Journal of Universal Computer Science, vol. 25, no. 9 (2019), 1109-1130
submitted: 29/1/19, accepted: 9/7/19, appeared: 28/9/19 © J.UCS

Mobile Agents for Detecting Network Attacks Using Timing Covert Channels

Jędrzej Bieniasz, Monika Stępkowska
Artur Janicki, and Krzysztof Szczypiorski
(Division of Cybersecurity, Institute of Telecommunications
Warsaw University of Technology, Poland
{j.bieniasz, m.stepkowska, a.janicki, k.szczypiorski}@tele.pw.edu.pl)

Abstract This article addresses the problem of network attacks using steganographic techniques based on the manipulation of time relationships between IP packets. In the study, an efficient method to detect such attacks is presented. The proposed algorithm is based on the Change Observation Theory, and employs two types of agents: base and flying ones. The agents observe the time parameters of the network traffic, using proposed meta-histograms and trained machine learning algorithms, in the node where they were installed. The results of experiments using various machine learning algorithm are presented and discussed. The study showed that the Random Forest and MLP classifiers achieved the best detection results, yielding an area under the ROC curve (AUC) above 0.85 for the evaluation data. We showed a proof-of-concept for an attack detection method that combined the classification algorithm, the proposed anomaly metrics and the mobile agents. We claim that due to a unique feature of self-regulation, realized by destroying unnecessary agents, the proposed method can establish a new type of multi-agent intrusion detection system that can be applied to a wider group of IT systems.

Key Words: network security, traffic analysis, anomaly detection, intrusion detection, steganography, multi-agent systems
Category: C.2.0, K.6.5, I.2.0

1 Introduction

Ensuring a high level of network security is a must in these times of global penetration of cyberattacks. Cybercriminals constantly improve their methods. They search for new targets that can be easily attacked. They work on new ways of delivering, installing and controlling malware, in a way that these processes are difficult to detect by security systems. They use hidden channels to control botnets, for example, to realize a DDoS attack. Last but not least, they create hidden channels of communication to allow the leakage of sensitive data.

In many of the above described actions, steganographic techniques are used. They allow the fact of communication itself to be hidden [Lubacz et al., 2014]. Steganographic methods exploit the unused fields in various protocol headers (e.g., TCP, UDP, RTP) [Murdoch and Lewis, 2005], modify payload data or time relationships between arriving packets [Mazurczyk et al., 2012]. While there are already various methods of steganalysis which try to detect protocol stegano-

graphy or payload manipulation (e.g., [Janicki et al., 2015]), not much research has been conducted to efficiently detect timing network steganography.

This article aims to fill this gap. We would like to propose a system that will employ mobile agents. They will move within the network according to the proposed algorithm, to get closer to the attacker, in line with the Change Observation Theory and the moving observer concept [Szczypiorski and Tyl, 2016]. We will present a proof-of-concept of our method applied to detecting timing network steganography.

1.1 Network intrusion detection

As a response to the growing activity of cybercriminals, in recent years a variety of countermeasures against network intrusions have emerged. These solutions are usually referred to as intrusion detection systems (IDS). In general, they are divided into network-based IDS systems (NIDS) and host-based IDS systems (HIDS) [Kwon et al., 2017]. The former group of systems analyses the network traffic in selected nodes of the computer network, while the latter one is installed locally on a machine, usually a server. HIDS systems are dedicated to the given operating system and even to the specific services offered by the host, e.g., they can be dedicated to a web server. NIDS systems are by far more universal, which is why they gain significant attention of the scientific world, and also in this case, a focus will be devoted to them.

Another division in IDS systems is based on the detection method. While signature-based IDS systems try to identify similarities between the analyzed data and the patterns in the threat database, the anomaly-based IDS systems aim to detect anomalies in the network traffic [Bhuyan et al., 2014, Casas et al., 2012]. In addition, the current study will follow the latter approach, as it does not rely on any signature database, and, therefore, is able to detect new threats.

Many IDS systems search for anomalies when analyzing various parameters of the network data [Ahmed et al., 2016]. Some researchers use the basic netflow set of the IP network traffic parameters, such as the number of packets in the flow, duration of the flow or used port numbers. However, several studies show that IDS systems gain much in their effectiveness if the basic netflow set of parameters is extended by their various derivatives. In [Su, 2011], the authors showed that adding detailed parameters, such as ICMP checksum error count or count of the TCP packets with a sequence number equal to 0, significantly improved the detection of DoS attacks. Careful selection of the feature space is crucial to reach high effectiveness in the intrusion detection. However, to the best of our knowledge, hardly any IDS employs the measurement of time relationships between packets.

Several researchers showed the applicability of histogram analysis for intrusion and anomaly detection. In [Kind et al., 2009], the authors described which

parameters can be most useful in detecting network anomalies when observing their distributions using histograms. They showed that, e.g., histograms of the source and the destination port numbers can help in the detection of network scans or worm propagation, while histograms of the TCP flags can inform about the SYN flooding attacks or server failures, while the histograms of the flow duration values can reveal a DoS attack. The authors showed that histograms can be a synthetic tool to capture statistical information about a given parameter. In [Miller et al., 2011], the researchers found histograms of the network parameters useful in the detection of various HTTP attacks.

The problem of intrusion detection is often addressed using a machine learning (ML) approach [Buczak and Guven, 2016]. A typical procedure consists of training a ML-based classifier using data, usually labeled, acquired from life or emulated environments. Next, the classifier is tested using evaluation data. A large variety of ML algorithms have been proposed by researchers, starting from the basic as k-Nearest Neighbors (k-NN) and ending with genetic algorithms or various types of neural networks, such as multilayer perceptrons (MLP), Kohonen maps [Tsai et al., 2009] or deep learning techniques [Kwon et al., 2017].

1.2 Multi-agent cyber defense solutions

The proposed system combines the idea of network intrusion detection systems with the concept of multi-agent systems. Such approach has been investigated earlier in literature, where the biological inspirations have been involved. [Haack et al., 2011] introduced the system implementing swarming-agent-based approach to network infrastructure defense. They implemented lower-level software agents to carry out tasks which administrators would not be able to perform quickly enough to mitigate security threats. They called this method as *Cooperative Infrastructure Defense*, where the ant-based approach enables cooperation between administrators and agents to foster a collaborative problem solving. The same team of researches developed their investigations in [Fink et al., 2014], by providing more results and developing the state-of-the-art solution in ant-based cyber defense.

