Decision Tree Rule Induction for Detecting Covert Timing Channels in TCP/IP Traffic

Félix Iglesias, Valentin Bernhardt, Robert Annessi, Tanja Zseby

Decision Tree Rule Induction for Detecting Covert Timing Channels in TCP/IP Traffic

Félix Iglesias, Valentin Bernhardt, Robert Annessi and Tanja Zseby

CN Group
Institute of Telecommunications, TU Wien
Gusshausstrae 25 / E389
1040, Vienna, Austria.
https://www.nt.tuwien.ac.at/research/communication-networks/
https://www.cn.tuwien.ac.at

Abstract. The detection of covert channels in communication networks is a current security challenge. By clandestinely transferring information, covert channels are able to circumvent security barriers, compromise systems, and facilitate data leakage. A set of statistical methods called DAT (Descriptive Analytics of Traffic) has been previously proposed as a general approach for detecting covert channels. In this paper, we implement and evaluate DAT detectors for the specific case of covert timing channels. Additionally, we propose machine learning models to induce classification rules and enable the fine parameterization of DAT detectors. A testbed has been created to reproduce main timing techniques published in the literature; consequently, the testbed allows the evaluation of covert channel detection techniques. We specifically applied Decision Trees to infer DAT-rules, achieving high accuracy and detection rates. This paper is a step forward for the actual implementation of effective covert channel detection plugins in modern network security devices.

Keywords: covert channels, decision trees, forensic analysis, machine learning, network communications, statistics

1 Introduction

Network communication platforms and protocols have been devised with clear structures and policies to allow fluent data transmission between different actors in networks. Covert channels profit from such structures and policies to send information in ways that are not in compliance with the original design of the communication schemes. The purpose of this unorthodox communication is that the covert data transfer remains generally unperceived but for the sender and receiver of the covert information. Ultimately, the pragmatic goal is usually—although not always—fraudulent or illicit, such as data exfiltration, or malware communication. For this reason, modern Intrusion Detection systems

(IDS) should incorporate covert channel detection capabilities to cope with those hidden communication methods.

In [11], detectors based on descriptive analytics of traffic (DAT) were presented as a broad-scope solution for covert channel identification in TCP/IP networks. DAT detectors operate by extracting meaningful information from network traffic and transform it into flow vectors. In such vectors, every susceptible TCP/IP field is represented by a set of features based on aggregations, statistics, autocorrelation, and multimodality estimations.

There exists a wide variety of ways to send covert information in IP networks. Diverse taxonomies and schemes to classify covert channels have been proposed in the literature over the years, e.g., [19] [30] [28]. More recent surveys are given in [24] and [11]. In [24], Wendzel et al. analyze 109 techniques developed between 1987 and 2013 and organize covert channel techniques as variations of 11 characteristic patterns. In [11], covert channels are differently classified from a detection perspective. The interested reader can find good overviews as well as links to multiple publications about the design and detection of covert channels in the cited papers. Beyond accurate classifications, from a global perspective covert channels were originally differentiated as: *timing* channels, if timing properties of communications mask the covert message, and *storage* channels, if covert data is somehow conveyed inside packets. In this work, we focus on *timing* channels.

In [11], DAT detectors were theoretically depicted. The evaluation was conducted with a proof of concept where the rules of a DAT detector prototype were adjusted solely based on experts knowledge, achieving moderate performance rates. Therefore, the main goals and contributions of this paper are:

1. To deeply evaluate DAT detectors for the specific case of covert timing channels. To this end, a testbed for generating and testing covert channels has been created. The evaluation has been conducted by implementing eight of the most popular covert timing techniques proposed in the literature.
2. To establish a methodology for tuning DAT detector parameters and rules based on Decision Trees learners.
3. To generalize a set of constituent rules for DAT detectors in the scope of covert timing channels, thus enabling the incorporation of DAT detectors in future IDS.

2 Related Work

Already in the '70s, covert timing channels were identified by the USA Air Force experts as a potential problem of secure communications even in end-to-end encryption scenarios [16]. Actually, devised without paying a special attention to security concerns, TCP/IP protocols entail many possibilities to conceal covert information. As a consequence, a considerable number of covert channel techniques have been presented in the related literature since then.

In the '90s, J. C. Wray noticed that the partition between *storage* and *timing* channels was not always appropriate, since there are covert channels that share properties of both types [25]. An example of half-timing half-storage channel is

given in [8], where a little delay is introduced in the packet delivery to make the lowest bit of TCP timestamps—usually seemingly random—to coincide with the covert bit to send. In this case, information is hidden in a TCP field, but the method invariably affects time properties in the communication. Note that, in this case, even if it is not possible for a warden to decode the message without checking packet fields, the detection by only analyzing packet inter-arrival times (henceforth *iats*) is theoretically possible. Another *hybrid* technique was proposed by W. Mazurczyk and K. Szczypiorski, who developed a method that inserts covert data in the payloads of VoIP packets that were intentionally delayed and normally dropped [15]. In [9], C. G. Girling tested a covert timing channel assuming that a sender could address a number of hosts in a network, and a wiretapper in the middle would interpret destination addresses as codifications (e.g., 16 addresses can conceal 4 bits of information).

Nevertheless, covert timing channels do not usually involve using information of packet fields. The attention is normally focused on iats. For example, in [3] the sender and receiver of the covert communication are synchronized and agree on a fixed sampling time interval. The existence or absence of a packet in the interval transports the covert information. V. Berk, A. Giani and G. Cybenko show a timing channel technique where covert 1s and 0s are deduced from two different packet delays [2]. Authors also discuss the *bandwidth commitment* in the design and implementation of covert timing channels, i.e., a low bandwidth is undesired for transmitting information but makes the covert channel stealthier.

