stealthML: Data-driven Malware for Stealthy Data Exfiltration

Keywhan Chung[1], Phuong Cao[2], Zbigniew T. Kalbarczyk[1], and Ravishankar K. Iyer[1]

[1]Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
[2]National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign

***Abstract*—The use of machine learning methods have been actively studied to detect and mitigate the consequences of malicious attacks. However, this sophisticated technology can become a threat when it falls into the wrong hands. This paper describes a new class of malware that employs machine learning to autonomously infer when and how to trigger an attack payload to maximize impact while minimizing attack traces. We designed, implemented, and demonstrated a smart malware that monitors the real-time network traffic flow of the victim system, analyzes the collected traffic data to forecast traffic and identify the most opportune time to trigger data extraction, and optimizes its strategy in planning the data exfiltration to minimize traces that might reveal the malware's presence.**

I. Introduction

In recent years, researchers and practitioners in the cyber security domain have been investigating and demonstrating the use of machine learning (ML) methods to detect and mitigate the consequences of malicious attacks. However, the possibility that adversaries might utilize the same ML technology to advance malicious intentions and conceal suspicious activities has been largely ignored. In this paper, we present a new threat model, stealthML, and demonstrate that data-driven malware can automatically reconnoiter the target system and customize the attack strategy to achieve the attacker's objectives (e.g., exfiltrate sensitive data) and mimic normal/benign user behavior so to hide attacker's presence in the system. In enabling this threat, we used an ensemble of ML methods to conduct monitoring, function development, and inference, which in the present case, is not possible for simple methods (e.g., at a comparable level of accuracy). We demonstrate, in the context of data breach attacks, that stealthML (substantially independent of the attacker) can autonomously train a model inside the victim (using LSTM) and adapt the model to dynamic changes in the system (using DQN).

In our hypothetical attack model, the goal of stealthML is to copy the sensitive data from the victim system to the attacker's data store while disguising the exfiltration traffic as legitimate network traffic (e.g., HTTP). StealthML consists of two machine learning modules. The *forecast module* tries to learn a model that can best predict the pattern of normal network activities in the victim system. This module employs LSTM (long short-term memory) recurrent neural network to forecast network traffic based on real-time system measurements. The *exfiltration module* derives the best exfiltration action given the forecast and past experience in interacting with the victim system. The module utilizes a deep Q-learning (DQN) method to empirically optimize a decision model. We encoded attacker intent and knowledge into a reward model.

To assess the performance of stealthML, we executed it in National Center for Supercomputing Applications's (NCSA) science demilitarized zone (DMZ) to contain the unforeseen impacts on the system and its user. Experimental results demonstrate that stealthML can conceal its malicious intent (under the noise of normal behavior) while accelerating the attack. This threat can challenge current protection methods that look for abnormal behavior in systems [16], [17]. Furthermore, stealthML not only reconnoitered and inferred actionable intelligence that fit the victim system but also transferred the intelligence and experience collected from one victim to another to improve the malware performance in future attacks. If such a threat becomes a reality, targeted attacks will no longer be restricted to targets that attackers can reach (i.e., communicate with); instead, intelligent malware will be able to spread itself automatically across multiple victims (like worms and viruses) and challenge the operators, as (i) the threat customization will be automatically driven by the victim's local data, and (ii) the intelligence will accumulate as the malware morphs and passes through multiple victims. Our findings reveal a need to proactively investigate new detection and mitigation methods that target such new threats.

II. Problem and Motivation

A. Motivation

Prior high-visibility data breach attacks (e.g., SolarWinds [6], Yahoo [8], Equifax [14], Target [15], and Capital One [7]) were successful in stealing and exposing sensitive data despite employing rudimentary malware or human-driven attack strategy. Furthermore, recent ransomware attacks – with severe impact across industries – are deploying double extortion where the attackers exfiltrate data before encryption for other purposes [12]. However, as more security monitors and policies are being deployed to protect systems, attackers face a higher risk of detection.