Cyber defense based on multi-agent systems is recognized as a modern and a very efficient approach. It gains interest of academics, industry, law enforcement agencies and even military. In [Kott, 2018], the author argues that intelligent autonomous agents will be the standard on the battlefield of the future. It means that intelligent autonomous cyber defense agents are going to become the main cyber fighters. The author introduced several novel ideas and summarized the other ones into a unified and reference architecture of such multi-agent system for cyber defense.

1.3 Attacks using steganography

Steganography conceals secret data by utilizing various features of different objects called *carriers*. It is generally recognized as the method of communication used by many kinds of adversaries or criminals. Other applications were considered as very specific or mostly theoretical. In recent years, evidence of the real applications of steganography has increased. To emphasize that steganography is a trending topic, recent reports warned about an emerging cyber threat of the application of steganography methods by malicious software designers [Kaspersky Lab, 2017, McAfee Labs, 2017, Jarvis, 2018].

Applying steganography for computer malware operations and communication enables to:

- avoid the operations of common security mechanisms, such as antivirus, IDS systems and firewalls, etc. Network traffic or multimedia files with hidden data would be recognized as benign network communication or data exchange, so they would be allowed.
- make detection more difficult or even evade forensic malware analysis.

Network steganography methods can be characterized by [Mazurczyk et al., 2016]:

- steganographic bandwidth – rate of transmitted secret data per time unit.
- undetectability – the level of inability of adversaries to compromise the method.
- robustness – the possible level for modifying the carrier of the steganographic data without distortion to that data.

Steganographic methods are subject to a trade-off between them following *a magic triangle* introduced in [Fridrich, 1999].

This paper focuses on one of the basic types of network steganography – so called *timing network steganography*. This kind of steganography utilizes varying timings of sending consecutive packets in the network streams. It means that for nearly every network protocol such a method can be applied. The parties in the communication secretly establish the algorithm of how to decode the embedded hidden data from the timing features of the network packet streams.

One of the examples of using time delays in steganography is the *Lost Audio Packets Steganography* (LACK) algorithm [Mazurczyk and Szczypiorski, 2008]. It utilizes intentionally delayed packets of VoIP streams to hide the steganograms. In the case of an extended delay, the endpoints of the VoIP communication consider such packets as lost and discard them. These packets do not take part in the reconstruction of a voice signal, and the steganograms can be retrieved from their payload.

1.4 Aim and structure of this study

In this study we propose an effective method for detecting attacks using timing covert channels, such as the ones presented in Section 1.3. Our method is based on mobile agents that can observe network traffic from various perspectives. According to [Mazurczyk et al., 2019], such a method can be classified as a combination of the passive warden concept [Anderson and Petitcolas, 1998] with capabilities of the network-aware active wardens [Lewandowski et al., 2007] in the scope of:

- sensing the topology of the network by agents and Central Unit;
- implementing interface to router logic in agents;
- defining states of detection logic and utilizing them into new decisions;

Using a testing environment, with various ML algorithms, we show experiments related to detecting timing network steganography attacks. For various testing scenarios, using a proof-of-concept, we will prove that the suggested method is able to effectively detect such a network attack. The proposed method is directly derived from the Change Observation Theory, which we presented in Section 2. Next, in Section 3 we propose our detection method. In Section 4 we describe the experimental setup used to prove out hypothesis. The results from the experiments are presented in Section 5. The article will end with conclusions described in Section 6.

2 Change Observation Theory

2.1 Introduction

Change Observation Theory combines ideas around objects, their features, changes in these features and observations of these changes. It introduces the universal framework for the evaluation of any algorithm for anomaly detection, which is one of the main efforts of academic and industrial research on cybersecurity. Anomaly detection is related to change detection events for which some of them are indicators of emerging cyber threats. The topic is well-established in the literature, but still applications to real problems such as detecting cyber threats are sought. One of the first, comprehensive introductions to this topic is presented in [Basseville and Nikiforov, 1993]. It established a unified approach for the design and the performance analysis of the algorithms for solving change detection problems defined as the problem of detecting a change in the parameters of a static or dynamic stochastic system.

The book [Poor and Hadjiliadis, 2008] is another contribution to the field of change observation theory basics. Its authors for the first time brought together

results from several sources. They defined an another unified treatment of several different approaches to the change detection problem. This section tries to summarize definitions and concepts of the Change Observation Theory, especially towards developing new anomaly detection algorithms for cybersecurity problems.

2.2 Definitions

In every object, features can be observed, which may be permanent, but some of them are under a process of change. The change is related to the transition of the feature into another. The observation of features is important if, and only if, information about the features' change can be utilized practically. It means that not all observations are equally important. Objects usually have many features, say N. We assume that the object has at least one feature ($N \geq 1$). An object without features cannot be the subject of observation, thus no change could be monitored.

In many cases it is impossible to be aware of all features of objects. The more complicated situation is when some features could become hidden whereas the others could appear. This aspect refers to the dynamism of features. Before any procedure of observation, the number M of features for practical measuring should be chosen, where $M \leq N$.

Let assume that the number is K, where $K \leq M \leq N$. In many cases, some of the features are not relevant to the observation of the change. The worst aspect would be the impossibility of measuring *the significance of the features*. Therefore, the initial phase of the problem could be stated as the effort to determine M number of features of an object and select a subset of size K of such features marked as the most significant for observation. Fig. 1a shows the presented idea of the object with four significant features (C_1, C_2, C_3, C_4), among which two are under observation (C_1, C_2).

