On the Performance of Anomaly Detection Systems Uncovering Traffic Mimicking Covert Channels

Johannes Bouché, Denis Hock, Martin Kappes
University of Applied Sciences Frankfurt am Main
Frankfurt am Main, Germany
e-mail: {johannes.bouche|dehock|kappes}@fb2.fra-uas.de

***Abstract*—Anomaly Detection Systems aim to construct accurate network traffic models with the objective to discover yet unknown malicious network traffic patterns. In this paper, we study the use of the same methods in order to create a covert channel which is not discovered by Anomaly Detection Systems and can be used to exfiltrate (malicous) traffic from a network. The channel is created by imitating current network traffic behaviour as detected by passive network analysis. Moreover, we present methods for calculating thresholds for the bandwidth of the channel such that, with high probability, the resulting traffic falls within the margins of the Anomaly Detection System under consideration. We also present results of practical experiments with commonly used Anomaly Detection Systems showing the practical applicability of our approach.**

***Keywords*—Anomaly Detection; Mimicry; Covert Channel;**

I. Introduction

Since the initial scientific publication of Dorothy Denning [1], the popularity of Aonmaly Detection is constantly increasing. Also, methods to subvert such systems have been proposed. Today, attackers utilize sophisticated heuristic algorithms and statistical methods in order to prevent the detection of their activities by 'Intrusion Detection Systems (IDS)' and network-based Anomaly Detection. One possibility to prevent detection is a so called 'Mimicry attack' which aims at mimicking legitimate user behaviour in order to bypass intrusion detection systems and to evade discovery. The objective of these technqiues is the unrecognized infiltration (e.g. malicious code) and exfiltration (e.g. sensitive information) of data using 'Covert Channels (CC)'. Jaskolka [2] defined them as: "(...) any communication channel that can be exploited to transfer information in a manner that violates the system's security policy". Thus, 'Mimicry-Attacks' are a special case of a 'Covert Channel'. In this paper, we assume that an attacker (or malicious insider) has already gained access to a victim's device including unrestricted access to network traffic and study the possibilities of hiding the attacker's traffic from detection by an Anomaly Detection System through mimicking legitimate traffic. As we will show, covert channels which perform even well enough to hide Botnet Command & Control traffic can be created, particularly if the methods used by the Anomaly Detection System are known.

The remainder of the paper is organized as follows. First, we present an overview of SnortAD and its prediction models for detecting anomalies. Then, we analyze the performance of covert channels in the presence of such Anomaly Detection Systems and highlight their limitations to detect covert channels with artificially generated data. In Section 2, we describe a practical experiment we conducted. By using the same prediction models as the target system, only acquired by passive traffic capturing, we calculate a traffic profile below the thresholds of the Anomaly Detection System and demonstrate that, indeed, the Anomaly Detection System does not report any anomalies.

II. Related Work

Anomaly Detection [1], has been been continuously expanded and improved. Two productivly used systems are SnortAD [3] and PHAD [4]. To the best of our knowlege, a performance evaluation of covert channels mimicking normal traffic based on packet rates and the introduction of a method to ensure that covert channel traffic does not exeed the threshold of an Anomaly Detection System has not been under investigation before.

Wendzel et al. [5] presents a recent overview of covert channels. There are also several detection methods, such as Reyes et al. [12] or Cabuk et al. [13] methods based on packet timings. Mimicry Attacks were first investigated in Wagners [6] work, who introduced 'Anagram'. The IDS 'Siren' [7] analyses how to find mimicry attacks by injecting human input into traffic, Pukkawanna et al. [8] uses a Kullback-Leibler (KL) divergence-based method on the port/pair distribution to detect Denial of Service attacks mimicking normal traffic. Wang et al. [9] detects suspicious payloads which mimic normal packet content with high-order n-grams. Casenove et al. [10] conducted a mimicry attack with covert channels by a so-called Polymorphic Blending Technique in order to exfiltrate data from a network. Wright et al. [11] is using a technique similar to the mimicry attack to prevent statistical analysis and extend the users privacy.

III. Method

To perform realtime Anomaly Detection in productive environments, Snort utilizes the preprocessor module SnortAD. In absence of other reliable solutions, we decided to use this well-known combination as a base for our experiments.

