Protocol-independent Detection of "Messaging Ordering" Network Covert Channels

Steffen Wendzel

Worms University of Applied Science, Centre of Technology and Transfer (ZTT)
contact: wendzel@hs-worms.de

Abstract

This preprint was submitted to CUING'19. The final version of this paper appeared in Proc. ARES 2019 and is available here: https://doi.org/10.1145/3339252.3341477. Enhanced measurements are provided in [17].

Detection methods are available for several known covert channels. However, a type of covert channel that received little attention within the last decade is the "message ordering" channel. Such a covert channel changes the order of PDUs (protocol data units, i.e. packets) transferred over the network to encode hidden information. The advantage of these channels is that they cannot be blocked easily as they do not modify header content but instead mimic typical network behavior such as TCP segments that arrive in a different order than they were sent.

Contribution: In this paper, we show a protocol-independent approach to detect message ordering channels. Our approach is based on a modified *compressibility score*. We analyze the detectability of message ordering channels and whether several types of message ordering channels differ in their detectability.

Results: Our results show that the detection of message ordering channels depends on their number of utilized PDUs. First, we performed a rough threshold selection by hand, which we later optimized using the C4.5 decision tree classifier. We were able to detect message ordering covert channels with an accuracy and F_1 score of $\geq 99.5\%$ and a false-positive rate $< 1\%$ and $< 0.1\%$ if they use sequences of 3 or 4 PDUs, respectively. Simpler channels that only manipulate a sequence of two PDUs were detectable with an accuracy and F_1 score of 94.5% and were linked to a false-positive rate of 5.19%. We thus consider our approach suitable for real-world detection scenarios with channels utilizing 3 or 4 PDUs while the detection of channels utilizing 2 PDUs should be improved further.

Keywords: Covert Channels, Steganography, Information Hiding, Stego-malware, Hiding Patterns, Message Ordering Pattern, PDU Order Pattern

1   Introduction

Covert channels are policy-breaking, stealthy communication channels that are reported to be utilized by cyber criminals to enable stealthy communications for malware and data exfiltration [10, 11, 13, 15]. While traditional steganography methods utilize digital media carriers to embed hidden information (e.g. image files or audio files [7]), network covert channels can be realized by hiding data in the meta-data of network traffic, e.g. timing of PDUs or modification of PDU content. Covert channels can utilize all OSI layers and were described for all popular communication protocols, e.g. DHCP, IPv4, IPv6, ICMP, TCP, UDP, DNS, HTTP, VoIP and others [19, 20, 12, 13, 8]. The detection of network covert channels is challenging, also for the law-enforcement and for professional administrators, and not all types of covert channels can be efficiently detected.

In recent years, detection approaches improved, rendering trivial covert channels easier to detect. For this reason, covert channels with untypical content, e.g. where usually reserved PDU bits are modified, can be detected with simple signatures. In contrast, some covert channels mimic the behavior of normal network traffic [13]. A rarely studied type of covert channel are those that can be summarized under the "message ordering" pattern [19, 12]. They do not modify the content of network packets but only their order. For instance, the order of TCP segments can be manipulated to encode a hidden message. While the content of the TCP segments would appear normal, the changed order of TCP segments can also be considered normal as such segments do not always arrive in the order they were originally sent. Message ordering channels can be realized by all protocols that can indicate the sequence of packets, e.g. TCP, AH and ESP.

In this paper we present an approach to detect message ordering channels in a protocol-independent manner. Our approach is tailored for the law-enforcement to analyze traffic recordings. However, our approach can be potentially applied to corporate networks with a low traffic volume in real-time, too. In particular, the contributions are as follows:
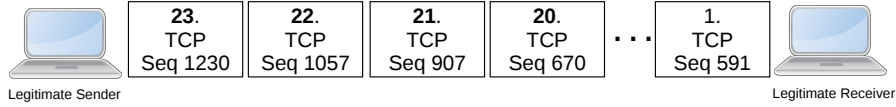
1. Perform a modification (*countermeasure variation*) of the so-called *compressibility* score that was originally introduced to detect covert channels that modify the timing between PDUs. Therefore, a suitable coding is presented. The presented detection approach is designed in a way to make it independent of the utilized network protocol.

2. Evaluating the detectability of message ordering channels that are utilizing a different number of PDUs in their coding.

The remainder of this paper is structured as follows. Sect. 2 covers fundamentals and discusses related work. Our detection approach and the related countermeasure variation are introduced in Sect. 3. We describe our experimental evaluation and discuss our results in Sect. 4. Sect. 5 concludes.

2 Fundamentals & Related Work

The functioning of message ordering channels is visualized in Fig. 1. A covert sender (CS) sends a sequence of messages to the covert receiver (CR). In comparison to legitimate traffic, the order of protocol data units (PDUs), i.e. network packets, is modified by a covert sender to encode a hidden message. Sender and receiver agree upon the coding a priori, e.g. it can be defined in a malware's executable before deployment.

1. Legitimate Transmission

	23. TCP Seq 1230	**22**. TCP Seq 1057	**21**. TCP Seq 907	**20**. TCP Seq 670	···	**1**. TCP Seq 591	

Legitimate Sender / Legitimate Receiver

2. Transmission with Message Ordering Pattern

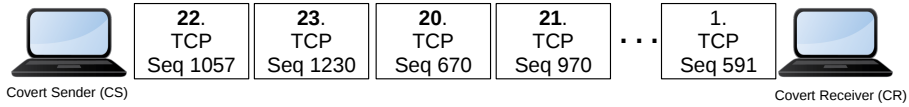|	**22**. TCP Seq 1057	**23**. TCP Seq 1230	**20**. TCP Seq 670	**21**. TCP Seq 970	···	**1**. TCP Seq 591	

Covert Sender (CS) / Covert Receiver (CR)

Figure 1: The functioning of the message ordering pattern.