Packet delays are also manipulated in [18], where a Jitterbug slightly modifies the time behaviour (in a millisecond scale) of hacked input devices, such as a keyboard, in order to allow an eavesdropper to guess the introduced data by observing the generated time sequence. This is a good example of how different malware can exploit covert channels. Video stream data is proposed as a carrier for sending binary covert information in [5], and a threshold for packet-iats is used to differentiate between 0s and 1s at the destination. Video on Demand services are used in [31] to transport covert information also by carefully manipulating iat values. In [26] message redundancy is exploited by applying Huffman coding to transform covert symbols into packet delays.

Given that detection methods are often based on statistics and distributions of the network usage, in [7] covert timing channels are created by matching time-distributions generated by legitimate services. In [13], Kiyavash and Coleman study a method for interactive traffic that relies on encoding mechanisms, statistical structures of network queues and packet iat for sending covert information. A different time-based approach is proposed in [14]. Here, bursts of packets transport the covert information, and the total number of packets in the burst represents the covert symbol. Additional timing-based convert channel techniques can be consulted, for example, in [10], [23] or [1].

The detection of covert timing channels has been classically faced from the perspective of the analysis of statistical properties. For instance, in [3], in addition to present some covert timing techniques, the detection of the proposed channels is conducted by means of calculations on variance patterns and mea-

suring similarity between adjacent iats. Entropy calculations are also used for such purpose, for example, Gianvecchio and Wang test entropy-based detection on three different techniques in [6]. More recently, in [4], *time-deterministic replay* has been introduced as a technique that can reproduce the precise timing of applications and, therefore, be used for disclosing covert timing channels. As for machine learning approaches, Support Vector Machines has been proposed for covert channel detection several times, e.g., in [21], and specifically for timing channels in [20], achieving good results against four popular timing techniques. Decision tree learners also obtain successful detection rates when tested with basic timing techniques in [29].

The detection framework presented in [11]—i.e., DAT detectors—focuses on the characterization of flows by means of a careful selection of statistical calculations and estimations. Therefore, it is a statistical approach in nature; nevertheless, in this paper we propose the application of machine learning to induce and adjust the rules embedded in DAT detectors, enhancing the basic proposal and making the most of the depicted framework. The excellent properties of Decision Trees for knowledge abstraction and generalization make them highly appropriated for the aimed purpose.

3 DAT Detector

DAT detectors are primarily introduced and widely described from a theoretical perspective in [11]. They are devised to be lightweight, fast traffic analyzers in network middleboxes. DAT detectors basically operate with descriptive analytics of traffic flows, creating flow vectors whose features are based on aggregations, statistics, autocorrelation, and multimodality estimations.

A DAT detector consists of three well-differentiated phases (Figure 1): preprocessing (P), feature extraction and transformation (T), and flow labeling and covert channel detection (D).

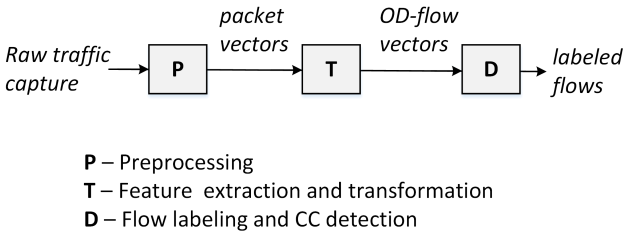

Fig. 1. DAT overall three-phase scheme.

1. *Preprocessing*
   The first phase takes raw traffic captures as inputs (e.g., PCAP files). Traffic

captures are parsed and packet vectors are formed with meaningful, homogeneous information for the subsequent analysis.
For the specific case of covert timing channels, the selected information must provide time characteristics and allow the identification of flows. Assuming that raw traffic is obtained as packet captures and a flow is simply defined by the source IP and destination IP tuple[1], the output of the preprocessing phase will contain packet vectors as shown in equation 1:

$$pkt_i = \{\text{timestamp}, \text{src.IP}, \text{dst.IP}\} \tag{1}$$

where pkt_i stands for the i^{th}-captured-packet, "timestamp" refers to the actual time when packet i was captured, and "src.IP" and "dst.IP" correspond to the flow source and destination IP addresses respectively.

2. *Feature extraction and transformation*
In this step, packet vectors are transformed into flows. At the same time, predefined calculations and estimations are conducted upon the flows. Finally, new two-level structured *OD-flow* vectors are created (origin-destination flow vectors). The first level corresponds to the packet fields to explore (e.g., IP Protocol, Time to Live, TCP Window, iat[2]), and the second level contains the calculations and estimations performed for every analyzed field. The general set of features proposed in [11] are:

U	number of unique values.
S_k	number of multimodality distribution peaks.
w_S	relative width of the main distribution.
S_s	number of main symbols.
Mo	statistical mode value.
$p(Mo)$	number of packets (or iats) with the mode value (percentage).
ρ_A	sum of autocorrelation coefficients.
μ_Δ	mean of differences.
$pkts$	total number of packets.