In the following, we address the mechanics of Snort and SnortAD with focus on the used AD models and most notable facts. We propose a theoretical background to infer the performance of payload injection in a covert channels and uncover weaknesses and limitations of Anomaly Detection affording to unveil those. We close this section unfolding to what degree we can utilize these findings improving the mimicry attack.

A. Snort and SnortAD

The rationale behind Snort is simple: A dedicated Snort host running in 'Single Sensor Mode' receives copies of all transported packets within the observed target network. At runtime these packets are passed through the preprocessing engine, enabling SnortAD to perform mandatory actions and raise alarms on occurrence of potentially anomalous network behavior. SnortAD can log packet volumes of several well-known protocols such as ARP, TCP, or DNS by fixed time intervals into a logfile. For each protocol, the total number of packets, bandwidth, amount of transferred bytes as well as the flow direction is collected and stored as a vector. The logfile is essentially a continuous set of vectors, representing a time series over several intervals. With a sufficient amount of data, the logfile can be used to predict future traffic with one of four different prediction models, described in the next subsection. These prediction models calculate and store a so called 'Confidence Band' - a minimum and maximum value representing the expectation for future traffic - for each protocol in an so called 'Profile'. After the generation of such the profile, SnortAD can be utilized to recognize anomalies outside the predicted Confidence Band.

B. SnortAD Prediction Models

Here, we briefly detail the working principles of the prediction models. SnortAD Profiles contain a list of predicted min/max packet volumes for each observed protocol, which is a effective and lightweight manner of storing and examining results. However, the aggregation also leads to various negative consequences as demonstrated in our practical evaluation. Depending on the underlying algorithm, a hostile machine could misuse this behavior to conduct a covert channel for data exfiltration.

Since SnortAD is a volume based Anomaly Detection System, meaning that any anomaly is based on extra ordinary packet volumes, we can define that an covert channel is successful when we can hide a Message M in an N-length packet communication without exceeding SnortAD's Confidence Band. SnortAD uses historical data to calculate a time series prediction of the next incoming packet frequencies. Which means, it measures at regular and discrete time intervals Δ_t the number of packets $p_1, p_2, ..., p_n \in \mathbb{N}$ where p_i is the measurement taken at $t_i = t_{i-1} + \Delta_t$. The chosen prediction model possesses major importance for the resulting detection accuracy. In the following subsection we briefly describe the used prediction models of SnortAD.

TABLE I
NOTATION IN THIS PAPER

Symbol	Description
p_i	Number of Packets on measurement i
t_i	Starttime of window i
α	smoothing factor for level
β	smoothing factor for trend
γ	smoothing factor for season
δ_i	parameter for autoregression

1) 'Moving Average' model (AVG): The 'Average' prediction model transforms the data represented in the logfile into a prediction model defined by arithmetic mean packet counts $\bar{p}$ for a moving window of size k within the logdata.

$$AVG_i = \frac{\sum_{j=i-k}^{i-1} p_j}{k} \tag{1}$$

Depending on the chosen size for the window, the weight of an individual outlier shrinks or grows. In that sense the model has no effective method to eliminate artifacts, negatively influencing the prediction result.

2) 'Holt-Winters' Prediction (HW): The 'Holt-Winters' model applies exponential smoothing to the supplied data and can be seen as an addition to the 'Moving Average' technique. It is used to smoothen the predicted values in a way, that the effect of collected outliers are lowered. To do that, the algorithm needs an scaling factor greater or equal to 1. A scaling factor of 1, produces just the given input as result and higher values will emphasize the distance between the resulting minimal and maximal values, effectively enlarging the region of accepted or normal network traffic. The major benefit of using this method is, that it does not need a minimum amount of input values and already works as expected with at least two observation points. The additive Holt-Winters model breaks the time series into level L_i (an approach to remove noise by subtracting the season), trend P_i (a forecast of changes in the level) and season S_i (an index for the expected level at t_i), and smooths each of the components with its own constant α, β and γ in range [0,1].

$$HW_i = P_i + L_i + S_i \tag{2}$$
$$P_i = \beta(L_i - L_i - 1) + (1 - \beta)P_{i-1} \tag{3}$$
$$L_i = \alpha(p_i - S(i - k)) + (1 - \alpha)(L_{i-1} + P_{i-1}) \tag{4}$$
$$S_i = \gamma(p_i - L_i) + (1 - \gamma)S_{i-k} \tag{5}$$