In general, a covert channel manipulating the order of n packets can be used to transfer $\log_2 n!$ bits [1, 9]. Thus, the more PDUs are utilized, the more bits can be transferred per PDU. For practicability, longer messages can also be split into m sequences of n PDUs each, allowing $m \cdot \log_2 n!$ bits to be transferred. For instance, if a transfer contains 2,000 packets, and if $n = 4$, then this would allow the transfer of $500 \cdot \log_2 4! = 2,292.5$ bits (more than 286 bytes), and it would be 2,762.8 bits (345 bytes) for $n = 5$ ($m = 400$). Message ordering channels can utilize the sequence numbers of TCP, AH or ESP headers, several application layer protocol's sequence numbers and the IPv4 Identifier field, among others [1, 9, 12].

Hiding patterns are abstract descriptions of covert channel hiding techniques; they were introduced in [19]. Each hiding pattern represents a set of covert channel techniques that follow the same general hiding technique. These hiding patterns form a taxonomy, where the message ordering pattern (former called *PDU order pattern*) is categorized as a timing channel (because manipulating the order of PDUs effectively alters their timings). In [12], the message ordering pattern was additionally categorized as *protocol-aware*, i.e. as a method *that require[s] the understanding of the carrier protocol*[1], and was finally renamed to *(manipulated) message ordering* pattern.

Covert channel detection is also well covered in the existing literature. In general, countermeasures either aim on detecting the presence of the channel itself or the involvement of a participant in the covert communication. Other

---

[1]In contrast, protocol-agnostic timing channels can modify the carrier blindly, e.g. by manipulating the inter-arrival times between packets.

countermeasures aim on limiting the channel capacity or on preventing the use/the existence of channels. Moreover channels can be audited [12]. Several detection methods exist and summaries can be found in [20, 12]. A recent trend is the detection of distributed covert channels [2]. Message ordering channels could also be realized in a distributed manner, e.g. split over multiple flows simultaneously. We do not explicitly focus on such a distributed scenario in this paper, but the presented approach could potentially detect distributed message ordering channels nevertheless on a per-flow basis. Moreover, digital media steganography has similar channels, e.g. methods that modify the order of HTML tags in websites, which can be detected with statistical methods [14]. However, no specific method was presented so far to detect network-based message ordering channels.

A detection approach for covert timing channels it the so-called *compressibility score* that was introduced by Cabuk et al. [4]. The compressibility score is applied to detect channels that modulate inter-arrival times (IAT) between succeeding network packets. These channels belong to the "Inter-packet Times" pattern [12]. Therefore, IAT values of a flow are recorded, and then rounded. The rounded values are encoded as short strings that are concatenated to a long string S. Afterwards, S is compressed with a compressor $\Im$ (e.g. *gzip*), i.e. $C = \Im(S)$. Finally, the value $\kappa = |C|/|S|$ is calculated. κ is used as an indicator for the presence of a covert timing channel. In general, a covert channel's artificial IAT values of the "inter-packet times" pattern would be similar within a given flow, rendering the results better compressible than legitimate traffic with more varying IAT values [4]. In comparison to the inter-packet times pattern, the message ordering pattern does not encode hidden information in the IAT values but in the pure order of PDUs, i.e. it does not matter how large the IAT values between PDUs are.

The concept of countermeasure variation was recently introduced in [18]. The idea is to take existing countermeasures, e.g. detection approaches as the one of Cabuk et al., and modify them slightly so that they can work for other patterns as originally intended. Countermeasure variation aids the problem of not having countermeasures for all patterns available. Previous work has shown that the compressibility score as well as the so-called ϵ-similarity[2] can be adjusted to work with the "size modulation" [18] and "artificial re-transmissions" patterns [21]. In this paper, we perform a countermeasure variation of the compressibility score so that it can work with the message ordering pattern.

3   Detection Approach

The development of countermeasures for criminal uses of covert channels and steganography has recently attracted increasing attention, cf. [10, 15]. Therefore, we consider a scenario where traffic recordings of criminal cases must be analyzed some time after they were recorded. This means that we do not focus on a real-time detection scenario, i.e. computing performance of our approach is

---

[2]The ϵ-similarity is described in [3, 4].

not a strong requirement. However, it can potentially be applied in low-volume networks, too, for real-time detection.

Our detection approach is designed to work for intermediate nodes (IM) located between covert sender (CS) and covert receiver (CR) that conduct traffic recordings. It can be a gateway or router. However, as an administrative system process, our detection approach could also be integrated into CS or CR as long as the network covert channel process and its user are separated from the detection process. This could be realized, e.g. in the form of a malware. However, in this case, the detection process needs direct access to the network interface, e.g. as a kernel-level routine. As it is beneficial to recognize *all* packets of a flow, it would even be optimal to locate the detection process on CS or CR since there might be multiple IMs for multiple routing paths, so one IM might not see *all* traffic exchanged between CS and CR.

We perform a countermeasure variation as follows. First, we modify the compressibility score of Cabuk et al. in a way that we provide it with a different input as for the algorithm. Second, we modify the generation process of the string S. Third, we determine optimal thresholds to differentiate between legitimate and covert channel traffic. The other steps of the compressibility score (compression and calculation of κ) remain as in the original approach. The modifications are explained in subsection 3.1.

3.1   Algorithm Input and String Coding

In the original approach by Cabuk et al., only one algorithm is shown to encode IAT values into strings. Instead, we encode sequence numbers and moreover experimented with four different approaches to encode these sequence numbers. Another difference to the original approach by Cabuk et al. is that IAT values cannot overrun but sequence numbers can indeed overrun. For this reason, we implemented a filter. We also defined new detection thresholds for κ.

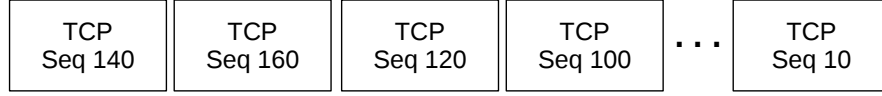The details of our approach are described in the following. Fig. 2 visualizes our approach.

In step 1, we record all flows between two or more hosts to be observed (Fig. 2-1). For each flow, we consider a window of 200 PDU sequences, i.e. 201 packets. For instance, such a flow could contain 201 TCP segments (or alternatively a flow using any other protocol with a header field that allows to determine the PDU order).[3]   For all these sequences, we assign each PDU a number between 1 and 200 based on its appearance in the order of packets in the window. For instance, if four following sequence numbers were 100, 120, 160, 140, for the packets 35 to 38, then we would record the order 35, 36, 38, 37 (see Fig. 2-2).

