Finding Rats in Cats: Detecting Stealthy Attacks using Group Anomaly Detection

Aditya Kuppa*†, Slawomir Grzonkowski*, Muhammad Rizwan Asghar‡, and Nhien-An Le-Khac†
*Symantec Corporation, Ireland
†School of Computer Science, University College Dublin, Ireland
‡School of Computer Science, The University of Auckland, New Zealand
aditya.kuppa@ucdconnect.ie, slawomir_grzonkowski@symantec.com, r.asghar@auckland.ac.nz, an.lekhac@ucd.ie

***Abstract*—Advanced attack campaigns span across multiple stages and stay stealthy for long time periods. There is a growing trend of attackers using off-the-shelf tools and pre-installed system applications (such as *powershell* and *wmic*) to evade the detection because the same tools are also used by system administrators and security analysts for legitimate purposes for their routine tasks. Such a dual nature of using these tools makes the analyst's task harder when it comes to spotting the difference between attack and benign activities. To start investigations, event logs can be collected from operational systems; however, these logs are generic enough and it often becomes impossible to attribute a potential attack to a specific attack group.**

Recent approaches in the literature have used anomaly detection techniques, which aim at distinguishing between malicious and normal behavior of computers or network systems. Unfortunately, anomaly detection systems based on point anomalies are too rigid in a sense that they could miss malicious activity and classify the attack, not an outlier. Therefore, there is a research challenge to make better detection of malicious activities. To address this challenge, in this paper, we leverage Group Anomaly Detection (GAD), which detects anomalous collections of individual data points.

Our approach is to build a neural network model utilizing Adversarial Autoencoder (AAE-α) in order to detect the activity of an attacker who leverages off-the-shelf tools and system applications. In addition, we also build *Behavior2Vec* and *Command2Vec* sentence embedding deep learning models specific for feature extraction tasks. We conduct extensive experiments to evaluate our models on real world datasets collected for a period of two months. Our method discovered 2 new attack tools used by targeted attack groups and multiple instances of the malicious activity. The empirical results demonstrate that our approach is effective and robust in discovering targeted attacks, pen-tests, and attack campaigns leveraging custom tools.

***Index Terms*—Group Anomaly Detection, Deep Learning, Advanced Threats, Log data analysis, Digital forensics**

I. Introduction

Today, with the increase in deployments of security controls inside enterprises, attackers are relying on tools already installed on the system for their attack campaigns [8]. For instance, intrusion kill chain [1] can lead to a successful penetration (e.g., a drive-by-download [7] or a spear-phishing [6]) or there could be one of the other intrusion stages including, for instance, reconnaissance, Command and Control (C&C) communications, privilege escalation, lateral movement through the network, and exfiltration of confidential information.

Different system tools are leveraged by attackers and cybercriminals alike to achieve their goal [8]. For example, tools like *net* and *powershell* can be used to discover information about the target environment, establish remote connections, run scripts to move laterally inside the victim environment, and exfiltrate sensitive data. Recent attacks against the Democratic National Committee (DNC) used *powershell* for lateral movement and discovery. The Odinaff group, which attacked SWIFT systems used in the financial services and banking sectors, used *mimikatz* tool to dump user passwords from memory. The network scanner allowed the group to identify other computers in the same local network. The dumped credentials were then used with *powershell* to start a new process on one of the identified remote computers.

