DAPT 2020 - Constructing a Benchmark Dataset for Advanced Persistent Threats

Sowmya Myneni*[1], Ankur Chowdhary*[1], Abdulhakim Sabur[1]
Sailik Sengupta[1], Garima Agrawal[1], Dijiang Huang[1], and Myong Kang[2] ⋆

[1] Arizona State University, Tempe, AZ, USA
{smyneni2,achaud16,asabur,ssengu15,gsindal,dijiang}@asu.edu
[2] US Naval Research Lab, USA
myong.kang@nrl.navy.mil

Abstract. Machine learning is being embraced by information security researchers and organizations alike for its potential in detecting attacks that an organization faces, specifically attacks that go undetected by traditional signature-based intrusion detection systems. Along with the ability to process large amounts of data, machine learning brings the potential to detect contextual and collective anomalies, an essential attribute of an ideal threat detection system. Datasets play a vital role in developing machine learning models that are capable of detecting complex and sophisticated threats like Advanced Persistent Threats (APT). However, there is currently no APT-dataset that can be used for modeling and detecting APT attacks. Characterized by the sophistication involved and the determined nature of the APT attackers, these threats are not only difficult to detect but also to model. Generic intrusion datasets have three key limitations - (1) They capture attack traffic at the external endpoints, limiting their usefulness in the context of APTs which comprise of attack vectors within the internal network as well (2) The difference between normal and anomalous behavior is quiet distinguishable in these datasets and thus fails to represent the sophisticated attackers' of APT attacks (3) The data imbalance in existing datasets do not reflect the real-world settings rendering themselves as a benchmark for supervised models and falling short of semi-supervised learning. To address these concerns, in this paper, we propose a dataset DAPT 2020 which consists of attacks that are part of Advanced Persistent Threats (APT). These attacks (1) are hard to distinguish from normal traffic flows but investigate the raw feature space and (2) comprise of traffic on both public-to-private interface and the internal (private) network. Due to the existence of severe class imbalance, we benchmark DAPT 2020 dataset on semi-supervised models and show that they perform poorly trying to detect attack traffic in the various stages of an APT.

Keywords: Advanced Persistent Threat, Benchmark Dataset, Stacked Autoencoder (SAE), Long Term Short Memory (LSTM), Anomaly Detection

⋆ Sowmya Myneni and Ankur Chowdhary have equally contributed to this work.

1 Introduction

Advanced Persistent Threat (APT) [29] is a form of a cybersecurity threat, posed by well-funded organizations, often to gain crucial information from the target organization. APT is defined by a combination of three words, namely (1) *Advanced:* APT attackers are advanced in terms of attack tools expertise, and attack methods. With attack vectors customized to the target, APT attackers organize the attack into multiple stages. (2) *Persistent:* APT attackers are determined to achieve the attack objective. The attack methods involve the use of evasive techniques to elude security agents deployed by the network defender. (3) *Threat:* The threat part of APT comes from the potential loss of sensitive data or mission-critical components. An APT attack usually consists of five main phases, (1) Reconnaissance (2) Foothold Establishment (3) Lateral Movement (4) Data Exfiltration, and (5) Post-Exfiltration [1].

A vast array of research exists in the area of anomaly detection [11], and traditional intrusion detection systems as a means of identification of slow and low attacks such as APT. These encompass methods to detect abnormal behaviors through the use of rule-based engines [55,28,15], machine learning algorithms [18], in general, and more recently, deep learning architectures [54] in particular. These methods focus mostly on detecting anomalies in external traffic packets, i.e., at the interface of the external and internal network. A survey of industry professionals conducted by Trend Micro [34] shows only 25.1% of the participants are familiar with APTs and 53.1% consider that APTs are similar to traditional attack vectors. However, the detection of Advanced Persistent Threat (APT) that involves the identification of long-term attack behavior both over the public and private channels is fundamentally different.

While attackers in the context of APTs leverage tools and techniques similar to those used in the external attack vectors, the mode of operation and the goal of these attacks is different from the traditional single-stage attacks. Traditional intrusion detection techniques such as pattern/signature matching, machine learning, etc. cannot detect APTs effectively because they are often designed to detect individual (known) attack patterns or methods as opposed to a threat that involves several interconnected malicious activities. Furthermore, performance on individual phases of APTs, as we show based on results of semi-supervised machine learning models (used in anomaly detection) is far from being effective (see auc-roc-pr). The *stealthiness*, *adaptability*, and *persistence* of APTs makes detection and prevention of such threats, by present methods, quite challenging [34].

Current research seeks to identify anomalous activities based on time series prediction, and machine learning-based correlation analysis [26] or threat scores based on a static set of rules, e.g., HOLMES [36]. Given the lack of APT datasets, these techniques (1) do not consider modeling the aspects of stealthiness, completeness, and persistence that are paramount in the case of APTs and (2) can neither identify nor leverage correlations across multiple phases of an APT. For example, given that a reconnaissance phase is essential before establishing a foothold, detecting attack traffic at an earlier stage could be useful in identifying the latter stages of an APT.

The use of current datasets for APT detection is limiting in the sense that (a) there is no APT pattern in the datasets. The data used for machine learning-based APT research works utilize existing datasets such as CAIDA [46], NSL-KDD [20], consists of individual network attacks (probe, DoS, User to Root (U2R)), which are all performed simultaneously. (b) The analysis of recent datasets used for APT detection such as CICIDS 2017 [45] and CICIDS 2018 [16] shows that the attack vectors are limited to a few categories of attack - reconnaissance, privilege escalation, etc. In this work, we created a custom dataset, called the *DAPT 2020* (Dataset for APT 2020), by simulating behavior that mimics APTs on a cloud network for five days. We also collect and provide data from the initial phase when attack vectors for APT were not injected into the system, thus providing a baseline for modeling benign traffic on the network. On subsequent days, we captured attack traffic, representative of different phases of APTs by skilled attackers. APT properties like persistence, and slow & low movement are key characteristics of our dataset.

The key contributions of this research work are as follows:

- We provide dataset DAPT 2020 that captures the various aspects of real-world APT attacks. These include (1) attack behavior both at the interface and inside the network. The threat model used for the creation of the APT dataset incorporates the four main phases of an APT attack - reconnaissance, foothold establishment, lateral movement, and data exfiltration and (2) the traffic features in DAPT 2020 encodes several latent characteristics, such as adaptability and stealthiness, of APTs. To the best of our knowledge, this is the first dataset that captures network behavior spanning *all* the stages of an APT.
- We compare and contrast the properties of our dataset to three popular intrusion detection datasets– the CICIDS 2018 [16], the CICIDS 2017 [45], and the UNB 2015 [37] dataset. We highlight the missing aspects of the current datasets and show that our dataset fills the gaps. Further, we propose the new task of identifying a multi-step attack as opposed to classifying one-off anomalies. We believe that the use of the proposed dataset in the future will help to set new frontiers for developing ML models for real-world cybersecurity scenarios.
- Given the data imbalance in cyber attack datasets, where attack traffic is significantly less than the benign traffic data, we consider the use of state-of-the-art semi-supervised approaches for constructing a representation of the legitimate network behavior and then using it for identifying anomalies. We show that, across the various stages of an APT, these models are hardly effective in detecting attack traffic.

The rest of the paper has been organized as follows. We discuss the key characteristics of current datasets and machine learning models used in APT research in section 2. The design of our dataset DAPT 2020, data collection methodology, and semi-supervised machine learning models used for benchmarking different phases of APT have been discussed in section 3. In section 4, we compare the performance of machine learning models on DAPT 2020, and existing datasets -CICIDS 2017 [45], CICIDS 2018 [16], and UNB 2015 [37]. We discuss the problem associated with the generalizability of existing datasets for APT detection and the need for better machine learning models in section 5. Finally, we conclude the paper in section 6 and provide directions for future research.