In the following, `diff` represents the relative difference between the current and the last PDU number. For the mentioned example, the values for `diff` would be +1, +2, and -1. `|x|` indicates the absolute value of the variable `x`.
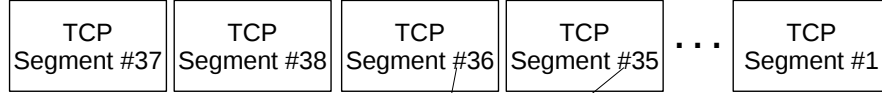
---

[3]Please note that the message ordering pattern requires protocols to have a numbering element. Thus, protocols without such a field are not of relevance for our research.
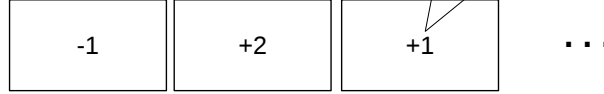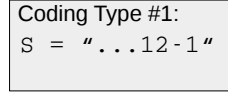
1. Create Traffic Recording

| TCP Seq 140 | TCP Seq 160 | TCP Seq 120 | TCP Seq 100 | **. . .** | TCP Seq 10 |

2. Enumerate Packets

| TCP Segment #37 | TCP Segment #38 | TCP Segment #36 | TCP Segment #35 | **. . .** | TCP Segment #1 |

3. Calculate Relative Differences

| -1 | +2 | +1 | **. . .** |

4. Concatenate Differences in a String

```
Coding Type #1:
S = "...12-1"
```

5. Compress String

```
C = compress(S)
```

6. Calculate Compressibility
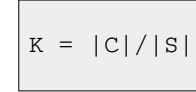
```
K = |C|/|S|
```

Figure 2: Our proposed five-step approach to detect message ordering channels (steps 3 and 4 are exemplified for one type of coding).

Moreover, `a || b` is the string concatenation of the strings `a` and `b`. We use `a || (y ? 't' : 'f')` to test whether the condition `y` is true. If it is true, the string `'t'` will be concatenated to the string `a`, otherwise the string `'f'` will be concatenated to `a`. The function `str(x)` returns the value of `x` in the form of a string.

We tested the following four different codings. Please note that only one string element is shown each, i.e., 200 of these values would be concatenated to create S.

1. `str(diff)`: concatenate the `diff` values of succeeding packets (exemplified in Fig. 2).

2. `str(|diff|)`: concatenate the absolute `diff` values of succeeding packets.

3. `str(diff) || ( |diff| % 2 ? 'A' : 'B' )`: for all succeeding packets, concatenate `diff` with an 'A' if the absolute value of `diff` mod 2 equals 1, otherwise with 'B'.

4. `str(|diff|) || ( |diff| % 2 ?  'A' : 'B' )`: for all succeeding packets, concatenate the absolute value of `diff` with an 'A' if the absolute value of `diff` mod 2 equals 1, otherwise with 'B'.

Fig. 2-3 and 4 exemplify coding type 1, i.e. a simple concatenation of the `diff` values. However, in result, coding type 4 performed best using our training data-set. Using coding 4, a perfect transmission without retransmissions and with no out-of-order packets would look like "**1A1A1A1A1A1A...1A1A1A**" with $|S| = 400$, resulting in $|C| = 27$ and thus $\kappa = 14.81481$ (two characters per difference). Indeed, we observed that several legitimate flows (approx. 28%) had exactly this κ value and the related string generate from 200 sequence numbers each.

Sometimes, sequence numbers overrun. For instance, if a sequence number field has eight bits, the sequence number following 255 would be 0. These overruns are rare, but we filtered them by checking whether the difference in the current PDU number is larger than 5 times the previous PDU number difference (`olddiff`) using the expression `olddiff > diff*5`. This filter slightly improved detection results additionally. Moreover can this filter handle typical sequence number fields. We tested the algorithm with TCP (32 bit sequence numbers) and the CCEAP covert channel[4] tool (8 bit sequence numbers).

Finally, we compressed the concatenated string S using $C = \Im(S)$ and then calculated $\kappa = |C|/|S|$ as done in the original approach by Cabuk et al. (Fig. 2-5 to 2-6).

3.2 Parameterization

For each a flow, in order to decide whether a covert channel or a legitimate data flow is present, a threshold for the value κ must be defined. Therefore, we extracted 454 flows with at least 201 sequence numbers from the NZIX data set[5] and calculated their compressibility scores. Please note that the NZIX data are coming from a real Internet environment and contain PDUs which are reordered due to legitimate reasons. We compared the NZIX scores with the compressibility scores of three different covert channel types, modifying the order of 2 to 4 PDUs, both randomly to represent the transmission of encrypted content. As shown in Fig. 3, all types of covert channels resulted in comparably κ values: channels with 2 PDUs had a mean of 3.983 while those with 3 PDUs had a mean of 2.621 and those with 4 PDUs had a mean of 2.259. The NZIX training data resulted in a mean κ value of 10.674. However, as visible in Fig. 3, there is a clear overlapping of compressibility results between especially NZIX training data and covert channels using 2 PDUs. For this reason, we decided to parametrize our detection approach with thresholds in the range of $\kappa = \langle 2; 14 \rangle$[6], while the range $\langle 2.5; 5.5 \rangle$ was studied in detail since only few covert

---

[4]`https://github.com/cdpxe/CCEAP/`

[5]`ftp://wits.cs.waikato.ac.nz/pma/long/nzix/2/`

[6]The maximum achievable compression using `gzip` was slightly above 14.8, thus rendering higher values insignificant. For compression, we write the string S into a file and run `gzip -9 --no-name <filename>` to calculate C.
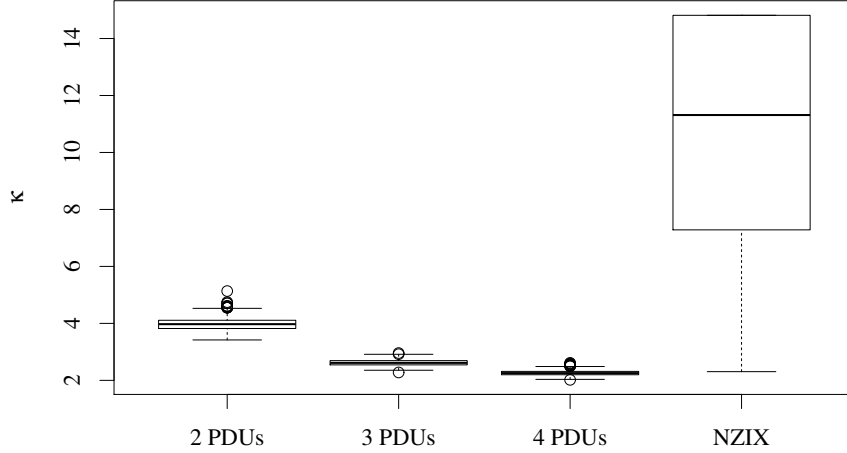
channel compressibility scores were above 5.5.

