzalis D, Lopez J (eds), *Emerging Challenges for Security, Privacy and Trust*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, 51–62.
17. Ovadia A, Ogen R, Mallah Y. *et al. Cross-Router Covert Channels in 13th USENIX Workshop on Offensive Technologies (WOOT 19)*. Santa Clara: USENIX Association, 2019.
18. El-Atawy A, Duan Q, Al-Shaer E. A novel class of robust covert channels using out-of-order packets. *IEEE Transact Dependable Secure Comput* 2017;**14**:116–29.
19. Rios R, Onieva JA, Lopez J. Covert communications through network configuration messages. *Computers & Security* 2013;**39**:34–46.
20. Qian Y, Sun T, Li J. *et al.* Design and analysis of the covert channel implemented by behaviors of network users. *Security Commun Networks* 2016;**9**:2359–70.
21. Backs P, Wendzel S, Keller J. Dynamic routing in covert channel overlays based on control protocols. In *2012 International Conference for Internet Technology and Secured Transactions*, pp. 32–39, 2012.
22. Szczypiorski K, Margasiński I, Mazurczyk W. *et al.* In: Meersman R, Tari Z (eds), *Trustmas: Trusted Communication Platform for Multi-Agent Systems in on the Move to Meaningful Internet Systems: OTM 2008*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, 1019–35.
23. Nagaraja S, Housmansadr A, Piyawongwisal P. *et al.* Stegobot: a covert social network botnet. In: *Proceedings of the 13th International Conference on Information Hiding*. Berlin, Heidelberg: Springer, 2011, 299–313.
24. Mazurczyk W, Wendzel S, Cabaj K. Towards deriving insights into data hiding methods using pattern-based approach. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. New York, NY: ACM, 2018, 10:1–10:10.
25. Cabaj K, Mazurczyk W, Nowakowski P. *et al.* Towards distributed network covert channels detection using data mining-based approach. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. New York, NY: ACM, 2018, 12:1–12:10.
26. Antonakakis M, April T, Bailey M. *et al.* Understanding the Mirai Botnet. In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, 1093–1110.
27. Hron M. *Are Smart Homes Vulnerable to Hacking?* 2008. https://blog.avast.com/mqtt-vulnerabilities-hacking-smart-homes (Last accessed: 28.10.2020).
28. Sikder AK, Petracca G, Aksu H. *et al. A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications*. *CoRR* abs/1802.02041. arXiv: 1802. 02041, 2018. http://arxiv.org/abs/1802.02041.
29. Caviglione L, Merlo A, Migliardi M. Covert channels in IoT deployments through data hiding techniques. In *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 559–563, 2018.
30. Patuck R, Hernandez-Castro J. Steganography using the Extensible Messaging and Presence Protocol (XMPP). *Computing Research Repository arXiv*, 1310.0524, 2013.
31. Wendzel S. *Covert and Side Channels in Buildings and the Prototype of a Building-aware Active Warden in Proceedings of the IEEE ICC*, pp. 6753–6758, 2012.
32. Mileva A, Velinov A, Stojanov D. *New Covert Channels in Internet of Things in Proceedings of the 12th International Conference on Emerging Security Information, Systems and Technologies - SECURWARE 2018, September 16-20, 2018, Venice, Italy*, pp. 30–6, 2018.
33. Wendzel S, Mazurczyk W, Haas G. *Don't You Touch My Nuts: Information Hiding in Cyber-physical Systems in Proceedings of the IEEE SPW 2017*, pp. 29–34, 2017.
34. Velinov A, Mileva A, Wendzel S. *et al.* Covert channels in the MQTT-Based Internet of Things. *IEEE Access* 2019;**7**:161899–915.
35. Ulz T, Feldbacher M, Pieber TW. *et al.* Sensing danger: exploiting sensors to build covert channels. In: *Proceedings of the 5th International*

Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019, 2019, pp. 100–113.
36. Tan Y-a, Zhang X, Sharif K, *et al.* Covert timing channels for IoT over mobile networks. *IEEE Wireless Commun* 2018;**25**:38–44.
37. Moskowitz IS, Russell S, Jalaian B. *Steganographic Internet of Things: Graph Topology Timing Channels* in *The Workshops of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2–7,* 2018, volume WS-18 of *AAAI Workshops,* pp. 252–9.
38. Herzberg A, Kfir Y. *The Leaky Actuator: A Provably-Covert Channel in Cyber Physical Systems in Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, pp. 87–98, New York, NY: Association for Computing Machinery, 2019
39. Herzberg A, Kfir Y. The chatty-sensor: a provably-covert channel in cyber physical systems. In: *Proceedings of the 35th Annual Computer Security Applications Conference,* Association for Computing Machinery, pp. 638–649, New York, NY, USA, 2019.
40. Wendzel S, Zander S, Fechner B. *et al.* Pattern-based survey and categorization of network covert channel techniques. *ACM Comput Surveys* 2015;**47**:1–26.
41. Subhedar MS, Mankar VH. Current status and key issues in image steganography: a survey. *Comput Sci Rev* 2014;**13–14**:95–113.
42. Frączek W, Mazurczyk W, Szczypiorski K. Hiding information in a stream control transmission protocol. *Comput Commun* 2012;**35**:159–169.
43. Luo X, Chan EWW, Chang RKC. In: Biskup J, López J. (eds), *Cloak: A Ten-Fold Way for Reliable Covert Communications. in Computer Security – ESORICS 2007*. Berlin, Heidelberg: Springer, 2007, 283–298.
44. Wendzel S, Zander S. Detecting protocol switching covert channels. In *37th IEEE Conf. on Local Computer Networks*, pp. 280–283, 2012.
45. Wendzel S, Keller J. Hidden and under control. *Ann Telecommun - Annal Télécommun* 2014;**69**:417–430.
46. Kaur J, Wendzel S, Eissa O. *et al.* Covert channel-internal control protocols: attacks and defense. *Sec Commun Networks* 2016;**9**:2986–2997.
47. Zander S, Armitage G, Branch P. An empirical evaluation of IP time to live covert channels. In: *15th IEEE International Conference on Networks*, Adelaide, SA, Australia, 2007.
48. Katzenbeisser S, Petitcolas FA. *Information Hiding Techniques for Steganography and Digital Watermarking*, 1st edn. USA: Artech House, Inc., 2000.
49. Dimitrova B, Mileva A. Steganography of hypertext transfer protocol version 2 (http/2). *J Computer Commun* 2017;**5**:98–111.

4.4 PAPER IV

Nowakowski Piotr, Żórawski Piotr, Cabaj Krzysztof, Mazurczyk Wojciech: Data Mining and Hierarchical Organisation of Frequent Sets, in: Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 2021, vol. 12, no. 1, pp.20-43. DOI:10.22667/JOWUA.2021.03.31.020

This is extended version of:

Nowakowski Piotr, Żórawski Piotr, Cabaj Krzysztof, Mazurczyk Wojciech: Network covert channels detection using data mining and hierarchical organisation of frequent sets: an initial study, in: ARES '20: Proceedings of the 15th International Conference on Availability, Reliability and Security, 2020, ISBN 978-1-4503-8833-7, pp. 1-10, DOI:10.1145/3407023.3409217

Detecting Network Covert Channels using Machine Learning, Data Mining and Hierarchical Organisation of Frequent Sets

Piotr Nowakowski, Piotr Żórawski, Krzysztof Cabaj*, and Wojciech Mazurczyk
Warsaw University of Technology, Warsaw, Poland
{P.Nowakowski, P.Zorawski, K.Cabaj, W.Mazurczyk}@ii.pw.edu.pl

Received: December 29. 2020; Accepted: February 18, 2021; Published: March 31, 2021

Abstract

Due to continuing improvements in defensive systems, malware developers create increasingly sophisticated techniques to remain undetected on the infected machine for as long as possible. One flavor of such methods are network covert channels, which, to transfer secret data, utilize subtle modifications to the legitimate network traffic. As currently there is no one-size-fits-all approach which would be effective in detecting covert communication in an efficient and scalable manner, more research effort is needed to devise a suitable solution. That is why, in this paper we propose to utilize machine learning and data mining accompanied by hierarchical organization of frequent sets to detect network covert channels: both distributed and undistributed. The obtained experimental results prove that the proposed approach is effective and efficient.

Keywords: Distributed Network Covert Channels (DNCCs), Network Security, Information Hiding, Data mining, Machine Learning.

1 Introduction

Cybercriminals are continuously devising novel methods that would enable them to delay or thwart the detection of their malicious activities on the infected device. The main reason for that is the constant improvement of the existing defensive systems. The most popular techniques used to cloak the attackers include [15]: multistage loading, fileless operation capabilities, encrypted and obfuscated payloads, anti-analysis mechanisms, and recently also various types of information hiding techniques [5].

During the last years, many types of data hiding methods have been devised, but lately an increased attention is brought towards techniques that, to transfer secrets, subtly modify network traffic. This subgroup of information hiding is called *network covert channels* and due to their nature they serve well purpose mostly due to their ephemeral nature. So far, in the literature a plethora of different techniques utilizing different network protocols have been proposed [18], [9], [14]. It must be also noted that, commonly, attackers use covert channels to cloak, e.g., communication with Command & Control (C&C) servers, exfiltration of stolen data, or downloading of additional malware modules or configurations [1].

Moreover, it is predicted that the trend of using data hiding techniques by attackers is likely to prevail in the near future [1] and soon we will experience a rise in the utilization of far more sophisticated network covert channels than are currently deployed [8], [5]. A recent branch of such complex data hiding methods that are increasingly gaining attention are called *distributed network covert channels* (DNCCs) [2]. They are defined as channels that during covert communication spread the secret data

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 12(1):20-43, Mar. 2021
DOI:*10.22667/JOWUA.2021.03.31.020*

*Corresponding author: Department of Electronics and Information Technology, Institute of Computer Science, Nowowiejska 15/19, 00-65 Warsaw, Poland, Tel: +48-22-234-77-11

among many flows/protocols/hosts or use multiple data hiding methods within the same flow or within various Protocol Data Units (PDUs) [6]. It is worth noting that, in contrast, the typical (undistributed) network covert channels are storage or timing channels that to embed secret data utilize PDUs of a single flow or protocol [8]. Utilization of the DNCC for the attacker can be profitable because it may improve the overall stealthiness and resulting data hiding capacity of the hidden communication. This is possible as in the DNCC smaller parts of secret data are transmitted using several covert channels, flows, or hosts.