2 Related Work

Most of the current research focuses on detecting and mitigating the network intrusion based on pattern matching, signature-based, and anomaly-based IDSs. However, these IDSs fail to detect attack variants, that use the system vulnerabilities, before they damage the system. Although there has been some research done on APT attacks, most of them either describe and analyze the APT attacks that were disclosed such as Stuxnet [12], Duqu [7] and Flame [6]. Research works [44,14] consider APT attacks as a two-player game between attacker and defender. These studies do not discuss solutions for the automatic detection of APTs [31]. Many works that have been surveyed in [48,31] use the information correlation from various sources such as host-based events and network-based events to generate the evidence graphs. The research works are however limited by the type of attack vectors present in the network traffic.

2.1 Analysis on existing datasets

Table (1) Analysis of phases of APT attack covered by attack vectors of existing works involving APT, network intrusion, and anomaly detection in cybersecurity. The table compares attack phases covered by datasets UNB-15 [37], CICIDS 2017 [45], NSL-KDD [20], Mawi [25], ISCX [46], DARPA [17], HERITRIX [53], and DAPT 2020 (our dataset).

APT Phase \ Dataset	UNB-15	CICIDS	NSL-KDD	Mawi	ISCX	DARPA	HERITRIX	DAPT 2020
Normal Traffic	✓	✓	✓	✓	✓	✓	✓	✓
Reconnaissance	✓	✓	✓		✓	✓		✓
Foothold Establishment		✓	✓	✓	✓	✓	✓	✓
Lateral Movement								✓
Data Exfiltration								✓

Our analysis considered the datasets involving security intrusions and anomaly detection. For instance, Pang *et. al.* [40] utilized deep anomaly detection based on deviation networks, and Moustafa *et. al.* [37] used UNSW-NB15 intrusion detection dataset. We considered different phases of the APT attack as measurement metrics. As can be seen in the Table 1, DARPA [17] only covers three phases of APT attack. None of the existing datasets cover data exfiltration, which is essential for the successful completion of an APT attack. Second, we analyzed the attack vectors utilized by different datasets, as described in the Table 2. The recent datasets such as UNB-15 lack attack vectors such as SQL Injection and Account Bruteforce. The datasets currently used for anomaly detection or machine learning-based research, targeting signature-based attacks or APT scenarios, lack a comprehensive set of attacks used for APT.

Table (2) Comparison between attack vectors of each dataset in terms of different attack vectors that is involved in APT attack. The table compares existance of every attack vectors by datasets UNB-15 [37], CICIDS 2018 [16] CICIDS 2017 [45], NSL-KDD [20], MAWI [25], ISCX [46], DARPA [17], HERITRIX [53], and DAPT 2020 (our dataset).

Dataset / **Attack**	**UNB-15**	**CICIDS 2018**	**CICIDS 2017**	**NSL-KDD**	**MAWI**	**ISCX**	**DARPA**	**HERITRIX**	**DAPT 2020**
Network Scan	✓	✓	✓	✓	✓	✓	✓		✓
Web Vulnerability Scan	✓	✓							✓
Account bruteforce		✓	✓	✓	✓	✓	✓		✓
SQL injection		✓	✓	✓		✓			✓
Malware Download	✓		✓						✓
Backdoor	✓	✓				✓			✓
Command Injection	✓	✓	✓				✓	✓	✓
DoS	✓	✓	✓	✓	✓	✓	✓		✓
CSRF		✓	✓			✓			✓
Privilege escalation			✓		✓	✓		✓	✓

2.2 Anomaly Detection and Machine Learning based APT Detection

Machine learning has been found and proven by many researchers as one of the promising solutions towards detecting APT attacks. Qu *et. al.* [41] have proposed an autoencoder model, with the gated-recurrent unit (GRU) as the basic unit, trained in an unsupervised approach towards detecting anomalies in web log data. They compare the accuracy of their model with Long Short Term Memory (LSTM) and Support Vector Machine (SVM) models. They used a clustering approach to reduce the feature space before giving it to the autoencoder. Bohara *et. al.* in [8] presented an unsupervised clustering approach on combined network and host logs to find any malicious activity. They claimed their approach can detect network scan attacks, flooding attacks, and the presence of malware on a host. Both these solutions embraced unsupervised machine learning approaches and are susceptible to high false positives and false negatives. Further, Bohara *et. al.* [8] uses a clustering approach that is affected by the initial seed and number of clusters. Du *et. al.* [22] proposed a DeepLog framework for anomaly detection based on system log where LSTM was utilized to derive a model trained on normal patterns with the ability to detect abnormal activities of DDoS attacks. Kumar et al. [30] proposed a framework to detect security intrusions using a hybrid approach of rules and machine learning techniques. Marchetti *et. al.* [32] proposed a supervised statistical approach based on network traffic logs and access information to detect APT activities after establishing foothold to exfiltration attempts including lateral movement and maintaining access. Siddiqui *et. al.* [47] have used K-Nearest Neighbor (KNN) machine-learning algorithm to detect the activities about the lateral movement and exfiltration stages of an APT attack. Cappers *et. al.* in [10] proposed a semi-supervised machine learning approach to detect APT activities from establishing a foothold stage to data ex-filtration stage by contextual analysis of network traffic alerts.

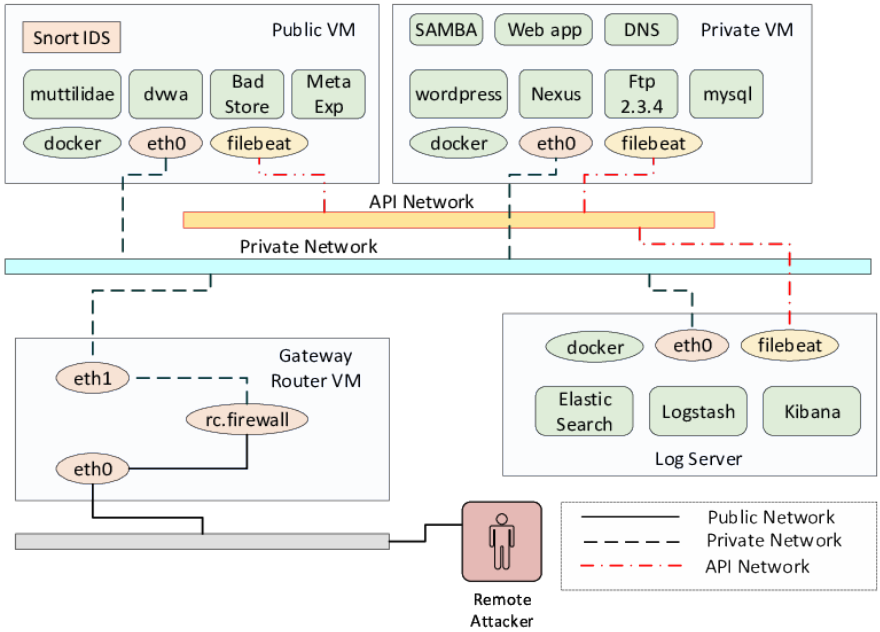

Fig. (1) System set used for construction of DAPT 2020 dataset. The attacker can access only public services exposed via firewall. Log Server (ELK Cluster) is used for collection of network and host logs.

3 DAPT 2020 Dataset Design

A key component lacking in current APT research is an APT dataset. A primary reason for this shortcoming is the legitimate skepticism amongst corporate organizations to share network attack data as it may reveal important aspects of the company. Further, the fear of disclosing personally identifiable information (PIO), and breaching the customer confidentiality agreement prevents companies from sharing this data. Hence, we try to construct an artificial dataset with characteristics of APT behavior as DAPT 2020.

In this section, we first provide an overview of the system-setup for facilitating data-collection. We then describe the data-collection process, giving an overview of the timeline and highlight the tools used for data-collection. Finally, we discuss state-of-the-art techniques that can be leveraged to distinguish between benign and malicious traffic.