Figure 3: Analysis of NZIX training data in comparison to covert channels using sequences of 2 and 4 PDUs.

4 Experiments

We used the aforementioned tool CCEAP to create message ordering covert channels. Therefore, we extended the tool so that randomized sequence numbers that represent encrypted content could be created. We generated between 100 and 4000 pairs of 2, 3 and 4 PDU each, and recorded the generated traffic. Each flow configuration was repeated 20 times and with different inter-arrival times between 10ms and 500ms. Overall, we created 2,160 covert channel flows (720 for every 2, 3 and 4 PDU channel). We extracted the sequence numbers as described in Sect. 3 and calculated the compressibility scores for all these flows.

Afterwards, we extracted flows with at least 200 sequence numbers from NZIX traffic recordings, resulting in approx. 8,500,000 packets (approx. 1,100 flows).

Next, we compared the detectability of every CCEAP configuration with the NZIX recordings. Each time, the same number of 720 legitimate and 720 covert channel flows were used to calculate the detectability. Thus, for all thresholds (2, 2.5, 2.75, 3, 3.25, 3.5, 3.75, 3.9, 4, 4.025, 4.05, 4.075, 4.1, 4.15, 4.2, 4.25,

4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5, 5.5, 6, 7, 8, 9, 10, 11, and 12) we measure the detectability for message ordering channels utilizing 2, 3 and 4 PDUs.

The measurement of the detectability was based on the metrics precision, recall, accuracy and F_1-score and all these measures are visualized in the following figures. *Precision* is amount of flows correctly classified as covert channel flows (true positives, TP) in comparison to all flows classified as covert channels (TP and false positives, FP) while *recall* is the number of correctly classified covert channel flows in comparison to the correctly classified covert channel flows plus those flows that were classified as legitimate traffic but were actually covert channel flows, too (false negatives, FN).

$$
\begin{aligned}
\text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\
\text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}.
\end{aligned}
$$

Accuracy is the number of true classifications (positive and negative, i.e. TP and TN) in comparison to the whole population of flows. Finally, the F_1-*score* is the harmonic mean of precision and recall.

$$
\begin{aligned}
\text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\
\text{F}_1-\text{score} &= \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.
\end{aligned}
$$

4.1   Experimental Results

First, we tested all parameters for covert channels using two symbols. The results are shown in Fig. 4. Here, the threshold of $\kappa = 4.6$ performed best (also see Tab. 1 for detailed results). However, the achieved accuracy of 94.513% and F_1 score of 94.756% are not satisfying for large volumes of traffic. The thresholds $\kappa < 4.4$ and $\kappa \geq 4.9$ were not leading to acceptable results (accuracy and F_1 score below 94%).

As shown in Fig. 5, the optimal threshold for channels that were utilizing 3 PDUs appears to be better for lower values of κ than in case of channels utilizing 2 PDUs. For the 3 PDU channels, acceptable values (both, accuracy and F_1 score $\geq$94%) were achieved for $\kappa = 3$ to $\kappa = 4.8$ with the best results being achieved for $\kappa = 3$ ($\geq$99.23% accuracy and F_1 score).

For a more sophisticated channel that would utilize 4 symbols, the optimal threshold of $\kappa = 2.75$ was again lower (Fig. 6 and Tab. 1). Results for $\kappa \geq 3$ were equal to channels utilizing 3 PDUs as the maximum compressibility scores of both channels were each below the same maximum value. Acceptable results of $\geq$94% accuracy and F_1 score were achieved for $2.5 \leq \kappa \leq 4.8$.

Finally, we investigated how well a mixture of covert channels utilizing 2, 3 or 4 PDUs can be detected. Therefore, we combined the legitimate NZIX flows with covert channel flows utilizing 2, 3 and 4 PDUs. We used 50% NZIX flows and

9

Seq. Items	Accuracy	F_1 **Score**
$\kappa = 2.5$:		
2	49.722%	0%
3	55.833%	21.673%
4	99.027%	99.022%
$\kappa = 2.75$:		
2	49.513%	0%
3	92.847%	92.375%
4	**99.513%**	**99.516%**
$\kappa = 3$:		
2	49.236%	0%
3	**99.236%**	**99.241%**
4	99.236%	99.241%
$\kappa = 3.25$:		
2	48.958%	0%
3 & 4	98.958%	98.968%
$\kappa = 4.25$:		
2	90.555%	90.38%
3 & 4	96.18%	96.32%
$\kappa = 4.3$:		
2	91.805%	91.815%
3 & 4	95.833%	95.999%
$\kappa = 4.4$:		
2	93.333%	93.494%
3 & 4	95.416%	95.617%
$\kappa = 4.5$:		
2	94.166%	94.384%
3 & 4	95.138%	95.364%
$\kappa = 4.6$:		
2	**94.513%**	**94.756%**
3 & 4	94.93%	95.174%
C4.5 Classifier:		
2	95.9% (+1.39%)	95.9% (+1.14%)
3	99.54% (+0.30%)	99.55% (+0.31%)
4	99.84% (+0.33%)	99.8% (+0.28%)