It must be emphasized that, up till now, in the existing literature only limited attempts have been made to detect distributed network covert channels [2]. Moreover, the proposed detection solutions were mostly focused on DNCCs that use many data hiding techniques simultaneously.

This work is aimed to improve this situation and it is an extended version of the previous conference paper [10] where we proposed a novel approach for network covert channel detection which utilizes multistep traffic processing combined with data mining and statistical techniques.

However, it must be noted that in this paper we significantly extend the initial concept and its evaluation. In more detail, the novel contributions of this work are:

- We introduce a new steganographic communication error detection and correction scheme for the DNCC,
- We add two more detection strategies which are based on machine learning (ML) – one utilizing only raw data from our preprocessing software and the second where we apply ML to the raw data enhanced by the metadata produced by our original concept,
- We conduct an extensive experimental evaluation where we determine the characteristics of all detection variants.

The rest of the paper is organized as follows. Section 2 presents the most notable works related to the detection of network covert channels. In section 3 the network covert channels we have selected for the experimental evaluation are presented. Next, in section 4 the proposed network covert channel detection strategy is introduced. Then, the methodology as well as the experimental testbed used are presented in section 5, while the obtained results are enclosed in section 6. Finally, section 7 summarizes our work and outlines potential future directions.

2 Related work

Nowadays, most of the research concerning the detection of information hiding techniques is devoted to discovering data hiding in various types of image, audio, or video files. However, as network covert channels become increasingly popular among attackers, this trend is slowly changing, especially that such techniques are used not only by human beings but they start appearing even in malware [1], as a technique, e.g., for protecting an attacker's C&C channel.

The simplest method that can be utilized for the detection of a network covert channel is to use the network warden [7]. Network warden can be treated as an Intrusion Detection System (IDS) which contains rules allowing the detection of well-known covert channels. During warden activity, all network traffic is subjected to inspection to find clues for network covert channel utilization. For example, we can consider a situation when in the IP packet Don't Fragment (DF) flag is set to '1', however the Fragment Offset (FO) field is carrying data other than zeros. This can be treated as a clear indication of the covert communication as under normal circumstances the FO field should contain nonzero values only if More Fragments (MF) flag is set to '1'. In this case, the network warden can normalize such disobeying fields and thus remove any secret data that is stored there. Detection of such 'misuse' of protocols' header fields is effective for well-known attacks and in the situation when security researchers provide rules.

However, such an approach is unsuitable for the detection of new, completely unknown methods. For such threats, behavioral analysis with anomaly detection should be used. In the literature, we can find numerous approaches which rely on various techniques. A brief description of the most relevant ones is presented below.

In [19] the authors utilize Markov model for the detection of anomalous traffic. Initially, they are using network traffic without covert channels to generate a model for normal TCP/IP stack activity, which they call TMM – TCP Markov Model. Furthermore, during the detection phase, unknown traffic, potentially harmful containing hidden data is compared to the previously 'clean' model using Kullback-Leibler (KL) statistical test. If the calculated value is greater than the chosen threshold, the traffic is marked as containing covert data.

Similar approach is described in [13], however, in this case the authors propose to use neural networks to learn the initial sequence number (ISN) generator. Such a model can be later used for the detection of covert channels that encode data in ISN number – for example, NUSHU tool. The proposed detection system observes the network traffic, predicts the next ISN number using the learned model, and compares it with the observed ISN carried in TCP segments. Enhanced neural networks, more specifically Recurrent Neural Network (RNN), are used in the [12]. What is interesting in this research, network covert channels are detected by performing an analysis of system calls taken from the monitored machine.

Other researchers propose to detect covert communication using machine learning-based approaches. In [11] the authors utilize Support Vector Machine (SVM) classifier for the detection of timing covert channels. As input data, various properties of inter-packet delay for the consequent 100 and 2000 packets are used. In [16] authors introduce two-step solution which uses density-based clustering. In the first step, the network data is clustered. Then, in the second step, the introduced classification algorithm can analyze the clusters and check if any of them is characterized with different density. In this case, it can be treated as outliers, i.e., as a sign of covert transmission.

When compared with existing works, the proposed detection method introduced in this paper uses machine learning together with data mining frequent pattern discovery. Such an approach has not been evaluated yet in the literature for network covert channel detection. The only work that deals with this topic is our previous papers published in [2] and [10]. However, in [2] only one *minimal support* parameter was used, while the method proposed in [10] utilizes multiple discovery executions and automatic analysis of the gathered results and is applicable to the DNCCs that rely on many data hiding techniques which are scattering covert data into many flows and hosts.

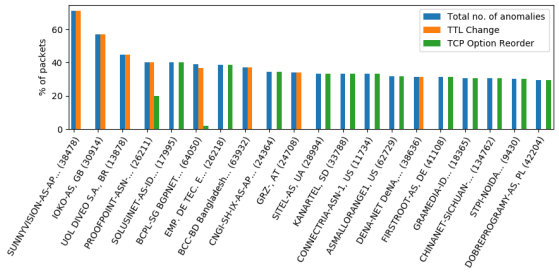
As already mentioned, this paper is a significantly extended version of the conference paper [10] where we improve several deficits of the initial approach as well as we apply machine learning techniques to further improve its detection performance and perform extensive experimental evaluation.

3 Network covert channels selected for the DNCC creation

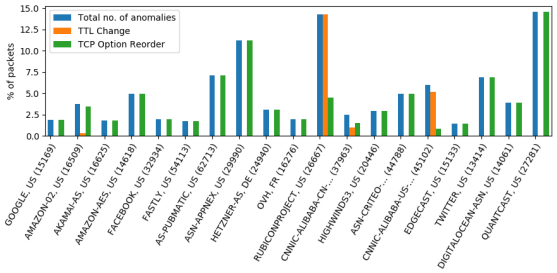
In this section, we first discuss our motivation behind using the three network covert channels that we chose for the experimental evaluation. Then, we present their detailed inner workings. Note that we deliberately selected network covert channels which rely on protocols that reside in three different layers of the TCP/IP stack, i.e., network (IPv4), transport (TCP), and application (HTTP).

3.1 Rationale behind covert channel selection

From the network traffic perspective, it is commonly believed that the utilization of covert channels would cause anomalies that should be easy to spot and that modern networks are no longer affected by



(a) Sorted by AS with the most frequently occurring anomalies.



(b) Sorted by most popular AS.

Figure 1: Network anomalies by AS, detected within web traffic to Alexa top 10,000 most popular websites (for the sake of readability, only top 20 positions are shown) [10].

routing or network protocol anomalies. A classical network anomaly example is a sudden change in the Time To Live (TTL) value (in the IPv4 header) within the same network connection. TTL-based mechanism is designed to prevent routing loops by limiting the number of network hops that a single packet can traverse. Each router on the communication path decreases the packet's TTL value and drops it if it reaches zero. However, if the TTL value has suddenly changed, it may also mean that the routing path has been changed, e.g., due to router failure and has been switched to the alternative routing path. In other cases, the TTL field can be used by simple covert channels [17], thus its sudden change may also indicate that the covert communication is taking place. If such an assumption is genuine, then each detected traffic anomaly should be treated as a clear indicator of the intentional packet manipulation performed either by the two hosts involved in network transmission or a malicious third party. In effect, a naive approach for covert channel detection would simply rely on enumeration of all identified traffic anomalies.

In this paper, we choose three covert channels that introduce anomalies to the normal overt network traffic. To verify whether these anomalies are occurring naturally in benign traffic, we have performed a very basic experiment. Specifically, we generated benign web traffic to the world's most popular websites and observed if any unexpected anomalies exist. The experiments were performed by launching an automated headless Chromium browser[1] to visit top 10,000 most popular websites listed by Alexa[2]. By analyzing the captured web traffic and grouping the responses' IPv4 source addresses by corresponding Autonomous Systems (AS), we were able to identify communication networks that exhibited a significant number of network anomalies. In the result, it was possible to establish a baseline using the most popular and commonly trusted networks.

The results of the conducted traffic measurements are presented in Figure 1. It is visible that for certain ASs up to 71% of the packets may contain an unexpected anomaly and even the most popular and benign networks exhibit up to 5% anomaly rates. For the readability purpose, the figures contain only 20 AS entries from all of the 1536 ASs observed during the experiment. Note that the complete results for the conducted measurements are presented in Table 1. Based on the obtained results, it can be observed that the dominant anomaly is TCP Options Reorder, which is prevalent in nearly all observed traffic. TTL changes are observed only with up to 15% of the traffic coming from examined ASs. Note that we were not able to verify whether TTL changes are caused by routing instability, load balancing mechanisms, or network traffic tampering. Unfortunately, based on the performed experiments, we were unable to analyze anomalies in the Application Layer. This is because the vast majority of the modern web servers

[1]https://github.com/pyppeteer/pyppeteer
[2]https://www.alexa.com/topsites

communicate using encrypted HTTPS traffic with the help of Transport Layer Security (TLS) protocol.

Based on the traffic analysis presented above, it is evident that the naive detection methods will be not effective as it will be difficult to differentiate between legitimate and covert traffic. Thus, a more sophisticated approach is needed. Note that we used the knowledge of the discovered anomalies to construct the network covert channels that we used during the performed experimental evaluation. We described the chosen data hiding techniques in the following subsection.

Observed anomalies [% of all pkts]	TTL changes	TCP Options reorder
Average value	0.789	4.685
Standard Deviation	4.131	6.029
Maximum value	71.154	40.000
Percentile		
5th	0.000	0.037
25th	0.000	0.618
50th	0.000	2.146
75th	0.000	6.698
80th	0.000	8.491
85th	0.073	10.496
90th	0.555	13.158
95th	3.522	16.667

Table 1: Distribution of the observed anomalies in network traffic from all observed ASNs (Alexa's top 10,000 websites) [10].

3.2 Network Covert Channels Functioning

During the research presented in this paper, we have developed and utilized three different steganographic methods functioning across three layers of the TCP/IP stack:

- TTL field modulation within IPv4 packet header (Network Layer),
- TCP Options reorder within TCP segment header (Transport Layer),
- HTTP Headers reorder within HTTP 1.1 requests (Application Layer).