The proposed features may vary depending on the field to analyze. In our experiments only iats are required, and we specifically redefine some features for the iat field. For example, μ_Δ and Mo are not used, and $\mu_{\omega S}$ is a novelty and replaces w_S. $\mu_{\omega S}$ is the mean of standard distributions of peaks obtained by kernel density estimations. Therefore, $\mu_{\omega S}$ allows to easily abstract the average distribution width. In addition, we add c, which depends on S_k and $pkts$. c is an estimation of the number of potential covert bytes and

[1] Note that the classic 5-tuple used to identify communication flows is not used here (i.e., *src.IP, dst.IP, Protocol, src.Port, dst.Port*). Unlike overt IP communications, covert channels can be constructed using different protocols, source and destination ports in the transmission of the same hidden message.
[2] *iat* is treated like a header field for DAT detectors.

is calculated by means of simple ad-hoc tables [11]. In the cited work, c is incorporated in the last detection phase, but it can perfectly be estimated in the OD-flow generation and included in the vector. Therefore, for the specific case of covert timing channel detection, a OD-flow vector is expressed as (equation 2)[3]:

$$\text{flow\_vec} = \{U, S_k, \mu_{\omega S}, S_s, p(Mo), \rho_A, pkts, c\} \tag{2}$$

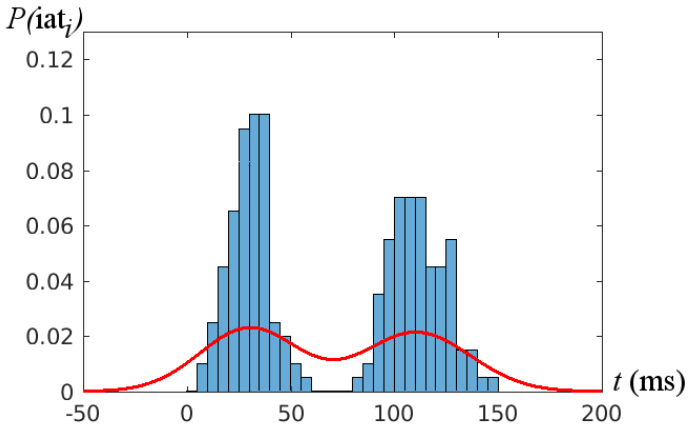

Fig. 2. iat-distribution of an example flow. *Probability* frequency diagram (blue) and kernel density estimation curve (red).

The meaning of the features can be better understood with an example. Imagine a flow of 200 packets from host A to host B (i.e., 199 iats). The detected iat-distribution matches Figure 2. Considering a granularity of 5 ms, out of 199 actual iats, the number of observed unique iat-values (U) is 25 (every bar of the frequency diagram or histogram). In Figure 2, the kernel density estimation (red curve) shows two peaks ($S_k = 2$) with an average standard distribution ($\mu_{\omega S}$) close to 20 ms (i.e., average width between 40ms and 50ms). S_s equals 13 according to the main symbols calculation proposed in [11] (i.e., it is approximately the number of outstanding histogram bars). The statistical mode for captured iats is 30 ms, which occurs 20 times, then $p(Mo) = \frac{20}{199} \approx 0.10$. The iats series is highly chaotic, showing $\rho_A = 0.05$. With $S_k = 2$ and $pkts = 200$, a proper estimation of c is 28 (according to the methods in [11]). The OD-vector would remain:

$$\text{flow\_vec}_{A \to B} = \{25, 2, 19.3\text{m}, 13, 0.10, 0.05, 199, 28\} \tag{3}$$

[3] Features in equation 2 should be annotated with the first-level field to which they correspond (i.e., U_{TTL}, $U_{src.Port}$, U_{iat},...). Since in this work we only use iats, we omit such subindices for the sake of clarity.

3. *Flow labeling and covert channel detection*
   A complete implementation of DAT detectors is expected to analyze traffic according to four different blocks or steps:
   - A *Packet Compliance Checker*, which, according to a set of fixed policies, detects traffic that is corrupted or does not comply to standard practices.
   - The *Intra-field Analysis* block, which checks TCP/IP header fields separately by examining the corresponding OD-flow vector values.
   - The *Inter-field Analysis* block, which considers combinations of OD-flow vector values in different TCP/IP header fields.
   - A *ML-based Detector* (machine-learning-based), which compares OD-flows by using a library that contains representative patterns (footprints) of OD-flows with covert channels. Such patterns are linked to known, published techniques.

   Here we aim to detect solely covert timing channels, so only a single TCP/IP flow field is to be analyzed: iats. It also means that only the *Intra-field Analysis* block must be adjusted and tested. Considering the specific case of covert timing channels, the evaluation of the *Intra-field Analysis*, the validation of Decision Trees as rule extractor-learners, and the proposal of a set of rules for actual implementations are the goals of this work.

4 Implemented Timing Techniques

We have created a testbed for the evaluation of covert channel detection techniques. Eight of the covert timing channels introduced in Section 2 have been implemented for the conducted experiments. We depict them with mnemonic names, using the first three letters of the author who originally published the technique and the corresponding reference. Note that covert channel generation manipulates packet delays in the communication source (aka inter departure times, henceforth abbreviated as *idts*), but detection uses iats; i.e., for two consecutive packets a and b, $iat_{ab} = idt_{ab} + (d_b - d_a)$, where d_a and d_b are the transmission delays of packet a and packet b.

The implemented covert timing techniques are:

- *Packet presence* (CAB)
  As presented in [3], the CAB technique requires synchronicity between the sender and the receiver, who agree on a fixed interval as sampling time. The absence or presence of a packet during an interval represents the binary symbol 0 or 1 respectively (Figure 3). In [3], different implementations for ensuring synchronization are suggested.

- *Differential/derivative* (ZAN)
  This differential covert channel was originally proposed for the IP-packet Time to Live field [27], but can be easily implemented by using packet delays. Given a basic idt (t_b), whenever a 1 is to be send the previous idt is

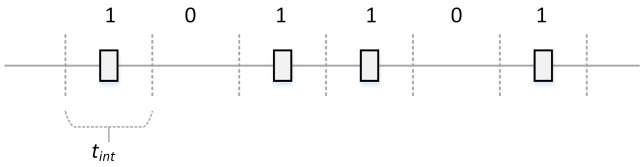

Fig. 3. CAB technique: presence or absence of a packet.

modified by adding or subtracting t_{inc}; in case of a 0, the last idt is kept the same, i.e., with no modification (Figure 4).