Table 1: Detection results, depending on the number of sequence items for selected thresholds (improvement of C4.5 classifier over best manually selected threshold).
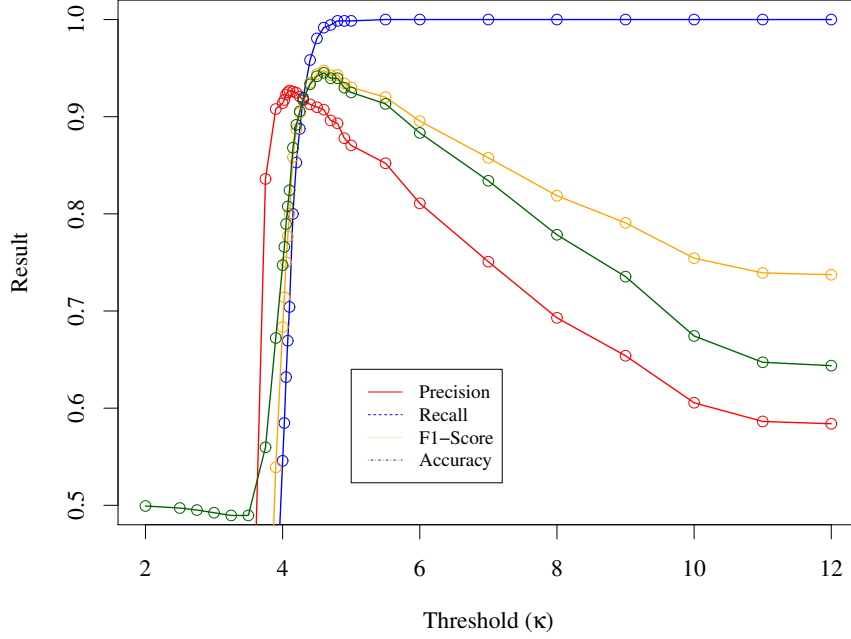
Figure 4: Detection results for 2-PDU message ordering channels.

50% covert channel flows, of which 1/3 were using the same number of PDUs each. Fig. 7 shows that the optimal threshold appears to be approximately around $\kappa = 4.5$. However, differences between $\kappa = 4.4$ to $\kappa = 4.7$ were only marginal (Tab. 2), indicating that the threshold is value is not highly sensitive.

4.2 Consideration of Realistic Conditions

Under realistic conditions, we can expect the percentage of covert channel traffic using the message ordering pattern to be very low. If we assume that amount of covert channel traffic is close 0%, then an important factor to be considered is the *false-positive rate* (FPR), i.e. the amount of legitimate traffic that was considered to be covert channel in comparison to all legitimate traffic:

$$\text{FPR} \quad = \quad \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

If the FPR is high, the number of false-positives could easily lead to extensive analysis work in an operational network.
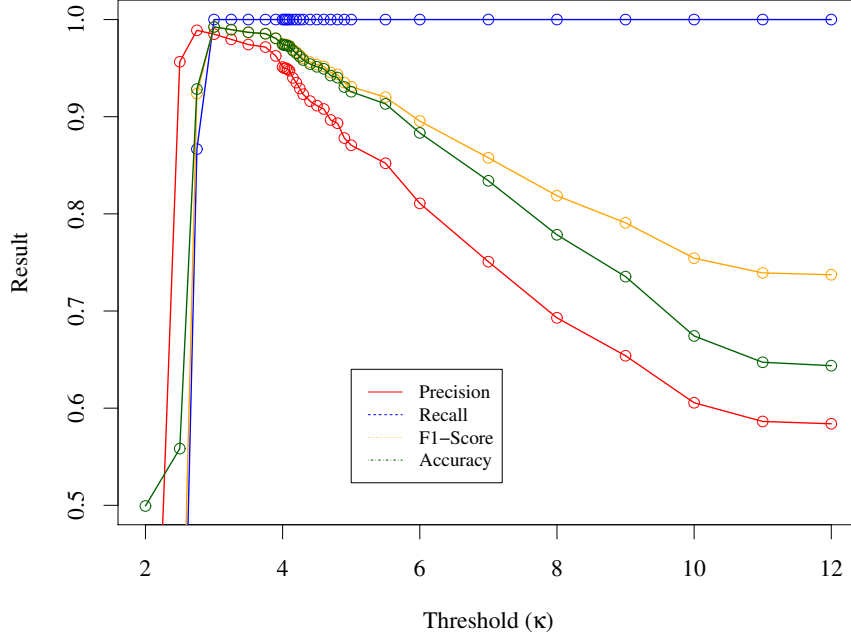
Figure 5: Detection results for 3-PDU message ordering channels.

We thus analyzed the FPR of legitimate-only traffic for different thresholds. Therefore, we analyzed 1,156 NZIX flows with ≥ 200 sequence numbers extracted from one split PCAP file with approximately 8,750,000 packets. The results are shown in Fig. 8. The FPR increases to more than 5% for thresholds $\kappa > 4.5$, rendering our approach impractical for these thresholds. However, for the thresholds 2 to 3.9 the FPR is between 0% and 3.719%.

If the optimal thresholds of $\kappa = 2.75$ to $\kappa = 3.0$ for channels using 3 or 4 PDUs are selected, the FPR is kept at 0.259% and 1.038%.

However, to detect channels that utilize only 2 PDUs, we optimally apply a threshold of $\kappa = 4.6$. This threshold leads to a FPR of 5.19% and thus would be impractical to handle in real-world scenarios as discussed in [16]. The overall optimal threshold for a *mixture* of covert channel types ($k = 4.5$) results in a FPR of 4.93%.

We thus conclude that our detection approach can be only considered practical under the following conditions:

- The detection of channels utilizing 2 or more PDUs is practical for very low traffic volumes such as in small enterprises or office networks due to
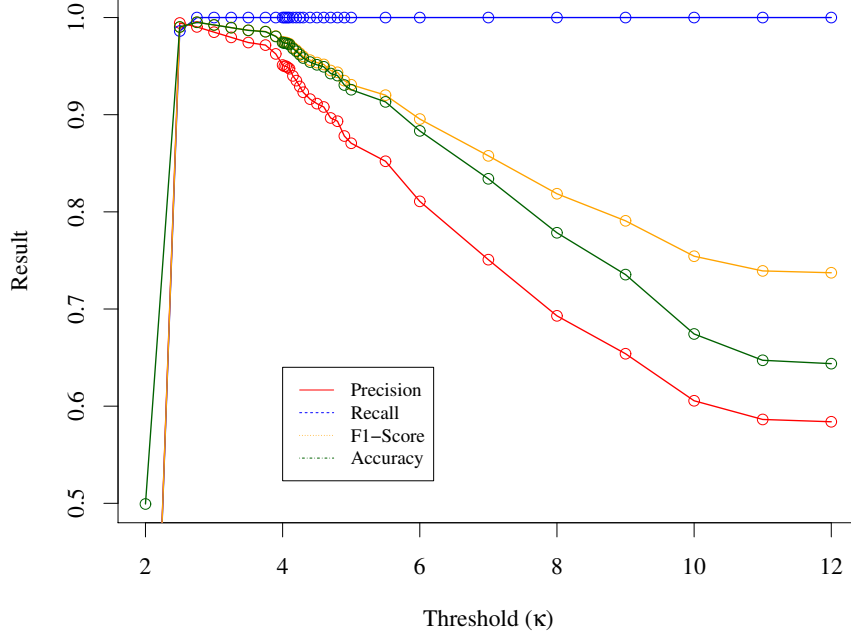
Figure 6: Detection results for 4-PDU message ordering channels.

the rather high FPR of 4.93%.

- The detection of channels utilizing 3 or more PDUs is practical in real-world scenarios in combination with a low-threshold due to a FPR of 0.259% and 1.038%. We expect that channels utilizing ≥ 5 PDUs are also providing low compressibility scores and should thus be detectable even easier with our approach.

In other words, channel types with the largest capacity can be detected the best while channels with the lowest capacity (utilizing only 2 PDUs) can be considered difficult to detect with this approach.

4.3 Further Optimization Using C4.5 Classifier

We trained the J48 classifier, which is an implementation of the C4.5 decision tree algorithm, using the tool `weka` with only one feature, i.e. the compressibility score of flows. We used only this single feature as no other traffic feature would directly be influenced by the message ordering pattern, potentially despite the
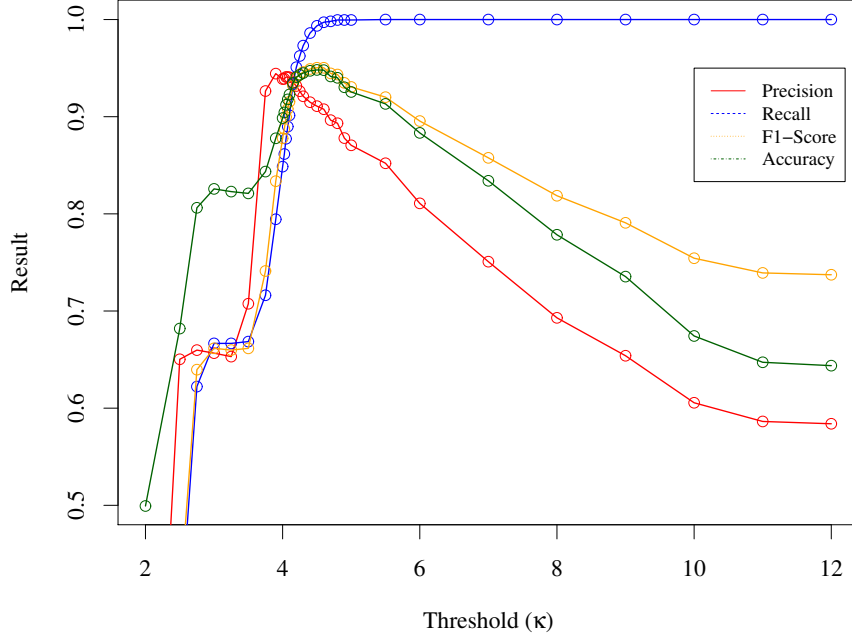
Figure 7: Detection results for a mixture of the three previously used covert channel types.

inter-arrival time of packets. However, even with only one feature, C4.5 can be used for two purposes:

1. To compare our selected threshold's performance with one or multiple thresholds automatically selected by C4.5. In this sense, we applied C4.5 as a means of quality control for our threshold-based detection approach. Our threshold-based approach provided good detection results but if our threshold was selected close to the optimum value, then its result should be only slightly lower than those of the C4.5 classifier that uses the same feature.[7]

2. To let C4.5 find suitable thresholds that we can later use for our threshold-based selection instead of our previously selected and discussed thresholds. Applying these new thresholds will further allow us to calculate the FPR.

---

[7]This is rooted in the fact that the classifier cannot only automatically determine a suitable threshold but can also divide the selection of legitimate and covert channel flows by applying several thresholds in a row.
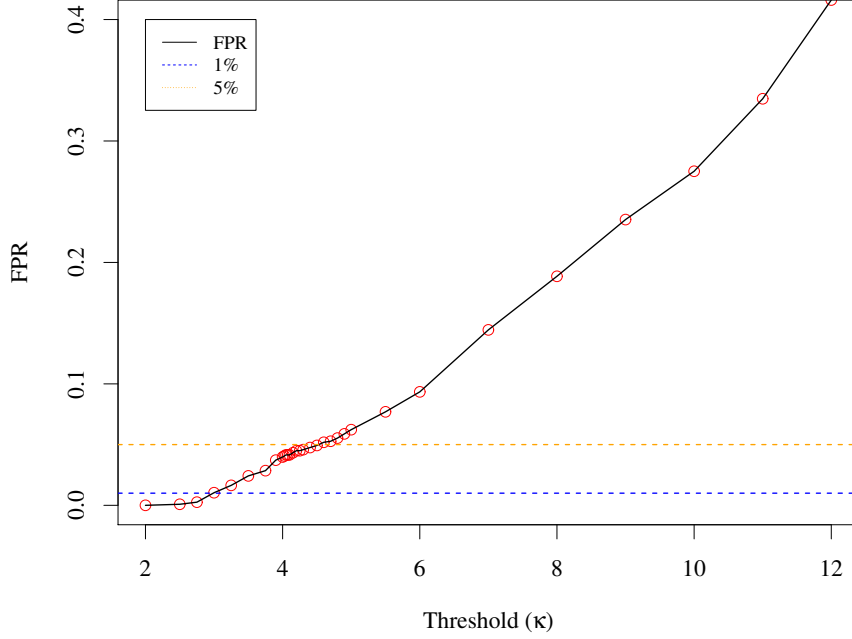
Figure 8: False-positive rate for our threshold-based detection.