3.1 System Setup

We utilized VMWare ESXi physical servers to host the virtual machines (VMs) with different services typical of an enterprise cloud network. As can be seen in the Figure 1, the Public VM comprised vulnerable services such as mutillidae [39], Damn Vulnerable Web Application (DVWA) [23], Metasploitable [43], and BadStore [51]. We utilized Snort, Network-based Intrusion

Table (3) Table with details on data collection on a multi-tenant cloud system with known and unknown vulnerabilities

Day	Activity	Tools Used	Details
Day 1, 8:00 AM-6:00 PM	Normal Traffic	ping, dig, GET, POST, curl, browsing, files upload, download	Baseline normal traffic based on user activities.
Day 2, 8:00 AM-6:00 PM	Reconnaissance	nmap, webscarab, sqlmap, dirbuster, nikto, burpsuite, application account discovery tools	Reconnaissance on public network, identification of vulnerabilities, directory structure, weak authentication, and authorization.
Day 3, 8:00 AM-6:00 PM	Foothold Establishment	PHP reverse shell, netcat, SQL vulnerability exploitation (sqlmap), XSS exploitation, authentication bypass, metasploitable	PHP reverse shell via DVWA, file upload, adding of malicious users was performed on badstore.
Day 4, 8:00 AM-6:00 PM	Lateral Movement	Nmap scan on local network, vsftpd 2.3.4 vulnerability, weak ssh authentication, mysql script for CVE-2012-2122, metasploit	Exploration of internal network from compromised VMs (Public VM), and obtaining foothold on critical local systems.
Day 5, 8:00 AM-6:00 PM	Data Exfiltration	Data exfiltration to C&C, SMB vulnerability CVE-2017-7494 used to obtain elevated privileges, Google Drive, PyExfil, ftp, scp	FTP put method from local machine to remote server, wput to remote location using anonymous user, scp large files to remote server, web based uploads to Google Drive.

Detection System (NIDS) [42] for checking the malicious traffic signatures. Each service was hosted as a separate Docker [33] container. The private VM was used to host services such as Samba, Wordpress website, FTP, MySQL, nexus (repository management). The private and public VMs were connected over the private network. Additionally, each VM had a packet and log capture feature. The ELK stack [13] based log server was used for log storage and filtering. The network and host logs were periodically shipped to the Log Server using filebeat agent as shown in Figure 1.

3.2 Data Collection

To mimic normal traffic seen on read-world cyber systems, a group of users was provided user and (some with) administrative credentials, for accessing public and private services of the network. They performed routine business operations throughout the week. For instance, admin performed some updates to a WordPress website, organized files, folders, users. On Monday, we ensured that no attack traffic was present on the network to generate a baseline for normal traffic. Then, as highlighted in Table 3, various attack methods were employed by our internal Red Team (team of experienced cyber-attackers). They performed a chain of attacks that mimic real-world APT attacks similar to the ones described by Alshamrani *et. al* [1]. On Tuesday, the Red Team attempted exploration (e.g. scanning and fingerprinting) of software present on public services. The team exploited vulnerabilities present on public services. On Wednesday the team used attack scripts and known attack tools such as metasploitable to establish a foothold and gain elevated privileges on the services present on the public network. In the next phase of the attack on Thursday, the Red Team employed lateral movement to exploit critical services in the network such as SMB, and FTP. Finally, the team used data exfiltration methods to send the data to external google drives, and FTP server on Friday. This completed the APT attack. Note, that an actual APT attack takes place over a longer duration, but the attack phases are quite similar to the experimental analysis performed by our internal team. A detailed description of attack tools used and findings during each phase of the attack are present on our public Gitlab repository [38].

Table (4) Comparison of attack methods employed in DAPT 2020 dataset against methods employed by real-world APT attacks - APT41 [21], Target APT Breach [52] and RSA SecureID Attack [49]

APT Phases	**DAPT 2020**	**APT41**	**Target Breach**	**RSA SecureID**
Reconnaissance	Network Scan			
	Application Scan			
	Account Bruteforce	✓		
Establish Foothold	CSRF	✓		
	SQL Injection	✓	✓	
	Malware Download		✓	
	Backdoor	✓		✓
	Reverse Shell	✓	✓	
	Command Injection			
Lateral Movement	Internal Scanning	✓	✓	
	Account Discovery		✓	
	Password Dumping	✓	✓	✓
	Credential Theft	✓	✓	✓
	Creation of user accounts	✓		
	Privelege Escalation	✓		✓
Data Exfiltration	Data Theft	✓	✓	✓

The normal users (students with basic knowledge of website maintenance and access), used shopping interface to checkout items, browse different options, create posts on the website, add comments on particular items, etc. The normal user operations continued over next few days. Attackers (advanced penetration testers) were instructed to be as stealthy as possible and perform attacks in a fashion that prevents any alarms triggered by security tools. The attackers were given access and used tools, techniques, and procedures (TTPs) similar to state of the art APT attacks to simulate APT attack. The data was collected from all the network interfaces in the form of (pcap) files, as well as logs from each host.

In particular, the host logs we collected were as follows:

- Log of system events (Syslog)
- MySQL access log
- Auditd host IDS logs
- Apache Access Logs
- Authentication Logs
- Logs from services - wordpress, docker, samba, ftp
- DNS logs

Constructing a dataset that represents real-world APT attacks is crucial to the success of the dataset and the models generated using that dataset. Our dataset, DAPT 2020, has been constructed by studying different APT attack groups and their methods. Table 4 compares the attack methods employed for constructing our dataset, DAPT 2020, with the attack methods of employed in real-world APT attacks.

3.3 Semi-Supervised Models for APT Detection

Understanding the normal behavior of systems within a network plays a crucial role in defending against APTs [1]. By developing a baseline for the normal behavior of a system, any deviation from this baseline, indicative of abnormal behavior, can be effectively identified. Semi-supervised approaches prevalent in anomaly detection leverage this idea to distinguish between normal and attack traffic at test time [11,9]. An advantage of using such semi-supervised techniques is that they are robust to the issue of data imbalance in network-traffic datasets. In real-world settings, which motivate the construction of such data-sets, the number of attack packets is considerably less than the number of normal traffic packets. For example, the proportion of users doing regular activities on a website like Google or Amazon or Facebook *vs.* the users trying to exploit it, is quite less. According to a study by F-Secure [5], 22% of companies did not detect a single attack in 2018 over 12 months, 20% of respondents detected only one type of attack over that period, whereas 31% companies reported 2-5 attacks. Although data to learn normal behavior is abundant, designing a full-fledged supervised classifier that can detect anomalies well, is quite challenging.

Our dataset consists of traffic data on an interconnected network of systems and is rich in contextual information. Regardless of the day on which the attacks are executed, the amount of attack traffic is a small fraction of the overall data. Thus, it makes sense for us to use semi-supervised learning approaches discussed in the literature. We will now discuss a few of these machine learning models that act as a benchmark for our proposed dataset and also the existing models which are considered later in our experiments.

- **One-Class Support Vector Machines (1-SVM)** are known to be particularly effective in scenarios where there is a large amount of normal traffic and a small fraction of anomalous traffic data [24,35]. The idea is to train the model on the labeled examples of the class that has more data. In our case, we trained the 1-SVM model on the abundant normal network traffic data. We then, at test time, using a pre-defined threshold, decide whether a reconstruction error is large enough to classify it as an anomaly.