The first method conceals secrets within the TTL field of the IPv4 header by encoding data symbols into deviations from the normal value of TTL field. This technique works as follows. When a new suitable network connection is established, the first few packets within the connection are left unmodified to establish a stable baseline value of TTL at the receiver's side. Then, the secret data is transmitted by incrementing or decrementing (i.e., modulating) the field's value using a predefined offset (i.e., symbol) which corresponds to a given data bit sequence. If there is no change to the TTL value, then it means that no secret data was transmitted. For example, the TTL value of IPv4 packets originating from a Linux host is set by default to 64. By passing through network routers, the TTL value is decreased by 1 for each network hop. At the covert receiver, the TTL value is observed with a smaller value (for example, 62) that should remain constant throughout the connection. If the TTL value changes unexpectedly from 62 to 60, then the symbol '-2' is transmitted, which can be mapped to a predefined data bit sequence.

Covert channel that relies on TCP Options reorder has been implemented by tampering with the order of 'Options' field within TCP segment's header. This field is an array which can carry variable values of optional parameters. Such an order is not strictly defined within the TCP standard aside from the terminating symbol 0, i.e., `END OF OPTIONS` and the array's size must be divisible by 32 bits. In our research, we have noticed that TCP segments usually contain a symbol 8 option, i.e., `TIMESTAMP` and multiple instances of symbol 1, i.e., `NO OPERATION` used to align the array length to 32-bit boundaries.

Because the order of options is ignored by the TCP/IP stack, it is possible to manipulate it to conceal secret information. In our implementation, we are swapping the position of the `TIMESTAMP` option with the first occurrence of the `NO OPERATION` to transmit a symbol or leave the order intact to send no data. This method is capable of sending only 1 symbol per TCP segment, so to transfer arbitrary binary data a specific signaling protocol is required. To transmit data, a two-step process is utilized. If there is no change to the options order, there is no hidden transmission. When the two options are swapped, we are signaling a control bit indicating that the next TCP segment will contain hidden data. If the following segment has also swapped the order of options, it means that bit '1' was transmitted, otherwise if no change is observed '0'.

Finally, HTTP header reorder covert channel utilizes HTTP protocol to transmit secret data by manipulating the order of headers contained within the request. The order of headers within the HTTP request is not strictly defined, thus it can be used to transfer covert data. The implemented technique works as follows. We assume that the repeated HTTP requests (sent by the same browser and with the same domain and URL) preserve the order of HTTP headers. If this is the case, then the first request is used by the covert communication parties to store the initial header order. Then, the difference in the header order in subsequent requests can be used to encode the covert data. Our algorithm uses index-based encoding. To transfer data, one of the headers is shifted to the beginning of the request. The index of this header in the original request is the symbol transferred (which corresponds to a group of bits). For example, we can send 2 bits of secret data per request by using four indexes: 1, 2, 3, 4 which map to 00, 01, 10, 11 secret data bits. If we consider, for example, the original HTTP request as:

```
    POST / HTTP/1.1
(0) Host: srv1-on.sdn
(1) User-Agent: python-requests/2.22.0
(2) Accept-Encoding: gzip, deflate
(3) Accept: */*
(4) Connection: keep-alive
(5) Content-Type: application/xml
(6) Content-Length: 406
```

and if we want to encode the symbol '2', the header with index 2 (Accept-Encoding) should be relocated to the beginning, resulting in a request:

```
    POST / HTTP/1.1
(0) Accept-Encoding: gzip, deflate
(1) Host: srv1-on.sdn
(2) User-Agent: python-requests/2.22.0
(3) Accept: */*
(4) Connection: keep-alive
(5) Content-Type: application/xml
(6) Content-Length: 406
```

3.3 Introduced error detection and correction scheme

In our previous work [10] the utilized covert transmission method used serial access to the bitstream of the linked input/output data file and the secret data bits were sent sequentially in the same order as they appeared in the data file. Such an approach was successful in cases where the covert transmission was not interrupted or altered. However, if a single secret bit was lost or erroneously inserted, then all secret data received later became shifted and unusable. This means that when the secret was transferred in the form of the ASCII content and only one bit was transmitted in the wrong order, then, in result, the whole extracted text at the receiving side was unreadable.

To summarize, the originally used serial method of transmitting secret data used in [10] has the following disadvantages:

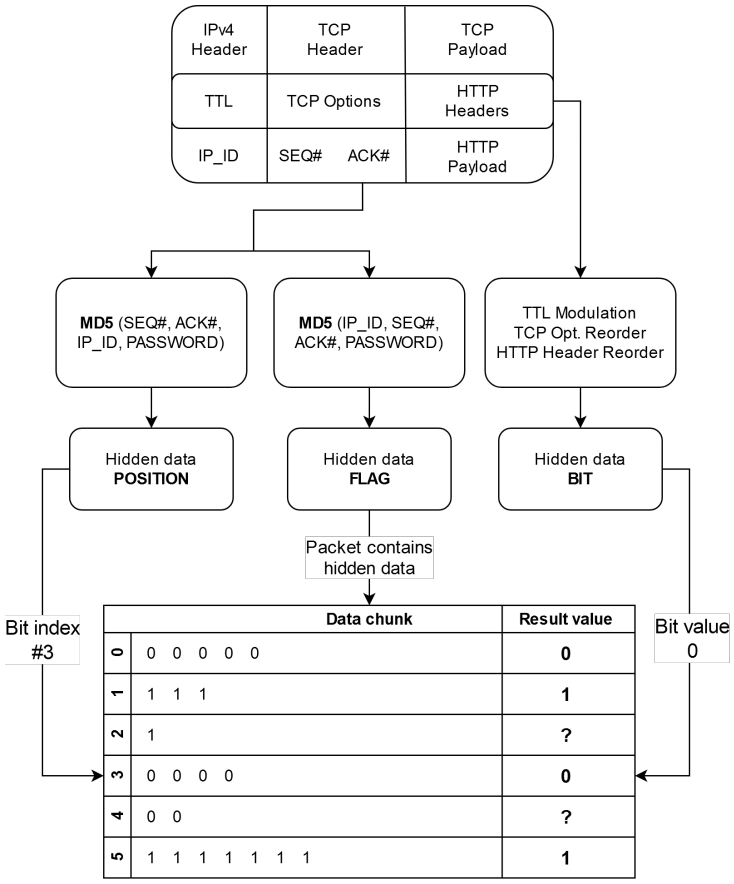


Figure 2: Simplified diagram of chunked data filtering, positioning, and data carriers. Selected IPv4 and TCP header fields are hashed with the MD5 algorithm to determine whether a packet should contain hidden data and the bit index. Modifications of the TTL value, TCP Options or HTTP Headers carry data bit.

- There was no indication of which packet contains steganographic data (it was achieved internally within the sender software), so the covert receiver had to closely monitor the whole incoming traffic for specific changes in packet fields,

- Due to the lack of synchronization and error correction mechanisms the resulting DNCC was highly sensitive to packet losses, reordering, or unexpected changes in the packet content which caused secret data corruptions. However, if a single secret bit was lost or erroneously inserted, the received secret data became altered,

- It required monitoring and modification of the order of options in all TCP packets to avoid data corruption in the TCP Options reorder module.

Considering the above, in this paper, we develop an upgraded transmission method which addresses all above-mentioned issues by introducing two improvements: *(i)* packet hashing to filter out packets that do not carry steganographic data and *(ii)* index data bit position within the data chunk regardless of the packet order sequence. Figure 2 visually explains the logic behind these improvements – both introduced mechanisms are described in detail below.

3.3.1 Mechanism for selecting packets to carry secret data

As already mentioned, in this mechanism we decided to use a hash-based solution. A group of fields (such as `IP ID`, `TCP ACK`, `TCP SYN`) is unlikely to be modified during the transmission. These values along with the chosen *salt* value are combined into a text string and then hashed using MD5 algorithm. The resulting hash can be then treated as a random value which allows to verify at the receiving side whether the packet carries the secret data bits or not. To have more control over the frequency of the secret data embedding, we evaluate if N lower bits of the hash, interpreted as a single integer, are less or equal to M. The N and M are selected to match as closely as possible the desired data frequency. For example, if $N = 3$, the possible values for the hash fragment are 0, 1, 2, 3, 4, 5, 6, 7. If $M = 5$, out of 8 possible values 6 will trigger packet modification, so the probability is $6/8 = 75\%$. In the experimental runs, N and M are set to match the desired packet modification probability (which is one of the considered parameters) as closely as possible. To determine if a packet contains secret data, the receiver has to follow the same procedure of generating the hash values as the covert sender.

3.3.2 Position-independent data transmission protocol

Instead of the raw serial transmission as used in [10], in this paper, the secret data is split into K-bit chunks which are transmitted sequentially (K is one of the tunable transmission parameters). The covert sender and the covert receiver spend a predefined amount of time on each chunk (a 'chunk time' interval) and then move on to the next one. The chunks themselves are transmitted using a technique similar to the hashmap implementation mentioned in the previous subsection. Just like for the packet selection mechanism, a group of fields from the packet are used to generate a hash value. The hash is used to select which bit of the chunk will be transmitted. The covert sender transmits bits until each bit is transmitted at least X times (1 is the minimum and a higher value can be set for increased reliability but at a cost of the lower 'usable' covert transmission speed). It should be also noted that, due to the randomness, some bits (i.e., positions in the current chunk) will be transmitted more often than the requested minimum. Thus, it is important to adjust the 'chunk time' interval (reflecting the throughput capability of the modules' configuration) to guarantee the transmission of the minimal acceptable number of bit repetitions for a chunk.

For modules which can transmit more than one secret bit per modified packet, the hash is used to place the first bit in the chunk normally, then it is used (again) as a seed for a pseudo-random number generator (a simple Linear Congruential Generator) to determine the positions for the rest of the bits. Similarly to the packet selection method described above, the covert receiver can calculate the bit positions by following the same algorithm as the covert sender. The value of each bit position is determined by the majority of the transmitted contents at this position (thus a simple error correction code is used). Note that the proposed algorithm is completely immune to packet reordering and offers adjustable protection against packet loss via duplication of the transmitted bits.

4 Frequent sets tree-based detection approach

In our previous work [10] we introduced a steganography detection method which is based on a specially constructed tree which stores the detected frequent sets for various values of *minimal support*. In the following sections, we briefly review the most important steps concerning raw data preprocessing, tree construction, and outlier finding. During the detailed presentation and discussion of the obtained results in section 6, we refer to this type detection strategy as 'Tree-based'.