First, we studied all three covert channel types separately. With 660 NZIX and 660 covert channel flows (each, i.e. per number of utilized PDUs) and a 10-fold cross-validation, we achieved a slightly higher detectability than using the previous thresholds. C4.5 resulted in slightly different thresholds for κ and most decision trees had only one decision element. For a channel with 3 symbols, the C4.5-selected κ-threshold was 2.88659 (previously 3.00), which resulted in a FPR of 0.605% (instead of the previous 1.038%). For a channel with four PDUs, the threshold for κ was 2.59047 (previously 2.75) and the linked FPR was even below 0.1% (0.086% instead of the previous 0.259%). This means that both thresholds were close to the original thresholds but improved the FPR values further.

The results increased even more for covert channels using two symbols (more than one percent improvement in accuracy and F_1 score). However, in comparison to the other two channel types, 2-PDU channels resulted in a larger decision tree and our results came from an non-pruned tree (over-fitted) and would decrease to the previous level after pruning.[8]

---

[8]The major threshold selected by C4.5 for the 2-PDU channel was 4.6 (i.e., exactly the

Fig. 9 visualizes the C4.5-based detectability for all 3 covert channel types depending on the number of utilized PDUs. For exact comparison, the results are also shown at the end of Tab. 1. Overall, the new thresholds provided an improvement of approx. 0.3% for accuracy and F_1 score of channels utilizing 3 or 4 PDUs.
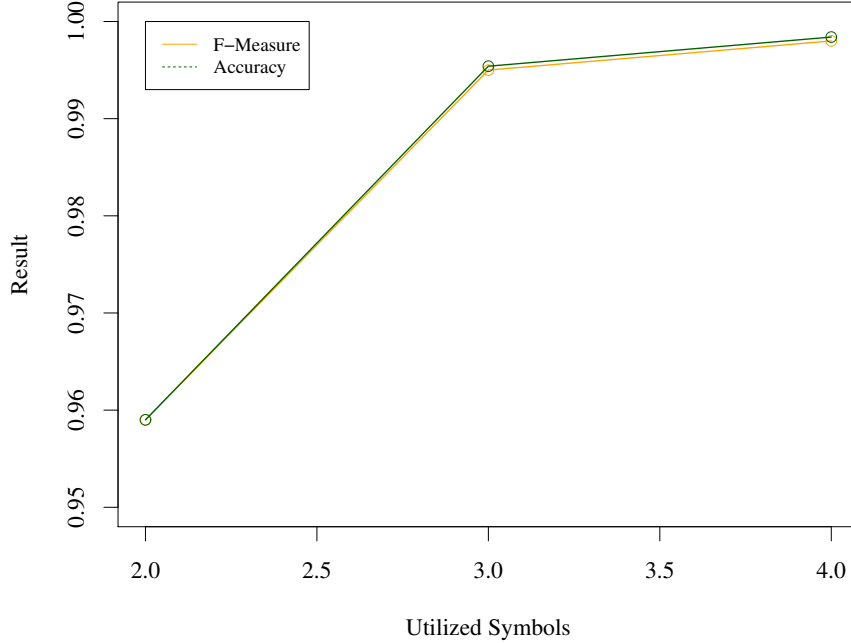

Figure 9: F_1 Score and Accuracy for D4.5, depending on the number of utilized symbols.

Finally, we compared the overall detectability of the threshold-based approach with the C4.5 classifier for all three covert channel types. As shown in Tab. 2, the C4.5 classifier outperformed our simpler threshold-based detection approach that used $\kappa = 4.5$ (improvement of 1.91% in accuracy and 1.66% in F_1 score). Again, we applied a non-pruned tree here to determine a maximum value. We can conclude that there is still slight room for improvement when it comes to the detection of a mixture of the three covert channels. However, even the detection results of C4.5 with an over-fitted tree were comparable to our simpler threshold-based detection approach.

one that we already selected).

16

Method	Accuracy	F_1 **Score**
Threshold, $\kappa = 4.4$	94.72%	94.91%
Threshold, $\kappa = 4.5$	**94.81%**	**95.04%**
Threshold, $\kappa = 4.6$	94.79%	95.04%
C4.5 Classifier	96.72% (+1.91%)	96.7% (+1.66%)

Table 2: Detection results over all covert channels, depending on the threshold.

4.4 Further Remarks

Implementation: Our detection approach can either be used on traffic recordings or in a real-time scenario. However, in a *real-time detection scenario*, our approach requires caching of potentially large flows since it uses a stateful algorithm (cf. [12, Ch. 8] and [6]). This renders a real-time detection for huge volumes of data difficult: an attacker could target the detection system by creating a high number of flows that must be cached by the warden, resulting in resource exhaustion, potentially leading to a DoS. However, if limits (e.g. on the number of cached flows/unit of time) are enforced, such attacks could be prevented.

On the other hand, if *traffic recordings* are provided and if they are not required to be processed in real-time, our detection approach can be easily used. This fact renders our approach suitable for forensic analyses, e.g. under the umbrella of the law-enforcement.

However, a key criterion that we observed during our experiments is the provision of enough RAM if large PCAP files must be processed. Our implementation (non-optimized shell scripts using basically a command pipeline with `traceconvert`, `tshark`, `grep`, `cut`, `sed` and `awk`[9]) was capable of handling PCAP files of few hundred MByte on an older SUN Fire X2100 server with 4 GByte of RAM, equipped with an AMD dual core Opteron 180 CPU (running Ubuntu 18.04 LTS) as long as the traffic recordings could be processed entirely in RAM. However, if large traffic recordings must be processed, then all flows and all sequence numbers of each particular flow must be enumerated, which is a computing-intense problem, consuming several days of computing time on our server.[10] Some NZIX recordings had a compressed size of 2 GByte and we needed to split these files into smaller pieces to process the files on our system (processing these files on a virtual machine with 8 GByte RAM was possible without splitting the files).

Application by LEAs: In general, we can assume that our simple heuristic provides good detection results for some of the studied channels. A detection

---

[9]Given enough RAM, an easy way of utilizing multiple cores would be to replace `grep` with `xargs`. Our implementation used only one CPU core.