Attackers performing data breach attacks put effort into disguising their data exfiltration traffic as normal user activity. For instance, one can utilize network protocols that are less obvious [6], [14], or try to choose an optimal time that makes detection challenging [15]. Reducing the data exfiltration rate

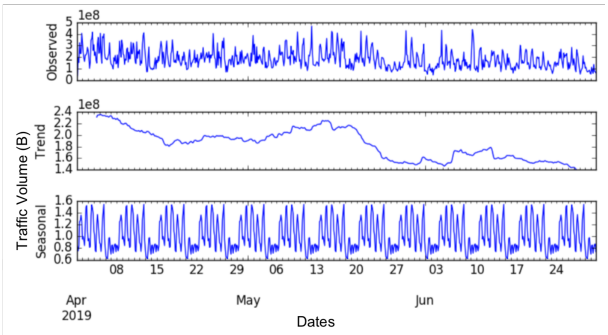

Fig. 1: STL decomposition of network traffic in 2019.

or size is another method commonly found in recent incidents [7], [8], [14]. Reconnaissance plays a critical role in deriving the strategy that best fits the specifics of the victim. However, such a manual procedure restricts the attacker in certain respects. For instance, an extended period of remote connection to a target system puts the attacker at risk of being exposed. Furthermore, with a human playing a critical role in the attack, it is only possible to target systems that are remotely accessible through the network; the threat cannot be generalized to extended targets. Fortunately, that limitation has restricted the impact of data breach attacks to certain victims. However, automation of the process would free the attacker from those restrictions, and the attack, encoded with learning methods, could travel across systems that a human attacker could not have reached.

A well-known intuition would be that there can be *seasonal trends and patterns* that are closely related to business hours and annual seasons [1], [9]. As depicted in Figure 1, we find that, in fact, the network traffic activity can be decomposed (using STL decomposition [5]) into what turns out to be daily and weekly seasonal patterns accompanied by trends. Such patterns, which match common intuition, could be considered either *global patterns* expected across multiple systems, or *local patterns* specific to each victim. A usual traffic pattern can be described as

$$y_t = \underbrace{S_{t,global} + T_{t,global}}_{\text{global pattern}} + \underbrace{S_{t,local} + T_{t,local}}_{\text{local pattern}} + R_t \quad (1)$$

where S_t, T_t, and R_t represent seasonal patterns, trends, and residuals, respectively. A larger residual indicates noisier traffic which challenges anomaly based detectors that try to represent traffic behavior as a model. If adversaries could design malware that automatically infers such patterns, they could craft their attack payloads *to remain within the victim's noise boundaries* to prevent the introduction of any anomaly that could hint at the attack's existence.

B. Threat model

In discussing stealthML, we assume that:

- The victim system is a web server and has (i) HTTP/HTTPS network packets going through its network interface as the normal network activity, and (ii) an IP address that is known to the attacker. Attackers can easily use network scanning to identify targets, their IP addresses, and the services that they offer.
- The attacker (i) can get privileged (remote) access (e.g., through phishing, stolen credentials, or insider attack) to the victim system, (ii) has set up a server to receive the extracted data, and (iii) can establish network connections between the victim system and the attacker's server to transmit the data.
- StealthML (i) can be installed and executed in the target system, (ii) has access to target data (that the malware would exfiltrate) in the victim system, (iii) has privileges to capture the network interface of the victim system to monitor the genuine network traffic and send extracted data through the interface, and (iv) does not have prior knowledge regarding the victim's detectors or defense systems.

III. System Overview

We consider the victim to be a typical computing system that stores a large amount of data. The attacker performs data breach attacks by setting up a server to receive network packets that contain the extracted data sent by the malware (*attacker data store*), gaining root privileges on the victim system with stolen credentials, and installing (and later executing) in the victim system stealthML, which is responsible for controlling the data exfiltration to the attacker's data store. Once stealthML is installed in the victim system, it automatically drives the attack.

Victim system. In this study, we consider a web server as an example of a typical victim system. Systems that host web services through the public network are often a tempting target for attackers trying to hide their existence behind legitimate users. A web server, in reality, might not host sensitive data. However, in recent data breach incidents [6], [14], we have seen situations in which attackers utilized HTTP to transfer data out of the victim. Otherwise, one can assume that the attacker (or malware) has collected data from other victims within the internal network of the target institution (often by lateral movement) and archived it in the web server (i.e., as the front end of the breach) for extraction.

The **attacker data store** is a system outside the monitoring boundary of the target institution, under full control of the attacker. The goal of the attacker (and the smart malware) is to copy the data in the victim system to the attacker's data store while disguising the exfiltration traffic as legitimate web traffic. To prevent a single IP from dominating a significant portion of the traffic (which could reveal the attack), one can distribute the exfiltration traffic to multiple data stores [6].

