Journal of Universal Computer Science, vol. 24, no. 5 (2018), 556-576
submitted: 23/1/18, accepted: 28/4/18, appeared: 28/5/18 © *J.UCS*

WoDiCoF – A Testbed for the Evaluation of (Parallel) Covert Channel Detection Algorithms

Ralf Keidel
(Center for Technology and Transfer (ZTT)
Worms University of Applied Sciences, Worms, Germany
keidel@ztt.hs-worms.de)

Steffen Wendzel
(Center for Technology and Transfer (ZTT)
Worms University of Applied Sciences, Worms, Germany
wendzel@hs-worms.de)

Sebastian Zillien
(Center for Technology and Transfer (ZTT)
Worms University of Applied Sciences, Worms, Germany
zillien@ztt.hs-worms.de)

Eric S. Conner
(Center for Technology and Transfer (ZTT)
Worms University of Applied Sciences, Worms, Germany
conner@ztt.hs-worms.de)

Georg Haas
(Center for Technology and Transfer (ZTT)
Worms University of Applied Sciences, Worms, Germany
haas@ztt.hs-worms.de)

Abstract: With the increasing number of steganography-capable malware and the increasing trend of stealthy data exfiltrations, network covert channels are becoming a crucial security threat – also for critical infrastructures (CIs): network covert channels enable the stealthy remote-control of malware nested in a CI and allow to exfiltrate sensitive data, such as sensor values, firmware or configuration parameters.
We present WoDiCoF, a distributed testbed, accessible for the international research community to perform a unified evaluation of detection algorithms for network covert channels. In comparison to existing works, our testbed is designed for upcoming big-data scenarios, in which huge traffic recordings must be analyzed for covert channels. It is the first testbed to allow the testing of parallel detection algorithms.
To evaluate WoDiCoF, we took a detection algorithm published in ACM CCS/TISSEC, verified several of the original results and enhanced the understanding of its performance by considering previously unconsidered parameters. By parallelizing the algorithm, we could moreover achieve a speed-up of 2.89 with three nodes.
Key Words: Covert Channels, Network Steganography, Information Hiding, Parallel Computing, Scientific Methodology, Testbeds.
Category: D.4.6, K.6.5, K.7.m

1 Introduction

Network covert channels are stealthy communication channels that enable secret data transfers and are dealt with in the research discipline called *network information hiding*, in particular *network steganography*. Many of these channels are created by hiding data in unused fields of protocol headers or by modulating the timing between network packets. Network covert channels have been studied intensively, resulting in several books and surveys, see e.g. [Mazurczyk et al., 2016; Mileva and Panajotov, 2014; Wendzel et al., 2015; Zander et al., 2007]. Due to the increasing trend of covert channel utilization in malware, new research on the topic is now also relevant for law-enforcement agencies (LEAs); moreover, information hiding-capable malware started to target Critical Infrastructure (CI)—a trend that can be expected to continue over the coming years [Mazurczyk and Caviglione, 2015; Mazurczyk and Wendzel, 2018]. In CI, covert channels can be used to secretly monitor equipment and leak data about sensitive CI processes. They can also be used as a command & control channel to secretly perform manipulations in critical equipment.

Recent work discusses the scientific fundamentals of network covert channels, e.g. cf. [Chen et al., 2017; Mazurczyk et al., 2016; Wendzel et al., 2016, 2017], and several publications address the scientific fundamentals of steganography itself, e.g. [Anderson, 1996; Anderson and Petitcolas, 1998; Katzenbeisser and Petitcolas, 2002]. However, none of these publications addresses two central problems of covert channel research: (1) network covert channel detection algorithms are not evaluated in a way that fosters reproducible experiments similarly as it is common practice in natural sciences, and (2) network covert channel detection methods are not designed for parallel algorithms.

In this article, we present WoDiCoF (*Worms Distributed Covert Channel Detection Framework*), a testbed that improves upon existing scientific methodology in network covert channels in the following ways:

- allows the evaluation of parallel network covert channel detection algorithms (and thus also aids big-data analysis);
- is remotely accessible for the scientific community;
- researchers can provide code and configuration files of their detection algorithms and covert channel techniques, which enables reproducible experiments under specified conditions;
- provides a traffic generator that is tailored for network covert channel analysis.

To illustrate the capabilities of WoDiCoF, we moreover enhance the current understanding of a well-cited covert channel detection algorithm published by [Cabuk et al., 2009] in CCS and TISSEC. We additionally parallelized the algorithm to achieve a speed-up.

WoDiCoF was designed and implemented by the *Center for Technology and Transfer* (ZTT) of the University of Applied Sciences in Worms and is additionally a project that runs under the umbrella of the CUING initiative[1].

The remainder of this article is structured as follows. In Section 2, we discuss related work. We present the design and the implementation of our testbed in Section 3 while Section 4 shows the evaluation of a sample covert channel detection using a well-known algorithm by Cabuk et al., for which we present enhanced insights. Potential drawbacks and limitations of WoDiCoF are discussed in Section 5 while Section 6 concludes and provides an outlook on future work.

2 Related Work

Our work is primarily related to network covert channel testbeds but also to fundamental aspects of information hiding research. We will cover these aspects separately.

Network Covert Channel Testbeds. Several testbeds for security research and testing have been presented. For instance, Benzel et al. designed and configured DETER [Benzel et al., 2007], a testbed comprised of hardware in combination with extensive control software to experimentally verify the effectiveness of attacks and defenses of malicious code. Siaterlis et al. designed and configured EPIC [Siaterlis et al., 2013], a testbed specifically designed for cyber security studies with multiple heterogeneous Networked Critical Infrastructure

However, only few testbeds on network covert channels have been presented so far. Zander developed CCHEF, the *Covert Channels Evaluation Framework*, cf. [Zander and Armitage, 2008]. CCHEF is an extensible tool that allows the creation of covert channels for computer networks. With CCHEF, traffic can be embedded using traffic recordings. CCHEF is set-up using configuration files, which would allow for reperforming conducted experiments.