Traditionally, observation is considered to be a time-oriented process. After distinguishing a feature, for example, C_1, a change in its properties is observed. A single unit of measurement is referred to as the state of a feature at the moment of measuring. Throughout the observation process, many samples of such states are collected and compared with one another. The essence of the described Change Observation Theory is to break the dependency of the observation process within a time domain. According to this theory, the key relationships between features are these that could be isolated from properties changing in time. The first step is to create an object model, and then determine the method for detecting change. The model of the object can introduce some inaccuracy in the mapping, which means the study of the object model does not coincide with the study of the object; however, it can provide relevant information. Features in the model

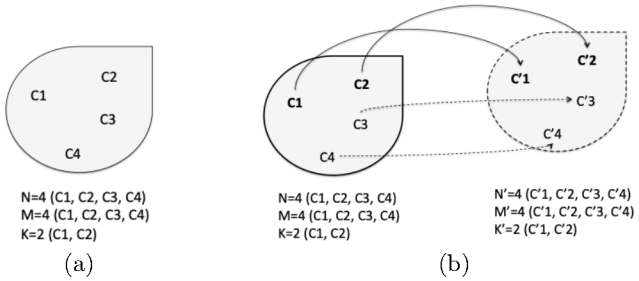

Figure 1: (a) Object with four features (C_1, C_2, C_3, C_4) among which two are under observation (C_1, C_2); (b) Model of object.

of the objects can be seen as relations. This idea is presented in Fig. 1b. For example, if C'_1 and C'_2 are integers, their relationship could be defined as:

- simple ratio of $\frac{C'_1}{C'_2}$;.
- correlation between these two values.

The correlation dependence between the features C'_1 and C'_2 is characterized by the fact that the values of one feature are strictly mapped to the mean values of the second feature. The purpose of the correlation analysis is to say whether there are any dependencies between the variables and the strength of this dependency. The established models of objects with relations between features can be used as a standard model in further applications. The appearance of other relationships between the features will be considered a change.

One example of applications of the presented theory to design a real anomaly detection method is the MoveSteg technique, as proposed in [Szczypiorski et al., 2015] and implemented in [Szczypiorski and Tyl, 2016]. It is based on the concept of *a moving observer* that can evaluate the measured features of the network flows to detect time-delay network attacks. The main result was to find the utilization of the observed facts in delay changes into the vector of the observer moving around the network. The aim of this was to discover the source of the attack. The results from the cited papers were the inspiration for the work in this paper to go further when designing a whole intrusion detection system based on Change Observation Theory.

3 Mobile agents – proposed approach

The main idea behind using mobile agents is to observe the network traffic from various points in the network, according to the Change Observation Theory, presented in Section 2. The agent will be responsible for collecting the data on

the traffic flowing through the router network interfaces and sending the preprocessed data to the central unit (CU) every T_{upload} time. The data preprocessing will include maintaining a network flow table and calculation of the statistics of the packet interarrival time values, e.g., by creating their histograms.

Two types of mobile agents are foreseen: *Base Agents* (BAs) – placed in the network nodes with the most intensive traffic, and *Flying Agents* (FAs) – sent temporarily to a network node by the CU or BA in case of a suspected anomaly, to perform a closer observation. Contrary to BAs, the FA's lifetime will be limited; when it has elapsed the FA instance will be deactivated, having obtained prior approval from the CU.

3.1 Observing time relationships between packets

To be able to capture anomalies in the packet interarrival times, a common approach is to analyze histograms of these values (see e.g., [Kind et al., 2009]), in which the frequency density for bin i, using Inverson bracket notation, is calculated as:

$$k_i = \frac{\sum_{j=0}^{N-1} [i \cdot \tau \leq t_j < (i+1) \cdot \tau]}{N} \tag{1}$$

where N denotes the number of packets within a certain time window, t denotes the time distance between the current packet j and the packet $j-1$, τ is the arbitrary set bin width.

However, our initial experiments revealed that such an approach did not allow the extraction of enough information to efficiently detect timing network steganography attacks. Therefore, we proposed that the mobile agents will calculate meta-histograms based on the original ones, using the following formula:

$$m_i = \begin{cases} k_c & \text{i} = 0 \\ k_{c-i} + k_{c+i} & \text{i} = \langle\ 1,\ \text{M}\ \rangle \\ k_{c-(i-M)} - k_{c+(i-M)} & \text{i} = \langle\ \text{M}{+}1,\ 2\text{M}\ \rangle \end{cases} \tag{2}$$

where c denotes the index of the gravity center of the original histogram and M denotes half the number of its bins. The proposed meta-histogram is supposed to capture in a better way the symmetry (or its lack) within the basic histogram.

A decision, whether a given meta-histogram represents a benign or a malicious network flow, can be made by a ML-based classifier. Prior to using it for detection purposes, the ML algorithm needs to be trained using labelled data.

If multiple network flows are analyzed, e.g., on a single router's interface, the ratio between positive decisions (*true positives* – *TP* and *false positives* – *FP*) of the classifier against the total number of the flows, can be treated as an anomaly metric d:

$$d = \frac{TP + FP}{all\ decisions} \tag{3}$$

This metric will be used later in this study.

3.2 Role of flying agents

When an FA is sent to a network router, it calculates the anomaly metric d, based on the algorithm proposed in Section 3.1, on each of the router's interfaces. If the d value on any of the router's interfaces is higher than in the node the FA came from (i.e., on the preceding router's interface), the FA continues to walk in the direction pointed by that interface, that is, towards the higher anomaly metrics. If the d value on all the router's interfaces is lower than the original value, the FA continues its walk in the random direction (different, however, from the agent's provenance).

Such a walk of the FA continues until the FA is **no longer** able to move, i.e., until it reaches a tree leaf (or gets to the last programmable router). It means that it is now likely to be close to the potential attacker. The FA continues to send its data to the CU. The CU, knowing that the FA is unable to walk any further, makes a decision based on the current d value. If it exceeds a certain d_{alarm} threshold, the CU detects a timing covert channel and performs the necessary actions. If the d_{alarm} threshold is not reached, the FA is deactivated.

3.3 Central unit

The central unit will be responsible for tracking the location of both types of agents (BAs and FAs), aggregating the data acquired from the mobile agents, and, last but not least, handling the main process of anomaly detection. The CU will implement the anomaly detection logic introduced in Sections 3.1 and 3.2. Its main role will consist of realization of the communication between the CU and the distributed agents. It will be in a major part hierarchical and realized in the following way:

- The CU allocates BAs to network nodes with the most intensive traffic.
- The agents observe the traffic, perform basic analyses and send the digested results to the CU.
- An FA can be sent to a network node close to a potential source of attack, following a request by the CU, or based on autonomous decision of a BA.
- The FA sends their observation results to the CU directly.
- The CU, based on the data acquired by the agents, may make a decision to trigger an alarm.

– When the T_{life} time passes, the FA asks the CU if it is allowed to deactivate. If the temporary observation is finished, the CU approves the request and removes the given FA from the registry.