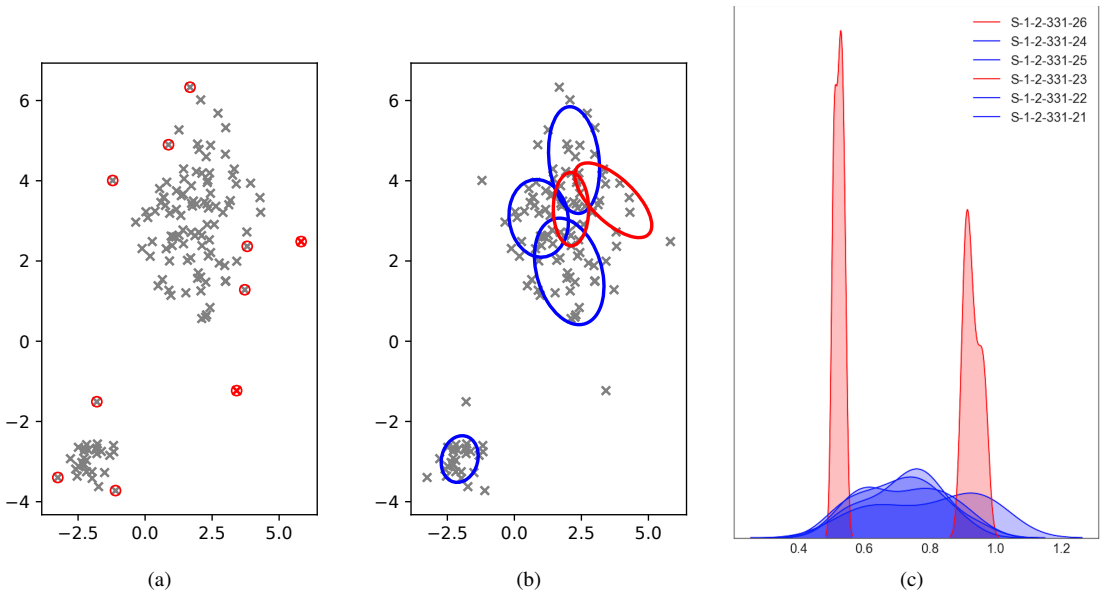
On the defensive side, the detection of patterns that differ from typical behavior is utterly important to detect new threats. This requirement has been satisfied by using algorithms that are capable of detecting point anomalies. Many of such approaches cannot detect a variety of different deviations that are evident in group datasets. For example, the activity of a domain admin on a machine can be similar to an attacker activity confusing any point anomaly detectors. Identifying attacker activities, in this case, require more specialized techniques for robustly differentiating such behavior.

Group Anomaly Detection (GAD) aims to identify groups that deviate from the regular group pattern [5]. Generally, a group consists of a collection of two or more points and group behavior could be described by a greater number of observations. GAD has been studied in various domains to find group anomalies where point-wise methods failed. For example, Muandet et al. [9] possibly discover Higgs bosons as a group of collision events in high energy particle physics; whereas, point-wise methods are unable to identify this anomalous behavior. Soleimani and Miller [10] characterize documents by topics and anomalous clusters of documents are discovered by their irregular topic mixtures. By incorporating additional information from pairwise connection data, Yu et al. [11] found potentially irregular communities of co-authors in various research communities. GAD has been applied to detect group anomalies in video [13] and image data [15]. To the best of our knowledge, GAD has not been applied to solve security problems we have addressed in this work.

In this paper, a Windows session – User's Security Identifier (SID) – is used as the grouping key and all activities occurring in that session are attributed to group behavior. Figure 1 illustrates an example of how group anomaly detection can

Table I
EACH ROW REPRESENTS COMMAND LINE EXECUTIONS IN A WINDOWS SESSION, COLOUR OF ROW INDICATES WHETHER THE SESSION IS AN **ATTACKER** SESSION OR A **NORMAL** ADMIN/USER SESSION.

Session ID	Commands Executed
S-1-2-331-21	cmd hostname,whoami query user ipconfig -all ping www.google.com net user, net view /domain tasklist /svc netstat -ano | find %TCP% msdtc [IP] [port]
S-1-2-331-22	ruby.exe" "tools%ruby%ridk%current%bin%rspec/cookbooks/configservice 1.exe" "--url" "ssh://access/ssh?team= iexplore.exe" SCODEF:XXXX CREDAT:XXXXXX /prefetch:X ping [IP] monitor.log
S-1-2-331-23	chrome.exe", cmd.exe /K RSSFeeds.bat cmd.exe" /C "backupupgrade.bat powershell.exe"Bypass-EncodedCommand JABwACAAPQAgAFMAdA.. powershell.exe iex (Text.Encoding..
S-1-2-331-24	inventory.exe "01-run.xml" winword.exe" /n "downloads%dada.doc" /o "" installer.exe" -o install -ip [localip] -u appdatalocaltemptmp0158.exe", /c net config workstation
S-1-2-331-25	setuptestcenterwin.lax" "appdata%local%temp%ladaesda.tmp, %Temp%EWH.bat, cmd.exe /Q /c powershell -nop -w hidden -encodedcommand JABzAD0AT, excel.exe" /e, appdata%roaming%tmpxxx.exe, appdatalocaltemp%tmpxxxx.exe", /c net config workstation
S-1-2-331-26	cmd.exe/c""apps%eclipse%lunaee_4_4_1%eclipse%lunaee.bat, ipsec.exe, ping -n 1 [IP], "cmd", query.exe" user, "cmd.exe" /c whoami, whoami.exe" /user temp%update.exe