Zseby et al. developed a network steganography testbed for higher education at the Technical University of Vienna [Zseby et al., 2016]. Their testbed involves CCHEF as a tool for covert channel creation and evaluation but also allows teaching students the statistical analysis of abnormal traffic patterns.

Recently (November 2017), Gunadi and Zander developed an extension for the *Bro* intrusion detection system. Their extension allows for detecting network covert channels [Gunadi and Zander, 2017b]. The authors utilize the concept of 'hiding patterns' that were introduced by [Wendzel et al., 2015] and work on the

[1] CUING (*Criminal Use of Information Hiding*) is an initiative supported by Europol's European Cybercrime Centre (EC3) cf. `http://www.cuing.org`

integration of detection modules for all known patterns. Implementation details are provided in [Gunadi and Zander, 2017a] specifically regarding the applied analysis methods: Kolmogorov-Smirnov test (KS test), entropy and corrected conditional entropy as well as multi modal analysis.

However, none of these testbeds address all of our major design goals, especially the support of parallel detection algorithms and the remote accessibility of the testbed by default.

Scientific fundamentals. Our work also addresses the scientific fundamentals of network information hiding since we foster the reproduction and verification of covert channel detection algorithms and experiment results. Scientific fundamentals of network information hiding recently became a topic in the scientific community. For instance, [Wendzel et al., 2017] discuss how the discipline could profit from the Science 2.0 paradigm. The authors see a need for experimental verification by re-executing experiments described in papers with data, parameters and tools of the particular authors. WoDiCoF aids this process by design.

[Chen et al., 2017] highlight several open problems and research challenges in covert channel research in which especially better metrics and a better understanding of the square root law in the context of noisy channels are demanded. The terminological inconsistencies between network covert channel research and network steganography research were unified by [Mazurczyk et al., 2016]. As pointed out in [Wendzel et al., 2016], network information hiding suffers from inconsistently described hiding methods. For this reason, a unified description method for such methods was proposed.

Fundamentals not specific to network information hiding but applied to the domain of steganography as a whole are also available. A first terminology for steganography and related disciplines, such as anonymity research, was developed under the umbrella of the first international workshop on information hiding in 1996, cf. [Pfitzmann, 1996]. During the same year, [Anderson, 1996] stated that public-key steganography is feasible. Later, he and Petitcolas *clarif[ied] what steganography is and what it can do* and that public-key steganography may be possible in the presence of an active warden [Anderson and Petitcolas, 1998]. [Katzenbeisser and Petitcolas, 2002] discuss how security can be defined in steganographic systems. In their publication, they also summarize previous approaches and discuss their limitations.

Relation to other research disciplines. We are aware of the fact that research on covert channel detection overlaps with research on anomaly detection, in which several approaches/heuristics have been proposed, also in a way that addresses both domains, covert channels and anomaly detection. For instance, [Bouché et al., 2016] performed a passive traffic analysis to mimic legitimate traffic with a covert channel, so that it is not detectable by Snort's Anomaly Detection plugin. Another recent idea was to evaluate only the first N bytes of network packets for anomaly detection [Erlacher and Dressler, 2017]; this idea could potentially also lead to efficient covert channel detection. An extensive survey on anomaly detection methods and tools can be found in [Bhuyan et al., 2014]. However, to the best of our knowledge, these approaches do not provide research testbeds with information hiding-specific tools as we provide them for WoDiCoF. Further, WoDiCoF is designed in a way that it can easily be extended with modules that implement the known anomaly detection methods.

3 Design & Implementation of WoDiCoF

In this section, we first provide an overview of WoDiCoF's design concept, followed by a detailed discussion of its implementation.

3.1 System Requirements

The decision for the open source framework *Apache Hadoop*[2] as our technology platform is based on the capability to support performance and memory space scaling by adding commodity hardware to our cluster or by simply renting cloud computing resources. Additionally it is our goal to offer low barriers to entry for experimentation and analysis. The ease of actually performing parallel tasks as well as no requirements on the development language beyond supporting standard input/output are crucial benefits in this regard. Finally, although there is a certain up-front administrative cost to running a Hadoop based system, we have learned that once successfully installed, only a minimum of care is needed.

Graphics processing unit (GPU) based systems might be faster for special applications, but we are bound to support a wide variety of as yet unknown detection algorithms. Some of these, such as ones based on breadth first or depth first traversals would perform extremely poorly on GPU based systems. We are confident that our approach offers good trade-offs regarding general applicability and performance in a wide range of scenarios.

3.2 Design Concept Overview

The design of WoDiCoF contains several components as depicted in Fig. 1. Traffic can either be generated using a traffic generator or by utilizing previously recorded traffic. Traffic recordings are accepted as PCAP and legacy ethernet format files (from NZIX, *New Zealand Internet Exchange*).

Following input, meta information is extracted from the traffic recordings. This includes: timing values of recorded frames, source and destination addresses,

[2] cf. `https://hadoop.apache.org`

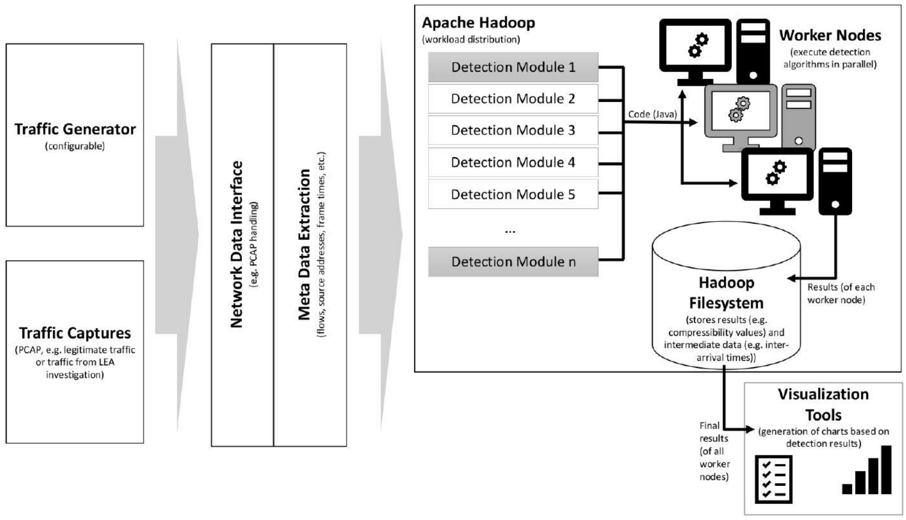

Figure 1: Overview of WoDiCoF.

employed protocols, source and destination ports and packet sizes. This information is then provided to an Apache Hadoop-based system for parallel preprocessing and analysis.