3.4 Setting decision threshold

The proposed mechanism of activation and deactivation of FAs implies that the decision threshold of the classifiers needs to be properly set. In our case, higher false positives (FPs) can be tolerated, but false negatives (FNs) need to be low. Therefore we will prioritize setups yielding high recall scores.

This is a consequence of the fact that our system offers another layer of FP verification – deactivation of less effective agents. Therefore, spawning unnecessary agents is an acceptable cost of this approach. On the other hand, the number of FNs should be minimized, as missing an attack is highly undesirable.

4 Experimental setup

In this section, the setup of the tools for testing the scenarios and emulating steganographic attacks is presented. It consists of a network emulator and applications for network packet stream generation. Furthermore, the process of generating the dataset for evaluating the proposed method is described. It considers the format of the collected raw samples of the network flows.

4.1 Testing scenarios

To test the proposed method we used a network consisting of 28 nodes, serving as routers, and 16 client machines, representing both clients and servers, with the exception of one that acts as the CU node.

Figure 2 shows a sample network topology. Both servers and hosts were used to generate network traffic considered as legitimate for the purpose of the tests and attackers #1 and #2 were used to generate illegitimate traffic.

We tested the effectiveness of the method proposed in the two separate scenarios involving either Attacker #1 or Attacker #2. They both attacked the same Victim machine (Server #2), but the attacks did not happen simultaneously. Those attacks used the following traffic path:

1. Attacker #1 → R13 → R12 → R8 → R1 → R15 → R16 → R18 → Victim
2. Attacker #2 → R28 → R26 → R22 → R15 → R16 → R18 → Victim

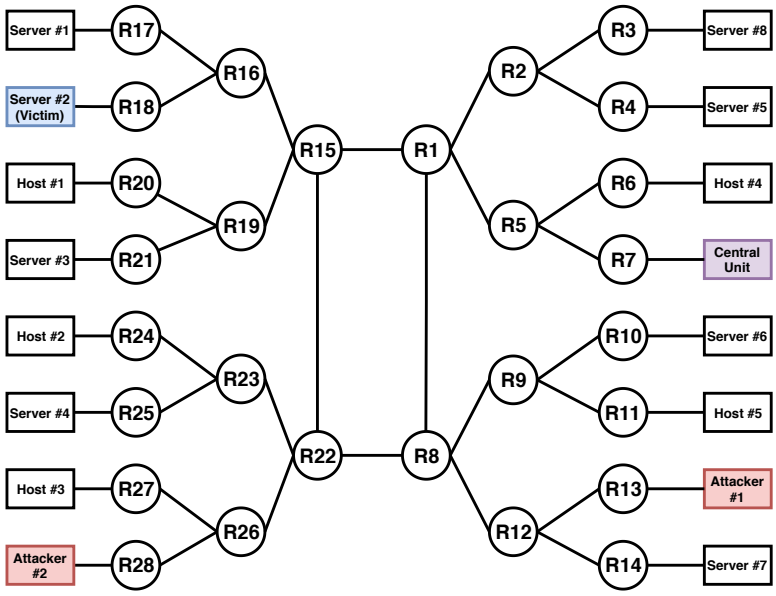

Figure 2: Sample network topology generated during emulations (blue denotes victim of attack, red – two separate attackers).

4.2 Emulating steganographic attacks

To emulate attack methods we used an open-source platform called Distributed Internet Traffic Generator (D-ITG) [Botta et al., 2012]. It was chosen due to its potential to define packet interarrival times. We emulated the steganography attacks with various percentages of traffic utilized for steganographic transmission, starting from 33% up to 100%. To execute this method we created a text file that consisted of values of interarrival times between packets. In Table 1 we present exemplary snippets of such files. They show values needed to create timing covert channels with an average interarrival time of 100ms. The value *null* means no steganographic bit assigned.

Mobile agents monitored the network traffic on the routers they were installed on and sent the data every minute to the CU. Apart from the meta-histogram data (see Section 3.1), the agents sent additional information, such as the timestamp, router ID, interface ID, source and destination addresses and ports etc.

The data, including both benign traffic and the malicious traffic were collected separately in two disjoint simulations with two attack scenarios in each of them, with 0.5% malicious traffic in the first simulation and 5.0% in the second

Table 1: Exemplary snippets of text files used by D-ITG to create steganography with different bitrate. Values in brackets denote mapping of time interval to steganographic bit.

Steganography 100%	**Steganography 66%**	**Steganography 33%**
50 [0]	50 [0]	50 [0]
150 [1]	100 [null]	100 [null]
50 [0]	150 [1]	100 [null]
150 [1]	50 [0]	100 [null]
50 [0]	100 [null]	100 [null]
150 [1]	150 [1]	150 [1]

Table 2: Datasets collected and utilized during experiments in this study.

Simulation	**Attack path**	**Dataset tag**	**Utilization of datasets**
Simulation 1	R13 - R18	Dataset 1	80% used for choosing hyperparameters (GridSearch with CV) and training classifiers 20% of used for testing classifiers
	R28 - R13	Dataset 2	80% used for training classifiers 20% used for testing
Simulation 2	R13 - R18	Dataset 3	100% used for: - final testing of classifiers - testing the proposed attack detection method
	R28 - R13	Dataset 4	

one. This resulted in four datasets in total (see Table 2 for summary). Datasets 1 and 2 were used for the training and tuning of the ML-based classifiers. Datasets 3 and 4 were used for the final testing of the trained classifiers and to verify if the proposed attack detection method was effective.

4.3 Machine learning algorithms

Several ML algorithms were considered as classifiers in the proposed method. After initial tests, the chosen MLs were SVM, Random Forest, Bernoulli Naïve Bayes, Gaussian Naïve Bayes and MLP, all from the supervised learning algorithm group. They were implemented using the Python `scikit-learn` library [Pedregosa et al., 2011].

Special attention was paid to selecting a data subset from the training data to train the ML algorithms, which is usually critical to the effectiveness of a ML-based classifier. Several options were considered, such as choosing the data that

were acquired in the direct vicinity of the attacker, remotely from the attacker, or as a mixture of the above cases.