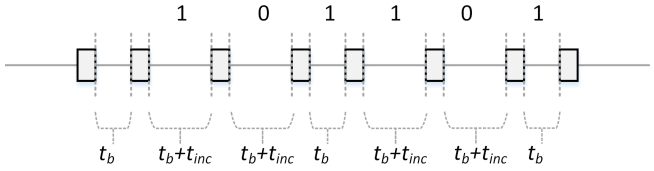

Fig. 4. ZAN technique: differential idt.

– *Fixed intervals* (BER)
Proposed in [2], this technique establishes fixed idts to represent 0s and 1s, respectively t_0 and t_1 (Figure 5).

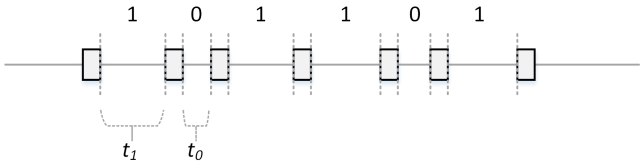

Fig. 5. BER technique: fixed idts.

– *Jitterbug/modulus* (SHA)
The Jitterbug [18] manipulates an existing transmission. It establishes a ground sampling time interval ω and adds a little delay to idts to make them divisible by ω or $\omega/2$ according to the covert symbol to send (Figure 6).

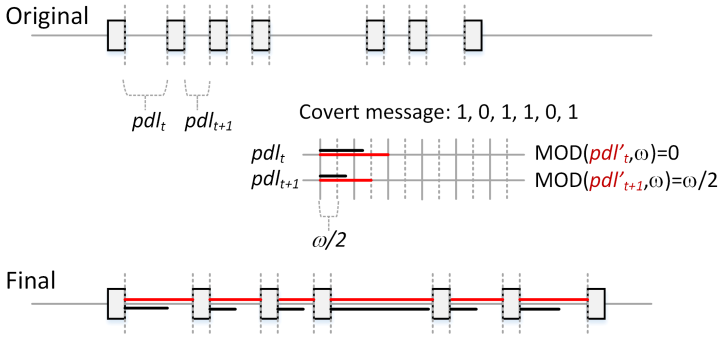

Fig. 6. SHA technique: jitterbug delay manipulation.

– *Huffman coding* (JIN)
This technique codes every covert symbol directly into a set of packets with different idts according to the frequency of the symbol and based on a Huffman codification. For the experiments we applied this technique only for plain text files and using the codification presented in [26] (Figure 7).

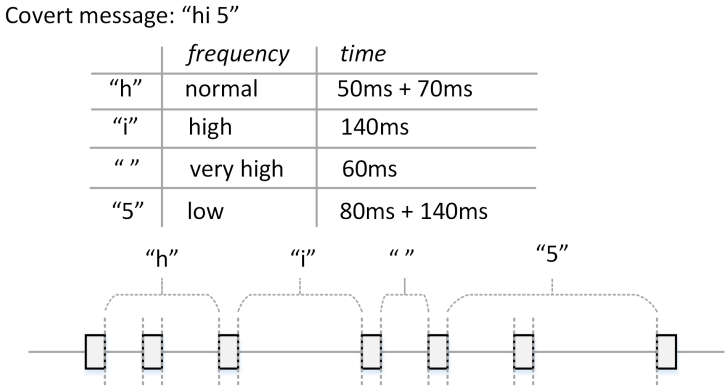

Fig. 7. JIN technique: Huffman coding.

– *Timestamp manipulation* (GIF)
By this technique [8], packet idts are manipulated based on the least significant bit (LSB) of the TCP timestamp. A minimum idt (t_b) between packets must be respected (authors propose, at least, t_b = 10ms). If the LSB coincides with the covert symbol, the packet is sent; if not, the condition is checked again after a time t_w (Figure 8).

– *One threshold* (GAS)
Devised for Android platforms and with video services as carriers, this tech-

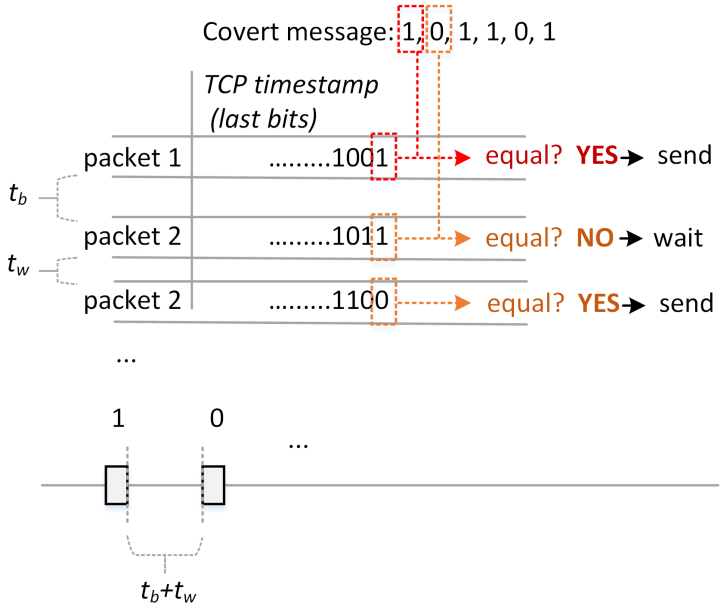

Fig. 8. GIF technique: timestamp manipulation.

nique establishes a threshold *th* for iats [5]. Delays above and below *th* will be considered 1s and 0s respectively (Figure 9).