In the Hadoop system, we sort and group the data into flows, derive flow-dependent *inter-arrival times* (IATs)[3] from the relative timing values of the recorded frames and hand these off for analysis. Large flows can be further split in advance into small windows (e.g. $N = 2,048$ packets) to better leverage parallelization advantages. Apache Hadoop performs the parallel execution of the covert channel detection algorithms based on the flow data. Such detection algorithms can be added by researchers using a Java-based interface.

Finally, results are provided on a *Hadoop Distributed Filesystem*[4] (HDFS) where they can be processed further by traditional tools, such as *Gnuplot* and Python libraries, to produce figures to aid the academic paper writing process.

3.3 Implementation Details

While the previous section provided a high-level overview of WoDiCoF, we will now highlight details of our implementation.

[3] IATs represent the elapsed time between two consecutive network packets of one or multiple flows. IATs are sometimes also called *inter-packet gaps* (IPGs).

[4] cf. HDFS Architecture Guide `https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html`.

3.3.1 Traffic Generator

The main purpose of our traffic generator is to create highly configurable synthetic network traffic with specific statistical characteristics.

While several tools for traffic generation exist, e.g. [Garcia and Fyodor, 2017; Haas, 2010; Klauser et al., 2017; Stefano Avallone and Ventre, 2003], all of these fail to *combine* both, flexible covert channel creation (e.g. choosing among different methods to manipulate header bits for information hiding) and scientific applicability (e.g. defining value distributions for header fields or packet timing in a way that is typical for covert channels). For this reason, WoDiCoF contains its own tailored traffic generator that allows both: the generation of covert channel traffic *and* the typical options of other traffic generators, i.e. to define a detailed configuration of several traffic characteristics, such as packet count and distribution of certain values. Our traffic generator is written in Python and uses YAML as a configuration markup language. It utilizes the Scapy packet generator[5] internally to provide a broad configuration interface and support for multiple protocols (TCP, UDP and ICMP).

For each protocol there are a number of configuration parameters available. In the case of TCP, it is possible to simulate a handshake and answers with acknowledgements while the TCP flags and IP options can be manipulated. For ICMP, it is also possible to simulate responses and to manipulate the IP options. For UDP, is it again possible to manipulate the IP options. The number of generated packets can be defined for all protocols.

As mentioned, a central functionality is the manipulation of IATs between packets. The IATs can be generated using a variety of statistical distributions or can be drawn from a user defined list with given probabilities. It is also possible to add another layer of noise on top of the IATs to simulate network jitter. This gives the user several options to simulate legitimate traffic and covert channels. To make the generated traffic reproducible, all used PRNGs are seeded and the seed can be defined in the configuration.

3.3.2 Network Data Interface and Metadata Extraction

The traffic recording input is converted into a comma separated value file (CSV) format that contains only the metadata that is relevant for the detection algorithms. The conversion is done using scripts. The CSV input data is then provided to the Apache Hadoop framework.

3.3.3 Utilization and Functionality of Apache Hadoop

The Apache Hadoop Library allows for the distributed parallel sorting and processing of arbitrary data on commodity systems. Possible clusters can be com-

[5] cf. `http://secdev.org/projects/scapy/`

posed of a few or of up to thousands of nodes. Hadoop employs the MapReduce programming model [Dean and Ghemawat, 2004], a distributed parallel data processing paradigm inspired by functional programming concepts. This enables sorting and processing of suitable big data with the benefit of reduced overall computation time.

The MapReduce programming model consists, unsurprisingly, of two stages: the mapper and the reducer, both of which are parallelized. We use the mapper stage to sort and group the input data into flows. A flow determines an IP-conversation between two endpoints. The covert channel analysis as well as some preprocessing, such as computing IATs from frame times, is performed in the reducer stage.

Each stage in the Hadoop system expects data to be provided as key/value pairs. In our mapper stage, the key is a tuple with the following structure: [*PCAP input filename, protocol name, IP source address, IP destination address, source port (if applicable), destination port (if applicable), frame time*] which together define a flow. The value is a tuple with the following structure: [*frame time, packet number, . . .*]. The value tuple is used for the actual covert channel detection in the reducer stage and can be extended and modified as needed to support different analysis methods. The frame time is part of the key as well as part of the value to make it possible to implement a Secondary Sort Algorithm [Gunarathne et al., 2015], which enables the reducer stage to provide IP-conversations with a list of chronologically sorted values. Arbitrary numbers of modules can be called in the reducer to process the data. The results are stored in CSV formatted files on the HDFS volume.

The input data are CSV files which are easy to split by the Hadoop framework to achieve best results in terms of overall computation time. HDFS distributes the input data within the cluster. Both mapper and reducer processes are replicated and distributed over the cluster. This setup works well using a default configuration but can be tuned further to achieve optimal results. Overall computation time can be adjusted by changing the cluster size.

3.3.4 Visualization

After the detection algorithms process the traffic metadata using Hadoop, all results are aggregated from files residing on HDFS to CSV formatted files on a Linux filesystem. We apply scripts to process these results to generate visualization output. This is currently done using shell scripts (`bash`, `awk`, `sed` and similar tools) as well as `gnuplot`, `weka`[6] and `Python`. By default, plots are generated for all given parameters and for each flow, resulting in Gigabytes of output plots, depending on the size of the input traffic and the number of flows. However, a

[6] `https://www.cs.waikato.ac.nz/ml/weka/`

pre-selection is possible (e.g. by defining thresholds for certain parameters in the scripts).