5 Experiments, results and their discussion

5.1 Evaluation metrics

Selecting the best set of hyperparameters for each ML-based classifier, training the models and testing the classification performance were based on standard metrics:

- Receiver Operator Characteristics (ROC) curve and area under ROC curve (AUC). The ROC curve shows the relationship of the TP rate (TPR) against FP rate (FPR) for various decision thresholds. An AUC value of 1 means a perfect classification; the lower the AUC, the more misclassifications that can be expected. An AUC exceeding 80% is often considered as acceptable [Tape, Thomas G., 2017].
- Equal Error Rate (EER) yields the error value for which the classifier has the same probability of misclassifying a positive as well as a negative sample.
- accuracy – the ratio of the number of correct detections to all detections.
- attack recall (TPR for attack detection) – the ratio of correctly detected attacks referred to all attack samples.
- precision – informs how often the classification was correct.
- F_1 score – harmonic mean of the precision and recall.

5.2 Training and tuning ML-based classifiers

To tune the selected classifiers, the methodology of data science analysis was applied. As the initial experiments revealed that the detection algorithm worked best if trained on data acquired close to the attacker, the data from the nodes adjoining the attackers from Dataset 1 and Dataset 2 were selected for training purposes.

The hyperparameters of the classifiers were tuned using a Grid Search algorithm with 10-fold cross validation (CV) procedure (for the SVM, the CV order was set to 3 as a higher value caused overfitting). This step ensured that all the algorithms worked in the best condition possible for the given data. GaussianNB was excluded from this process as it did not have parameters that could be tuned. The best hyperparameters for each classifier were chosen according to the mean scores of the metrics in the following order of importance:

Table 3: Evaluation metric results from hyperparameters optimization process of GRID SEARCH with cross validation.

	Mean					
ML model	**AUC**	**Accuracy**	**Recall**	**Precision**	F_1	**Hyperparameters results**

BERNOULLINB	0.9036	0.9416	0.7550	0.4128	0.5337	`alpha: 0,`
						`binarize: 0,`
						`fit_prior: True`

GAUSSIANNB	0.7500	0.5409	0.9363	0.0829	0.1534	`default parameters`

SVM	0.9130	0.9562	0.1445	0.4108	0.1990	`C: 1,`
						`decision_function_shape: ovo,`
						`gamma: 2,`
						`kernel: poly,`
						`shrinking: False`

RANDOMFOREST	0.9776	0.9818	0.6307	0.9355	0.7507	`bootstrap: True,`
						`criterion: entropy,`
						`max_depth: 48,`
						`max_features: log2,`
						`min_samples_leaf: 2,`
						`min_samples_split: 2,`
						`n_estimators: 36`

MLP	0.9681	0.9812	0.7385	0.8196	0.7753	`activation: tanh,`
						`alpha: 1e-05,`
						`hidden_layer_sizes: (100, 40),`
						`learning_rate: adaptive,`
						`max_iter: 1000,`
						`solver: adam`

Table 4: Classification results for testing part of DATASET 1 for various ML algorithms trained on data from R13.

ML model	**AUC**	**Accuracy**	**Recall**	**Precision**	F_1
RANDOMFOREST	0.8652	0.9834	0.7361	0.8548	0.7910
BERNOULLINB	0.8493	0.9403	0.7500	0.3941	0.5167
GAUSSIANNB	0.7226	0.5325	0.9305	0.0785	0.1448
SVM	0.8404	0.9739	0.6944	0.6944	0.6944
MLP	0.8591	0.9716	0.7361	0.6463	0.6883

1. AUC, as most universal metrics.
2. Recall, as we wanted to minimize number of FNs.
3. F_1 and precision, which were less important in our case.

The best hyperparameters identified for the considered ML classifiers are presented in Table 3. The classification results for these classifiers are presented in Table 4 for the testing part of DATASET 1 for node R13. Figure 3 shows a

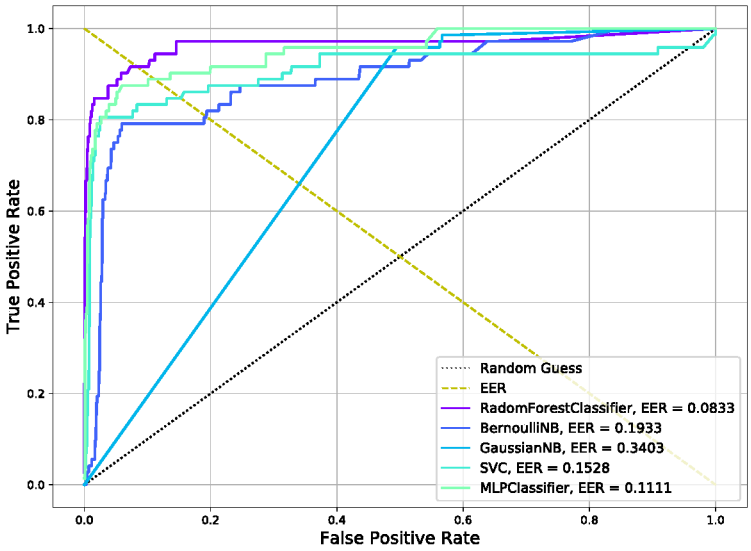

Figure 3: ROC curves calculated on testing part of DATASET 1 data for classifiers trained on R13 data from training part of DATASET 1.

sample ROC curve with EERs for the testing of classifiers for models trained on the training part of DATASET 1 for node R13.

The best detection was achieved for the MLP, Random Forrest, SVM and Bernoulli Naïve Bayes classifiers – they all yielded high values of AUC (above 80%), accuracy (above 90%) and recall (70-75%). It is noteworthy that the results for the testing data are only slightly inferior to those for the training data, still yielding high recognition scores for these classifiers. In contrast, the Gaussian Naïve Bayes algorithm performed worse, with ca. 72% AUC and accuracy of 55% only.

5.3 Results for detection of timing network steganography attacks by mobile agents

We analyzed how the trained classifiers would perform if used for detecting timing network steganography attacks by mobile agents. We compared two sets of models: trained on R13 data and trained on R28 data (i.e., using DATASET 1 and DATASET 2, respectively). Therefore we considered four cases:

1. Attack path from R13 to R18 with model trained on DATASET 1.

Table 5: Anomaly metric d along attack path and on other routers in cases 1 and 2 with ML classifiers: (A) BernoulliNB, (B) GaussianNB, (C) SVM, (D) RandomForest and (E) MLP.