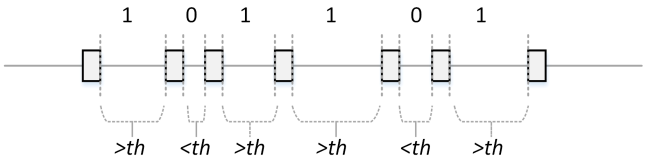

Fig. 9. GAS technique: one threshold.

– *Packet bursts* (LUO)
By this technique presented in [14], packet bursts are sent separated by a waiting time interval t_w. The number of packets in a burst directly represents the covert symbol or piece of code to send. For example, three packets in a burst are used to send the symbol "3" (Figure 10).

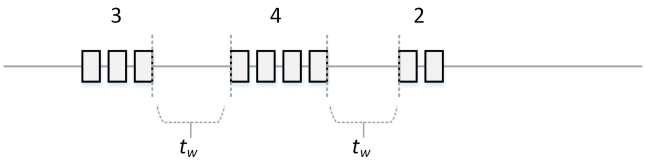

Fig. 10. LUO technique: packet bursts.

5 Experiments

The objective of the experiments was to create and refine DAT rules for the detection of covert timing channels. For this purpose, training and testing processes were performed with known—i.e., labeled—datasets. Figure 11 illustrates the training process, which consisted of three main phases: (1) dataset generation, (2) model obtaining, and (3) rule generalization.

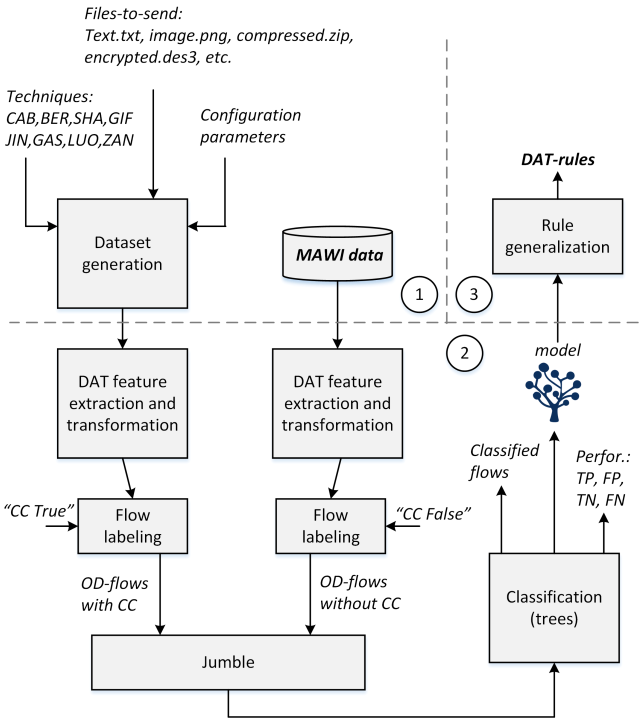

Fig. 11. Phases of the training process: dataset generation, model obtaining, and rule generation.

1. *Dataset Generation*
A testbed for traffic generation was built for creating IP flows containing covert channels as defined by the techniques published in the literature and briefly described in Section 4. In addition to selecting techniques, inputs included the files to covertly send and configuration parameters necessary for the setup. We have created two datasets (*cc_training* and *cc_testing*) with different files and random seeds. Selected files included different kinds of plain-text documents (poems, list of passwords, technical reports and programming scripts), images (PNG and JPEG), compressed files (in ZIP and GZ formats) and 3DES encrypted files. Parameters for the automated generation of flows with covert channels were randomly selected within the ranges defined in Table 1. The parameter ranges were adjusted according to the papers in which the techniques were originally presented or, if not documented, based on the knowledge of network traffic measurement experts.

Table 1. Parameters for the random generation of covert channels.

Technique	Parameters
d_i (delay)	By default, transmissions delays are modeled with a Lomax (Pareto Type II) distribution with $\alpha = 3$ ms and $\lambda = 10$ ms
CAB	$t_{int} \in [60, 140]$ ms
BER	$t_0 \in [10, 50]$ ms, $t_1 \in [80, 220]$ ms
SHA	pkt-dist.: Gamma with $k \in [40, 760]$ ms, $\phi \in [40, 360]$ ms $\omega \in [10, 90]$ ms
GAS	0-idts pkt-dist.: $t_0 = th - t_s$ 1-idts pkt-dist.: $t_1 = th - t_s + t_a$ $th \in [100, 300]$ ms, $t_s \in [60, 140]$ms, $t_a \in [20, 80]$ ms
JIN	Codification from [26]
LUO	$t_w \in [50, 250]$ ms Packets in a burst are sent every ms
ZAN	$t_b \in [30, 70]$ ms, $t_{inc} \in [30, 70]$ ms
GIF	$t_b \in [10, 30]$ ms, $t_w \in [4, 12]$ ms

As for the training sets that are free of covert channels, real network data was downloaded from the MAWI Working Group Traffic Archive. In the MAWI database[4], 15 minutes of real backbone traffic is published every day for research purposes. We downloaded traffic corresponding to Sun Feb. 5, 2017, and divided into three 5-minute datasets, from which we used one dataset for training (*nocc_training*), one for testing (*nocc_testing*), and discarded the third one.