3) 'Brutlag' Prediction (BL): The 'Brutlag' method utilizes the 'Holt-Winters' model to provide predictions. This method compares actual data of the last period with fitted 'Holt-Winters' values for the same point of time [3]. This technique tries to equalize the impact of seasonality, by utilizing the trend of past periods:

$$BL_i = HW_i + m \cdot d_{i-k} \tag{6}$$

, where d is a predicted deviation and m a scaling factor. This type of prediction is also able to distinguish between different kinds of periodicity, e.g. daily or weekly and introduces therefore a more general prediction, less prone to outliers and singular events. The result is then again scaled with an scaling factor suggested to be chosen between 2 and 3.

4) 'Autoregression' Prediction (AR): The 'Autoregression' prediction method utilizes linear regression for variables and their past values and predicts an error value, which is used to define the upper and lower border of a future Confidence Band. Therefore the method compares actual data of one observation point, compared to all past observations of the same period and category. In the context of SnortAD this model tries to emulate the past behavior and occurred patterns. The model accepts a scale factor passed as input variable which will not affect these patterns but will change the scale of the series itself. This approach is therefore also not aware of outliers and seasonal effects. We forecast each point using a linear combination of past values, where δ denotes a parameter to affect the output.

$$AR_i = \delta_0 + \sum_{j=1}^{k} \delta_j p_{t-j} \tag{7}$$

All models above conduct a time-series analysis and operate on a given set of input data and at least a scale factor as parameter. They are in general agnostic to any semantics of the analyzed data and they are all prone to singular events, such as outliers.

C. Limits for Payload Injection in a Covert Channel

In the last section we described the general working principles of all models used by SnortAD. Despite the fact, that every algorithm has it's own purpose and objective, they still have a common vulnerability to covert channels. This limitation is exemplary described on the 'Holt-Winters (HW)' algorithm, but the given assumptions hold for the other prediction algorithms as well. Hereinafter, we provide the theoretical background of traffic mimicking attacks against SnortAD and describe in which way methods for data exfiltration could be determined.

Based on the output curve of a HW model, SnortAD calculates two thresholds for each category of network traffic – a minimum and a maximum – simply be subtracting and adding the standard deviation multiplied with a scale factor from the training data d. We reconstruct this Anomaly Detection method to check the theoretical performances. Based on these experiments we can estimate the maximum rate of hidden transmission for M.

For an initial experiment we can generate a random set of normal distributed data and use the Holt Winters Algorithm to reproduce a smoothed curve as a prediction of the future 50 values. Figure 1 shows a data set with $N = 10000$, $\mu = 100$ and $\sigma = 0.1$. The learned data is black, while the output of the Hold Winters prediction grey dotted. Figure 2 shows a detailed view of the Holt Winters prediction, including the minimum and maximum range, calculated with a scale factor of 1 and the standard deviation of the original data. Note, that the minimum and maximum range would also appear in all other SnortAD models, the used model only affects the trends based on the previous input.

The advantage of a time series prediction in contrast to threshold values based on average frequencies is the adaption to daily and weekly traffic trends. The netflow traces in Figure 3 have been captured on a German medium sized ISP and show the typical traffic trends for one week. The y-axis shows the amount of flows, the x-axis is the time (24 hours). Each circle shows the amount of netflows for one minute, while the dotted line shows the mean amount of flows. Another typical behavior is the change of variance in the amount of traffic.