		Metrics d along attack path							**Metrics d on other routers**						
Case	**Model**	**R13**	**R12**	**R8**	**R1**	**R15**	**R16**	**R18**	**R14**	**R22**	**R9**	**R2**	**R5**	**R19**	**R17**
1	**(A)**	32.42	15.30	13.62	11.65	10.64	19.94	29.79	9.75	6.17	6.03	5.06	5.16	2.73	0.00
	(B)	45.94	41.36	40.33	36.85	36.40	33.75	31.39	40.72	37.31	42.50	37.05	42.53	37.96	0.00
	(C)	1.70	0.24	0.20	0.32	0.54	0.42	0.38	0.24	0.30	0.35	0.46	0.20	1.01	0.00
	(D)	6.86	1.46	0.24	0.09	0.07	0.22	0.19	0.02	0.08	0.09	0.01	0.00	0.07	0.00
	(E)	3.50	1.18	0.47	0.51	0.63	0.74	0.58	0.59	0.53	0.58	0.70	0.41	1.14	0.00
2	**(A)**	34.12	15.56	13.85	12.62	10.92	20.78	31.66	9.42	6.12	6.033	5.14	5.16	2.73	0.00
	(B)	45.94	41.36	40.33	36.85	36.40	33.75	31.39	40.72	37.31	42.50	37.05	42.53	37.96	0.00
	(C)	6.25	1.08	0.64	0.78	1.04	0.95	0.86	0.88	0.72	1.08	1.09	0.49	2.00	0.00
	(D)	13.50	3.84	1.80	1.06	1.09	1.74	2.07	0.38	0.58	0.60	0.56	0.21	0.42	0.00
	(E)	10.87	3.54	1.62	1.22	1.15	1.56	1.30	1.32	1.13	1.30	1.25	0.74	1.30	0.00

2. Attack path from R13 to R18 with model trained on Dataset 2.
3. Attack path from R28 to R18 with model trained on Dataset 2.
4. Attack path from R28 to R18 with model trained on Dataset 1.

Tables 5 and 6 show the anomaly metrics d for various classifiers for these cases. Figure 4a-4f display these results graphically – as an example for case 3 and the RandomForest classifier, which was among the best performing ones. The red color shows the nodes where the mobile agents yielded the highest results. The yellow color marks nodes where the FAs were spawned according to the proposed algorithm. The green nodes denote routers where the FAs would be deactivated, as the anomaly metric d would be lower than on the node from which the FA was sent. It can be seen from the last figure that red nodes were perfectly aligned with the path leading to the attacker.

5.4 Discussion

The first step – selecting and tuning the ML-based classifiers – was very important for the correct functioning of the attack detection algorithm, as the proposed anomaly metric d is based on the classifier's positive decisions (both TPs and FPs). Therefore, they need to be as accurate as possible. RandomForest and MLP turned out to perform best in our case. Naïve Bayes algorithms gave the worst results and would not work properly in our solution. The other models presented some drawbacks, e.g., too low a number of positive detections (SVM) which could result in missing an attack. The GaussianNB classifier should also be avoided, as its low accuracy would spawn too many unnecessary FAs. Even though our proposed system is able to regulate this by the mechanism of the

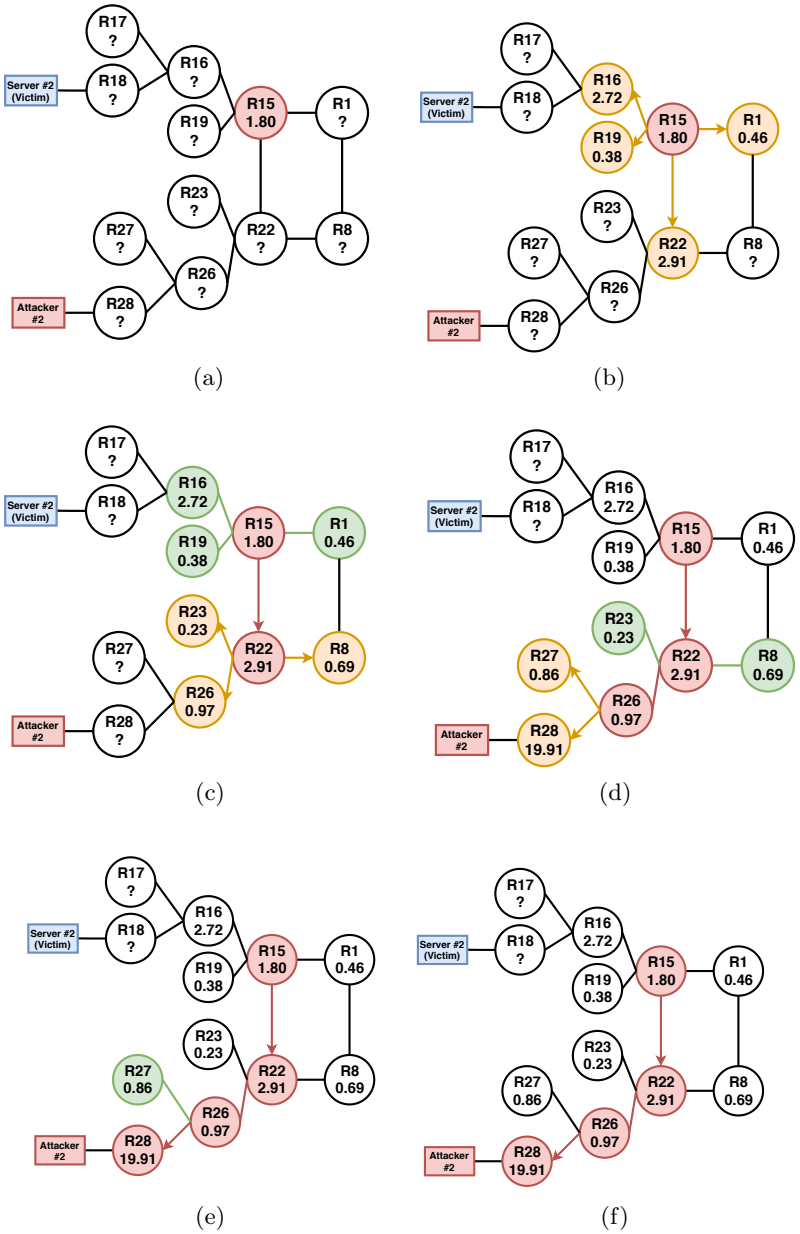

Figure 4: Anomaly metric results from the attack by R28 to R18, with RandomForest model trained on Dataset 2. (a)-(f) present consecutive phases of BAs and FAs cooperation to identify the attacker. Red nodes mean that BA or FA yielded a high value for metric d and raised the attention. Yellow node means spawning of FA. Green node means that a FA ends its life.