[4] http://mawi.wide.ad.jp/mawi/

2. *Model Obtaining*
   During the second phase, OD-flows were extracted from the datasets, labeled, and jumbled before undergoing classification. To speed up classification, all flows with less than one packet were already discarded in this phase. For all explored datasets (with and without covert channels), the maximum observation window for a OD-flow was set to five minutes (the rest was simply discarded). The interested reader can freely download the used OD-flow datasets from [22].

   The objective in this phase was to obtain a classification model that could differentiate flows with covert channels from flows without. Since generalization was desired, Decision Trees were selected as classifiers. Decision Trees are known to be efficient learners, with good accuracy rates and, more importantly, they allow easy interpretation and rule extraction from results [12]. We used basic Decision Tree algorithms based on recursive partitioning instead of more complex options (e.g., Random Forest) because they are more interpretable and predictions are easier to explain and generalize. Another characteristic of Decision Trees that make them eligible for the current application is that they are embedded feature selection methods [17], i.e., they can potentially ignore features that are redundant or irrelevant.

   During the experiments, pre- and postpruning were performed to avoid overfitting and favor generalization.[5] In addition, a 10-fold cross-validation process was performed to reinforce disclosed models. During the training phase, models were obtained from the *cc_training* and *nocc_training* datasets.

3. *Rule Generalization*
   In the final phase, rules were extracted from the Decision Tree *complete* model. These rules were manually checked later and adjusted according to generalization criteria. An enhanced set of rules was obtained and implemented in DAT detectors.

   Once new rules were embedded, DAT detectors underwent an additional testing phase with reserved datasets. Such validation was conducted with completely new data (not used before): *cc_testing* and *nocc_testing*.

It is worth remarking that, even not among the purposes of the current work, the presented testbed enables the modeling of covert timing channel techniques and the obtaining of patterns for the *ML-based Detector* block introduced in Section 3.

[5] Details of the conducted parametrization are: Decision Trees used Information Gain (i.e., entropy-based) as splitting criterion; the minimal size for splitting was four samples; the minimal leaf size was two samples, allowing a maximal tree depth of 20 levels; the minimal gain for splitting a node was 0.1; the confidence level used for the pessimistic error calculation of pruning was 0.25, whereas the number of prepruning alternatives was three.

6 Results

The outcomes of the tests conducted in Section 5 were used to evaluate the performance of DAT detectors and the validity of the obtained constituent decision rules. Results were divided into two distinct phases:

1. Obtaining a rule-based decision model based on Decision Tree induction.
2. Validation of the generalized model with testing datasets.

6.1 Rule extraction experiments

Decision Trees generated the model illustrated in Figure 12. The corresponding performances indices are shown in Table 2. Results in Table 2 discloses that Decision Trees were able to capture significant patterns to discriminate between flows with and without covert channels. Provided training data is representative enough, the use of cross validation ensures the robustness of the obtained model and indices.

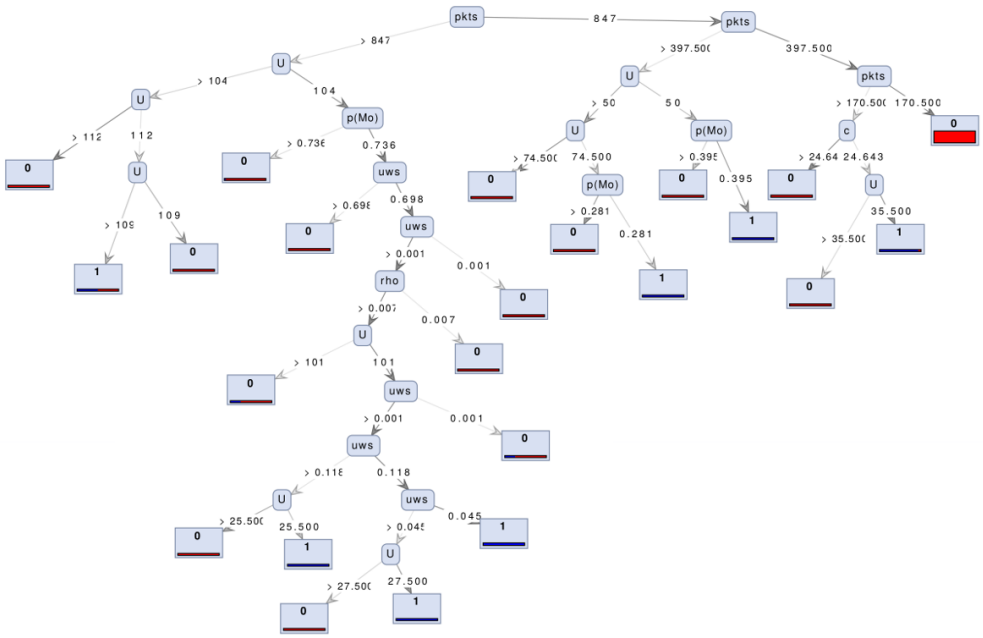

Fig. 12. Decision Tree obtained during the training phase. '1' corresponds to leaves that identify covert channels, '0' for leaves that identify normal traffic.

One of the first aspects that draws attention from Figure 12 is the absence of ρ_A, S_s and S_k in the solution model. Not using ρ_A is not surprising as iats

Table 2. Performance Indices of the complete model during training (datasets: *cc_training* and *nocc_training*).