Network traffic data. Our work is based on real data collected from a large-scale computing infrastructure (NCSA). The data consist of logs on network connections and HTTP traffic. The data were collected by network monitors deployed next to the border router of the infrastructure from April to June 2019, and from January to October 2020. On average, 23K connections were made each day between ∼3,000 unique hosts. Using the HTTP logs collected by network monitors at NCSA, we replayed the web requests and collected packet-level traffic

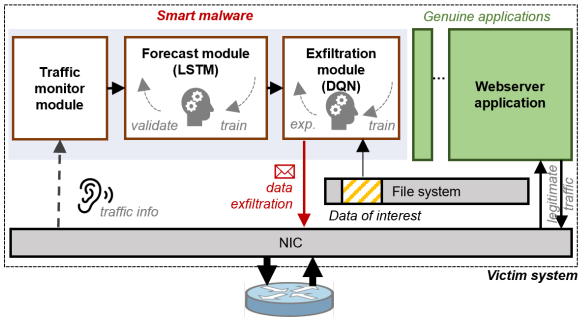

Fig. 2: Smart data exfiltration malware logic.

information for this study. We discuss details of the traffic replay in Section V.

IV. Approach: Stealthy Data Exfiltration

We consider the activity of the malware and its interaction with the victim system in the context of game theory and seek the optimal solution using reinforcement learning (RL). The victim system (or its detection system, in particular) is the environment. Benign users and malware are the players in this game. In our scenario, unlike a conventional game, benign users have no conflict of interest with other players, and there is no decision making in taking actions. On the other hand, malware has a clear goal to derive the best action in the context of *when* and *how much data* to exfiltrate while minimizing the risk of transitioning into a malicious state. If the combined actions exceed what the operator considers normal by more than the noise boundary for a number of consecutive time steps, the operator will consider it abnormal and investigate the case, which will likely reveal the malware. In making the decision on the size of data exfiltration, stealthML is provided with only *partial information*. To be exact, the malware is not aware of the current state.

Smart malware logic. As shown in Figure 2, stealthML consists of three modules: the *traffic monitor module*, the *forecast module*, and the *data exfiltration module*. The traffic monitor module parses measurements (*observations*) from the system that can benefit the forecasting module. The forecasting module trains a model that can predict the likely *state*, given the observations from the traffic monitor. The exfiltration module manages the decision model to determine the optimal action, given the observations and the state prediction. Each module is implemented in a separate thread such that with multithreading, all of the modules can execute simultaneously.

A. Network traffic monitor

As the malware executes, the network monitor (using `tcpdump`) listens to the network interface of the victim system and collects network packets of type HTTP/HTTPS that are being received or sent by the victim server.

The packet parser transforms the network packet logs and stores them in shared memory so that the forecasting module can read its input (i.e., traffic rate per minute). To reduce memory usage and improve computational efficiency, we fixed the shared memory size. Data are stored as a queue in the shared memory whose length is equal to the length of our observation window (which has length N, see Figure 3) of the network traffic. When the observation window moves forward in time, the earliest observations are dequeued, and observations are enqueued.

B. Forecast module

The goal of the forecast module is to analyze the normal traffic in the victim system and predict what is likely to be considered *normal traffic* in the next time interval. Given the observation passed by the traffic monitor module as input, the forecast model performs real-time analysis to predict the traffic for the next time interval.

Long Short Term Memory (LSTM). To train the forecast model, we utilized LSTM, a well-studied machine learning method that is commonly used in time series forecasting [10]. LSTM is a specific kind of recurrent neural network (RNN) that addresses the vanishing gradient problem of RNNs, and hence is less sensitive to the gap length. Because of that property, LSTM is often described as a neural network architecture that remembers values over arbitrary time intervals. Such remembrance is enabled by a set of gates that compose a single memory cell: an input gate, an output gate, and a forget gate. A generic representation of an LSTM memory cell can be found in Figure 3.

Objective. The architecture of our LSTM and its mapping to the training data set are depicted in Figure 3. Time series data are the data recorded by the traffic monitor module in the shared memory. A data point X_t at time t consists of features used for training the model. The features are generic to any system, such that a model trained in one system could be transferred and used in any other victim.