Table 6: Anomaly metric d along attack path and on other routers in cases 3 and 4 with ML classifiers: (A) BernoulliNB, (B) GaussianNB, (C) SVM, (D) RandomForest and (E) MLP;

		Metrics d along attack path						**Metrics d on other routers**					
Case	**Model**	**R28**	**R26**	**R22**	**R15**	**R16**	**R18**	**R27**	**R23**	**R8**	**R1**	**R19**	**R17**
3	**(A)**	23.60	19.92	11.80	8.79	17.14	24.96	7.40	4.95	6.87	5.29	2.70	0.00
	(B)	49.81	47.53	41.55	40.51	41.28	41.44	43.48	40.49	45.21	39.77	42.44	0.00
	(C)	12.65	3.49	0.82	1.22	1.14	1.05	1.53	0.81	0.86	0.92	2.28	0.00
	(D)	16.91	9.77	2.91	1.80	2.72	2.78	0.86	0.23	0.69	0.46	0.38	0.00
	(E)	16.90	9.19	2.70	1.88	2.29	2.16	1.52	0.77	1.28	1.03	1.46	0.00
4	**(A)**	23.26	19.61	11.61	8.67	16.69	24.18	7.32	4.95	6.93	5.29	2.69	0.00
	(B)	50.00	47.88	43.79	41.62	41.36	40.83	44.59	44.68	43.82	40.39	43.18	0.00
	(C)	3.86	0.46	0.30	0.63	0.50	0.52	0.95	0.34	0.34	0.43	1.21	0.00
	(D)	12.43	5.43	0.76	0.25	0.26	0.31	0.12	0.00	0.07	0.05	0.05	0.00
	(E)	9.43	4.34	0.71	0.82	0.81	0.77	1.16	0.50	0.55	0.57	1.31	0.00

programmed death of FAs, in this case it would introduce too high a load on the system. In addition, too high a number of false detections could lead to incorrect CU decisions and a FA could be sent in a wrong direction, and, as a consequence, it would not be able find the source of the attack.

Analyses of Tables 5 and 6 reveal that the anomaly metric d increases the closer the agent is to the source. Neighboring nodes to the ones along the attack path have this value significantly lower, so by using this metric the CU should easily send the FAs towards the growing anomaly, i.e., towards the attacker. It is noteworthy, that it is not required for the FAs in the nodes outside of the attack path to return $d = 0$ (this would be the case for perfect classifiers) – it is enough that the FAs are send in the direction of the growing d (which, by the way, resembles behavior of gradient methods). This shows a great advantage of the proposed method: even using non-perfect classifiers, together with the mechanism of moving agents, it can create a highly effective method of network attack detection.

Figure 5 presents the anomaly metric d as a function of distance to the source of an attack (RandomForest classifier was used). The x-axis represents the distance (in nodes) from the current node to the attacker (value 1 denotes the node closest to the attacker). It shows that in all cases d is the highest for the shortest distance to the attacker; this value decreases with the growing distance. However, it can be observed that it increases slightly again the closer to the victim. It is mainly a result of less traffic going through these nodes what decreases the denominator in the d formula. It should not, however, negatively impact neither the quality nor the effectiveness of the method. It additionally proves that the model can be more universal. It means that creating models

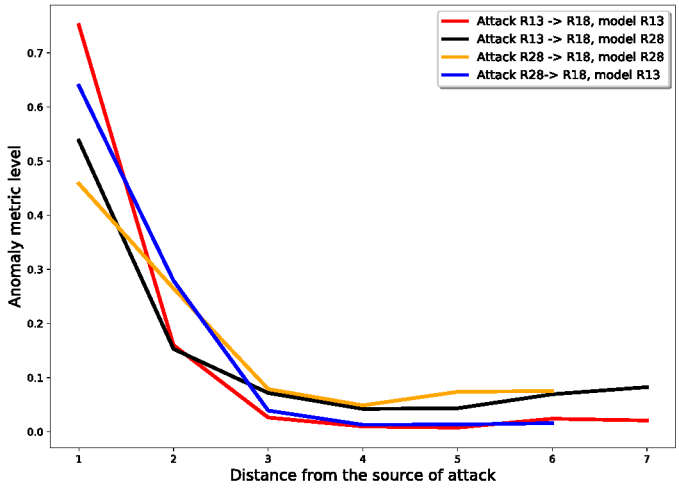

Figure 5: Distribution of anomaly metric value as function of distance from source of attack.

from data close to the possible attackers (i.e., from the nodes to which the end devices are connected) will provide the method with a proper model that should work regardless of where the attacker is placed or how the network has changed.

As probably there would be several BAs in the network, the CU should have even more information to determine if the network is under an attack, and, if it is suspected, from which node to start spawning the FAs. If the d value is high in several places in the network, the CU will need to decide if this is caused by a single attack with a single source, a single attack with multiple sources or multiple, unrelated to one another, network attacks.

6 Conclusions and future work

This paper presented results on applying the Change Observation Theory and the moving observer concept to a new kind of intrusion detection system. It is characterized by a distributed character of observation in cyberspace, cooperation between observing entities and using their mobility to identify the sources of anomalies and attacks.

We applied the proposed method to detecting attacks using timing network steganography. Therefore, as feature vectors we used meta-histograms with statistics of packet interarrival time. We trained various classifiers and proposed

an anomaly metrics based on their classification decisions. We showed, using a proof-of-concept, that this method allowed the mobile agents to move towards the source of an attack, and therefore the proposed method is effective in detecting timing network steganography. However, to apply the method to other types of attacks, the feature vector should be extended.

Our future efforts will be focused on improvements in emulating various network topologies and attacks other than those using timing network steganography. The modeling and data analysis will also be strengthened by (a) introducing new features, (b) selecting the most important ones, (b) optimizations of data size and (d) experimenting with other ML-based algorithms. In further research we will also try to take into consideration the fact that a node is visited by a flying agent several times, without a detection of an attack. We are going to investigate if this may be an indicator of a potential security breach in the given node.