4.1 Extraction of packet anomalies from traffic traces

The first step of the traffic analysis is performed directly on packet capture dumps, where each transmitted packet is simultaneously tracked as Layer 3, Layer 4, and Layer 7 connections. By grouping packets into individual multilayer connections, we can compare all subsequent pairs of neighboring packets within the connection. The values of all header fields are compared between packets and any differences are listed in a JSON log file. If the protocol specification allows a given field to have a variable list of values, then we compare the order of specified values to determine whether any value pairs were reordered. The resulting JSON file contains all discovered differences between packets within network connections. However, it must be noted that such an anomaly detection technique is viable only for the header fields that are relatively constant within network connections, e.g., TTL value in the IPv4 header. Values that are completely changed with each passing packet (for instance, TCP Sequence number) are out of the scope of this work. The resulting JSON file contains only these packets which exhibit unexpected changes in the header fields. Figure 3 graphically illustrates the preprocessing steps described in this section.

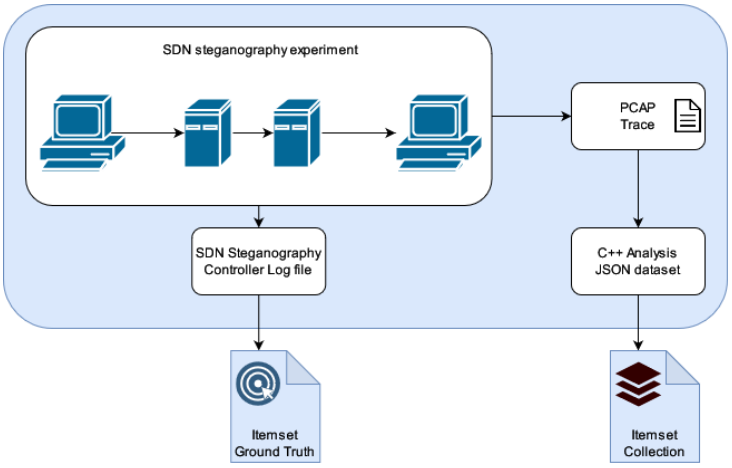


Figure 3: Shared pipeline scheme – from the logs of SDN controllers and network traces gathered during an experiment the item set collection and the associated ground truth (steganograhic/not steganographic traffic) are generated.

4.2 Frequent set-based detection of traffic anomalies

The obtained file contains only information concerning changes between consecutive packet pairs. Even for benign traffic, this file may contain a significant amount of data. On top of that, the obtained information cannot be processed directly by statistical techniques because the detected anomalies are context-specific and thus applying such means will remove any context from the results.

In our previous research, we utilized the frequent set mining technique to identify network connections with recurring anomalies (to which we refer as 'FIM' later in this paper). The discovery process takes as an input a JSON file obtained during the previously described step and uses the implementation from the PyFIM library[3]. Each detected header change is supplied with the layer-specific connection metadata as a single transaction in the dataset and the resulting frequent sets were analyzed. In our previous approach [2], we tried to apply the FIM algorithm directly as a detection method. The rationale

[3]http://www.borgelt.net/pyfim.html

behind such an approach was that with the proper parameters' tuning, the algorithm should return no results if there was no covert communication in the dataset, otherwise it should return one or more frequent item sets corresponding to the type of covert channels present in the dataset.

However, in this paper, we propose a different approach, where the FIM algorithm is executed multiple times with various *minimal support* parameters which control the minimum acceptable support of the discovered item sets. This changes the output significantly – with the higher *minimal support* discovered item sets are more general and correspond to a larger portion of the initial data (per item set). This 'cross-section' view is beneficial as some anomaly traits may not be evident at the lower level while obvious when using higher-level representation.

The obtained frequent sets from multiple runs of the PyFIM library are grouped by the *minimal support* parameter and stored in a JSON file. Exemplary detected frequent item sets are presented in Listing 1. Each detected item set contains the field names and values which provide metadata describing the network connection and the detected anomaly. As an example, set *A* describes the detected anomaly where the TCP Options order was changed unexpectedly within TCP connection from `192.168.1.30:36824` to `192.168.2.3:80`. In addition, the *support* field denotes the detected item set support.

```
A = {"IP_Addr_Dst": "192.168.2.3",
     "IP_Addr_Src": "192.168.1.30",
     "TCP_Port_Dst": 80, "TCP_Port_Src": 36824,
     "IHP:TCP_OPTIONS_REORDER": "TCP_OPTIONS_REORDER",
     "IP_Protocol": 6,
     "support": 1},
B = {"IP_Addr_Dst": "192.168.2.5",
     "IP_Addr_Src": "192.168.1.50",
     "TCP_Port_Dst": 80, "TCP_Port_Src": 48536,
     "IHP:TCP_OPTIONS_REORDER": "TCP_OPTIONS_REORDER",
     "IP_Protocol": 6,
     "support": 1},
C = {"IHP:HTTP_HEADER_REORDER": "HTTP_HEADER_REORDER",
     "IP_Addr_Dst": "192.168.2.3",
     "IP_Addr_Src": "192.168.1.30",
     "TCP_Port_Dst": 80, "TCP_Port_Src": 53796,
     "IP_Protocol": 6,
     "support": 1},
D = {"IP_Addr_Dst": "192.168.2.3",
     "IP_Addr_Src": "192.168.1.30",
     "TCP_Port_Dst": 80,
     "IHP:TCP_OPTIONS_REORDER": "TCP_OPTIONS_REORDER",
     "IP_Protocol": 6,
     "support": 305},
E = {"IP_Addr_Dst": "192.168.2.3",
     "IP_Addr_Src": "192.168.1.30",
     "TCP_Port_Dst": 80,
     "IP_Protocol": 6,
     "support": 1050},
```

Listing 1: Exemplary detected frequent item sets [10].

4.3 Grouping the detected frequent sets into a hierarchical tree

Because with the increasing *minimal support* parameter sets discovered by the PyFIM library tend to be less specific, it is reasonable to organize them into hierarchical tree with the least specific set as the root node of the tree and the most specific nodes as leaves. By organizing sets into a hierarchical tree, an additional relationship context is introduced into the dataset, which helps with data analysis. Figure 4 showcases all steps performed during frequent item sets discovery and tree construction.

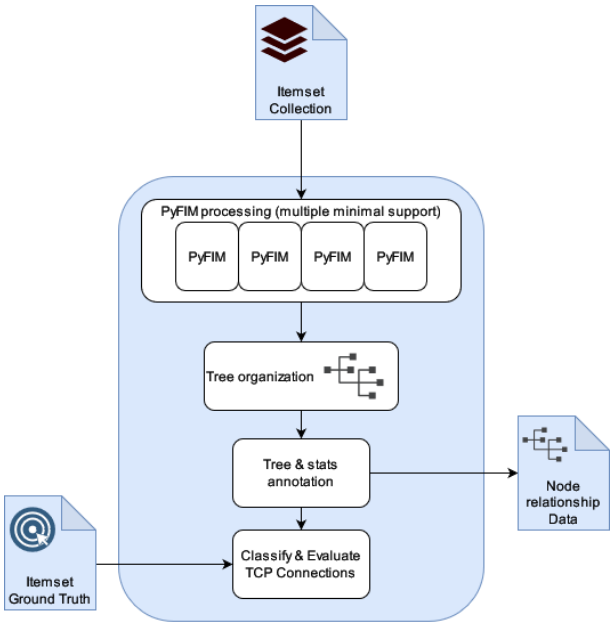


Figure 4: Tree-based approach pipeline: the individual item sets are grouped by the Frequent Itemsets algorithm and this grouping is performed in multiple attempts with different minimal support parameter.

To build a hierarchical tree, it is required to introduce relationship operators to compare frequent sets. The tree nodes are organized into parent-child node relationships when the following criteria are met:

- The parent node has a greater *support* value,
- The parent node key set is a subset of the child key set,
- The overlapping key values are equal in both parent and child (excluding *support* value).

The hierarchical tree building algorithm is executed using top-down approach from the highest *minimal support* and all discovered frequent item sets are attempted to fit into the tree. If a frequent set does not match the criteria for any node in the tree, it is withheld for later iterations. If all possibilities of fitting withheld nodes are exceeded, the node is attached to a new tree root.

By following the criteria and the algorithm provided above, the exemplary sets A, B, C, D, and E presented in the Listing 1 can be organized into the tree structure presented in Figure 5.

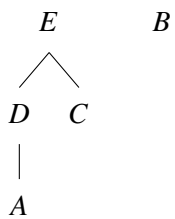


Figure 5: Tree structure constructed from the exemplary frequent sets [10].

The most important benefit of organizing the frequent item sets into a hierarchical structure is the possibility to quickly summarize the type of traffic by briefly examining root nodes, and when further details are required, the tree can be followed to more specific nodes.

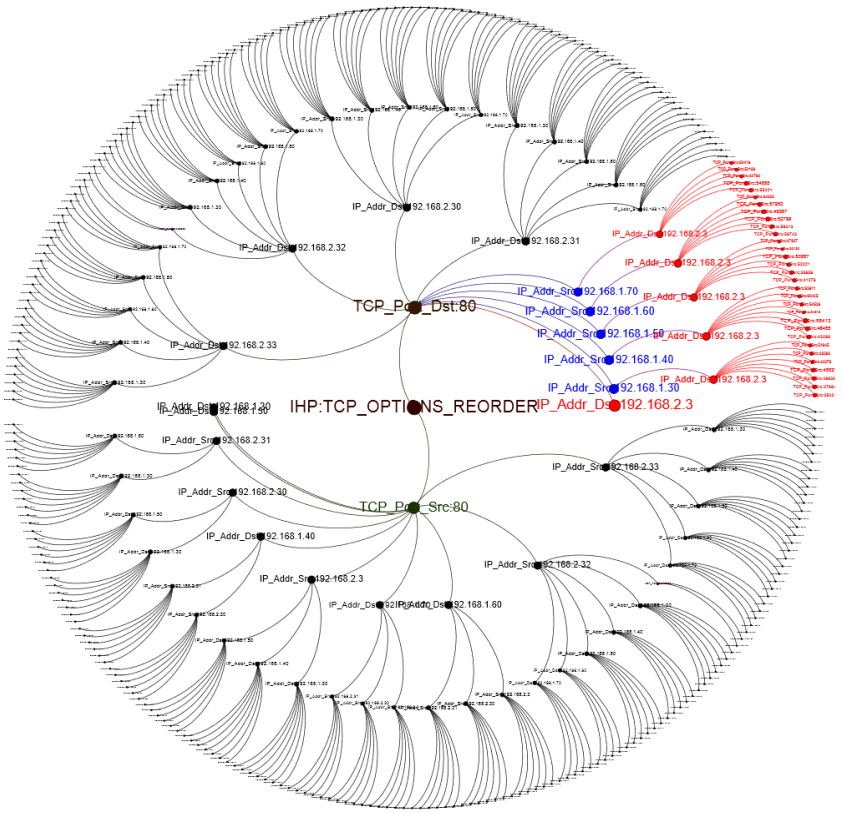


Figure 6: Exemplary hierarchical tree with classified nodes for the scenario with the DNCC using TCP Options Reorder covert channel between 5 clients and 1 server (red color denotes nodes classified as *outliers*, blue color denotes *compound outlier* nodes) [10].