The usage of scripts in combination with the mentioned UNIX tools and frameworks gives rapid results but is error-prone and cumbersome if parameters have to be adjusted. We are currently evaluating approaches to offer user interfaces for parameter adjustments which are self-explanatory and workflows to speed up the generation of plots for immediate visual results. We are targeting a professional audience with limited programming skills as well as an academic audience.

4 Evaluation of a Sample Approach

We evaluated the functionality of WoDiCoF by implementing a well-known detection algorithm presented by [Cabuk et al., 2004, 2009]. The paper was initially published in ACM CCS 2004. In 2009, an extended version of the paper was published in ACM Trans. Inf. Syst. Security (TISSEC). In sum, Google Scholar reveals more than 500 citations of these publicaitons.

In the following, we first introduce the selected detection algorithm by Cabuk et al. and then explain how we extended the evaluation of their algorithm in comparison to the original work of the authors using WoDiCoF and compare our evaluation results with those of Cabuk et al. To the best of our knowledge, this is the first re-evaluation of a major covert channel detection algorithm. Enabling such re-evaluations (or proofs), as proposed in [Wendzel et al., 2017], was a major goal of WoDiCoF.

4.1 Description of the Used Detection Approach

The underling scenario of Cabuk et al. are storage and timing channels that transfer data in the following ways:[7]

1. *storage channel:* either send data within pre-defined time-frames of duration τ, or not (indicating a zero or one bit, respectively).[8]

2. *timing channel:* Hidden data is encoded in IATs where the IAT τ encodes a zero bit and the IAT 2τ encodes a one bit.

[7] In the original paper by [Cabuk et al., 2009], this approach is described in Section 4.1.2.

[8] Note that this understanding of a storage channel is not perfectly aligned with the typical representation of a network covert storage channel as e.g. defined in [Mazurczyk et al., 2016]. However, in order to stay within the terminology of the original paper, we apply the distinction used by Cabuk et al. in the remainder of this article.

The detection algorithm operates as follows: First, the IATs between network packets $\mathrm{IAT}_1 \ldots \mathrm{IAT}_n$ are recorded for every flow. IAT values > 1 are dropped to reduce noise. Secondly, the IATs are encoded into strings using a function which we call E. E first rounds the IAT to the first two significant digits behind leading zeros. E also encodes zero digits as an alphabetical character. For instance, the authors translate the IAT 0.00247s to 'B25' ('B' represents two zeros) and 0.0247s to 'A25' ('A' represents one zero). All string-encoded IATs of a flow are afterwards concatenated to a string S, i.e.

$$S = E(\mathrm{IAT}_1) \mid\mid E(\mathrm{IAT}_2) \mid\mid \ \ldots \mid\mid E(\mathrm{IAT}_n),$$

with $||$ representing a string concatenation.

Thirdly, the authors approximate the Kolmogorov complexity of S, which is the maximum compression of a string, by compressing the string with the *gzip* algorithm, i.e. $C = \mathrm{gzip}(S)$, and then calculating the *compressibility* (compression rate):

$$\kappa(S) = \frac{|S|}{|C|},$$

with $|x|$ representing the length of the string x.

Cabuk et al. compare the compressibility of covert channel flows with non-covert channel flows. Their results indicate that covert channel flows and non-covert channel flows can be distinguished in specific cases that were presented in the original paper (we will discuss their results in Sect. 4.3).

Cabuk et al. transferred a textual string over their covert channel, using a connection between Purdue and Georgetown universities with τ set to 0.04, 0.06 and 0.08 sec. Their goal was to determine, whether the compressibility approach allows to *accurately detect a covert channel in a short window* [Cabuk et al., 2009]. The authors used a window size of $N = 2,000$ packets. They performed a detection of covert channel traffic and compared their results with detectability results for legitimate traffic, including noise.

4.2 WoDiCoF-based Enhancement of the Analysis

In comparison to the original analysis by Cabuk et al., we did not reperform *all* of their conducted experiments. However, we extended the evaluation of their algorithm for the following aspects to underpin the capabilities of WoDiCoF:

1. We parallelized the algorithm to evaluate its performance depending on the number of nodes operating in parallel.

2. We implemented the coding of string S using different precisions of digits behind leading zeros of the IATs, i.e. not solely considering the first two significant digits behind leading zeros.

3. We evaluated the algorithm based on different content transferred over the covert channel. In particular, we transferred repetitive strings in the form (i) "ABABAB...", (ii) "The quick brown fox jumps over the lazy dog.", (iii) the first page of Goethe's *Faust*[9], as well as Faust compressed using (iv) GZIP, (v) ZIP, and (vi) BZIP2. No error-correcting codes were applied.

4. We analyzed covert channel traffic using different connections: a connection over local ethernet switches and a connection with a remote host in another autonomous system.

4.3 Evaluation Results

In the following, we provide our results for the parameters mentioned in the previous section and compare our results with results provided by Cabuk et al. (if available).

4.3.1 Parallel vs. Sequential Performance

One of the design goals of WoDiCoF was to enable parallel execution of detection algorithms using the Apache Hadoop framework. To achieve parallelization, input data are dynamically split by Hadoop and fed into the system. Hadoop automatically monitors the utilization of each node and ensures an equal workload distribution. As expected, this simple parallelization already achieves a speed-up as shown in Fig 2.[10] We calculated the speed-up S_n of the algorithm running on n nodes by dividing the serial execution time T_1 by the time the algorithm required to complete its execution on n nodes (T_n):

$$S_n = T_1/T_n$$

This calculation is analog to the classical speed-up calculation described in [Tanenbaum and Bos, 2015], where the theoretical speed-up is calculated depending on the available processors. However, in our case, T_1 and T_n were determined via time measurements.

We processed the NZIX-II recordings[11] that were used by Cabuk et al. and which contained 835 Mio. packets with the detection algorithm described above. If the code was executed on only one node, it took 3:05 hrs to process all data. When two nodes were used, the overall runtime was roughly halved (1:36 hrs,

[9] The text of Faust is available under `http://gutenberg.spiegel.de/buch/-3664/4`.