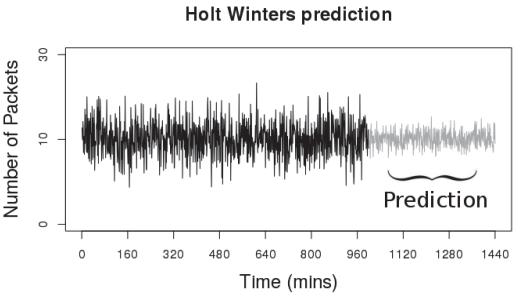

Fig. 1. Normal distributed test data and Holt Winters prediction

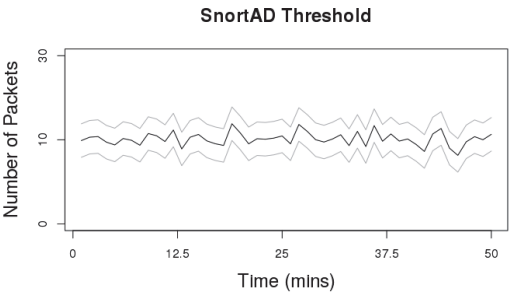

Fig. 2. Holt Winters prediction and min/max range

This fact is interesting, because SnortAD relies on the standard deviation to calculate the thresholds. However, the min/max ranges are consistent throughout the complete predicted interval. Hence, the high variance areas also increase the ranges for low variance intervals. Figure 4 shows two generated time series, where the black curve represents 10.000 points of learned data and the bright curve represents the predicted min and max areas for the next 5000 values calculated with the Holt Winters model. The area of high variance only affects the global min/max range of the output. As the authors of SnortAD did, we define the region between all minimum and all depending maximum values for a given set intervals as 'Confidence band (CB)' [3].

We can conclude, that these drifts from low to high variance either produce false positives or bloat the min/max range unnecessary which provides target for malicious traffic. In that sense, we define values, exceeding thresholds and generating false positives, as outlier. Based on the mean-value μ and standard deviation σ of the training data d, an outlier

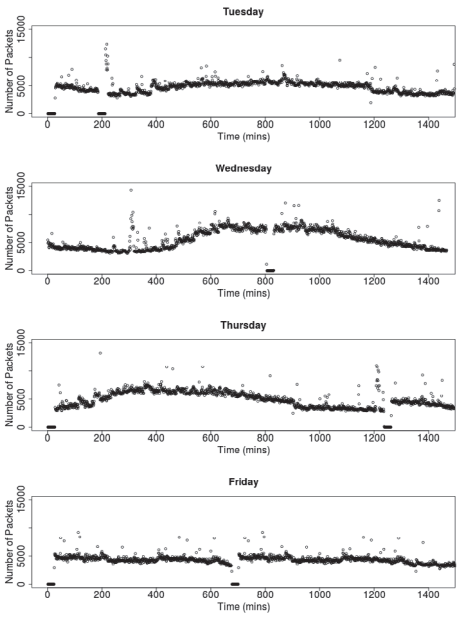

Fig. 3. Daily traffic trends (Tue-Fri)

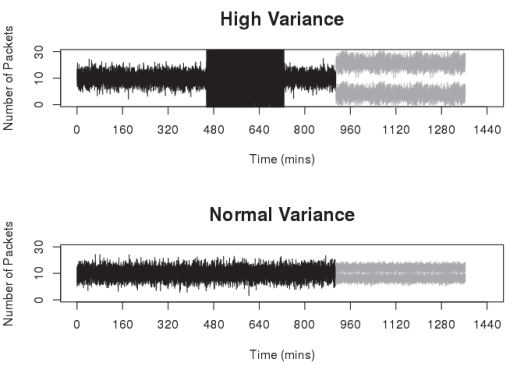

Fig. 4. Holt Winters prediction with and without a high variance area

o is any value included in the following set (8):

$$o = \{p|p > \mu + \sigma\} \wedge \{p|p < \mu - \sigma\} \tag{8}$$

Based on this conclusion, we calculate the minimal scale factor to ensure that SnortAD does not produce false positives. A naive approach to calculate the scale factor would be to use the distance to mean from our global extrema. Using the highest absolute value (11) calculated from the maximal positive (9) and negative (10) distance to the arithmetic mean $\bar{x}$ of the training data d, we can use the standard deviation s (12) to receive the scale factor.

$$maxDist_{pos} = max(d) - \bar{x} \tag{9}$$

$$maxDist_{neg} = min(d) - \bar{x} \tag{10}$$

$$maxDist = max(|maxDist_{pos}|, |maxDist_{neg}|) \tag{11}$$

$$sf = \frac{maxDist}{s} \tag{12}$$

The scale factor sf depends upon outliers, or more precisely the distribution of values. The effect of skewed distributions on standard deviation and arithmetic mean is well known.

Figure 5 shows how many values are within the allowed area (mean +- standard deviation). If the distribution is more skewed we observe less outlier, which are the further away from our allowed area.