[10]A more efficient sorting could most likely be achieved if *Apache Hadoop* (or similar distributed solutions) would be applied instead. However, performance was not a concern in our experimental evaluation.

approach using neural networks or similar concepts of AI research could potentially lead to better performing algorithms but also to more opaque ones [5], which can be considered a drawback, especially when explainable, court-proof evidence must be provided by LEA users. For this reason, we can consider the application of our simple algorithm useful for LEAs when the following aspects are taken into account: channels using 2 PDUs were linked to a FPR that is not useful for criminal investigations (5.19%). Even the small FPRs of 0.6% and 0.08% for 3 and 4 PDU channels could provide false evidence for non-criminals due to the chance for false-positives. However, we can assume that it is rather unlikely that only one message ordering flow would be exchanged over a longer period of several days between CS and CR. For this reason, larger traffic volumes should result in several 'positives'. However, the lower the compressibility score of a 'positive' flow, the more likely a covert channel is present. Thus, for criminal investigations, we recommend not only to determine the number of 'positives' but also their compressibility values. Moreover, the *more* positives, the higher the chance that CS and CR did in fact exchange covert information using message ordering.

5 Conclusion

Covert channels that manipulate the order of PDUs in a network flow appear as legitimate traffic as they do not modify any message's content that would not be modified by legitimate transmissions, too. This quality renders such channels attractive for malware and data exfiltration.

In this paper, we have provided the first countermeasure variation and threshold-based detection approach for such message ordering channels. Our detection method can be applied to every network protocol that can be exploited for a message ordering channel.

Our results have shown that the detectability depends on the number of utilized PDUs of the covert channel and different thresholds should be applied for these channels.

Covert channels utilizing 3 or 4 PDUs could be detected with more than 99.5% accuracy and F_1 score. The low false-positive rate of <1% and <0.1%, respectively, for these channels renders our approach even useful for scenarios with high volumes of traffic. This is important for a real-world application since these two channels can transfer the most information per packet. However, *channels utilizing 2 PDUs* could only be detected with an accuracy and F_1 score of 94.51% while their optimal threshold was linked to a false-positive rate of approx. 5.1%, rendering our approach not suitable in practice (except for targeted scenarios with a low traffic volume or when a lower true-positive rate would be acceptable).

In future work, we plan to evaluate whether the so-called ϵ-similarity can lead to higher detection results for the message ordering channels.

References

[1] AHSAN, K., AND KUNDUR, D. Practical data hiding in tcp/ip. In *Proc. Workshop on Multimedia Security at ACM Multimedia* (2002), vol. 2(7), ACM.

[2] CABAJ, K., MAZURCZYK, W., NOWAKOWSKI, P., AND ŻÓRAWSKI, P. 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, USA, 2018), ARES 2018, ACM, pp. 12:1–12:10.

[3] CABUK, S., BRODLEY, C. E., AND SHIELDS, C. IP covert timing channels: design and detection. In *Proc. 11th ACM conference on Computer and Communications Security (CCS'04)* (2004), ACM, pp. 178–187.

[4] CABUK, S., BRODLEY, C. E., AND SHIELDS, C. IP covert channel detection. *ACM Trans. Inf. Syst. Secur. 12*, 4 (Apr. 2009), 22:1–22:29.

[5] DANIEL E. GEER, JR. Unknowable unknowns. *IEEE Security and Privacy 17*, 2 (2019), 79–80.

[6] HANDLEY, M., PAXSON, V., AND KREIBICH, C. Network intrusion detection: evasion, traffic normalization, and end-to-end protocol semantics. In *10th USENIX Security Symposium* (2001).

[7] KATZENBEISSER, S., AND PETITCOLAS, F. A., Eds. *Information Hiding Techniques for Steganography and Digital Watermarking*, 1st ed. Artech House, Inc., Norwood, MA, USA, 2000.

[8] KRAETZER, C., AND DITTMANN, J. Mel-cepstrum based steganalysis for voip steganography. In *SPIE Proceedings* (2007), vol. 6505: Security, Steganography, and Watermarking of Multimedia Contents IX.

[9] KUNDUR, D., AND AHSAN, K. Practical internet steganography: data hiding in IP.

[10] MAZURCZYK, W., AND CAVIGLIONE, L. Information hiding as a challenge for malware detection. *IEEE Security Privacy 13*, 2 (Mar 2015), 89–93.

[11] MAZURCZYK, W., AND CAVIGLIONE, L. Steganography in modern smartphones and mitigation techniques. *IEEE Communications Surveys & Tutorials 17*, 1 (2015), 334–357.

[12] MAZURCZYK, W., WENDZEL, S., ZANDER, S., HOUMANSADR, A., AND SZCZYPIORSKI, K. *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. Wiley-IEEE, 2016.

[13] PANAJOTOV, B., AND MILEVA, A. Covert channels in TCP/IP protocol stack. *ICT innovations web proceedings* (2013), 190–199.

[14] Polak, L., and Kotulski, Z. Sending hidden data through WWW pages: detection and prevention. *Engineering Transactions 58*, 1-2 (2010), 75–89.

[15] Project, C. Criminal use of information hiding project website, 2019. http://www.cuing.org.

[16] Steinebach, M., Ester, A., and Liu, H. Channel steganalysis. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (New York, NY, USA, 2018), ARES 2018, ACM, pp. 9:1–9:8.

[17] Wendzel, S. *Network information hiding: Terminology, taxonomy, methodology and countermeasures.* Habilitation thesis, Department of Mathematics & Computer Science, Unversity of Hagen. http://www.wendzel.de/dr.org/files/Papers/thesis_with_cover.pdf.

[18] Wendzel, S., Eller, D., and Mazurczyk, W. One countermeasure, multiple patterns: Countermeasure variation for covert channels.

[19] Wendzel, S., Zander, S., Fechner, B., and Herdin, C. Pattern-based survey and categorization of network covert channel techniques. *Computing Surveys (CSUR) 47*, 3 (2015).

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

[21] Zillien, S., and Wendzel, S. Detection of covert channels in TCP retransmissions. In *Proc. 23rd Nordic Conference on Secure IT Systems (NordSec)* (2018), vol. 11252 of *LNCS*, Springer, pp. 203–218.