(a) (b) (c)

Figure 1. (a) Feature space of 6 sessions projected in 2-D latent space, point anomalies are marked in circles (b) Known Group membership of features per session Attacker session, normal/user session. (c) Latent distribution Z of 6 sessions, Attacker sessions and normal/user session.

be applied to discover attacker activity. For example, out of six sessions observed on an endpoint (see Table I, four are normal behavior and two sessions are of real attacker activity. Existing state-of-the-art point anomaly detectors (Figure 1-(a)), miss the attacker activity as the attacker's session does not exhibit any outlier characteristics when compared to normal activities. When we apply distribution (Figure 1-(c)) based group anomaly method, with carefully chosen grouping (Figure 1-(b)) and deviance functions (Equation 2), it clearly differentiates between benign and attacker sessions.

The main contributions of this paper are as follows:

- We apply GAD to identify anomalies and propose a new deep generative model: Adversarial Autoencoders-α (AAE-α) based on Adversarial Autoencoders (AAE) to detect attacker activities. Although anomaly detection has been applied to solve multiple problems in the security domain, discovering advanced attackers leveraging off-the-shelf tools and system applications in their campaigns has not been thoroughly explored for real world datasets.
- We build two novel deep learning embedding models: *Behavior2Vec* and *Command2Vec* for feature extraction specific to the security domain. The low-level behavior observed on an endpoint is mapped to a subset of MITRE ATT&CK tactics that are used as features to build the models.
- The method was tested on real world data collected from 12000 enterprises with more than 100k endpoints for a period of two months. The empirical results demonstrate that our approach is effective and robust in detecting attacks including targeted attacks, pen-tests, and attack campaigns leveraging custom tools.

The rest of the paper is organized as follows. We review related work in Section II. We describe the methodology and elaborate on our proposed solution in Section III. Description of datasets, feature extraction methods, and motivations are presented in Section IV. In Section V, we present the results

of our experiments. Finally, Section VI concludes our work and gives some recommendations for further research.

II. Related Work

The field of anomaly detection has been thoroughly studied since decades [3], [4], [40], [42]. The usefulness of such algorithms to host protection has been confirmed long-time ago [16] as this is the main tool to detect previously unknown attacks.

Several approaches use log correlation to perform detection by clustering similar logs and by identifying causal relationships between logs [30], [31]. Beehive [34] identifies potential security threats from logs by unsupervised clustering of data-specific features, and then manually labeling outliers. Oprea [35] uses belief propagation to detect early-stage enterprise infection from DNS logs. BotHunter [32] employs an anomaly-based approach to correlate dialog between internal and external hosts in a network. HERCULE [33] leverages community discovery techniques to correlate attack steps that may be dispersed across multiple logs. All the aforementioned techniques either find point anomalies or group similar logs by clustering. In our approach, we discover attacker activity that is similar to normal activity and not a point anomaly.

Recent advances in Recurrent Neural Networks, motivated additional work in this field, for example, Du et al. [17] proposed DeepLog that uses Long Short-Term Memory (LSTM) networks. This approach is able to capture high dimension non-linear dependencies. In contrast to our work, their approach uses raw system logs that contain unstructured information, typically in an unordered manner, due to concurrency reasons; and also, the idea has been used to detect only one type of threat: Denial of Service (DoS). The problem has been addressed by Brown et al. [18] who performed unsupervised anomaly detection by extending a typical LSTM architecture with attention mechanism. The approach provides an output if the input log is malicious or not, without specifying generated threat category.

Other recent approaches proposed to detect anomalies include energy-based models [24] and Generative Adversarial Network (GAN) [36]. Next, Zhou and Paffenroth [19] proposed a method to train robust autoencoders, drawing inspiration from robust statistics [20] and more specifically robust Principal Component Analysis (PCA). Yang et al. [21] focus on clustering and train autoencoders that generate latent representations that are friendly for k-means.

A recent survey by Toth et al. [5] on group anomaly detection and group change detection describes developments in the application of GAD in static and dynamic situations. Xiong [22] provides a more detailed description of current state-of-the-art GAD methods. Yu et al. [23] further review GAD techniques, where group structures are not previously known. However, clusters are inferred based on additional information of pairwise relationships between data instances.