To conclude this section, we summarize that the examined algorithm (HW) provides in general an upper and lower border (min/max ranges) for future values, received from a scale factor and an input set of intervals, containing observed packet amounts. These future min/max ranges are used by SnortAD to distinct between an legitimate or anomalous traffic amount and the distance between a min and max value is negatively influenced by singular events in the input data. To compensate that, the underlying scale factor has to be adjusted accordingly, in order to ensure that all legitimate values remain inside of the predicted min/max range for a given interval. Therefore a naive approach to determine an appropriate scale factor was given, including a description of the implications for inappropriately chosen scale factors.

D. Exploiting limitations of prediction models

The last part of this section, we focus on the implementation and practical evaluation of our observations. We present a method to calculate a packet volume threshold for a hidden message to in order to be unnoticeable. To do that, we detail considerations of network administrators and show how to calculate the estimated duration and bandwidth for a covert channel with fixed message length.

Since an attacker trying to evade detection by an IDS, can use our described drawbacks to his advantage, we assume a scenario where an malicious insider wants to exfiltrate sensitive data from the victim network to the outside without being noticed by SnortAD. We also assume that an already infected host can collect the same network traffic as the Snort sensor does. In contrast to the administrators of the victim network, which have to consider their specific demands when they choose prediction model and parameter, an attacker could, instead of guessing the correct algorithm and scale factor, easily create profiles for all available models and choose very restrictive or conservative parameter. Since the predictions are essentially min/max ranges for future time series, the administrators have to provide scale factors resulting in confidence bands, which are large enough to accept all legitimate traffic (including legitimate outliers) without triggering false-positive alarms. Furthermore, they need scale factors to be small enough to only accept the legitimate traffic. In our initial experiments we have chosen

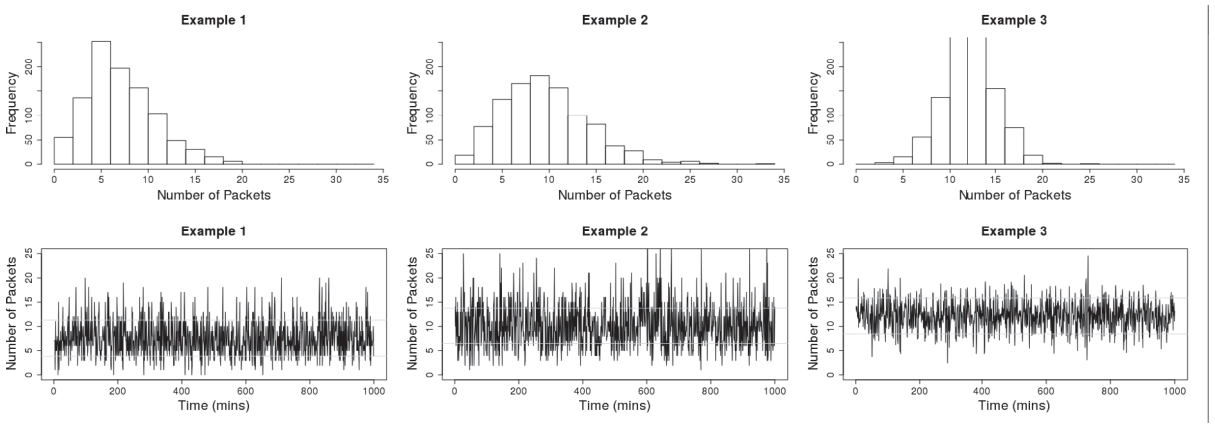

Fig. 5. Outliers in contrast to distribution

rather restrictive scale factors of 1.3 (AVG, HW, AR) or 2 (BL). In the previous section we also explained, how a reasonable scale factor could be predicted more realistically. Since we can guess the necessary scale factor conservatively, we are in the situation to generate profiles for all used algorithms and combine them to a fifth 'Covert Channel profile (CCP)', containing the minima of all MAX-values as well as the maxima of all min-values. Through this combination we obtain a confidence band in which we most likely can send data without conflicting the rules of the observing IDS. This is the case, since the administrators definitely have chosen one of the four provided algorithms and most likely have chosen a scale factor larger than ours, in order to avoid false positive alerts. If that is the case, and the attacker is still able to monitor the current network traffic, he can choose the desired protocol as well as a given message for exfiltration and utilizing the created CCP to exactly determine when he can send unnoticed packets to our destination and how. The amount of packets which can be send, is obtained by subtracting the packet counters of the involved protocol categories from protocol categories in the CCP. For a message of fixed size, the attacker can then calculate the estimated duration and necessary packet counts for a undetected data exfiltration.