4.4 Finding outliers within the tree structure

Independently of the tree building, the frequent item sets can be sorted into collections based on the types of features inside them. For example, all frequent sets which contain items associated with the source IP address, destination IP address, source port, and destination port can be treated as one collection, regardless of the actual values of these fields. As an example, both sets A and B from the previous subsection belong to the same collection because they feature the same keys, despite their values being different.

To find the outliers, firstly the median of frequent sets supports for each collection is calculated. Then, a frequent set can be classified as an outlier based on the deviation of the discovered frequent set support from the collection's median (with an adjustable threshold). More precisely, a frequent set is considered normal if it satisfies the following equation:

$$1.0 - T < \frac{S}{M} \leq 1.0 + T$$

where S is the frequent set support, M is the median support for the collection, and T is the adjustable threshold. Otherwise, it is considered as an outlier.

The introduced method generally marks frequent sets of high specificity. This is understandable, as naturally the number of frequent sets decreases if the specificity decreases – there is just not enough data to generate statistics for their proper classification. Thus, a second step of the algorithm is needed to mark

less specific frequent sets and it works as follows. After the initial outlier marking, each node of the tree is recursively checked if it can become a *compound outlier*. A node is considered as *compound outlier* if more than *X%* (a configurable parameter) of its children are outliers, either normal or compound. This action marks additional nodes closer to the root of the tree as outliers which were missed during the previous step.

Finally, the steganographic classification can be performed. For each leaf (which represents the most specific frequent sets describing TCP connections), we recursively go up through the tree, looking for a parent node which was marked as either a compound or a normal outlier. If no such node was found when the root was reached, the leaf (the TCP connection) is considered as negative, i.e., no covert communication was found. Otherwise, as soon as such a node is discovered, the leaf is marked as positive, i.e., steganographic modifications were present and the classification process for this leaf ends.

The result of such a detection algorithm is a hierarchical tree with annotated outlier nodes. Each node within the tree contains an instance of frequent set alongside the support and metadata regarding the set collection it belongs to. The tree can be traversed top-down to list nodes classified as steganographic — each frequent set node describes a portion of the network traffic.

An exemplary tree has been depicted in Figure 6. In this scenario, the DNCC is using TCP Options Reorder data hiding method between 5 clients and 1 server. Red color denotes nodes classified as *outliers*, blue color denotes *compound outliers*. By following the tree structure from the root node, it is evident that the traffic has been grouped with the leading TCP Options Reorder the root node, which at the next level was evenly divided by TCP source and destination ports – HTTP requests (Destination port = 80) and responses (Source port = 80). At the lower level, the destination port was divided into source and destination addresses. Here nodes related to IPv4 source addresses 192.168.1.30, 192.168.1.40, 192.168.1.50, 192.168.1.60, 192.168.1.70 were classified as *compound outliers* and further down, destination address 192.168.2.3 and individual TCP connections were marked as *outliers*. This shows that the proposed classification approach was able to perfectly identify the DNCC scheme utilized in this scenario.

5 Experimental methodology and testbed

During the conducted experiments, we want to evaluate and compare the performances of various steganography detection approaches proposed in this paper. In more detail, we present our results, findings, and conclusions concerning three detection strategies:

- **Tree-based**: this is our previous strategy originally proposed in [10] which uses only the tree of item sets (Figure 4),
- **Basic ML**: it is a Machine Learning approach which relies only on the preprocessed data of the network traffic (Figure 7),
- **Hybrid**: this is also a Machine Learning approach, however, in this case it operates on the preprocessed data and additionally on the information obtained from the tree of item sets (Figure 7) – so it combines both detection methods mentioned previously above.

Note that the two detection methods which utilize ML techniques (i.e., 'ML Basic' and 'Hybrid') use the trained Machine Learning model obtained via the Driverless AI (DAI)[4]. DAI is an automatic machine learning platform which can handle automatic feature engineering, model selection, tuning, validation, and deployment. We have decided to use it during the experimental evaluation as it has been previously successfully deployed in a similar cybersecurity scenario [4].

[4]https://www.h2o.ai/products/h2o-driverless-ai

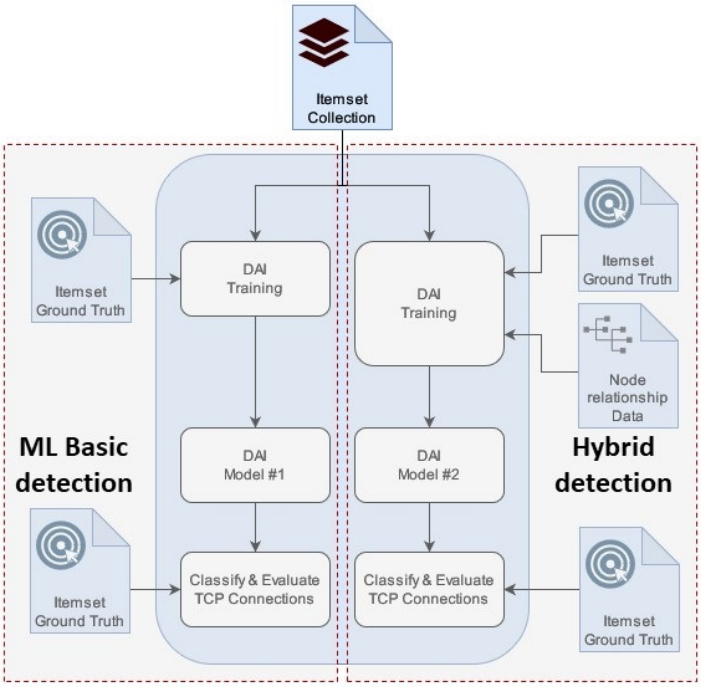


Figure 7: CSV pipeline scheme – two AI models are trained on the data: the first one only with the information from the pcap item set ('ML Basic' approach), the second one with the information from the pcap item set and from the tree node relationship ('Hybrid' method).

All three above-mentioned detection strategies are evaluated using raw network traffic as an input, which simulates various types of DNCC channels in the IoT environment consisting of temperature sensors. During the conducted experiments, we compare decisions of the detection algorithms (i.e., whether the traffic is considered as containing or not steganographic data embedded) – with the ground truth taken from the configuration of the recorded network traffic. The ground truth contains the information which connections, in this particular test run had the steganographic transmission enabled. In effect, for each of the three strategies mentioned above, we are able to calculate true positives and false positives. Using these results, we can prepare ROC (Receiver Operating Characteristic) curves and calculate AUC (Area Under Curve) parameter used in the final comparison of detection methods.

Experimental traffic was generated during our research in the testbed, which is described in detail in subsection 5.1. During the conducted experiments, we introduce various network anomalies, which are described in section 5.2. This part of the experiment evaluates how well our coding scheme (see section 3.3.2) performs as well as what impact on the detection rate such network conditions have. Moreover, during the conducted experiments we examine various configurations of the DNCC, which in effect produce a test set containing more than 7600 test runs. All details concerning the generated test set which is used later in the evaluation process are enclosed in subsection 5.3.

5.1 Test-bed

Our experiment scenario consists of two separated LANs (Figure 8). The first one consists of machines that mimic IoT devices and the second one which contains a central HTTP server. This emulates the real-world setup, where such LANs would be connected via a WAN or Internet. Due to the fact that our research mainly focuses on DNCC and not on the detailed analysis of the particular data hiding method

used for its creation, all experiments are performed in the virtual environment. Moreover, for the sake of simplicity of the utilized topology, the default routers of both LANs are connected directly by a virtual link. In such scenario, the clients in the first network have data that needs to be secretly transmitted to the second network using data hiding techniques. In this testbed, covert transmission is performed by a custom SDN application running on a SDN controller host. During our experiments, we change the number of clients which represent IoT devices, sending data to one or multiple central servers.

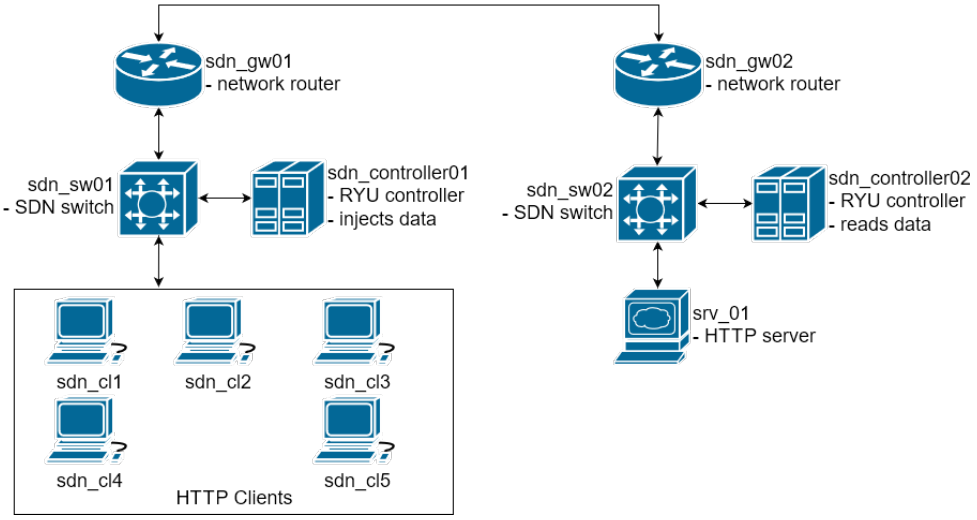


Figure 8: Utilized experimental test-bed [10].