[10] Please note that this flow-based workload assignment will not work for detection algorithms that need to consider data of multiple flows, cf. Sect. 5. Such algorithms are, for instance, required in the case of protocol switching covert channels (PSCC).

[11] `https://wand.net.nz/wits/nzix/2/`

speed-up: 1.927). For three nodes, the computing time was 1:04 hrs (speed-up: 2.890), indicating small amount of serial code. However, these values solely include the parallel execution of the detection algorithm, not the pre-processing of the NZIX-II data, i.e., the initial extraction of meta-data, which can take multiple minutes for such large traffic recordings.

All nodes were equipped with the similar resources (AMD Opteron 4180 or Intel Xeon E5603/E31260L CPUs, with eight cores each (running eight threads for Apache Hadoop), 4-8 GBytes of RAM, 1 GBit/s network link).

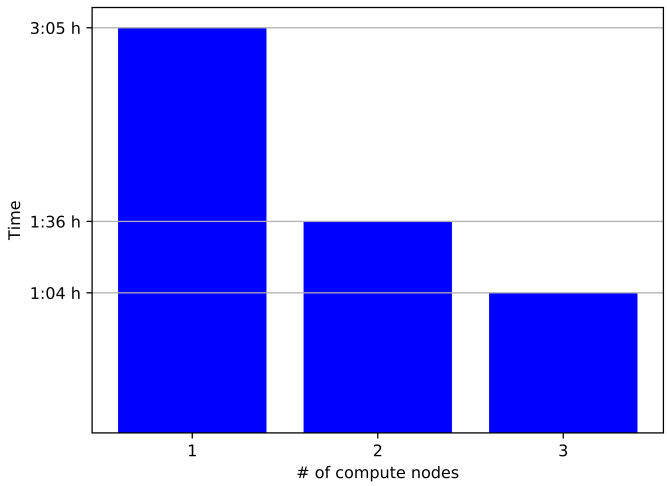

Figure 2: Comparison of computing time for compressibility-based detection depending on the number of nodes operating in parallel.

4.3.2 Compressibility-dependence on String Coding Precision

Cabuk et al. utilized a precision of two, i.e. the first two significant bits of the IAT where taken into account to perform a string compression. While Cabuk et al. selected only a few flows which all featured a κ value lower than those of their covert storage channels, our results in Fig. 3 show that the precision (number of utilized relevant digits behind leading zeros) clearly influences κ. As also visible, the legitimate traffic in the NZIX-II recordings results in κ values that strongly overlap with the κ values of covert channel traffic for all tested protocols, i.e. legitimate UDP, TCP and ICMP traffic. However, for a higher precision, κ of legitimate traffic is smaller (below seven for precission 4). Moreover, storage covert channel (SCC) and timing covert channel (TCC) traffic overlap in their κ values.

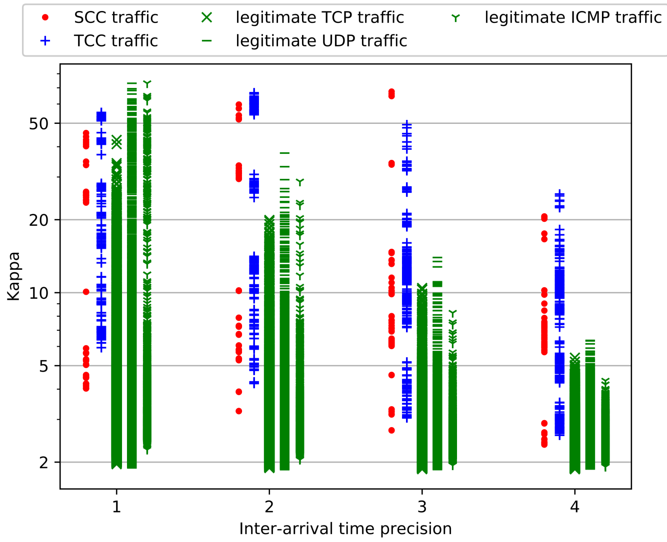

Figure 3: Dependence of κ on the type of applied precision and traffic type.

4.3.3 Compressibility-dependence on Transferred Data Type

Cabuk et al. did not distinguish how the transferred data of the covert channel influences the compressibility (κ). As long as encrypted, random or compressed content is transferred, our compressibility results for storage channels are at least similar to those of Cabuk et al., i.e. our results partially confirm the results of Cabuk et al. (see Fig. 4). However, we also transferred different types of plaintext over the covert channels. In particular, we additionally transferred the text Faust, the repeating string "ABABAB...", the GZip, BZip2 and ZIP compressed version of Faust and the text "The quick brown fox jumps over the lazy dog" in loops over the channel. Each traffic was transferred using the τ values used by Cabuk et al. i.e. 0.04 sec, 0.06 sec, and 0.08 sec.

Fig. 4 shows the results for a storage channel (SCC). For $\tau = 0.04$ sec, our results match those of Cabuk et al. if the data is compressed with ZIP or GZip. We achieve a lower compressibility for BZip2 and significantly larger compressibility scores for Faust and highly-compressible content ("ABABAB" and "The quick brown fox ...").

For $\tau = 0.06$ and for $\tau = 0.08$, we cannot confirm consistency with the result of Cabuk et al. in our experiment as the compressibility of Cabuk et al. matches approximately the compressibility of uncompressed the Faust text. For all compressed data, we achieve lower κ values.

The texts "ABABAB..." and "The quick brown fox" resulted in high compressibility scores for all tested values of τ.

Cabuk et al. observed that with an *increase [of] timing interval for IP SCC, compressibility increases*, too. Our results cannot confirm this observation as shown in the difference for our values and the values by Cabuk et al. in Fig. 4. Instead, our compressibility results slightly decreased or remained stable. However, it must be noted, that we did not transfer exactly the same text that was used by Cabuk et al. (but several other texts) and used our own covert channel tool to generate the covert traffic (no error-correcting codes were applied).