– **Stacked Auto Encoder (SAE)** - Auto-encoders are a specific kind of feed-forward neural network [50] that are meant to find a compact latent-space representation of the input which can be leveraged for reconstruction. Autoencoders have one hidden layer and compression occurs between the input and hidden layer while reconstruction occurs between the hidden and the output layer. In Stacked Auto-encoders, the compression function followed by the reconstruction is done with a deep neural network as opposed to a single non-linear layer. During training, the output of an SAE is forced to mimic the input; thus, the loss function seeks to minimize the distance between the original input and the reconstructed output. We first train an SAE on normal traffic data, followed by testing on both normal and anomalous data. The expectation is that although SAE can accurately reconstruct the normal data, it fails to do so effectively for the abnormal data and has higher reconstruction error [2]. This makes it easy for a classifier to detect anomalous network traffic data by comparing the reconstruction error to a pre-defined threshold.
– **Stacked Auto Encoder with Long Short-Term Memory (LSTM-SAE)** - While a regular stacked auto-encoders have been used in many research works, the SAE is not capable of detecting contextual anomalies, which is of great significance in the context of APTs. This is because the input layers of an SAE only accept a single network packet as input. To solve this issue, we use a stacked auto-encoder that uses LSTM cells instead of hidden layer cells of SAEs. LSTMs, which have been successful in time-series analysis [4]. LSTM allows us to consider data across multiple time steps. The modified SAE, termed as LSTM-SAE, helps us to compress network traffic packets in multiple consecutive time-steps and then, reconstruct it. By using the same mechanism of training on abundant normal data and testing on both attack and normal data, we can detect attacks that are executed in parts, i.e. spread across multiple packets. This provides both a good benchmark for our dataset and it is a promising first-step for contextual anomaly detection.

4 Evaluation

In this section, we compare the performance of the different models, mentioned above, on three existing datasets– the CICIDS 2017 [45], CICIDS 2018 [16], UNB 2015 [37]– and our proposed dataset DAPT 2020. The goal is to show that similar semi-supervised learning methods, i.e. similar semi-supervised architectures with similar training hyper-parameters, can detect anomalies better in the case of existing data-sets in comparison to detecting anomalies in our dataset DAPT 2020.

The anomaly detection models we have used are based on the Stacked Auto-Encoder (SAE), the LSTM Stacked Auto-Encoder (LSTM-SAE), and a single-class Support Vector Machine. The key idea behind using these models for anomaly detection briefly highlighted in section 3 is to train these models on the normal traffic data on the network. At test time, given an input, we pass it thorough the auto-encoder and check if the normalized reconstruction error is above a certain threshold. If so, we classify it as an anomalous traffic packet. Otherwise, we classify it as normal traffic.

A metric to gauge the effectiveness of anomaly detection systems in settings that have class imbalance issues, such as anomaly prediction in the context of cyber-attacks, is the Precision-Recall (PR) curve as opposed to more popular measures such as accuracy and Receiver Operating Characteristics (ROC) [19]. First, given that attack representation is often less than 2% in the test-set, even a naive classifier that classifies all data to the majority class will have a 98% accuracy. Further, the difference between algorithms on a dataset (or between datasets using the same algorithm) is harder to reason about within the 2% scale. Second, although both the ROC and the PR curve use the Recall (or the True Positive Rate), ROC uses the False Positive Rate (FPR) in comparison to the PR curve's Precision. The FPR rate is less sensitive to changes in the number of false positives (i.e. normal traffic being classified as attacks) while Precision looks at only the set of samples that are predicted to be positive. Thus, it provides a much better metric when a particular class is severely underrepresented in comparison to another class. A detailed discussion on this topic can be found in [19]. Third, comparing the performance of an algorithm on different data-sets using accuracy or the ROC curve becomes quite misleading in our context because of the different degree to which anomalies are under-represented in the data. For example, the ratio of attack traffic in the case of brute-force attacks for CICIDS 2018 is $\approx 22\%$ while for UNB 2015 it is $\approx 14\%$. Hence, the baselines for the two datasets (i.e., a naive classifier that classifies everything to the majority class label) will have 75% and 86% accuracy respectively. Hence putting them side by side on the accuracy table or the AUC of a ROC curve does not help quantify the effectiveness of an algorithm to classify the data. For completeness, we will discuss the AUC-ROC and AUC-PR data in the subsection 4.4. We now briefly discuss how the PR-curves were constructed and should be interpreted.

In our setting, the model outputs a confidence value of p after normalizing the reconstruction error across all test examples between $[0, 1]$. When the reconstruction error is large, the value of p is close to 1. Thus, p indicates the confidence with which the model predicts an input as an anomaly (i.e. belongs to class 1). To plot a point in the PR-curve, we first set a threshold of τ. We then, for each test inputs, find p, and if $p < \tau$ we classify it as normal traffic (and as anomalous traffic otherwise). By doing this for all test inputs, we can come up with a confusion matrix (that showcases the True/False Positives/Negatives). Finally, we obtain the Precision and Recall for the particular τ and plot it on the PR-curve. The ideal classifier should be able to correctly predict the test label of each input with complete confidence, i.e. for anomalies, it outputs $p = 1$, while for normal, it predicts $p = 0$. Such a classifier plots the line $y = 1$ and then stretches from $(-1, 1)$ to the point $(1, \text{frac. of anomalous examples})$ in the PR curve. On the other hand, a No-Skill (NS) classifier that outputs $p = 1$ on all input data can be plotted using

Fig. (2) Precision-Recall (PR) curves for detecting attacks across the various stages of an APT for the various datasets using the Stacked Auto-encoder (SAE).

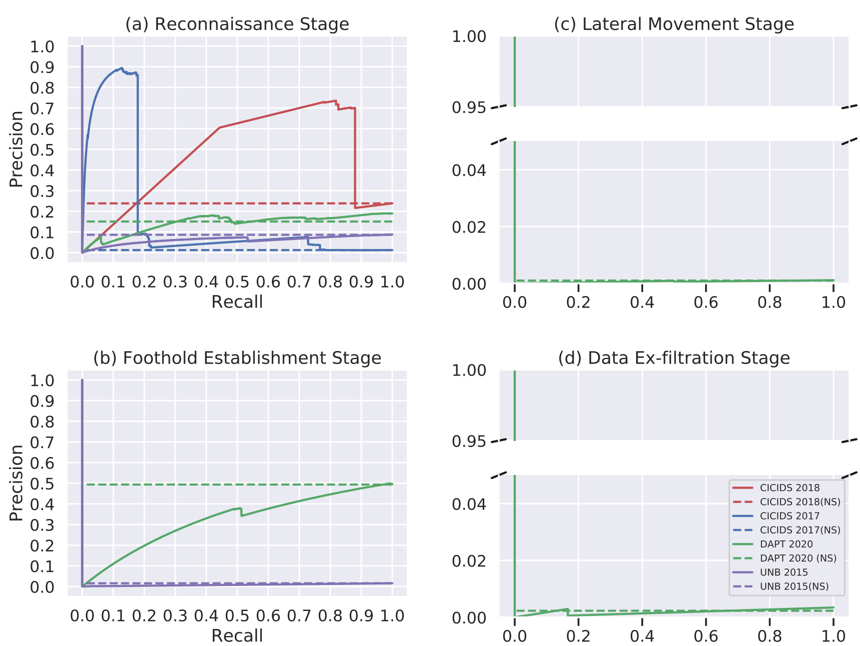

Fig. (3) Precision-Recall (PR) curves for detecting attacks across the various stages of an APT for the various datasets using the Stacked Auto-encoder with LSTM cells (LSTM-SAE).