Gateway machines are connected to the Internet, which allows communication between LANs. The IoT sensors periodically transmit measurements to the server machine, which create a flow of packets that passes through both SDN switches, where the covert messages are injected (at the first SDN switch) and extracted (at the second SDN switch).

In our testbed we used the following software:

- All Virtual Machines are running Centos 7.5.1804 (Core),
- Open vSwitch v 2.10.1 on the SDN Switches,
- RYU Python SDN controller,
- Apache HTTPD v 2.4.6 on the HTTP Server machine.

Devices `sdn_cl1`, `sdn_cl2`, `sdn_cl3`, `sdn_cl4`, `sdn_cl5` act as the smart temperature sensor nodes. A Python script is used to push sensor readings via HTTP `POST` messages to the central HTTP server running on `sdn_srv01`. Machine `sdn_sw01` runs Open vSwitch network switch which is managed by the `sdn_controller01` machine running RYU Controller and the device `sdn_gw01` acts as a router for this network. A similar configuration is used on the receiving side of the test-bed, with `sdn_sw02` running another Open vSwitch controlled by RYU running on `sdn_controller02` machine. Machine `sdn_gw02` acts as a router for the receiver network.

As the ML-based platform, as already mentioned, we chose *Driveless AI dai*-1.8.0 which is an automatic Machine Learning platform and *CUDA 10.2*, i.e., the parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs)[5]. One

[5]https://developer.nvidia.com/cuda-zone

of the most important features of DAI we utilize is the auto-tuning of the ML algorithms, which allows a focus on the main problem, i.e., the covert transmission detection and not on the tuning of ML parameters.

5.2 Introducing anomalies to the network traffic

In modern computer networks, network anomalies occur sporadically and are typically introduced by differences in network stack implementations, network routing issues, or undefined behavior of network protocols. In general, network anomalies are introduced along the routing path and longer paths are more likely to contain anomalies. As already mentioned, the testbed utilized in this research consists of two Local Area Networks directly connected to each other through a short routing path. This makes the traffic exchanged using the above-mentioned testbed very unlikely to contain any sporadic network anomalies (which, as stated above, occur typically in communication networks). On the other hand, the steganographic methods utilized in this testbed intentionally introduce network anomalies as a data carrier for covert channel transmission. In result, the vast majority of anomalies within the test-bed traffic are introduced by steganography, thus the classification process could be greatly simplified.

To make the classification process nontrivial, we have decided to introduce benign network anomalies into our testbed to replicate the anomaly distribution typical to what we discovered in WAN traffic. The introduction of benign anomalies makes the steganographic data transmission more prone to data corruption and in more aggressive cases, the covert transmission is impossible to continue due to the large amount of errors. Thus, to make the steganographic data transmission more resilient to data corruption, we developed a dedicated error detection and correction scheme which was described in detail in section 3.3. With the utilization of such mechanism, we are able to introduce an arbitrary number of benign network anomalies while maintaining high steganographic transmission accuracy. In effect, the covert channel detection algorithms can be validated in our testbed in an environment close to the real-world case.

To simulate various network anomalies, we utilize existing steganography modules to introduce random data corruption in the covert transmission layer, namely, random data bits:

- **insertion**: non-steganographic packets are randomly modified during transmission to introduce a random steganographic value that can be read by the receiver,
- **deletion**: steganographic packets are randomly deleted during transmission to simulate data loss,
- **flipping**: steganographic packets are randomly modified to invert the data bits carried in the message.

The chosen data corruption methods make it difficult to synchronize the covert data stream, thus classic error correction codes cannot be directly applied to the transmission, as the stream can be shifted and the meaning of the individual data and parity bits may be lost after random bit insertion, deletion, or flipping. Considering the above, to ensure the correct data transmission, we have designed and implemented solutions to achieve reliable covert data transmission in a channel with high network anomaly noise. In more detail, we use: *(i)* false network anomaly filtering, *(ii)* data bit indexing within continuous network stream, and *(iii)* error correction through redundancy. These solutions have been explained in detail in subsection 3.3.

5.3 Test set

During preparation of the test set, we have evaluated different DNCC configurations as well as various conditions of the network by introducing traffic anomalies which are similar to these observed in real-world communication networks (see section 5.2). As a result, we have produced a test set which contains

7657 test runs. The following summarizes the details of the test set configuration. Each experiment lasted 60 seconds and during experiments the simulated data corruption was set to 2, 5, 6.6, 10, and 20%, respectively. Moreover, the experiments have been recorded with 1, 2, or 3 steganographic modules enabled and they were instructed to modify roughly every second (50% transmission rates) or fifth (20% transmission rate) packet.

The experimental scenario involved client-server communication 1-to-1, 1-to-5, 5-to-1, and 5-to-5. In 5-to-1 and 5-to-5 scenarios, the number of actual steganographic clients can be 1, 2, or 5 (i.e., there could be 4, 3 or 0 clients which never had their packets modified). Similarly, in the 1-to-5 and 5-to-5 scenarios, the number of steganographic servers could be 1, 2, or 5 (i.e., there could be 4, 3 or 0 servers which never received modified packets). Only the network traffic from a steganographic client to a steganographic server is considered by the algorithm for packet modification. Moreover, clients were generating network traffic to the servers via HTTP requests sent with varying frequency, i.e., every 1, 5, or 10 seconds. Finally, some scenarios intentionally had the steganography disabled (the number of covert communication clients equals 0).

6 Results

During the first part of the conducted research, we measure the overall accuracy of each of the detection strategies, i.e., without differentiation of any specific type of DNCC traffic. Figure 9 presents ROC curves for the various variants of the Tree-based detection approach. During the performed experiments, we explore how the results change for various ratios of suspicious children in the tree necessary to mark a parent node as suspicious and, thus, in effect decide that the steganographic traffic appears in this test run. In Figure 9 we present the results when this parameter changes in the range from 0.1 to 0.9. As it can be seen, practically for all values, the resulting AUC is greater than 0.8 and the best results are achieved for 0.5 (AUC=0.845).

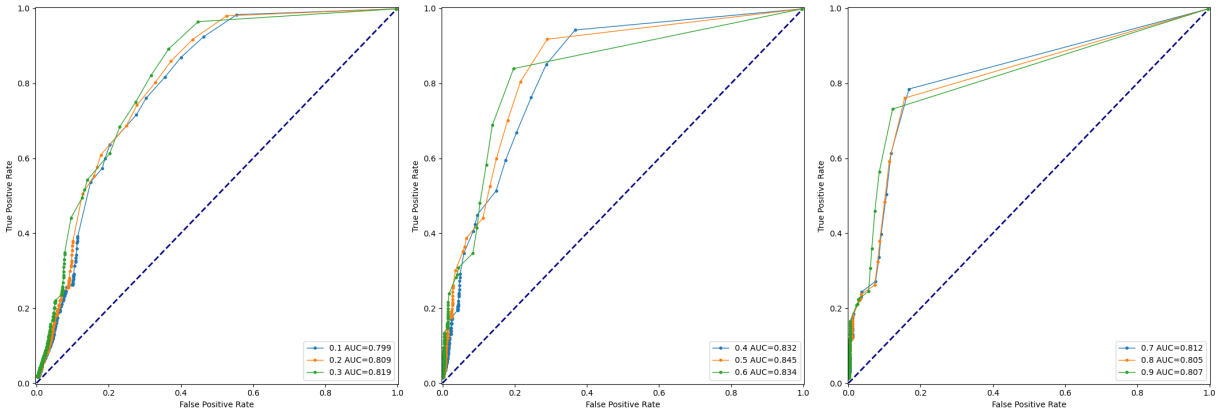
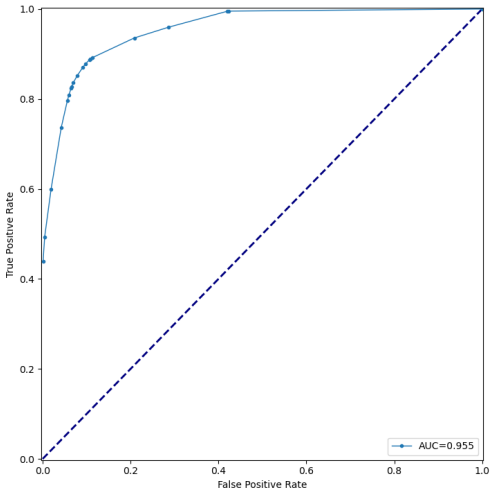
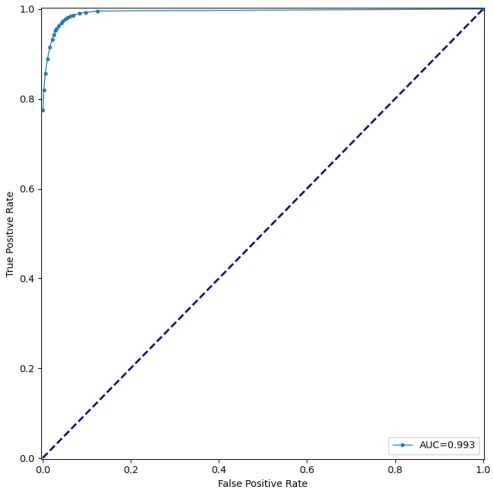


Figure 9: ROC plots for the Tree-based detection method for various ratios of suspicious children in the tree.

Next, Figure 10 illustrates ROC curves for detection strategies which utilize Machine Learning implemented in the DAI software. The subfigure (a) presents detection results for the 'Basic ML' approach where only preprocessed data are used for training and classification. On the other hand, the subfigure (b) showcases results of the 'Hybrid' method when the ML algorithm utilizes data enhanced with meta-data derived from the Tree-based solution (thus it can be considered as a combined concept). In both ML-based approaches (i.e., 'Basic ML' and 'Hybrid') the DAI software automatically tests various classification algorithms from the presented list: XGBoost GBM, LightGBM Random Forest, XGBoost



(a) 'ML Basic' approach utilizing only the pre-processed data.



(b) 'Hybrid' method using pre-processed data enhanced by the information from the tree of item-sets.

Figure 10: ROC plots presenting detection results for the ML-based detection strategies.