III. Methodology

A group is a collection of two or more related data instances [5]. Groups which deviate significantly when compared with other groups are known as group anomalies. Group anomalies can be point based or distribution based. Point-based anomalous groups are where all members are also point-wise anomalies. In a distribution-based group anomaly, a collection of points differs from expected group patterns; however, individual data instances may not seem anomalous. We formulate the problem of discovering attacker activity that may be similar to normal admin/user activity as a distribution based group anomaly detection. Attacker behavior may be similar to normal admin user, but when we compare across groups of admins/users activities they significantly deviate by their distributions. To capture group behaviors, we need an objective function that minimizes intra-group variations while simultaneously achieving maximal inter-group separation.

First, consider a set of groups $\mathcal{G} = \{\mathbf{G}_m\}_{m=1}^{M}$, where the mth group contains N_m observations with

$$\mathbf{G}_m = \big(X_{nv}\big) \in \mathbb{R}^{N_m \times V} \tag{1}$$

where X_{nv} is the vth feature (v = 1, 2, ..., V) of observation n (n = 1,2, .., N_m) in the group G_m, R is a continuous value domain. The total number of individual observations is $N = \sum_{m=1}^{M} N_m$.

For inter-group similarity, we measure the distance between groups by restricting the group boundaries by α, The parameter α in Eq. 2 determines the group spread and helps to achieve uniform inter-group variations. Our visualisation of the learnt features in Fig. 1 demonstrate that the attacker activity mainly falls in the tail end of distributions. For our proposed method, normal activity in groups is directly related with the value of parameter α. We perform experiment on dataset for different values of the the parameter $\alpha = \{1, 20\}$.

$$\mathcal{G}^{(exp)} = max\Big[\exp\Big(-\frac{\parallel \mathbf{G}_i - \mathbf{G}_j \parallel^2}{\alpha}\Big)\Big] \tag{2}$$

The results in Fig. 4 show that the optimal performance is achieved for values of α between 10.

To learn intra-group compactness of features we design a loss function

$$L_{gg} = \sum_{z} \max\big(0, \lambda + d(\mathbf{f}_i, \mathbf{G}_z) - d(\mathbf{f}_i, \mathbf{G}_y)\big) \ : z \neq y, \tag{3}$$

where $\mathbf{f}_i$ is activity feature observed in a group $\mathbf{G}_z$, $d(\mathbf{f}_i, \mathbf{G}_z)$ is the similarity of the $\mathbf{f}_i$ with other activity features in the same group, $d(\mathbf{f}_i, \mathbf{G}_y)$ is similarity of the activity feature $\mathbf{f}_i$ with other groups represented by $\mathbf{G}_y$), and λ is the enforced margin.

Then, a distance metric $d(\cdot, \cdot) \geq 0$ is applied to measure the deviation of a particular group from the other groups. The distance score $d\Big(\mathcal{G}^{(exp)}, \mathbf{G}_m\Big)$ quantifies the deviance of the mth group from the expected group pattern, where larger values are associated with more anomalous groups. In summary the proposed method provides us: **(a)** the adjustability to enforce margin constraints on group distributions, **(b)** capture exact group boundaries for attacker and normal activity, and **(c)** control the variance of learned features in a group and therefore enhancing intra-group compactness.

Algorithm 1: Group anomaly detection using AAE-α

Input : Groups $\mathcal{G} = \{\mathbf{G}_m\}_{m=1}^{M}$

Output: Group anomaly scores **S** for input groups $\mathcal{G}$

begin
  Draw a random latent representation $z - z\_tail \sim f_\phi(z|\mathcal{G})$ Reconstruct sample using decoder $g_\psi(\mathcal{G}|z)$
  for *(m = 1 to M)* **do**
    Compute the score $s_m = d\left(\mathcal{G}^{(exp)}, \mathbf{G}_m\right)$
  end
  Sort scores in descending order **S**=$\{s_{(M)} > \cdots > s_{(1)}\}$
  $\{s_{(m)}\}_{m=1}^{M}$ Groups that are furthest from $\mathcal{G}^{(ref)}$ are more anomalous.
  return S
end

Adversarial Autoencoders (AAEs). An AAE is a generative model that is trained with dual objectives (i.e., a traditional reconstruction error criterion and an adversarial training criterion [14]). The encoder in AAE learns to convert the data distribution to a latent representation with an arbitrary prior distribution, attempting to minimize the reconstruction error. In other words, a GAN is attached to the latent layer.