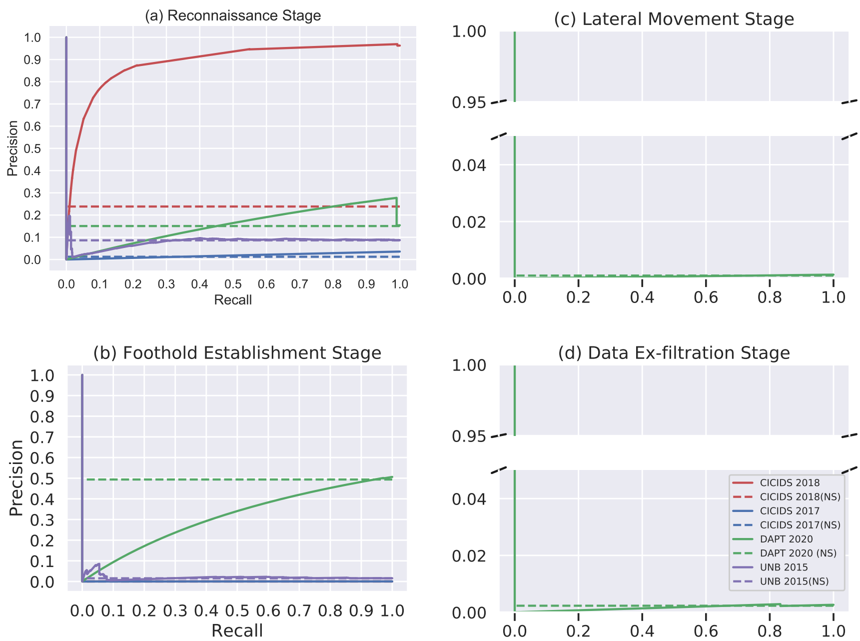

Fig. (4) Precision-Recall (PR) curves for detecting attacks across the various stages of an APT for the various datasets using 1-class Support Vector Machine (1-SVM).

the line,

$$y = \text{Precison} = \frac{TP}{TP + NP} = \frac{\#(\text{Anomalous Traffic})}{\#(\text{Dataset})}$$

where $\#(\cdot)$ denotes the cardinality. For all our PR-curves that follow the no-skill classifier's performance is shown using dotted lines and, in the legend, indexed using the suffix NS. In the PR-space, a curve that is higher compared to another curve or closer to the top-right corner of the unit-square (with corners at $(0, 0)$ and $(1, 1)$) is considered to represent better anomaly detection.

As opposed to considering the detection of anomalies as a whole, which is common in all existing works, we break the anomalies down into the four stages of APTs– Reconnaissance, Foothold Establishment, Lateral Movement, and Data Exfiltration. This helps us highlight the various characteristics of attacks (data imbalance ratios, lack of data) across the different APT stages that make the semi-supervised learning task difficult. We are also able to show that even in the context of existing data, the abundance of attack data in one-phase helps the accuracy of the overall anomaly detection system, which may be highly unreliable in another context.

We divide the results of our experiments into three subsections– one for each of the semi-supervised learning methods. Due to the lack of particular attack vectors in each of the existing datasets, highlighted in Table 1, UNB 2015, CICIDS 2017, and CICIDS 2018 data-sets are only used to detect attack in the Reconnaissance and the Foothold Establishment stages. Each row of figures represents the PR-curves for a particular anomaly detection model and are arranged as per the detection result of reconnaissance data on the left and data-exfiltration on the right. The ordering is the representation of the way APT attacks progress through the system.

Attack Vector Details. The following attack vectors were selected for benchmarking models on the different stages present in the DAPT 2020 dataset:

1. Reconnaissance - Web Application Scan, Port Scan, Account Discovery
2. Foothold Establishment - Directory Bruteforce, SQL Injection, Malware Download
3. Lateral Movement - Port Scan (on a private network), backdoor, SQL Injection (on a private network)
4. Data Exfiltration - exfiltration to a remote FTP server, Google Drive upload.

The training set for individual attack stages comprised of all normal traffic data seen on weekdays when attack vectors belonging to the particular attack stage was absent. On the other hand, the test set comprised of all the traffic data– both attack and normal–on days attack vectors indicative of the attack stage was executed. For example, training data for lateral movement in DAPT 2020 consisted of normal data from Monday, Tuesday, Wednesday, and Friday (days on which there was no lateral movement), whereas the test set consisted of normal and attack data from Thursday.

The results are shown in Figure 2, 3, and 4, mostly portray the failure of the semi-supervised system as being able to detect attack vectors in APT scenarios. Further, the data imbalance also makes it hard for supervised learners to perform well in this context. We sincerely hope that the benchmarking results act as an encouragement for the research community to propose better methods that are better at anomaly detection in the context of real-world APTs.

4.1 Anomaly Detection with SAE

The anomaly detection results using the Stacked Auto-Encoder (SAE) are shown in Figure 2. As can be seen, the classifier performs satisfactorily in only two settings– (1) detecting reconnaissance attacks on the CICIDS 2018 and (2) detecting foothold establishment attacks in our DAPT 2020 dataset. Beyond these cases, the AUC-PR of the classifiers, as shown in Table 5 is highly unsatisfactory and often as bad as the no-skill classifier. We do not compare the results for different datasets across the various stages of the APT.

Reconnaissance The anomaly detection on CICIDS 2018, for which the no-skill classifier has a precision value of 0.23, corresponding to the fraction of anomalies, is far better than on any of the other datasets with the highest precision value of 0.87 for some threshold. For the other datasets, performance on our DAPT 2020 and the UNB 2015 dataset do not differ significantly.

This is because the baseline of the no-skill classifier is higher in our case as opposed to UNB 2015 and thus, similar improvements produce the PR curves plotted in Figure 2(a). Performance on the CICIDS 2017 is the worst with the maximum precision value reaching to barely 0.1. Not surprisingly, the AUC-PR for the CICIDS 2017, shown in Table 5, is the worst for this setting.

Foothold Establishment The fraction of attack data in the context of Foothold Establishment is significantly less in the existing datasets with only a section of the brute force and sparse SQL Injection attack vectors in the haystack of normal data. In our case, nearly 50% of the traffic consists of attacks that try to gain a foothold in the network. The performance of SAE in the context of our dataset dominates the performance on other datasets by a significant margin. Further, the scanty traffic in the other datasets is not easily distinguishable from the normal data, resulting in the SAE behaving as bad as a no-skill classifier in this setting.

Lateral Movement and Data Ex-filtration The y-scale of the graph in this setting ranges show activity between $[0, 0.15]$ instead of the usual $[0, 1]$ scale in the context of other attacks. The plot acts as proof to show that attack data representing lateral movement exists in the dataset. The performance of the SAE is extremely poor, reaching a maximum precision of 0.1. As can be observed, re-scaling the axis is needed for all plots on the detection of lateral movement and data-ex-filtration data, but the performance of other models deteriorates even further.

4.2 Anomaly Detection with LSTM-SAE

The anomaly detection results using the LSTM-based Stacked Auto-encoder (LSTM-SAE) are plotted in Figure 3. The results for LSTM-SAE is only promising for the CICIDS 2017 and the CICIDS 2018 dataset in the context of reconnaissance attacks. A more detailed discussion follows.

Reconnaissance An interesting observation, in comparison to the performance of SAE on the CICIDS-2017 dataset, is that the performance of LSTM-SAE shows significant improvement, jumping from a max precision value of 0.1 in the former case to a value of $\approx$ 0.9 in the latter plot. This is, to an extent, indicative that there exists contextual information that is the CICIDS-2018 dataset that can be leveraged by LSTM-SAEs to better detect anomalies. This might result because of a particular pattern that was used to inject attack data for this dataset. Looking at the performance on the other datasets, it seems that the addition of contextual information makes the distinct representation of the attack vectors difficult, making them close to normal representation and in turn, reducing the effectiveness of anomaly detection.

Foothold Establishment In this setting, the LSTM-SAE turns out to be the worst classifier. It is as bad as using a no-skill classifier for all the datasets concerned. As can be observed from the data, compared to SAE or 1-SVM, even with a sufficiently large fraction of attack data, it cannot perform any better. This gives a clear indication that the use of contextual information (of up to 3 timesteps) dilutes attack data for our dataset. We discuss this further when analyzing on the AUC values for the PR curves.

Lateral Movement and Data Ex-filtration The LSTM-SAEs performance is worse than that of SAE on the Lateral Movement and Data Exfiltration data compared to other stages of the APT attack. This shows that the distribution of contextualized attack vectors in these stages is almost the same as that of normal traffic.

4.3 Anomaly Detection with 1-SVM

In Figure 4, we highlight the performance of a one-class Support Vector Machine on the different data-sets. Other than the case of detecting Foothold establishment attacks on our dataset DAPT 2020, the 1-SVM performs poorly in all the cases.