Let X_t represent the observation at time t. The problem is to train an LSTM model, $M(X_t)$, that forecasts the traffic volume for the next time interval, given the observations from the past N time intervals. In training the model, the goal is to capture the global pattern shared across victims (see equation 1) such that when provided with the same feature space from victims i and j, χ_i, the learning agent infers models that satisfy

$$M_{global_i}(X) \approx M_{global_j}(X) \tag{2}$$

Then, given a model inferred from *victim i*, the training process only needs to tune the transferred model to capture the local patterns specific to the new victim. Hence, one can accelerate the process of inferring the model for *victim j* (relative to inferring it from scratch) by utilizing the model learned from *victim i* and fitting the pre-trained model to the new victim's specifics (i.e., $M_{local}+\epsilon$). Further, as the malware accumulates observations from multiple victims, the global model inferred from them becomes more generic.

Training. Regardless of whether we start from scratch or start with a pre-trained model transferred from another system, we need to fit the model to the specifics of the local system (just as an attacker would reconnoiter the system and customize the attack payload). For each training instance, the model is provided with training data from an observation window of length M (see Figure 3). After each training instance, the

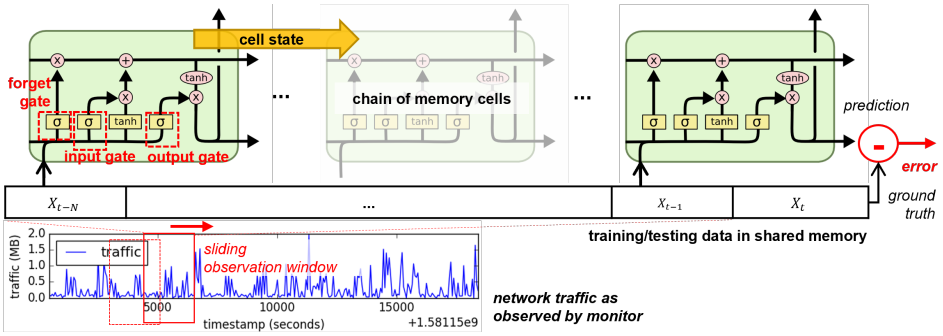

Fig. 3: LSTM model and its training/validation.

window is shifted by one time interval. Given the data set, the forecasting module normalizes the measurements so that all features are within the same range. Because the absolute error is in normalized scale, we inverse-transformed it to its original unit. The training process is repeated until the absolute error reported during the training process reports negligible change.

C. Data exfiltration

Deep Q-Learning (DQN). To derive the optimal action that satisfies the attacker's intent, we used Deep Q-Network. In Q-learning, we model the decision process in the context of quality of states ($Q(S, a)$), an abstraction of the rewards (i.e., immediate reward, $R(S, a)$, and future rewards combined) one can expect by taking an action in the current state. Given the decision model and current state, the DQN agent chooses the action, a, that maximizes $Q(S, a)$. Because stealthML has access to limited information (partial observation, X), we cannot define the full decision model for the malware. Instead, we utilize a Deep Q-network which represents the decision model as a neural network, $\hat{Q}(X, a)$, and tune the network through training.

Objective. The goal of the forecast module is to predict the normal traffic pattern at its best. Depending on various factors, including the significance of the pattern (e.g., seasonality or trend), the performance of the forecast engine could vary. To determine the optimal action for the malware at each time instant, we utilized a decision model trained using deep reinforcement learning (DRL). We encoded the attacker's preference into a reward model that evaluates and determines the consequences of the data exfiltration module's action. As depicted in Figure 4, our DRL agent learns how much weight (or trust) to put on the forecast module's suggestion from experience (i.e., the collection of previous actions taken and resulting rewards). And using this model, the exfiltration module determines the action with the most promising predicted result and performs the exfiltration accordingly. Our agent's action set consists of discounting the forecast, exfiltrating more than the forecast, or deciding not to exfiltrate data.

V. Experiment Design

To assess stealthML, we set up a testbed in a production system of NCSA. All network packets were passed through the same pipeline as for any other production system, and the

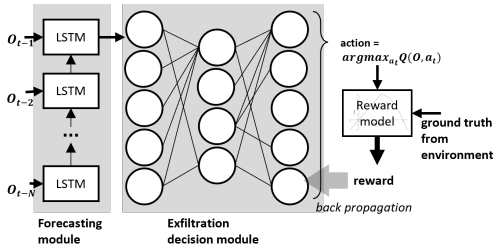

Fig. 4: Use of DRL to determine an optimal exfiltration action.