The concept of using classifiers (even non-ideal ones) with the decision threshold biased towards increased TP rates, but combined with the deactivation procedure for the mobile agents, appears to be highly novel. It proved to be very promising and will continue to be researched in the future.

Acknowledgements

This work was supported by the National Centre for Research and Development (agreement No. CYBERSECIDENT/369532/I/NCBR/2017) within the CyberSecIdent Programme. The authors wish to thank Piotr Januszewski, MSc, for his contribution to setting up the experiments.

References

[Ahmed et al., 2016] Ahmed, M., Mahmood, A. N., and Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31.

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

[Basseville and Nikiforov, 1993] Basseville, M. and Nikiforov, I. V. (1993). *Detection of Abrupt Changes - Theory and Application*. Prentice Hall, Inc. - http://people.irisa.fr/Michele.Basseville/kniga/.

[Bhuyan et al., 2014] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2014). Network anomaly detection: methods, systems and tools. *IEEE communications surveys & tutorials*, 16(1):303–336.

[Botta et al., 2012] Botta, A., Dainotti, A., and PescapÃľ, A. (2012). A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531 – 3547.

[Buczak and Guven, 2016] Buczak, A. L. and Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176.

[Casas et al., 2012] Casas, P., Mazel, J., and Owezarski, P. (2012). Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Computer Communications*, 35(7):772–783.

[Fink et al., 2014] Fink, G. A., Haack, J. N., McKinnon, A. D., and Fulp, E. W. (2014). Defense on the move: Ant-based cyber defense. *IEEE Security Privacy*, 12(2):36–43.

[Fridrich, 1999] Fridrich, J. (1999). Applications of data hiding in digital images. In *ISSPA '99. Proceedings of the Fifth International Symposium on Signal Processing and its Applications (IEEE Cat. No.99EX359)*, volume 1, pages 9 vol.1–.

[Haack et al., 2011] Haack, J. N., Fink, G. A., Maiden, W. M., McKinnon, A. D., Templeton, S. J., and Fulp, E. W. (2011). Ant-based cyber security. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 918–926.

[Janicki et al., 2015] Janicki, A., Mazurczyk, W., and Szczypiorski, K. (2015). On the undetectability of transcoding steganography. *Security and Communication Networks*, 8(18):3804–3814.

[Jarvis, 2018] Jarvis, J. (2018). Steganography: Combatting Threats Hiding in Plain Sight. `https://www.fortinet.com/blog/threat-research/steganography--combatting-threats-hiding-in-plain-sight.html`. [Online; accessed 10.01.2019].

[Kaspersky Lab, 2017] Kaspersky Lab (2017). Kaspersky Lab Identifies Worrying Trend in Hackers Using Steganography. `https://usa.kaspersky.com/about/press-releases/2017_kaspersky-lab-identifies-worrying-trend-in-hackers-using-steganography`. [Online; accessed 10.01.2019].

[Kind et al., 2009] Kind, A., Stoecklin, M. P., and Dimitropoulos, X. (2009). Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management*, 6(2):110–121.

[Kott, 2018] Kott, A. (2018). Intelligent autonomous agents are key to cyber defense of the future army networks. *The Cyber Defense Review*, 3(3):57–70.

[Kwon et al., 2017] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., and Kim, K. J. (2017). A survey of deep learning-based network anomaly detection. *Cluster Computing*.

[Lewandowski et al., 2007] Lewandowski, G., Lucena, N. B., and Chapin, S. J. (2007). Analyzing network-aware active wardens in ipv6. In Camenisch, J. L., Collberg, C. S., Johnson, N. F., and Sallee, P., editors, *Information Hiding*, pages 58–77, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Lubacz et al., 2014] Lubacz, J., Mazurczyk, W., and Szczypiorski, K. (2014). Principles and overview of network steganography. *IEEE Communications Magazine*, 52(5):225–229.

[Mazurczyk et al., 2012] Mazurczyk, W., Cabaj, K., and Szczypiorski, K. (2012). What are suspicious VoIP delays? *Multimedia Tools and Applications*, 57(1):109–126.

[Mazurczyk and Szczypiorski, 2008] Mazurczyk, W. and Szczypiorski, K. (2008). Steganography of VoIP streams. In Meersman, R. and Tari, Z., editors, *On the Move to Meaningful Internet Systems: OTM 2008*, pages 1001–1018, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

[Mazurczyk et al., 2016] Mazurczyk, W., Wendzel, S., Zander, S., Houmansadr, A., and Szczypiorski, K. (2016). *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. Wiley-IEEE Press, 1st edition.

[McAfee Labs, 2017] McAfee Labs (2017). McAfee Labs Threats Report. `https://www.mcafee.com/enterprise/en-us/assets/reports/`

rp-quarterly-threats-jun-2017.pdf. [Online; accessed 10.01.2019].

[Miller et al., 2011] Miller, Z., Deitrick, W., and Hu, W. (2011). Anomalous network packet detection using data stream mining. *Journal of Information Security*, 2(04):158.

[Murdoch and Lewis, 2005] Murdoch, S. J. and Lewis, S. (2005). Embedding covert channels into tcp/ip. In *International Workshop on Information Hiding*, pages 247–261. Springer.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Poor and Hadjiliadis, 2008] Poor, H. V. and Hadjiliadis, O. (2008). *Quickest Detection*. Cambridge University Press.

[Su, 2011] Su, M.-Y. (2011). Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers. *Expert Systems with Applications*, 38(4):3492–3498.

[Szczypiorski et al., 2015] Szczypiorski, K., Janicki, A., and Wendzel, S. (2015). The Good, The Bad And The Ugly: Evaluation of Wi-Fi Steganography. *Journal of Communications*, 10(10):749–750.

[Szczypiorski and Tyl, 2016] Szczypiorski, K. and Tyl, T. (2016). MoveSteg: A Method of Network Steganography Detection. *International Journal of Electronics and Telecommunications*, 62(4):335–341.

[Tape, Thomas G., 2017] Tape, Thomas G. (2017). The Area Under an ROC Curve. http://gim.unmc.edu/dxtests/roc3.htm. [Online; accessed 10.01.2019].

[Tsai et al., 2009] Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., and Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000.