Reconnaissance The learning classifier performs as bad as the no-skill classifier for all the datasets except CICIDS 2018. In the case of CICIDS 2018, it performs quite well compared to CICIDS 2017, UNB 2015, and DAPT 2020 dataset. It essentially implies that for the CICIDS 2018 dataset, it essentially means that the classifier can identify the anomalous traffic correctly. This can be explained by the fact that a large percent of attack traffic in CICIDS included quite observable attempts by the attacker to perform reconnaissance. The classifier is, however, not able to identify the anomalous events in DAPT 2020 and other datasets, given that the percentage of attack traffic is quite low.

Foothold establishment The 1-SVM performs the best for detecting attacks in the foothold establishment stage in our dataset going up to a 0.9 on precision value while it performs almost as bad as the no-skill classifiers for the other datasets. Since foothold establishment is a key stage associated with any APT attack, this was expected behavior.

Lateral Movement and Data Ex-filtration Similar to the anomaly detection behavior seen in the case of SAE and LSTM-SAE, the PR curves for the 1-SVM show extremely poor performance for detecting attacks in the two final stages of APTs. This shows that attacks were quite stealthy and almost identical to the normal traffic. It is clear that the reliable detection of these attack phases on an actual APT attack is quite difficult with existing anomaly detection models.

Table (5) AUC-ROC and AUC-PR results for machine learning models - SAE, SAE-LSTM, 1-SVM.

Dataset	AUC-ROC			AUC-PR		
	Reconnaissance			**Reconnaissance**		
	SAE	**SAE-LSTM**	**1-SVM**	**SAE**	**SAE-LSTM**	**1-SVM**
UNB 2015	**0.601**	0.352	0.489	0.158	0.061	0.079
CICIDS 2017	0.499	**0.727**	0.66	0.0263	0.173	0.018
CICIDS 2018	**0.832**	0.799	0.99	0.592	0.457	0.88
DAPT 2020	**0.641**	0.525	0.54	0.262	0.143	0.15
Dataset	**Foothold Establishment**			**Foothold Establishment**		
	SAE	**SAE-LSTM**	**1-SVM**	**SAE**	**SAE-LSTM**	**1-SVM**
UNB 2015	**0.602**	0.09	0.547	0.0280	0.009	0.019
CICIDS 2017	**0.365**	0.34	0.670	0.000001	0.00001	0.00018
CICIDS 2018	**0.674**	0.665	0.540	0.0001	0.00001	0.000001
DAPT 2020	**0.846**	0.386	0.058	0.498	0.323	0.313
Dataset	**Lateral Movement**			**Lateral Movement**		
	SAE	**SAE-LSTM**	**1-SVM**	**SAE**	**SAE-LSTM**	**1-SVM**
UNB 2015	NA	NA	NA	NA	NA	NA
CICIDS 2017	NA	NA	NA	NA	NA	NA
CICIDS 2018	NA	NA	NA	NA	NA	NA
DAPT 2020	**0.634**	0.28	0.25	0.0136	0.0006	0.0006
Dataset	**Data Exfiltration**			**Data Exfiltration**		
	SAE	**SAE-LSTM**	**1-SVM**	**SAE**	**SAE-LSTM**	**1-SVM**
UNB 2015	NA	NA	NA	NA	NA	NA
CICIDS 2017	NA	NA	NA	NA	NA	NA
CICIDS 2018	NA	NA	NA	NA	NA	NA
DAPT 2020	**0.685**	0.386	0.298	0.0034	0.0027	0.0015

4.4 Analysis of Performance on AUC

Stacked autoencoder performed well on reconnaissance, and foothold establishment phase, whereas 1-SVM performed better on lateral movement, and data-exfiltration phase. LSTM-SAE performed quite poorly on all phases of APT. AUC-PR values are quite low suggesting anomalies to be sparsely distributed.

We enumerate the Area Under Curve - Receiver Operating Characteristics (AUC-ROC), and AUC Precision-Recall (AUC-PR) values of the classifiers on the dataset UNB 2015, CICIDS 2017, CICIDS 2018, and DAPT 2020 in Table 5. AUC-ROC summarizes the ROC curves of true positives against false positives, while AUC-PR is a summarizes the curve of precision against the recall. If the AUC-ROC value is quite high (close to 1), it implies good performance, whereas AUC-ROC value close to 0.5 means a random ranking of the objects. The choice of performance metric AUC-ROC and AUC-PR depends on the goal of the anomaly detection method. AUC-ROC is used because of good interpretability, however, if the anomaly detection mechanism is sensitive to the performance on the positive class as opposed to negative class, AUC-PR is a good performance metric for anomaly detection. If the anomalies are distributed unevenly in the dataset, the value of AUC-PR is generally low.

Reconnaissance The SAE model performed quite well on most of the datasets in the reconnaissance phase, achieving *0.601* AUC-ROC value, in case of UNB 2015, 0.832 in case of CICIDS 2018, and 0.641 in the case of DAPT 2020 dataset. Surprisingly, the SAE-LSTM showcased better AUC value compared to SAE, and 1-SVM on CICIDS dataset. The AUC-PR values were consistently low for all the algorithms in the reconnaissance phase. This means the attack distribution was quite sparse during the reconnaissance phase and thereby, difficult to detect.

Foothold establishment The SAE showed good performance in the case of DAPT 2020 for foothold establishment (0.846). These results are on the lines of PR curves observed in the previous section for the foothold establishment stage. The AUC-PR values for foothold establishment are higher for DAPT 2020 dataset compared to other datasets, since attack traffic was bit higher compared to other datasets, 0.323 for LSTM-SAE, and 0.313 for 1-SVM.

Lateral Movement and Data Ex-filtration The foothold establishment and lateral movement phases were missing on the existing datasets. The different machine learning models when evaluated on DAPT 2020 consistently showed poor performance, for both lateral movement and data exfiltration. Moreover, the consistently low values of all unsupervised learning algorithms on these phases of APT show that the attack vectors employed in our dataset are highly stealthy and difficult to detect using existing classifiers.

5 Discussion

As already highlighted, the availability of data on APT is difficult because of (1) privacy concerns pertaining to an organization and its customers and (2) spending effort in creating a data-set like DAPT 2020 is both time-consuming and expensive. Given that we now propose the DAPT 2020 dataset, it is natural to consider the use of data-augmentation techniques prevalent in the machine learning community [3]. However, there are two concerns in using data-augmentation techniques. First, the current dataset reflects an APT attack with thin lines between normal and abnormal behaviors of the systems within the network. As a result, ensuring that augmentation of the normal traffic still represents normal behavior of the systems is quiet challenging. Second, unlike regular intrusion detection datasets, this dataset represents an APT attack where in consecutive attack vectors are inter-related. Data augmentation can potentially affect these dependency relations with the generated attack data failing to capture the APT attackers' movement in the network. Further, GANs are known to exacerbate biases and thus, generated data may induce mode collapse, repeating a particular patterns present in the attack data to generate synthetic attack traffic, in turn reducing its rich diversity [27]. We believe, the effectiveness of data augmentation in regards to APT traffic needs to be investigated, and we intend to move in this direction in near future. We plan to consider GAN based models to identify better machine learning models in context of APT attacks.

While machine learning models are known to be less effective when test data is out-of-distribution (OOD), the problem amplifies further in the context of cyber-security. Each system is highly specific to the software it uses, the inputs (or outputs) it expects (or generates), the traffic patterns seen etc. Hence, a model trained on a particular dataset might not be as effective when used in a different context. DAPT 2020 helps bridge this gap and provides motivation for the development of machine learning technologies suitable for APT attack detection.