We specifically choose an AAE over other variants in our work for the following reasons:

- In order to capture the true distribution of the latent space, we want to filter out latent vectors near the tail-end of the latent distribution in the scoring logic. For other GAN variants, it is non-trivial to encode an arbitrary sample back into the latent space.
- Since session data follows seasonal patterns with AAEs, we can capture a variety of prior distributions on the latent space without knowing the exact functional form in advance.

We calculate the magnitude of each encoded latent vector as measured from the latent mean and filter vectors with magnitudes below the α percentile for calculating inter group similarity as described in Equation 2. In our system, we set α to 10th percentile norm of the training set vectors, as it showed the most robust behavior throughout all experiments. One can also fix the α , e.g., to a specified number of standard deviations, without optimizing on the training set.

The process of how we calculate group anomaly score is further explained in Algorithm 1. Group anomalies are effectively detected when functions f and g respectively capture properties of group distributions and appropriately combine information into a group reference.

Training. Activities are recorded by the agent installed on the endpoint. These activities include process, file and command creation, termination and executions. We use concatenation of feature vectors of the group as the input layer in the AAE specifically, $\boldsymbol{X}_m = \mathbf{concat}(\mathbf{x_1}, \ldots, \mathbf{x_n})$ where $\boldsymbol{x}_i$ is ith feature vector of a group.

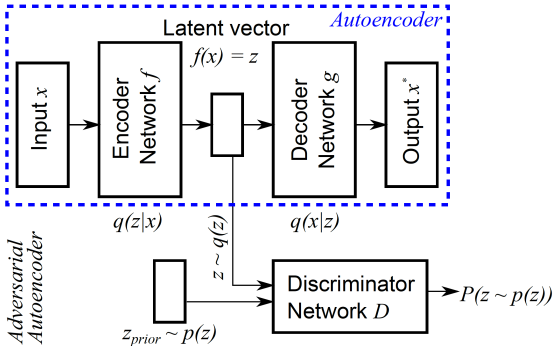


Figure 2. The architecture of Adversarial Autoencoder (AAE -α) for finding group anomalies. AAE -α is trained with objective function that minimizes intra-group variations and inter-group separation.

The AAE-α architecture in Figure 2 is trained according to the loss function given in Equation (3). The objective function is optimized using a standard backpropagation algorithm. Given known group memberships, AAE is fully trained on input groups to obtain a representative group expectation score $\mathcal{G}^{(exp)}$. We tuned via grid search additional hyper-parameters, including the number of hidden-layer nodes $H \in \{120, 60, 320\}$, and regularization λ within range $[0, 100]$. The learning drop out rates and regularization parameter μ were sampled from a uniform distribution in the range $[0.05, 0.1]$. The output scores are sorted according to descending order, where groups that are furthest from $\mathcal{G}^{(exp)}$ are considered anomalous.

IV. Feature Extraction and Pre-processing

In our system, a group is defined by a set of activities observed in a single windows session. Session activity consists of signature names, command line text, session properties, file, and process information. We derive multiple features including density-based feature, session property based features, static, dynamic, reputation and prevalence based features from file and process information. For a given day, We group all raw events by session ID field in the dataset. Next, we extract relevant features from each group. Below we give a brief overview of our feature extraction pipeline.

Command2Vec. An Agent installed on the endpoint matches a set of signatures/rules based on the activity and generates a log. Logs contain the signature name, timestamp, command line text executed by a different process in a given session. At training time, we group all command lines by signature name from the data set. Next, we use command line text to train a sentence2vec model[37] with 200-dimensional sentence embeddings, with window size 20 and by setting negative examples to 10 for each signature. Finally, at feature extraction time for a given session we (a) concatenate all command line text for each signature; (b) use the signature model to extract vectors for the concatenated text; (c) adopt a 3 layer autoencoder on the all the vectors to reduce the dimension.

For example, let us assume in a given session, $Session1$ can be denoted as a group of signatures and command lines.

$$Session1[[s_1, [c_{s1_1}..c_{s1_n}], [s_2, [c_{s2_1}..c_{s2_n}]..[s_m, [c_{sm_1}..c_{sm_n}]]$$

We concatenate command line text $[c_{s1_1}..c_{s1_n}]$ as one blob of text and feed it to already trained signature model to extract feature vectors. Each signature model generates 200 x 1 vector if in a session we have M signatures then the final vector length will be 200 x M. Now, this 200 x M is fed into a 3 layer autoencoder to reduce the dimensionality from 200 x M to 200 x 1.