Dart, GLM, RuleFit, LightGBM, and FRTL. Results from the reports generated via DAI experiments show that for the final evaluation XGBoost GBM model has been chosen as the best performing one [3]. The measured performances of these models on the training data set are presented in Tables 2 and 3.

Table 2: Detection results for the 'ML Basic' method.

Scorer	Final ensemble scores on validation	Final ensemble standard deviation on validation	Final test scores	Final test standard deviation
ACCURACY	0.6828	0.0014	0.6833	0.0026
AUC	0.9399	0.0005	0.9404	0.0009
F1	0.6828	0.0014	0.6833	0.0026
F2	0.6828	0.0014	0.6833	0.0026
GINI	0.8799	0.0011	0.8809	0.0019
MCC	0.6375	0.0016	0.6381	0.0029

Table 3: Detection results for the 'Hybrid' approach.

Scorer	Final ensemble scores on validation	Final ensemble standard deviation on validation	Final test scores	Final test standard deviation
ACCURACY	0.7681	0.0018	0.7739	0.0018
AUC	0.9741	0.0003	0.9754	0.0003
F1	0.7689	0.0018	0.7739	0.0018
F2	0.7689	0.0018	0.7739	0.0018
GINI	0.9481	0.0007	0.9508	0.0007
MCC	0.7349	0.0021	0.7416	0.0021

The results presented above prove that the best detection method is the 'Hybrid' approach. This variant achieved AUC=0.993 and it outperformed 'Basic ML' scenario (AUC=0.955) and the previously proposed Tree-based scheme (AUC=0.845).

The next investigated issue concerns the detection accuracy for various steganographic transmission rates. Figure 11 showcases the accuracy results depending on the number of steganographically modified

packets. In this figure, we present data gathered from all conducted experiments (background plot). Because the most interesting features are presented for lower values on the X-axis, in the small (foreground plot), we demonstrate also the magnified part of the plot for this area. The same approach will be used for other plots presented in this section as well. Due to the fact that during the presented research we utilized an additional error detection and correction scheme (see section 3.3 for a detailed description) on the X axis we indicate the number of modified packets and not the effective steganographic data rate. However, these two features are correlated as the more network packets contain secret data the higher resulting overall covert channel bandwidth.

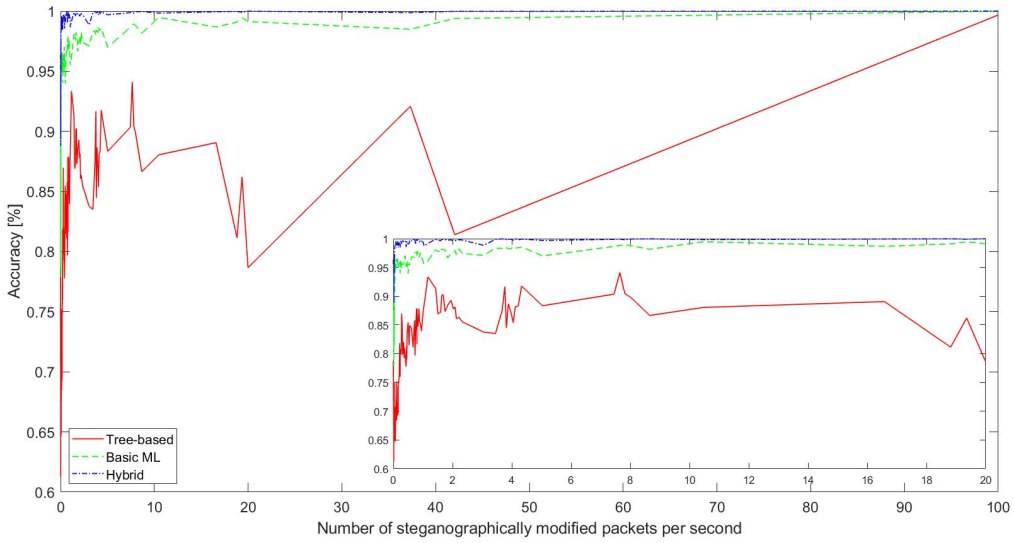


Figure 11: Accuracy of the detection depending on the number of the steganographically modified packets.

The presented results clearly show that higher steganographic rates are detected almost in all cases. However, it must be noted that reducing the steganographic rate allows the steganographer to 'stay under the radar' and remain undetected. Like in the first part of the conducted experiments, the best results have been obtained for the 'Hybrid' detection method.

The red solid curve in Figure 11 represents the results of the Tree-based detection method. We can observe that this approach has some instabilities which are associated with the detection of new frequent sets and reorganization of the tree. However, what is interesting to note is that such behavior is not transferred when using the same data for the 'Hybrid' detection strategy.

The last investigated issue concerns the detection accuracy for various numbers of steganographic clients depending on the number of modified packets. Figures 12-14 demonstrate the accuracy of different detection strategies for the varied number of steganographic clients (1, 2, or 5). In the next three plots, we present the results separately for each detection method, i.e., 'Tree-based', 'Basic ML', and 'Hybrid'. In the first plot, due to the high instability of the introduced Tree-based method, we cannot easily determine if this detection strategy performs better for a smaller or greater number of steganographic clients. However, as our previous research shows, the easiest to detect should be the DNCC configuration only with one client. In fact, such a situation is illustrated in Figure 12, only in the range from around 5 to 20 packet modifications per second, where this method achieves the best detection accuracy.

For the remaining two approaches, i.e., 'Basic ML' and 'Hybrid' results are much more stable and it can be easily observed that the resulting detection accuracy for five clients (blue curve in all three

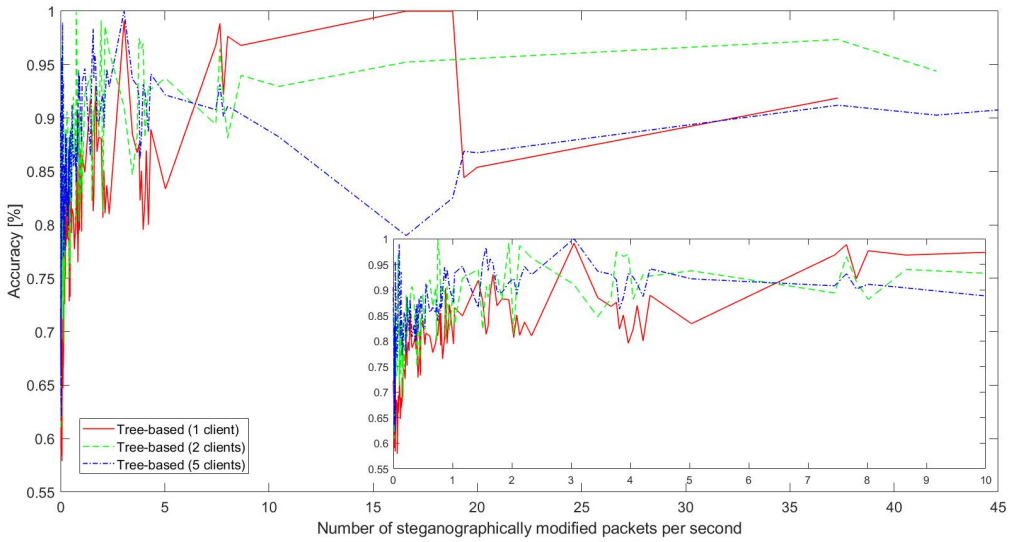


Figure 12: Accuracy of the detection depending on the number of the steganographically modified packets ('Tree-based' method).

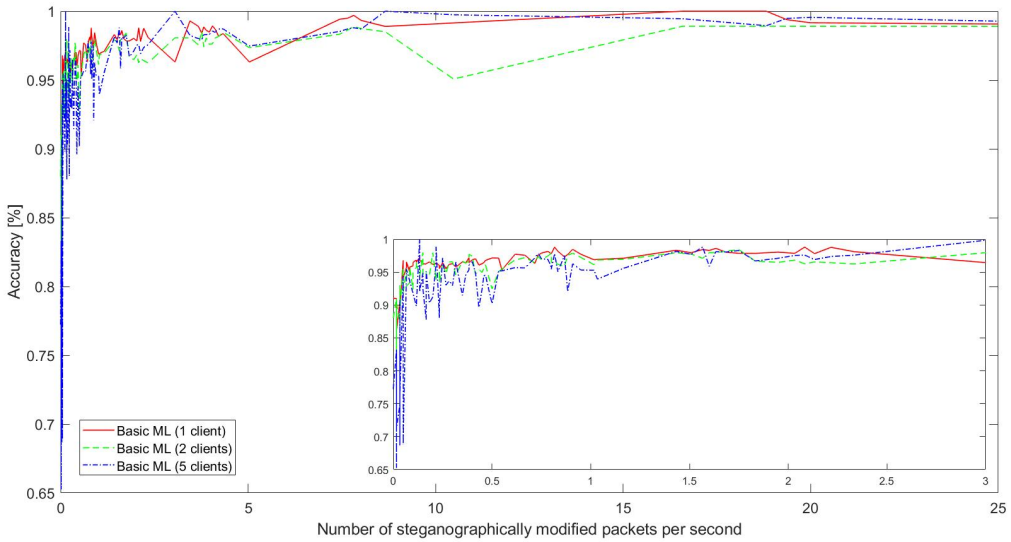


Figure 13: Accuracy of the detection depending on the number of the steganographically modified packets ('Basic ML' method).

plots) is the worst. As it was discussed earlier in this section, despite the high instability of the Tree-based method, the 'Hybrid' approach is one with almost invisible instability. These results confirm our previous research predictions that the utilization of the DNCC with a higher number of steganographic clients is harder to detect.