	Predicted CC	**Predicted Normal**
Real CC	1015 (TP)	17 (FN)
Real Normal	7 (FP)	16302 (TN)

Accuracy	99.86% ± 0.05%
Precision	99.96% ± 0.04%
Recall	99.90% ± 0.06%
AUC (area under ROC)	0.996 ± 0.004

time series are chaotic and low self-correlated regardless of the class. This is due to the fact that, given a sampling resolution of 1 ms, delays in generation and transport invariably entail scenarios with a considerable entropy. The absence of S_s is also understandable due to same reason, i.e., transmission delays make iats to take approximate but different values even when coming from the same idt by design. In this respect, S_k is a better estimation of main symbols for the iat-case but, even so, it does not appear in the Tree either. Actually, S_k is indirectly affecting the model through other parameters, such as c and $\mu_{\omega S}$ (which depend on S_k). Decision Trees just avoided redundancy with respect to this variable (given to its capabilities for feature selection), but the use of multimodality by kernel density estimation was determinant for the detection.

6.2 DAT testing after generalization

The close examination of the model in Figure 12 allows to carry out a rule generalization like the one exposed in Figure 13. The generalized Decision Tree in Figure 13 condenses the detection in two main common patterns (green leaves). One of these patterns corresponds to flows that, during the observation scope (5 minutes), show a considerable amount of packets ($pkts > 170$) but low potential covert bytes ($c < 25$). This is idiosyncratic of flows with most iats very close to one center value, making S_k and S_s equal one or close to one (for instance, the packet bursts technique, LUO). However, the behaviour emphasized by this decision branch can also be matched by some legitimate services and is prone to generate some false positives (see Table 3).

The second branch, which concentrates most of the discovered covert channels, reveals that covert channels show distributions where packet-iats are in a range between 1 ms and 50 ms around the center values; also, the iat-mode-value represents less that 70% of iats, the number of estimated possible covert bytes must be high ($c > 50$) and the number of unique values (in ms resolution) is somehow below the estimated covert bytes ($U/c < 0.9$). This revealing last property suggests the footprint of a structure behind iat-values, which are generated by algorithmic patterns.

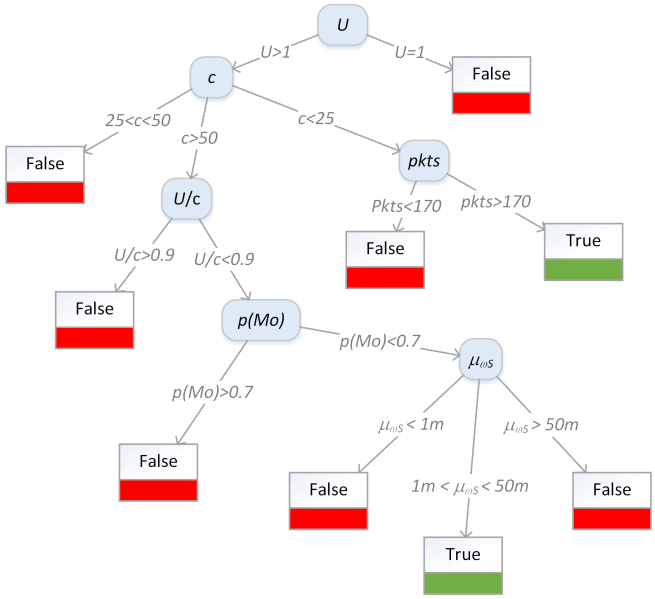

Fig. 13. Generalized Decision Tree implemented in final DAT detectors.

Note that in Figure 13 the $U > 1$ condition has been included, as such condition is not inferred by Decision Trees but previously imposed in the experiments (i.e., a flow with only one iat value cannot hide a covert timing channel). Except for this specific feature U, a certain fuzziness is naturally assumed when establishing rule thresholds. Therefore, in DAT detectors, instead of strictly applying the thresholds discovered by Decision Trees with a step function, the fitting of every feature to the given rules has been adjusted by using a smooth step function. Later, the final evaluation is obtained with the product t-norm among the different branch conditions.

Table 3 shows the results for the final validation experiments. Indices are promising for detectors that are designed to be lightweight detection barriers, although reveal some false positives to be corrected in final applications. Future implementations will gain robustness by the incorporation of fuzzy controllers in the DAT-decision making, whose capabilities to deal with vague thresholds make them perfect for the application. Detection accuracy is also expected to improve by the complementary analysis carried out by the *ML-based Detector* block, which stores a library with patterns related to known covert channel techniques. The development of such pattern library can be easily performed by the same testbed used together with Decision Trees rule extractors.

Table 3. Performance Indices of the generalized model during evaluation and testing.

DATA: *cc_training* dataset and *nocc_training* dataset

	Predicted CC	**Predicted Normal**
Real CC	990 (TP)	33 (FN)
Real Normal	94 (FP)	16320 (TN)

Accuracy	99.23%
Precision	91.33%
Recall	96.77%

DATA: *cc_testing* dataset and *nocc_testing* dataset

	Predicted CC	**Predicted Normal**
Real CC	995 (TP)	42 (FN)
Real Normal	99 (FP)	16013 (TN)

Accuracy	99.18%
Precision	90.95%
Recall	95.95%

7 Conclusion

This work tested and validated statistics-based framework—DAT detectors—for the detection of covert timing channels. To this end, multiple cover channels have been generated based on a set of eight popular covert timing techniques, which were then mixed up with real traffic captures. Decision Trees classifiers were used to infer rules from training data to embed in final implementations of the DAT detector decision making. The final testing with new datasets corroborated the fitness of DAT detectors. In short, this work complements and satisfactorily proves the theoretical foundations exposed in [11], paving the way for an easy integration of fast covert channel detection in modern IDSs.

Acknowledgments

The research leading to these results has been partially funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129, "BigDAMA".