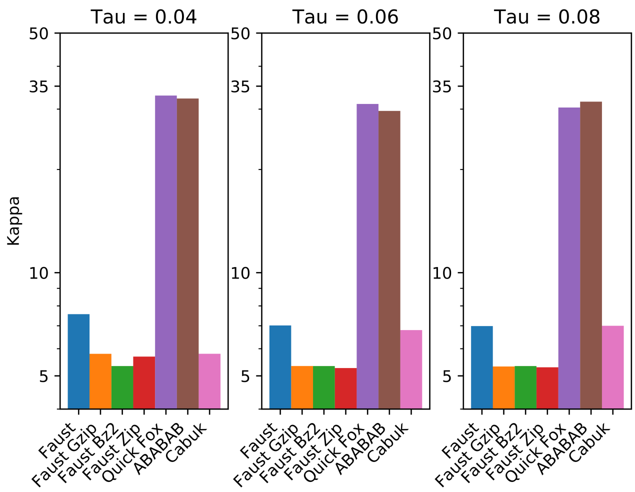

Figure 4: Dependence of κ on the type of transferred content for SCC, in comparison to results provided by Cabuk et al.

Unfortunately, Cabuk et al. only provided SCC results and no TCC results in their paper. We performed the necessary experiments to show how κ is influenced for timing channels, depending on content type and τ value (Fig. 5). As shown in the figure, most of the κ values for TCC are in a range of 10-15. Depending on τ, we could also observe a strong difference for the repeating sentence "The quick brown fox ..." (colored in lilac).

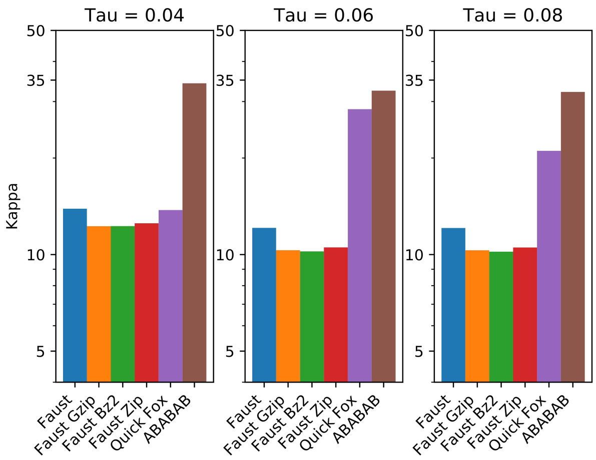

Figure 5: Dependence of κ on the type of transferred content for TCC.

We also analyzed how the κ values for legitimate traffic are distributed for the NZIX-II data recordings. We applied a precision of two since it was used by Cabuk et al. As shown in Fig. 6, the largest part of the legitimate traffic (TCP, UDP, ICMP) results in κ values below ten. However, a significant amount of legitimate traffic results in larger values. Given that κ values between 5-7 were presented as SCC traffic by Cabuk et al. and that our experiments showed that most of the TCC traffic has values below 10, a significant amount of the legitimate traffic would result in false positives if a detection would be performed based on the compressibility.

4.3.4 Compressibility-dependence on Network Connection

Finally, we compared, how the type of the network connection influences the compressibility as different connections provide different jitter and performance. We transferred traffic over the local university network in Worms (two hops over ethernet) and between an ISP-hosted server in Frankfurt and our university network (seven hops for most routing paths). Again, we applied the previously used τ values of Cabuk et al. (0.04, 0.06 and 0.08 sec) with a precision of two. We transferred both, TCC and SCC traffic for all combinations of parameters.

As shown in Fig. 7, remote traffic resulted in overall lower compressibility scores (4-14 instead of 5-68), which we assume is influenced by the increased network jitter. This statement applies for all three τ values as well as for both covert channel types, SCC and TCC.

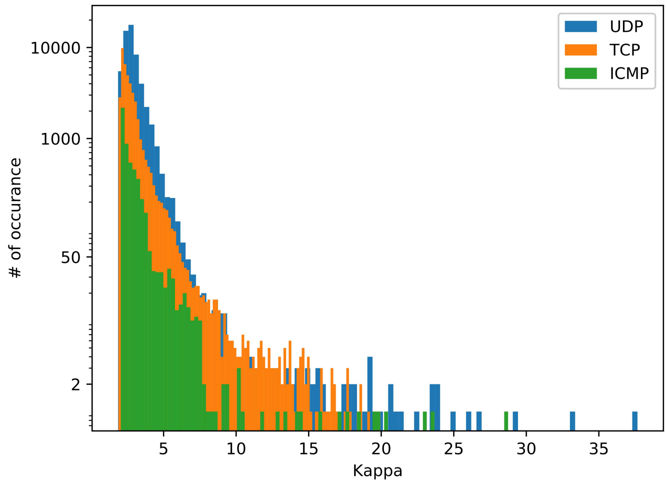

Figure 6: Histogram of κ values for legitimate traffic from the NZIX-II recordings.

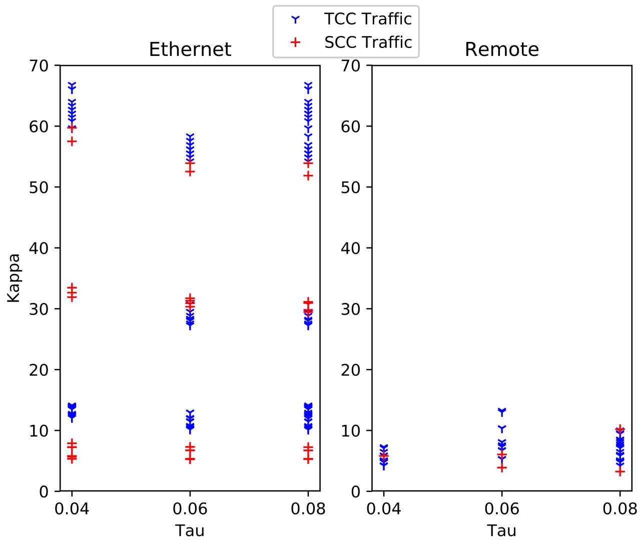

Figure 7: Dependence of κ on the utilized network connection and τ parameters for both, TCC and SCC traffic.