6 Conclusion

Advanced Persistent Threats are one of the most challenging attacks to defend against. Several machine learning research works have tried to address the APT detection problem. They are, however, limited by the attack vectors, and the attack phases critical for an APT attack. We propose a new DAPT 2020 dataset, and benchmark existing anomaly detection models on our dataset. The performance of anomaly detection models in terms of precision-recall (PR) values, AUC-ROC, and AUC-PR values is consistently low. This shows that reliable detection of APT attacks using existing machine learning models is very difficult, and more effort needs to be invested towards creating better learning models for APT detection. Further, a key component that is required for defending against APTs is a correlation model that correlates the anomalies detected. We believe this DAPT 2020 dataset instigates development of fine correlation models that help detect a threat in its entirety and not just the individual attack vectors. The code and dataset for this research work can be found at `https://gitlab.thothlab.org/Advanced-Persistent-Threat/apt-2020/tree/master` (along-with a detailed description of DAPT-20 [38]).

7 Acknowledgement

This research is supported in part by following research grants: Naval Research Lab N0017319-1-G002, NSF DGE-1723440, OAC-1642031. Sailik Sengupta was supported in part by the IBM Ph.D. Fellowship.

References

1. Alshamrani, A., Myneni, S., Chowdhary, A., Huang, D.: A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. IEEE Communications Surveys & Tutorials **21**(2), 1851–1877 (2019)
2. An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE **2**(1), 1–18 (2015)
3. Antoniou, A., Storkey, A., Edwards, H.: Data augmentation generative adversarial networks. arXiv preprint arXiv:1711.04340 (2017)

4. Bao, W., Yue, J., Rao, Y.: A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one **12**(7) (2017)
5. Barker, I.: Attack traffic up 32 percent in 2018 (2018), https://betanews.com/2019/03/05/attack-traffic-increase/
6. Bencsáth, B., Pék, G., Buttyán, L., Felegyhazi, M.: The cousins of stuxnet: Duqu, flame, and gauss. Future Internet **4**(4), 971–1003 (2012)
7. Bencsáth, B., Pék, G., Buttyán, L., Félegyházi, M.: Duqu: Analysis, detection, and lessons learned. In: ACM European Workshop on System Security (EuroSec). vol. 2012 (2012)
8. Bohara, A., Thakore, U., Sanders, W.H.: Intrusion detection in enterprise systems by combining and clustering diverse monitor data. In: Proceedings of the Symposium and Bootcamp on the Science of Security. pp. 7–16. ACM (2016)
9. Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., Benini, L.: A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems. Engineering Applications of Artificial Intelligence **85**, 634–644 (2019)
10. Cappers, B.C., van Wijk, J.J.: Understanding the context of network traffic alerts. In: Visualization for Cyber Security (VizSec), 2016 IEEE Symposium on. pp. 1–8. IEEE (2016)
11. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 15 (2009)
12. Chen, T.M., Abu-Nimeh, S.: Lessons from stuxnet. Computer **44**(4), 91–93 (2011)
13. Chhajed, S.: Learning ELK Stack. Packt Publishing Ltd (2015)
14. Chowdhary, A., Sengupta, S., Huang, D., Kambhampati, S.: Markov game modeling of moving target defense for strategic detection of threats in cloud networks. AAAI Workshop on Artificial Intelligence for Cyber-Security (2018)
15. Chung, C.J., Khatkar, P., Xing, T., Lee, J., Huang, D.: Nice: Network intrusion detection and countermeasure selection in virtual network systems. IEEE transactions on dependable and secure computing **10**(4), 198–211 (2013)
16. CSE-CIC-IDS2018: A collaborative project between the communications security establishment (cse) and the canadian institute for cybersecurity (cic) (2018), https://www.unb.ca/cic/datasets/ids-2018.html
17. Cunningham, R.K., Lippmann, R.P., Fried, D.J., Garfinkel, S.L., Graf, I., Kendall, K.R., Webster, S.E., Wyschogrod, D., Zissman, M.A.: Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation. Tech. rep., MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB (1999)
18. DARPA: Darpa scalable network monitoring (snm) program traffic (11/03/2009 to 11/12/2009) (2012). https://doi.org/10.23721/111/1354735
19. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd international conference on Machine learning. pp. 233–240 (2006)
20. Dhanabal, L., Shantharajah, S.: A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. International Journal of Advanced Research in Computer and Communication Engineering **4**(6), 446–452 (2015)
21. Dragon, D.: Double dragon: Apt41, a dual espionage and cyber crime operation. https://content.fireeye.com/apt-41/rpt-apt41, (Accessed on 07/29/2020)
22. Du, M., Li, F., Zheng, G., Srikumar, V.: Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1285–1298. ACM (2017)

23. DVWA, U.: Damn vulnerable web application (2020), http://www.dvwa.co.uk/
24. Emmott, A.F., Das, S., Dietterich, T., Fern, A., Wong, W.K.: Systematic construction of anomaly detection benchmarks from real data. In: Proceedings of the ACM SIGKDD workshop on outlier detection and description. pp. 16–21 (2013)
25. Fontugne, R., Borgnat, P., Abry, P., Fukuda, K.: Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In: Proceedings of the 6th International COnference. p. 8. ACM (2010)
26. Ghafir, I., Hammoudeh, M., Prenosil, V., Han, L., Hegarty, R., Rabie, K., Aparicio-Navarro, F.J.: Detection of advanced persistent threat using machine-learning correlation analysis. Future Generation Computer Systems **89**, 349–359 (2018)
27. Jain, N., Olmo, A., Sengupta, S., Manikonda, L., Kambhampati, S.: Imperfect imaganation: Implications of gans exacerbating biases on facial data augmentation and snapchat selfie lenses. arXiv preprint arXiv:2001.09528 (2020)
28. Kim, H., Kim, J., Kim, I., Chung, T.m.: Behavior-based anomaly detection on big data (2015)
29. Kissel, R.: Glossary of key information security terms. Diane Publishing (2011)
30. Kumar, R.S.S., Wicker, A., Swann, M.: Practical machine learning for cloud intrusion detection: Challenges and the way forward. arXiv preprint arXiv:1709.07095 (2017)
31. Marchetti, M., Pierazzi, F., Colajanni, M., Guido, A.: Analysis of high volumes of network traffic for advanced persistent threat detection. Computer Networks **109**, 127–141 (2016)
32. Marchetti, M., Pierazzi, F., Guido, A., Colajanni, M.: Countering advanced persistent threats through security intelligence and big data analytics. In: Cyber Conflict (CyCon), 2016 8th International Conference on. pp. 243–261. IEEE (2016)
33. Merkel, D.: Docker: lightweight linux containers for consistent development and deployment. Linux journal **2014**(239), 2 (2014)
34. Micro, T.: Advanced persistent threat awareness (2018), https://www.trendmicro.it/media/misc/apt-survey-report-en.pdf
35. Microsoft: One-class support vector machine (2019), https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/one-class-support-vector-machine
36. Milajerdi, S.M., Gjomemo, R., Eshete, B., Sekar, R., Venkatakrishnan, V.: Holmes: real-time apt detection through correlation of suspicious information flows. arXiv preprint arXiv:1810.01594 (2018)
37. Moustafa, N., Slay, J.: Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 military communications and information systems conference (MilCIS). pp. 1–6. IEEE (2015)
38. Myneni, S., Chowdhary, A.: Apt dataset detailed description (March 2020), https://gitlab.thothlab.org/Advanced-Persistent-Threat/apt-2020/
39. OWASP: Owasp mutillidae 2 project (2020), https://wiki.owasp.org/index.php/
40. Pang, G., Hengel, A.v.d., Shen, C.: Weakly-supervised deep anomaly detection with pairwise relation learning. arXiv preprint arXiv:1910.13601 (2019)
41. Qu, Z., Su, L., Wang, X., Zheng, S., Song, X., Song, X.: A unsupervised learning method of anomaly detection using gru. In: 2018 IEEE International Conference on Big Data and Smart Computing (BigComp). pp. 685–688. IEEE (2018)
42. Roesch, M., et al.: Snort: Lightweight intrusion detection for networks. In: Lisa. vol. 99, pp. 229–238 (1999)