References

1. Archibald, R., Ghosal, D.: A covert timing channel based on fountain codes. In: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications. pp. 970–977 (2012)

2. Berk, V., Giani, A., Cybenko, G., Hanover, N.: Detection of covert channel encoding in network packet delays. Rapport technique TR536, de lUniversité de Dartmouth p. 19 (2005)
3. Cabuk, S., Brodley, C.E., Shields, C.: IP covert timing channels: Design and detection. In: Proceedings of the 11th ACM Conference on Computer and Communications Security. pp. 178–187. CCS'04, ACM, New York, NY, USA (2004)
4. Chen, A., Moore, W.B., Xiao, H., Haeberlen, A., Phan, L.T.X., Sherr, M., Zhou, W.: Detecting covert timing channels with time-deterministic replay. In: Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation. pp. 541–554. OSDI'14 (2014)
5. Gasior, W., Yang, L.: Network covert channels on the android platform. In: Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research. pp. 61:1–61:1. CSIIRW '11, ACM, New York, NY, USA (2011)
6. Gianvecchio, S., Wang, H.: An entropy-based approach to detecting covert timing channels. IEEE Transactions on Dependable and Secure Computing 8(6), 785–797 (2011)
7. Gianvecchio, S., Wang, H., Wijesekera, D., Jajodia, S.: Model-based covert timing channels: Automated modeling and evasion. In: Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection. pp. 211–230. RAID '08, Springer-Verlag, Berlin, Heidelberg (2008)
8. Giffin, J., Greenstadt, R., Litwack, P., Tibbetts, R.: Covert Messaging through TCP Timestamps, pp. 194–208. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
9. Girling, C.G.: Covert channels in lan's. IEEE Trans. Softw. Eng. 13(2) (Feb 1987)
10. Holloway, R., Beyah, R.: Covert dcf: A dcf-based covert timing channel in 802.11 networks. In: 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems. pp. 570–579 (2011)
11. Iglesias, F., Annessi, R., Zseby, T.: DAT detectors: uncovering TCP/IP covert channels by descriptive analytics. Security and Communication Networks 9(15), 3011–3029 (2016), sec.1531
12. Kamber, M., Winstone, L., Gong, W., Cheng, S., Han, J.: Generalization and decision tree induction: efficient classification in data mining. In: Proceedings Seventh International Workshop on Research Issues in Data Engineering. High Performance Database Management for Large-Scale Applications. pp. 111–120 (1997)
13. Kiyavash, N., Coleman, T.: Covert timing channels codes for communication over interactive traffic. Acoustics, Speech, and Signal Processing, IEEE International Conference on 00, 1485–1488 (2009)
14. Luo, X., Chan, E.W.W., Chang, R.K.C.: Tcp covert timing channels: Design and detection. In: IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN). pp. 420–429 (June 2008)
15. Mazurczyk, W., Szczypiorski, K.: Steganography of VoIP Streams, pp. 1001–1018. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
16. Padlipsky, M.A., Snow, D.W., Karger, P.A.: Limitations of end-to-end encryption in secure computer networks, eSD-TR-78-158
17. Saeys, Y., Inza, I., Larraaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics 23(19), 2507–2517 (2007)
18. Shah, G., Molina, A., Blaze, M.: Keyboards and covert channels. In: Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15. USENIX-SS'06, USENIX Association, Berkeley, CA, USA (2006)

19. Shen, J., Qing, S., Shen, Q., Li, L.: Optimization of covert channel identification. In: Security in Storage Workshop, 2005. SISW '05. Third IEEE International. pp. 13 pp.–95 (Dec 2005)
20. Shrestha, P.L., Hempel, M., Rezaei, F., Sharif, H.: A support vector machine-based framework for detection of covert timing channels. IEEE Transactions on Dependable and Secure Computing 13(2), 274–283 (2016)
21. Sohn, T., Seo, J., Moon, J.: A Study on the Covert Channel Detection of TCP/IP Header Using Support Vector Machine, pp. 313–324. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
22. TU Wien CN Group: Data Analysis and Algorithms (2017), https://www.cn.tuwien.ac.at/public/data.html
23. Walls, R.J., Kothari, K., Wright, M.: Liquid: A detection-resistant covert timing channel based on {IPD} shaping. Computer Networks 55(6), 1217 – 1228 (2011)
24. Wendzel, S., Zander, S., Fechner, B., Herdin, C.: Pattern-based survey and categorization of network covert channel techniques. ACM Comput. Surv. 47(3), 50:1–50:26 (Apr 2015)
25. Wray, J.C.: An analysis of covert timing channels. Journal of Computer Security 1(3-4), 219–232 (1992)
26. Wu, J., Wang, Y., Ding, L., Liao, X.: Improving performance of network covert timing channel through huffman coding. Mathematical and Computer Modelling 55(1–2), 69–79 (2012), Advanced Theory and Practice for Cryptography and Future Security
27. Zander, S., Armitage, G., Branch, P.: An empirical evaluation of ip time to live covert channels. In: 2007 15th IEEE International Conference on Networks. pp. 42–47 (Nov 2007)
28. Zander, S., Armitage, G., Branch, P.: A survey of covert channels and countermeasures in computer network protocols. Commun. Surveys Tut. 9(3), 44–57 (2007)
29. Zander, S., Armitage, G., Branch, P.: Stealthier inter-packet timing covert channels. In: Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I. pp. 458–470. NETWORKING'11 (2011)
30. Zhiyong, C., Yong, Z.: Entropy based taxonomy of network convert channels. In: Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference on. vol. 1, pp. 451–455 (Dec 2009)
31. Zi, X., Yao, L., Pan, L., Li, J.: Implementing a passive network covert timing channel. Comput. Secur. 29(6), 686–696 (Sep 2010)