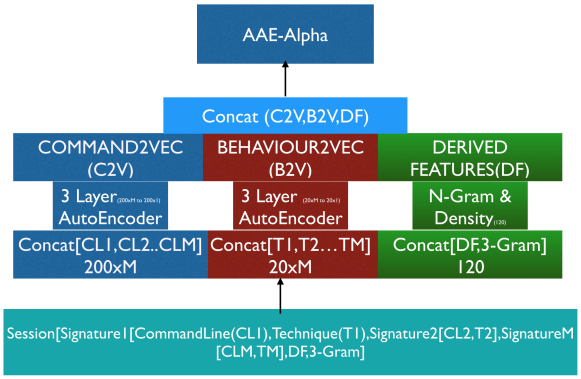


Figure 3. Feature extraction pipeline: (a) For a given session- aggregate all signatures and commandline text and extract embeddings; (b) Run dimensionality reduction on aggregated embeddings to reduce to size 200x1 and 20x1; (c) Feed the resulting embeddings to AAE network as features.

Behavior2Vec. MITRE's ATT&CK framework[39] is a community-curated knowledge base of adversary Tactics, Techniques, and Procedures (TTP) observed in the real world. Each TTP defines one possible way to realize a particular high-level capability. For instance, the capability of persistence in a compromised system can be achieved using 11 distinct TTPs, each of which represents a possible sequence of lower level actions in the ATT&CK framework, e.g., installation of a rootkit, modification of boot scripts, and so on. First, we map the AV signature rules to high-level behaviors manually. Table II provides an overview of mappings between signature rules to high-level techniques from MITRE's ATT&CK framework [39]. We group all logs for a given session ordered by timestamp and collect all the signatures in the session. Next, we create a sequence of MITRE techniques for a given session by looking up from II. We use these sequences to train a sentence2vec model with 20-dimensional sentence embeddings with window size 2 and by setting negative examples to 2. The advantage of high-level mapping is two-fold: (a) we can communicate to the human analyst in an intuitive way instead of just black box anomaly score; (b) we use these mappings to build a sentence2vec model for feature extraction.

For strings such as (a) File, process and directory names and paths; (b) File Signer Information(Subject, Issuer); (c) Process thread names, function names we use tri-gram model for feature extraction. Figure 3 summarizes feature extraction pipeline.

Table II
HIGH-LEVEL BEHAVIORS MAPPED TO A SUBSET OF MITRE TECHNIQUES.

High-level Behaviors	**Technique**
Bits Admin activity	BITSJobs
Command launches by *msiexe, WMI, powershell*, network activity by command line	Command-LineInterface
Memory access of *LSASS, Crypt-Dll*	CredentialDumping and LSASS-Driver
Query for Security tools, Registry changes to trusted process	DisablingSecurityTools
Password dumping and Key logging activity	ExploitationforCredentialAccess
Random folder creation, Proxy changes, Registry and startup folder changes	HiddenFilesandDirectories
WMIC activity	WMIC
WScript runs, Powershell launches other process	Scripting
Build tools, script execution by common tools with network traffic	TrustedDeveloperUtilities
Scheduled tasks added or launched	ScheduledTask
Process injections with network activity	ProcessInjection
Client tools with network activity other then browsers	ExploitationforClientExecution
Suspicious Browser Helper Object modification KeyLogger activity, Suspected screen capture attempted	Collection

Density based Features. These features capture the inherent properties of a session (group), for example, how popular a session is in the enterprise, type of activity in that group - backdoor, attacker, helpdesk, remote admin. For a given session, we extract (a)number of machines on which the session was seen; (b)behaviors which are shared across other sessions; (c)command lines shared across other sessions. A system admin may use same commands across multiple endpoints to update software or apply a patch while using the same user account. Some attackers may also reuse the same session across multiple organizations to conduct their campaigns.

Session Features. Typically, targeted attackers reuse, migrate or create a new user or session for lateral movement and data exfiltration. Session properties like (a) remote vs local; (b) admin vs non-admin; (c) user was idle vs active give insights into the type of user activity in a given session.

Static and Dynamic Features. Static and dynamic features of the files and process executing in a given session help us understand the type of file or process used to create a registry entry and from which place in the file system. Typically, attackers use temporary folders in the file system for malicious file downloads and use schedule tasks and registry entries for persistence. We extract (a)Signer information of the file; (b) Entropy; (c)Import functions and Sections; (d)File Header information; (e)Export functions used; (f)File Paths, and directory; (g) Process names.