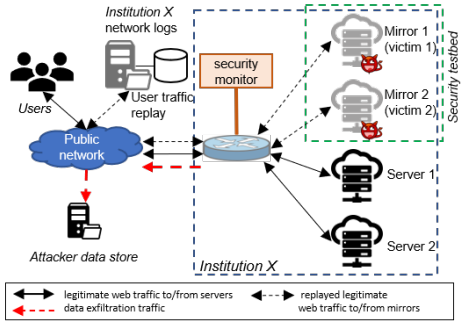

Fig. 5: Experimental setup.

activities in the testbed were monitored by the security team. Figure 5 depicts the experimental setup.

Security testbed. In experimenting on our threat model, we utilize a security testbed that runs alongside production systems at NCSA. The testbed supports features for running host machines with specific configurations (and vulnerabilities) while assuring that any impact introduced by our experiments is contained within the testbed. Using the testbed, we can run our experiments in a realistic setting while assuring that normal users are not affected.

Victim system setup. In the testbed, we set up two web servers, which were copies of web servers in service at NCSA. The servers were the top two in web traffic volume among systems at NCSA that were accessible from the public network without authentication. In our experiment, these servers were the victims in which the malware was installed. In addition to including copies of the real servers' web pages, we also placed a large file (a 5 GB binary image) in each system to represent the data of interest that the adversaries are trying to exfiltrate from the corporate network.

TABLE I: Comparison of data exfiltration methods.

Attack strategy	Detector performance (false negative / false positive) (%)				Exfiltration time (hr)
	z=1	z=0.1	ARIMA	LSTM	
Benign traffic	N/A[a] / 14.9	N/A / 38.6	N/A / 21.1	N/A / 14.8	N/A
Brute-force	0 / 0[b]	0 / 0	0 / 0	0 / 0	0.14
Low-and-slow	90 / 14.2	88.8 / 37.7	12.73 / 19.8	5.67 / 14.1	164
LSTM only	98.1 / 12.2	58.8 / 37.8	52.2 / 18.2	15.7 / 14.3	171
LSTM + DQN	95.6 / 10.4	55.8 / 31.2	51.1 / 17.1	19.2 / 16.5	147

[a]Does not include attack traffic, hence, no positive case to detect.
[b]Attack is always taking place and, hence, there cannot be a false alarm.

Traffic replay. We implemented a traffic replay tool that took security logs for inbound HTTP traffic (collected using the Zeek [18] IDS) as inputs and replayed the web requests while preserving the latency between requests. In replaying the requests, we set the victim system (in the testbed) to serve as a proxy so that the traffic would be sent to the victim system. In our experiment, the traffic replay tool represented the benign users in the original systems. Note that in our experiments, because (i) we only had HTTP traffic that we can replay to represent regular system activity, and (ii) academic systems attract less web traffic than commercial web servers, the traffic data available to the malware was relatively sparse. This difference between our experimental system and commercial web servers has implications for the performance of stealthML as discussed in detail in Section VI-B.

Anomaly detectors. We considered three representative anomaly detection methods: a *z-score*-based anomaly detector, an *ARIMA*-based detector, and an *LSTM*-based detector. A z-score-based detector considers an observation that is z standard deviations from the mean to be an anomaly. While z-score based detector is more of a rule-based detector with a fixed threshold, Autoregressive integrated moving average (ARIMA) is a well-established statistical method, known to be effective in predicting the next value in a time series. For an LSTM-based detector, we take advantage of the fact that the defender is less restricted (than an intruder who has to hide its presence) to the use of computational resources for which there is a strong need to detect anomalies. Hence, we utilized richer (i.e., longer memory-cell chains and more layers in the neural network) LSTM architecture for detecting anomalies.

VI. Results

A. Smart data exfiltration vs. current approaches

In Table I, we compare the performance of different attack strategies. As a baseline, we present how the detectors perform against *benign traffic*. Ideally, the detectors should not raise an alert for benign traffic. Because of the noisy nature of web traffic, we see that the detectors are challenged with high false positive rates. *Brute-force* attack had an advantage in dramatically reducing the exfiltration time. For this attack, even the simplest z-score based detector would not miss a single action of the attack. However, because of the exceptional speed, the attacker or malware could have accomplished its goal and left the victim system, by the time the system operator takes action. Hence, such a strategy would be effective if there is a fix-sized data file (of reasonable size) that the adversary is targeting but not if the attacker is expecting a stream of new data generated every day and wants to keep a foothold on the system for an extended period. A "*low-and-slow*" strategy significantly reduced the risk of revealing malicious activities. However, as in the Target data breach attack [15], which took 2 weeks to exfiltrate 11 GB of data, it took roughly 7 days to exfiltrate our 5 GB of synthetic data.