While Cabuk et al. did not compare different types of network connections, they introduced several degrees of noise (10%-50%) to *legitimate* traffic and observed a decrease in compressibility. This is similar to our experiment which compares a seven-hop high-jitter connection with a two-hop low-jitter connection. We thus consider our observation as an indicator that confirms these findings of Cabuk et al.

5 Discussion

Despite the discussed advantages, WoDiCoF is linked to a number of drawbacks.

First of all, WoDiCoF currently allows no live-traffic analysis as it requires traffic recordings in the form of PCAP files or NZIX' legacy ethernet format files.

Our testbed is currently flow-oriented, i.e. it does not support detection algorithms for inter-protocol steganography or protocol switching covert channels [Jankowski et al., 2013; Mazurczyk et al., 2016; Wendzel and Keller, 2012, Chapter 4]. Such channels could, however, be detected if each protocol/sub-carrier would be analyzed separately and results would then be merged in an additional step.

What can also be considered a drawback of WoDiCoF is that it requires a fundamental understanding of Apache Hadoop. The requirements are low if researchers or professionals (e.g. LEA users) use the testbed with the already implemented algorithms. However, a deeper understanding of Apache Hadoop is required if new detection algorithms shall be added.

Currently, the visualization is based on a simple heuristic (e.g. threshold values). This means that for a larger traffic recording with thousands of flows, graphs are generated for all flows in combination with used parameters (e.g. for all τ values and all configured precisions of significant digits in inter-arrival times), resulting in several GBytes of graph outputs. The results are challenging to select by hand and thus need decision-making support. We plan to add a visual analytics component for WoDiCoF to aid this problem.

There is also a limitation regarding the evaluation of the compressibility algorithm by Cabuk et al. as we applied neither error-detecting codes nor error-correcting codes. However, this could be added and would most likely not reveal new key insights that are significantly different from the provided content types. Tools such as CCHEF could be used as a part of WoDiCoF to generate such traffic.

Further, Cabuk et al. enhanced their approach by combining different measures for more accurate detectability and also modified their compressibility approach to an approach with sliding windows. We did not performed measurements to compare our results with these extended versions.

Finally, it must be noted that Cabuk et al. discuss several limitations of their work and the problem of finding an optimal window size parameter in their original work. Their compressibility measure can thus be seen as a sophisticated detection approach. However, our results underpin that if certain parameters change (e.g. TCC instead of SCC traffic or change of transferred content), the compressibility scores highly vary. Moreover, we could show that the compressibility values of legitimate traffic can significantly overlap with the compressibility values of covert traffic. Overlapping compressibility values of legitimate and covert traffic would result in false-positives or false-negatives, depending on whether a compressibility threshold for detection is set too low or too high.

6 Conclusion and Future Work

Firstly, we present WoDiCoF (*Worms Distributed Covert Channel Detection Framework*), a testbed for experiment verification and parallelization of detection algorithms in network information hiding. Our testbed allows the implementation and evaluation of parallel and sequential detection algorithms and the generation of tailored traffic for research.

With WoDiCoF, we were secondly able to provide additional insights in a well-cited covert channel detection algorithm presented by Cabuk et al. The tested algorithm is based on a compressibility score. We performed analyses with several additional parameters not originally tested by the authors and plan to analyze more parameters in future work, especially sliding window sizes. Our results show that we could confirm some of the evaluation results provided by Cabuk et al. but that under varying conditions (e.g. different content type or different type of covert channel (timing vs. storage)), results significantly differ. This underpins the importance of experiment verification in network information hiding.

Thirdly, Cabuk et al. state that more parameters could be analyzed to identify covert channels but that this would *require additional hardware or processing time, and is best done in an offline manner*. This is a key aspect provided by WoDiCoF. We have tested the parallel efficiency of our testbed and could determine a speed-up for the algorithm by Cabuk et al.

Currently, we are working on the implementation of additional detection algorithms for both, covert timing and storage channels. In particular, we plan to implement detection algorithms for more 'hiding patterns'. Moreover are we planing to extend WoDiCoF with a visual analytics component to aid LEA users' analysis phase. WoDiCoF will further be jointly developed together with partners of the CUING initiative and aims to provide a remotely accessible testbed especially for academic and LEA users.

References

[Anderson, 1996] Anderson, R. (1996). "Stretching the limits of steganography"; R. Anderson, ed., Proc. Information Hiding: First International Workshop

Cambridge, U.K.; 39–48; Springer Berlin Heidelberg, Berlin, Heidelberg.

[Anderson and Petitcolas, 1998] Anderson, R. J., Petitcolas, F. A. P. (1998). “On the limits of steganography”; IEEE Journal on Selected Areas in Communications; 16, 4, 474–481.

[Benzel et al., 2007] Benzel, T., Braden, R., Kim, D., Joseph, A. D., Neuman, B. C., Ostrenga, R., Schwab, S., Sklower, K. (2007). “Design, deployment, and use of the DETER testbed”; Proc. DETER Community Workshop on Cyber-Security and Test; USENIX Assoc.

[Bhuyan et al., 2014] Bhuyan, M. H., Bhattacharyya, D. K., Kalita, J. K. (2014). “Network anomaly detection: Methods, systems and tools”; IEEE Communications Surveys & Tutorials; 16, 1, 303–336.

[Bouché et al., 2016] Bouché, J., Hock, D., Kappes, M. (2016). “On the performance of anomaly detection systems uncovering traffic mimicking covert channels”; Proc. 11th International Network Converence (INC 2016); 19–24; Plymouth University.

[Cabuk et al., 2004] Cabuk, S., Brodley, C. E., Shields, C. (2004). “IP covert timing channels: design and detection”; Proc. 11th ACM conference on Computer and Communications Security (CCS’04); 178–187; ACM.