IV. Results

For the following evaluation, we created a virtual target network - including an malicious host, a snort sensor and a gateway connected to the Internet - to prove and verify our assumptions. We generated a symbolized normal communication of our virtual environment, in the further section referred to as background traffic, by on-the-fly re-injecting prerecorded traffic samples. This artificial background traffic has been used during the conducted learning phase of SnortAD. The resulting profiles for all available algorithms (see Section 2) were obtained with a scale factor of 2 and a daily periodicity over a total period of four weeks. Altogether, we conducted three tests containing low ('Set 1'), medium ('Set 2') and high ('Set 3') variance background traffic, which was created by re-injecting the prerecorded traffic samples with differing sending rates. According to our defined scenario, the malicious host captures the same traffic as the snort sensor to process an equivalent model of network traffic in order to secretly exfiltrate sensitive data to an controlled host on the Internet. We proceeded to this scenario by creating a combined profile of all implemented algorithms. Using the method described in the previous chapter, we obtained a conservative scale factor of 1.3 on the malicious host, which was used to predict future send rates over a length of 60 seconds. In consonance with the captured background traffic, our protocol of choice to establish the covert channel was HTTP, which was abundant and showed a reasonably high variance in terms of sending rate and total amount of send packets.

Table II provides the obtained average min/max packet counts over a period of 28 days for each test set, as well as the utilized network bandwidth. As can be seen, the total amount of packets, as well as the utilized bandwidth roughly doubles with each set. Table III shows an excerpt of the SnortAD

TABLE II
Captured packets counts and traffic utilization

	Total #Pkt	HTTP #Pkts	HTTP Up	HTTP Down
Set1	9079-9277	8768-8921	12.4-12.6 KB/s	46.6-47.9 KB/s
Set2	9250-18078	8913-17432	12.6-24.6 KB/s	47.5-93.3 KB/s
Set3	9242-736039	8984-34674	12.6 48.9 KB/s	47.5-186.5 KB/s

profiles, the average minimum and maximum packet counts for the category 'HTTP' defining the confidence band for each algorithm (Section 2) are used by the snort sensor to determine anomalous behavior. In that sense packet amounts lower than the min-value or higher than the max-value, will be considered as anomaly and lead to a preprocessor alert inside snort. Table IV shows the combined profile as created by the malicious host. To obtain the minimal Covert Channel

TABLE III
SNORT SENSOR ALGORITHM PREDICTIONS (HTTP)

	AVG	HW	BL	AR
Set1	8605-9074	8605-9072	8601-9074	8606-9073
Set2	3851-17120	2909-19248	4634-17559	9528-17122
Set3	0-56153	0-56201	0-17823	0-56099

Profile (CCP), we simply choose the highest minimum value and smallest maximum of each interval and all of the above algorithms. The amount of packets per interval predicted for the CCP, represent the upper and lower border of a corridor in which an attacker can most likely send undetected traffic. These borders were obtained by scaling the CCP down to 10 percent. The 'Center' of the Covert Channel Profile, which is the maximum amount of packets the malicious host sends, is defined by subtracting the min-value from the max-value and through the division of the result by two. Defining a fixed message size of 1MB as exfiltration data and by assuming a payload of 1000 bytes per send packet, we can calculate the time needed to transfer a Message M via our covert channel. Recapitulatory, our results show that an malicious insider is

TABLE IV
COMBINED PROFILE AND COVERT CHANNEL PREDICTION FOR 1MB MESSAGE (HTTP)

	Combined	Covert Channel	Center	Duration
Set1	8726 - 8954	872 - 895	3	9.38h
Set2	6931 - 14355	693 - 1435	248	0.13h
Set3	34700 - 40051	347 - 4005	524	0.03h

able to find a sufficient confidence band (CB) to extract traffic unseen. The maximum performance of the covert channel is predfined by the variance in the underlying traffic and influences the required time to send all data. Since we can assume at least a decent variance in most protocols, we assume that an attacker is always able to find such a channel, given that the time is not a critical factor for the exfiltration. The proper knowledge of actual payload data is not essential to perform this operation, since SnortAD is merely a paket counter, solely relying on statistical data obtained from the network, not taking into account the semantics or payloads of transported data. Therefore an attacker has to mimick only the bandwidth of the monitored network and does not have to care about actual contents of single packets. In that sense, the creation of such an covert channel is also possible on any other network protocol (e. g. HTTPS, SMTP, FTP) , providing enough variance at a given time window.