StealthML reduces further the attack traces. For instance, it was barely detected by the $z = 1$ detector with a miss rate of 95% and performed better than "low-and-slow" when played against detectors with a more sophisticated method. The exfiltration time, unexpectedly, was longer than the "low-and-slow" approach when we deployed LSTM forecast only, but as we discuss in Section VI-B, such results are specific to the characteristics of the victim system. Furthermore, by deploying the DQN-based decision module, we can accelerate the data exfiltration while still hiding attack traces (see *LSTM+DQN*). The DQN achieves such improvement in performance by taking a more aggressive action (i.e., more data exfiltrated per action than suggested by the LSTM forecast) when expecting less benign traffic.

In our set of experiments, LSTM-based detectors outperformed detectors based on simpler methods, such as threshold-based methods or ARIMA, in detecting abnormal traffic patterns. The difference in performance becomes more significant as the attacker deploys a more sophisticated method. Our experimental results demonstrate how sophisticated detection methods would result in requiring attackers to explore complicated methods to bypass detection. Hence, we need to be prepared and should be proactive rather than reactive.

B. Data exfiltration time vs. traffic volume

In Section VI-A, we did not observe a significant difference in the data exfiltration rate for the three attack strategies (i.e., low-and-slow, LSTM forecast only, and LSTM + DQN) presented in Table I. For example, LSTM+DQN strategy reduces the data exfiltration time by about 10% as compared with low-and-slow approach which took 164 hrs. We find the reason for this relatively small difference between the two attack strategies from the low utilization of the overall network bandwidth due to very limited user activities. The systems that we mirrored as a victim are located at an academic institution, and the volume of benign web activities is relatively smaller as compared with busy commercial web servers. To validate our conjecture, we assessed how the time to data exfiltration changes for amplified benign traffic. As depicted in Figure 6, we find that stealthML (unlike brute-force and "low-and-slow" attack strategies) accelerates exfiltration. For instance, as we amplify the benign traffic volume by a factor of five,[1] our LSTM+DQN attack strategy reduces the data exfiltration time by 80% from 147 to 29. Hence, when deployed in a busy

[1]To put such amplification in perspective, a single Apache server can handle 160 requests per second. On average we have less than one request per second in our data set. Hence, even our amplified traffic is merely a fraction of the usual web traffic in commercial servers.

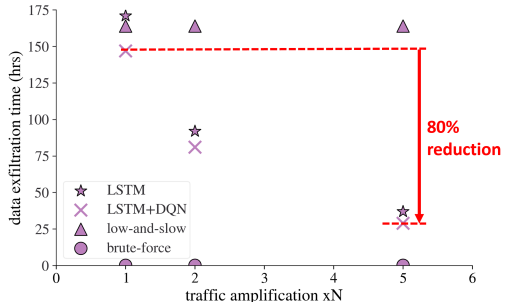

Fig. 6: Data exfiltration time for amplified benign traffic.

system with sufficient user activity, stealthML can achieve an exfiltration rate comparable to the brute-force approach while maintaining the low visibility to security monitoring tools.

C. Minimal footprint on system behavior

Despite the use of sophisticated methods, our approach does not introduce significant traces that can reveal the malware. In designing the trainers, we restricted the queue size and, hence, the memory usage does not exceed 1 GB throughout the process. Furthermore, because the model is updated only once every time interval (i.e., 5 minutes), the most computation-heavy thread for training the model remains idle 99% of the time during the attack, waiting for the traffic monitor to collect the traffic information for the next time window.

VII. Related Work

A number of previous works have considered ML-based threats on cyber infrastructure. However, unlike stealthML, such an approach was either limited in adapting their attack strategy to the specifics of the target system or the learning was conducted offline. In [3], the authors exploit vulnerabilities of a surgical robot and make use of ML to mimic an accidental failure so that the malicious intent is hard for the system administrator to identify. Similarly, in [2], the authors take advantage of statistical analysis to disguise malicious actions as accidental failures by inferring attack strategies from CPS measurement data. DeepLocker [13] leverages a deep neural network to detect the target and disguises itself as benign software before a target is detected. A thorough overview of machine learning-based malware can be found in [4], [11].