[Cabuk et al., 2009] Cabuk, S., Brodley, C. E., Shields, C. (2009). “IP covert channel detection”; ACM Trans. Inf. Syst. Secur.; 12, 4, 22:1–22:29.

[Chen et al., 2017] Chen, O., Meadows, C., Trivedi, G. (2017). “Stealthy protocols: Metrics and open problems”; Concurrency, Security, and Puzzles: Essays Dedicated to Andrew William Roscoe on the Occasion of His 60th Birthday; 1–17; Springer International Publishing, Cham.

[Dean and Ghemawat, 2004] Dean, J., Ghemawat, S. (2004). “Mapreduce: Simplified data processing on large clusters”; Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6; OSDI’04; 10–10; USENIX Association, Berkeley, CA, USA.

[Erlacher and Dressler, 2017] Erlacher, F., Dressler, F. (2017). “High performance intrusion detection using http-based payload aggregation”; Proc. 2017 IEEE 42nd Conference on Local Computer Networks (LCN); 418–425; IEEE.

[Garcia and Fyodor, 2017] Garcia, L. M., Fyodor (2017). “Nping”; `https://nmap.org/nping/`.

[Gunadi and Zander, 2017a] Gunadi, H., Zander, S. (2017a). “Bro covert channel detection (BroCCaDe) framework: Design and implementation”; Technical Report IT NSRG TR 20171117B; Murdoch University.

[Gunadi and Zander, 2017b] Gunadi, H., Zander, S. (2017b). “Bro covert channel detection (BroCCaDe) framework: Scope and background”; Technical Report IT NSRG TR 20171117A; Murdoch University.

[Gunarathne et al., 2015] Gunarathne, T., Blaminsky, J., Gordon, E., Lalwani, P., Paiva, A., Subramanian, L. (2015). Hadoop MapReduce v2 Cookbook; Packt Publishing; 2 edition.

[Haas, 2010] Haas, H. (2010). "Mausezahn"; `https://github.com/uweber/mausezahn`.

[Jankowski et al., 2013] Jankowski, B., Mazurczyk, W., Szczypiorski, K. (2013). "Padsteg: introducing inter-protocol steganography"; Telecommunication Systems; 52, 2, 1101–1111.

[Katzenbeisser and Petitcolas, 2002] Katzenbeisser, S., Petitcolas, F. A. P. (2002). "Defining security in steganographic systems"; Proc. Security and Watermarking of Multimedia Contents IVSPIE 2002; volume 4675; doi: `http://dx.doi.org/10.1117/12.465313`.

[Klauser et al., 2017] Klauser, T., Borkmann, D., et al. (2017). "Trafgen (part of netsniff-ng toolkit)"; `http://netsniff-ng.org/`.

[Mazurczyk and Caviglione, 2015] Mazurczyk, W., Caviglione, L. (2015) "Information hiding as a challenge for malware detection"; IEEE Security & Privacy; 13, 2, 89–93.

[Mazurczyk and Wendzel, 2018] Mazurczyk, W., Wendzel, S. (2018). "Information hiding: Challenges for forensic experts"; Communications of the ACM (CACM); 61, 1, 86–94; `https://dl.acm.org/citation.cfm?id=3158416`.

[Mazurczyk et al., 2016] Mazurczyk, W., Wendzel, S., Zander, S., Houmansadr, A., Szczypiorski, K. (2016). Information Hiding in Communication Networks: Fundamentals, Mechanisms, and Applications; Wiley-IEEE.

[Mileva and Panajotov, 2014] Mileva, A., Panajotov, B. (2014). "Covert channels in TCP/IP protocol stack - extended version-"; Central European Journal of Computer Science; 4, 2, 45–66.

[Pfitzmann, 1996] Pfitzmann, B. (1996). "Information hiding terminology"; Proc. First International Workshop on Information Hiding; volume 1174 of LNCS; 347–350; Springer.

[Siaterlis et al., 2013] Siaterlis, C., Genge, B., Hohenadel, M. (2013). "Epic: A testbed for scientifically rigorous cyber-physical security experimentation"; IEEE Transactions on Emerging Topics in Computing; 319–330.

[Stefano Avallone and Ventre, 2003] Stefano Avallone, A. P., Ventre, G. (2003). "Distributed internet traffic generator (D-ITG): analysis and experimentation over heterogeneous networks"; Int. Conference on Network Protocols (ICNP'03) poster proceedings.

[Tanenbaum and Bos, 2015] Tanenbaum, A. S., Bos, H. (2015). Modern Operating Systems; Pearson; 4 edition.

[Wendzel and Keller, 2012] Wendzel, S., Keller, J. (2012) "Preventing protocol switching covert channels"; International Journal On Advances in Security; 5, 3 and 4, 81–93.

[Wendzel et al., 2015] Wendzel, S., Zander, S., Fechner, B., Herdin, C. (2015)

"Pattern-based survey and categorization of network covert channel techniques"; ACM Computing Surveys (CSUR); 47, 3, article 50.

[Wendzel et al., 2016] Wendzel, S., Mazurczyk, W., Zander, S. (2016). "Unified description for network information hiding methods"; Journal of Universal Computer Science (J.UCS); 22, 11, 1456–1486; http://www.jucs.org/jucs_22_11/unified_description_for_network.

[Wendzel et al., 2017] Wendzel, S., Caviglione, L., Mazurczyk, W., Lalande, J.-F. (2017). "Network information hiding and Science 2.0: Can it be a match?"; Int. Journal of Electronics and Telecommunications; 62, 2, 217–222.

[Zander and Armitage, 2008] Zander, S., Armitage, G. (2008). "CCHEF–covert channels evaluation framework design and implementation"; Technical report; Swinburne University of Technology. Centre for Advanced Internet Architectures.

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

[Zseby et al., 2016] Zseby, T., Vázquez, F. I., Bernhardt, V., Frkat, D., Annessi, R. (2016). "A network steganography lab on detecting TCP/IP covert channels"; IEEE Transactions on Education; 59, 3, 224–232.