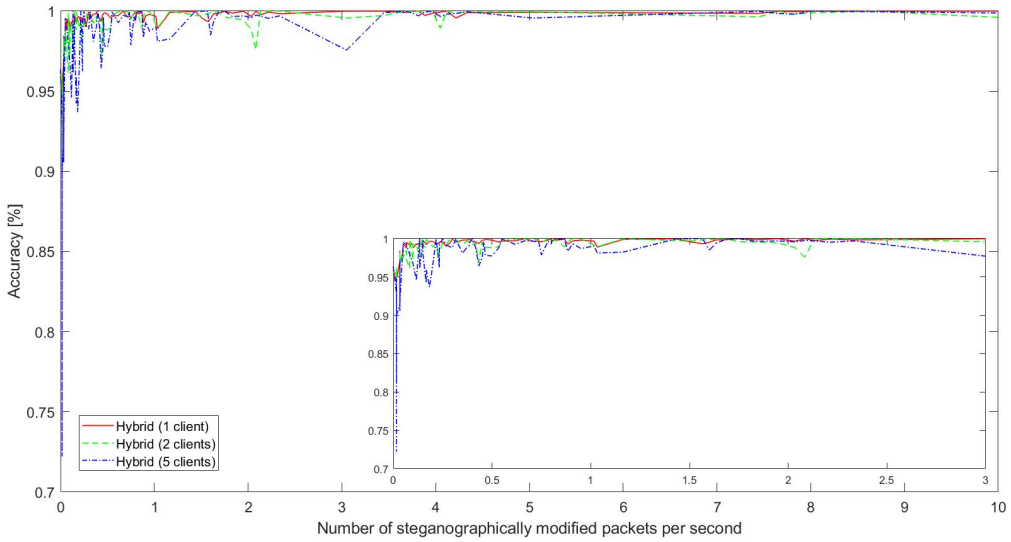


Figure 14: Accuracy of the detection depending on the number of the steganographically modified packets ('Hybrid' method).

7 Conclusions

In this paper, we present our extended research concerning the detection of steganographic transmissions. The proposed method can be used for the detection of simple covert communication attempts, i.e., where only one data hiding method in one network stream is used, as well as for more stealthy solutions, e.g., DNCC channels. To improve the detection accuracy after preprocessing phase data mining frequent sets are mined multiple times with various values of the *minimal support* parameters. Using the obtained results, a special tree structure which contains the discovered frequent item sets is constructed. In this paper, we introduced and described the detection method which is based on the analysis of such item sets tree. During the conducted experiments, we evaluated the proposed approach using steganographic traffic which was generated in the simulated IoT environment. In more detail, we proposed three detection strategies – one as described above and two others which utilize machine learning algorithms implemented in the DAI software. During these experiments, we compared ML algorithms – one which uses only preprocessed data and the second one which utilizes preprocessed data enhanced with the metadata obtained from the previously introduced tree-based method. Obtained research results proved that both ML-based approaches outperformed the original solution. However, it must be noted that the addition of the metadata generated by our original approach increases the resulting detection rate of ML-based detection. The measured AUC parameter for such a combined ('Hybrid') approach increased from 0.845 to 0.993. This results in a rise in performance by almost 15%. Our future work involves a more extensive study of both ML-based and hybrid solutions for steganographic transmission detection. First of all, we plan to investigate the accuracy and performance of other classes of ML algorithms. Secondly, we would like to utilize other data mining pattern discovery algorithms, which could produce more meta-data further used by ML solutions.

Acknowledgments

This material is based upon the work supported by the Air Force Office of Scientific Research under the award number FA9550-17-1-0254. The supported project is named CoCoDe – Covert Communication Detection. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

References

[1] K. Cabaj, L. Caviglione, W. Mazurczyk, S. Wendzel, A. Woodward, and S. Zander. The new threats of information hiding: The road ahead. *IT Professional*, 20(3):31–39, June 2018.

[2] K. Cabaj, W. Mazurczyk, P. Nowakowski, and P. Żórawski. Towards distributed network covert channels detection using data mining-based approach. In *Proc. of the 13th International Conference on Availability, Reliability and Security (ARES'18), Hamburg, Germany*, pages 12:1–12:10. ACM, August 2018.

[3] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16), San Francisco, California, USA*, page 785–794. ACM, August 2016.

[4] M. Gregorczyk, P. Żórawski, P. Nowakowski, K. Cabaj, and W. Mazurczyk. Sniffing detection based on network traffic probing and machine learning. *IEEE Access*, 8:149255–149269, August 2020.

[5] W. Mazurczyk and S. Wendzel. Information hiding: Challenges for forensic experts. *Communication of the ACM*, 61(1):86—94, December 2017.

[6] W. Mazurczyk, S. Wendzel, and K. Cabaj. Towards deriving insights into data hiding methods using pattern-based approach. In *Proc. of the 13th International Conference on Availability, Reliability and Security (ARES'18), Hamburg, Germany*, pages 1–10. ACM, August 2018.

[7] W. Mazurczyk, S. Wendzel, M. Chourib, and J. Keller. Countering adaptive network covert communication with dynamic wardens. *Future Generation Computer Systems*, 94:712–725, November 2019.

[8] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, and K. Szczypiorski. *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. John Wiley & Sons, February 2016.

[9] A. Mileva and B. Panajotov. Covert channels in tcp/ip protocol stack - extended version. *Central European Journal of Computer Science*, 4(2):45–66, June 2014.

[10] P. Nowakowski, P. Zórawski, K. Cabaj, and W. Mazurczyk. Network covert channels detection using data mining and hierarchical organisation of frequent sets: An initial study. In *Proc. of the 15th International Conference on Availability, Reliability and Security (ARES'20), Virtual Event, Ireland*, pages 1–10. ACM, August 2020.

[11] P. L. Shrestha, M. Hempel, F. Rezaei, and H. Sharif. A support vector machine-based framework for detection of covert timing channels. *IEEE Transactions on Dependable and Secure Computing*, 13(2):274–283, April 2016.

[12] Y. Sun, L. Zhang, and C. Zhao. A study of network covert channel detection based on deep learning. In *Proc. of the 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC'18), Xi'an , China*, pages 637–641. IEEE, May 2018.

[13] E. Tumoian and M. Anikeev. Network based detection of passive covert channels in tcp/ip. In *Proc. of the 2005 IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05), Sydney, NSW, Australia*, pages 802–809. IEEE, November 2005.

[14] S. Wendzel, S. Zander, B. Fechner, and C. Herdin. Pattern-based survey and categorization of network covert channel techniques. *ACM Computing Surveys*, 47(3):1–26, April 2015.

[15] I. You and K. Yim. Malware obfuscation techniques: A brief survey. In *Proc. of the 2010 International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA'10), Fukuoka, Japan*, pages 297–300. IEEE, November 2010.

[16] Q. Yuwen, S. Huaju, S. Chao, W. Xi, and L. Linjie. Network covert channel detection with cluster based on hierarchy and density. *Procedia Engineering*, 29:4175–4180, January 2012.

[17] S. Zander, G. Armitage, and P. Branch. An empirical evaluation of ip time to live covert channels. In *Proc. of the 2007 15th IEEE International Conference on Networks (ICON'07), Adelaide, South Australia, Australia*, pages 42–47. IEEE, November 2007.

[18] S. Zander, G. Armitage, and P. Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys & Tutorials*, 9(3):44–57, September 2007.

[19] J. Zhai, G. Liu, and Y. Dai. A covert channel detection algorithm based on tcp markov model. In *Proc. of the 2010 International Conferrence on Multimedia Information Networking and Security (MINES'10), Nanjing, China*, pages 893–897. IEEE, November 2010.

Author Biography

Piotr Nowakowski received his B.Sc. in electronics (2016) and M.Sc. in computer science (2018) from the Faculty of Electronics and Information Technology at Warsaw University of Technology (WUT), where he currently pursues a PhD degree. His research interests include: network security, reverse engineering, parallel processing and computer graphics.

Piotr Żórawski received the B.Sc. and M.Sc. degrees in computer science from the Faculty of Electronics and Information Technology, Warsaw University of Technology (WUT), in 2016 and 2019, respectively, where he is currently working as a teaching assistant. His research interests include computer and network security, dynamic malware analysis, and reverse engineering. He took part in projects for EU and U.S. Air Force.

Krzysztof Cabaj holds M.Sc (2004), Ph.D. (2009) and D.Sc. (habilitation) (2019) in computer science from Faculty of Electronics and Information Technology, Warsaw University of Technology (WUT). He is currently hired as University Professor at Institute of Computer Science, WUT. Former instructor of Cisco certificated Academy courses: CCNA Routing & Switching, CCNA Security and CCNP at International Telecommunication Union Internet Training Centre (ITU-ITC). His research interests include: network security, honeypots, dynamic malware analysis, data-mining techniques, IoT and Industrial Control Systems security. He is author or co-author of over 70 publications, and supervisor of more than twenty five M.Sc. and B.Sc. degree theses in the field of information security. He took part in over a dozen research projects, among others for EU, ESA, Samsung, US Army and US Air Force. Co-leader of Computer Systems Security Group at Institute of Computer Science.

Wojciech Mazurczyk received the B.Sc., M.Sc., Ph.D. (Hons.), and D.Sc. (habilitation) degrees in telecommunications from the Warsaw University of Technology (WUT), Warsaw, Poland, in 2003, 2004, 2009, and 2014, respectively. He is currently a University Professor with the Institute of Computer Science at WUT and a head of the Computer Systems Security Group. He also works as a Researcher at the Parallelism and VLSI Group at Faculty of Mathematics and Computer Science at FernUniversitaet, Germany. His research interests include bio-inspired cybersecurity and networking, information hiding, and network security. He is involved in the technical program committee of many international conferences and also serves as a reviewer for major international magazines and journals. From 2016 he is Editor-in-Chief of an open access Journal of Cyber Security and Mobility, and from 2018 he is serving as an Associate Editor of the IEEE Transactions on Information Forensics and Security. He is also a Senior Member of IEEE.

5 Conclusions

During the CoCoDe project, we have conducted research related to the distributed network covert channels and we investigated their nature from different perspectives. First of all, we developed the classification of DNCCs and analyzed theoretically their performance features. Then, using the developed testbed, we were able to create several different networking scenarios from a common IP-based network to one IoT-specific environment. This allowed us to perform extensive experimental evaluations of different configurations of DNCCs to investigate their characteristics and, moreover, to determine how undetectable they really are. In particular, we analyzed how the tuning of the parameters used by each single steganographic method forming DNCC could lead to the more stealth DNCC, which is in turn makes it much harder to be disclosed. On the other hand, we have also developed several detection methods that are based on data mining as well as machine learning. The obtained results prove that when used naively, DNCC can be quite easily detected, however, when the parameters of DNCC are conveniently depicted and proper data hiding methods are used, then covert communication is limited but still possible. This urges for more research efforts in this area.

The most important results from the conducted experiments are covered in the six reviewed research papers: four of them were published at the ARES conference while the remaining two in the prestigious scientific journals. We have also prepared and shared a dataset of network traffic containing captures of DNCC, which can be further used by the security community for further research purposes.