VIII. Discussion

Parameter optimization. The goal of our project was to demonstrate how machine learning can advance cyber threats. Therefore, the parameters (e.g., length of the observation window, number of hidden layers, training data size, and number of epochs per training instance) were not fully optimized. Despite the lack of optimization, experimental results show that the trained model could effectively disguise the data exfiltration traffic within the noisy regular traffic.

Monitoring at the enterprise level. StealthML targeted the normal behavior of a particular victim system. However, in practice, network-level monitoring audits network activities (especially their volume) at the border router at an aggregated scale. Hence, the traffic utilized to exfiltrate the data is even more likely to be buried under the aggregated noise from the entire enterprise. Therefore, to provide visibility to the traffic associated to attacks like stealthML, the monitors or detectors must be placed closer to the victim systems.

Limitations. Our victims (and the data sets collected from them) were limited to (copies of) systems in an academic institution in which the HTTP traffic was relatively sparse, compared to commercial web servers. (E.g., in our data set for victim B, we encountered zero HTTP traffic 62% of the time.) This limitation resulted in weak trends and patterns in user traffic. We expect that stealthML would be more efficient in a commercial web server with heavy web activities.

References

[1] Roberto Casado-Vara, Angel Martin del Rey, Daniel Pérez-Palau, Luis de-la Fuente-Valentín, and Juan M Corchado. Web Traffic Time Series Forecasting Using LSTM Neural Networks with Distributed Asynchronous Training. *Mathematics*, 9(4):421, 2021.

[2] Keywhan Chung, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. Availability Attacks on Computing Systems through Alteration of Environmental Control: Smart Malware Approach. In *Proc. of the ACM/IEEE Intl. Conf. on Cyber-Physical Systems*, pages 1–12. ACM, 2019.

[3] Keywhan Chung et al. Smart Malware that Uses Leaked Control Data of Robotic Applications: The Case of Raven-II Surgical Robots. In *Proc. of the Intl. Symp. on Research in Attacks, Intrusions and Defenses*. USENIX, 2019.

[4] Keywhan Chung et al. Machine Learning in the Hands of a Malicious Adversary: A Near Future If Not Reality. In *Game Theory and Machine Learning for Cyber Security*. Wiley, 2021.

[5] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. STL: A Seasonal-trend Decomposition. *Journal of official statistics*, 6(1):3–73, 1990.

[6] FireEye. Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor, 2020.

[7] Karen Flitter, Emilu; Weise. Capital One Data Breach Compromises Data of Over 100 Million, 2019.

[8] Nicole Goel, Vindu; Perlroth. Yahoo Says 1 Billion User Accounts Were Hacked, 2016.

[9] Hao He and Niklas Karlsson. Identification of seasonality in internet traffic to support control of online advertising. In *Proc. of the American Control Conference*, pages 3835–3840, 2019.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[11] Nektaria Kaloudi and Jingyue Li. The AI-Based Cyber Threat Landscape : A Survey. *ACM Computer Survey*, 53(1):1–34, 2020.

[12] Quintin Kerns, Bryson Payne, and Tamirat Abegaz. Double-Extortion Ransomware: A Technical Analysis of Maze Ransomware. In *Proc. of the Future Technologies Conference*, pages 82–94. Springer, 2021.

[13] Dhilung Kirat, Jiyoung Jang, and Marc Ph. Stoecklin. Deeplocker – Concealing Targeted Attacks with AI Locksmithing. In *Blackhat USA*, 2018.

[14] Nick Marinos and Michael Clements. Actions Taken by Equifax and Federal Agencies in Response to the 2017 Breach. Technical report, New York, NY, USA, 2018.

[15] Committee on Commerce Science and Transportation. A "Kill Chain" Analysis of the 2013 Target Data Breach. In *The Target Store Data Breaches: Examination and Insight*, pages 41–60. 2014.

[16] Md Salik Parwez, Danda B. Rawat, and Moses Garuba. Big Data Analytics for User-activity Analysis and User-anomaly Detection in Mobile Wireless Network. *IEEE Trans. on Industrial Informatics*, 2017.

[17] Duygu Sinanc Terzi, Ramazan Terzi, and Seref Sagiroglu. Big Data Analytics for Network Anomaly Detection from Netflow Data. In *Proc. of the Intl. Conf. on Comupter Science and Engineering*, 2017.

[18] The Zeek Project. The Zeek Network Security Monitor, 2020.