V. Conclusion and Future Work

Many Anomaly Detection Systems rely on the assumption that malicious traffic is different from the norm. A sophisticated stealthy attack, such as our proposed mimicry attack can be very challenging to detect. Our aim was to study how efficiently mimicking traffic can hide covert channels from Anomaly Detection Systems. We showed that we can indeed easily hide traffic. We outlined the mechanics of the Anomaly Detection Plug-in SnortAD and concluded how the variance of legitimate network traffic can negatively affect the confidence band. We detailed how to exploit this weakness by calculating how much additional volume of network traffic can be send unseen and were able to estimated duration and bandwidth for a covert channel with fixed message length. Our results showed that even small networks offer enough variance to easily hide data equivalent to several HTTP pages, which is sufficient to hide botnet command and control traffic, within a small amount of time. Since SnortAD is based purely on packetcounts, we did not focus much on our method to inject specific payloads. However, we assume that the mentioned techniques are not only possible within the HTTP protocol, but also within encrypted protocols such as HTTPS. Therefore we state, that the time to transport hidden messages is in fact the only constraint to conduct such an attack successfully. We are well aware that our exemplarily used Anomaly Detection System SnortAD represents only one of many possible systems but we are confident that other systems can be circumvented in a similar way whenever someone with malicious intend has the same network insights as the Anomaly Detection System and other mitigation methods are not implemented. While we deliberately wanted to keep the initial experiment simple and representative for a wide range of network traffic, a future prospect is to evaluate this approach in industrial environments and the associated protocols which are lately threatened by 'Advanced Persistent Threats (APT)' such as stealthy botnets and other malware.

VI. References

[1] Dorothy E Denning. An intrusion-detection model. *Software Engineering, IEEE Transactions on*, (2):222–232, 1987.

[2] Jason Jaskolka and Ridha Khedri. Exploring covert channels. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–10. IEEE, 2011.

[3] Maciej Szmit, Slawomir Adamus, Sebastian Bugala, and Anna Szmit. Implementation of brutlag's algorithm in anomaly detection 3.0. In *FedCSIS*, pages 685–691, 2012.

[4] Matthew V Mahoney and Philip K Chan. Phad: Packet header anomaly detection for identifying hostile network traffic. 2001.

[5] Steffen Wendzel and Jörg Keller. Hidden and under control. *annals of telecommunications-annales des télécommunications*, 69(7-8):417–430, 2014.

[6] David Wagner and Drew Dean. Intrusion detection via static analysis. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 156–168. IEEE, 2001.

[7] Kevin Borders, Xin Zhao, and Atul Prakash. Siren: Catching evasive malware. In *Security and Privacy, 2006 IEEE Symposium on*, pages 6–pp. IEEE, 2006.

[8] Sirikarn Pukkawanna, Youki Kadobayashi, and Suguru Yamaguchi. Network-based mimicry anomaly detection using divergence measures. 2015.

[9] Ke Wang, Janak J Parekh, and Salvatore J Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Recent Advances in Intrusion Detection*, pages 226–248. Springer, 2006.

[10] Matteo Casenove. Exfiltrations using polymorphic blending techniques: Analysis and countermeasures. In *Cyber Conflict: Architectures in Cyberspace (CyCon), 2015 7th International Conference on*, pages 217–230. IEEE, 2015.

[11] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, 2009.

[12] Guido Pineda Reyes and Pepijn Jansen. Covert channel detection using flow-data. 2014.

[13] Serdar Cabuk, Carla E Brodley, and Clay Shields. Ip covert channel detection. *ACM Transactions on Information and System Security (TISSEC)*, 12(4):22, 2009.