43. Security, O.: Metasploitable unleashed (2020), https://www.offensive-security.com/metasploit-unleashed/requirements/
44. Sengupta, S., Chowdhary, A., Huang, D., Kambhampati, S.: General sum markov games for strategic detection of advanced persistent threats using moving target defense in cloud networks. In: International Conference on Decision and Game Theory for Security. pp. 492–512. Springer (2019)
45. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: A detailed analysis of the cicids2017 data set. In: International Conference on Information Systems Security and Privacy. pp. 172–188. Springer (2018)
46. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. computers & security **31**(3), 357–374 (2012)
47. Siddiqui, S., Khan, M.S., Ferens, K., Kinsner, W.: Detecting advanced persistent threats using fractal dimension based machine learning classification. In: Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics. pp. 64–69. ACM (2016)
48. Singh, S., Sharma, P.K., Moon, S.Y., Moon, D., Park, J.H.: A comprehensive study on apt attacks and countermeasures for future networks and communications: challenges and solutions. The Journal of Supercomputing pp. 1–32 (2016)
49. TheRegister: Rsa explains how attackers breached its systems • the register. https://www.theregister.com/2011/04/04/rsa_hack_howdunnit/, (Accessed on 07/29/2020)
50. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research **11**(Dec), 3371–3408 (2010)
51. Vulnhub: Vulnhub badstore (2020), https://www.vulnhub.com/entry/badstore-123,41/
52. Wagner, R., Fredrikson, M., Garlan, D.: An advanced persistent threat exemplar. MONTH (2017)
53. Wang, Y., Cai, W.d., Wei, P.c.: A deep learning approach for detecting malicious javascript code. security and communication networks **9**(11), 1520–1534 (2016)
54. Yuan, X.: Phd forum: Deep learning-based real-time malware detection with multi-stage analysis. In: Smart Computing (SMARTCOMP), 2017 IEEE International Conference on. pp. 1–2. IEEE (2017)
55. Zhao, G., Xu, K., Xu, L., Wu, B.: Detecting apt malware infections based on malicious dns and traffic analysis. IEEE Access **3**, 1132–1142 (2015)

A Appendix

A.1 APT Attack Phases

The detailed description of different APT phases are as follows:

Reconnaissance

- **Scan Applications** - Nessus, Web Scarab, Burp Suite. Find vulnerabilities such as XSS, XSRF, SQL Injection etc.
- **Scan Network** - NMap, Portsweep, Mscan, Satan, Ipsweep, Saint. Find systems' fingerprints, network architecture information etc. Firewall should log deny event. If multiple denies are seen against unique destination ports from the same origin host within a small windows of time, it is safe to assume that some sort of port scanning activity is taking place.

Establish Foothold

- **Download or Install Malware** - Scanbox, Backdoor Sogu, PoisonIvy, KeyLoggers.
- **R2L** - Guess_Password, Ftp_Write, Imap, Phf, Multihop, Warezmaster, Warezclient, SpyXlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named.
- **C&C Communication** - Send communication to external server that the malware has been installed. Monitor network traffic originating from a system to an external server, after a download of a file or similar network activitiy.

Lateral Movement

- **Credential Compromise** - Key Loggers, Hash retrieval, LDAP, Metasploit.
- **Privilege Escalation (U2R)** - Buffer_Overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, PS.

Internal Reconnaissance Same as Reconnaissance above, just from different source in search of data. IP range might be probed for port 1433 in case of enumerating SQL servers. Ports 135-139 are usually probed by attackers when in search of network shares.

Data Exfiltration Uploading to Google Drive, Dropbox, AWS or any such cloud. Need to baseline against the normal activity of a system.

Cover Up Deletion of log files, modification of log files etc. Needs host based intrusion detection agent. OUT OF SCOPE for current research.

A.2 APT Feature Description

We collected the following features from network and host logs. The details of features extracted extracted from the data collected are present in the Table 6.

Table (6) APT20 Feature Description

fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	Total size of packet in forward direction
fw_pkt_l_max	Maximum size of packet in forward direction
fw_pkt_l_min	Minimum size of packet in forward direction
fw_pkt_l_avg	Average size of packet in forward direction
fw_pkt_l_std	Standard deviation size of packet in forward direction
Bw_pkt_l_max	Maximum size of packet in backward direction
Bw_pkt_l_min	Minimum size of packet in backward direction
Bw_pkt_l_avg	Mean size of packet in backward direction
Bw_pkt_l_std	Standard deviation size of packet in backward direction
fl_byt_s	Flow byte rate that is number of packets transferred per second
fl_pkt_s	Flow packets rate that is number of packets transferred per second
fl_iat_avg	Average time between two flows
fl_iat_std	Standard deviation time two flows
fl_iat_max	Maximum time between two flows
fl_iat_min	Minimum time between two flows
fw_iat_tot	Total time between two packets sent in the forward direction
fw_iat_avg	Mean time between two packets sent in the forward direction
fw_iat_std	Standard deviation time between two packets sent in the forward direction
fw_iat_max	Maximum time between two packets sent in the forward direction
fw_iat_min	Minimum time between two packets sent in the forward direction
bw_iat_tot	Total time between two packets sent in the backward direction
bw_iat_avg	Mean time between two packets sent in the backward direction
bw_iat_std	Standard deviation time between two packets sent in the backward direction
bw_iat_max	Maximum time between two packets sent in the backward direction
bw_iat_min	Minimum time between two packets sent in the backward direction
fw_psh_flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
bw_psh_flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
fw_urg_flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)

bw_urg_flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
fw_hdr_len	Total bytes used for headers in the forward direction
bw_hdr_len	Total bytes used for headers in the forward direction
fw_pkt_s	Number of forward packets per second
bw_pkt_s	Number of backward packets per second
pkt_len_min	Minimum length of a flow
pkt_len_max	Maximum length of a flow
pkt_len_avg	Mean length of a flow
pkt_len_std	Standard deviation length of a flow
pkt_len_va	Minimum inter-arrival time of packet
fin_cnt	Number of packets with FIN
syn_cnt	Number of packets with SYN
rst_cnt	Number of packets with RST
pst_cnt	Number of packets with PUSH
ack_cnt	Number of packets with ACK
urg_cnt	Number of packets with URG
cwe_cnt	Number of packets with CWE
ece_cnt	Number of packets with ECE
down_up_ratio	Download and upload ratio
pkt_size_avg	Average size of packet
fw_seg_avg	Average size observed in the forward direction
bw_seg_avg	Average size observed in the backward direction
fw_byt_blk_avg	Average number of bytes bulk rate in the forward direction
fw_pkt_blk_avg	Average number of packets bulk rate in the forward direction
fw_blk_rate_avg	Average number of bulk rate in the forward direction
bw_byt_blk_avg	Average number of bytes bulk rate in the backward direction
bw_pkt_blk_avg	Average number of packets bulk rate in the backward direction
bw_blk_rate_avg	Average number of bulk rate in the backward direction
subfl_fw_pk	The average number of packets in a sub flow in the forward direction
subfl_fw_byt	The average number of bytes in a sub flow in the forward direction
subfl_bw_pkt	The average number of packets in a sub flow in the backward direction
subfl_bw_byt	The average number of bytes in a sub flow in the backward direction
fw_win_byt	Number of bytes sent in initial window in the forward direction
bw_win_byt	Number of bytes sent in initial window in the backward direction

fw_act_pk	Number of packets with at least 1 byte of TCP data payload in the forward direction
fw_seg_min	Minimum segment size observed in the forward direction
atv_avg	Mean time a flow was active before becoming idle
atv_std	Standard deviation time a flow was active before becoming idle
atv_max	Maximum time a flow was active before becoming idle
atv_min	Minimum time a flow was active before becoming idle
idl_avg	Mean time a flow was idle before becoming active
idl_std	Standard deviation time a flow was idle before becoming active
idl_max	Maximum time a flow was idle before becoming active
idl_min	Minimum time a flow was idle before becoming active