Reputation and Prevalence based Features. Popularity and reputation of file or process inside an enterprise and across enterprises help us capturing the toolset of the attackers. Attackers reuse the tools in multiple attack campaigns. Reputation and prevalence of a file change over time, so in our sys-

tem, we use histograms collected over a period of 60 days as one of the features. Histogram features include (a)Prevalence of file, and process across population and inside an enterprise; (b)File signer subject reputation; (c)File reputation. We want to note that this data was provided by Anti-Virus firm for all the file hashes in the dataset.

V. Experiments and Results

In this section, we discuss the results of experiments for the proposed method.

A. Datasets

The dataset consists of behavioral events (e.g., registry value changes, process executed, command lines responsible for the activity, and windows session IDs occurring on endpoints of enterprise customers for two month time period. A large antivirus company that opted in to share their data, such that through large-scale data analysis new methodologies can be developed to increase the existing advanced threat detection capabilities. To protect customer identities, all sensitive information is anonymized. The behavioral events are collected from more than 12000 enterprises with a total of 100k endpoints, which is only a subset of data processed by the antivirus company. Even when this data set is small our results show that it is sufficient to model attacker activity.

The fields we are particularly interested in are: (a) (anonymized) machine and customer identifiers; (b) static features of file or process; (c) reputation and prevalence of the file or process in enterprise; (e) command lines executed by process/file; (f) file name and directory; (g) timestamps (in UTC) of activity; (h) low level behaviors. Table III gives an overview of different statistics of the dataset.

Table III
Statistics of the datasets.

Type	Unique counts
Sessions	5998744
Known Malicious Sessions	926 (0.0154%)
Command lines	3004454
Signatures	992
Endpoints	100233
Enterprises	12000
Features Extracted per session	340

We compare our approach with both point-wise anomaly detectors and group anomaly detectors, which we briefly describe below.

One Class Support Vector Machines (OC-SVM). [25] are a classic kernel method for novelty detection that learns a decision boundary around normal examples. We use the radial basis function kernel in our experiments. The ν parameter is set to the expected anomaly proportion in the dataset, i.e., 0.00015, which is assumed to be known, whereas the γ parameter is set to $1/m$ where m (340) is the number of input features.

Isolation Forests (IF). [26] are a newer classic machine learning technique that isolates anomalies instead of modeling the distribution of normal data. The method proceeds by first building trees using randomly selected split values across randomly chosen features. Then, the anomaly score is defined to be the average path length from a particular sample to the root. We use default parameters provided by the scikit-learn [27] package in our experiments.

Deep Autoencoding Gaussian Mixture Model (DAGMM). [28] is a state-of-the-art autoencoder based method for anomaly detection. The method trains a deep autoencoder and uses its latent representations, together with its reconstruction error, as input to a second network, which is used to predict the membership of each data instance to a Gaussian Mixture Model (GMM). At test time, the likelihood of a sample's latent and reconstruction features as determined using the learned GMM is used as the anomaly detection metric. The estimation network considers a GMM with 4 mixture components for the best performance. For the experiments, compression network with 3 dimensional input to the estimation network, where one is the reduced dimension and the other two are from the reconstruction error, the compression network runs with FC(340,120), FC(120, 60, tanh)-FC(60, 30, tanh)-FC(30, 10, tanh)-FC(10, 1, none)-FC(1, 10, tanh)-FC(10, 30, tanh)-FC(30, 60, tanh)-FC(60, 120, tanh)–FC(120, 340, tanh), and the estimation network performs with FC(3, 10, tanh)-Drop(0.5)-FC(10, 4, softmax).

Mixture of Gaussian Mixture Models (MGMM). [29] is a hierarchical generative approach for detecting group anomalies. The data generating process in MGMM assumes that each group follows a Gaussian mixture, where more than one regular mixture proportion is possible. An anomalous group is defined by an irregular mixture of features. The number of regular group behaviors T=1 and number of Gaussian mixtures L=4 was used for training.

One Class Support Measure Machines (OCSMM). [9] maximize the margin that separates regular class of group behaviors from anomalous groups. Each group is first characterized by a mean embedding function then group representations are separated by a parameterized hyperplane. The kernel bandwidth smoothing parameter in OCSMM [9] is chosen as median$\left\{||\mathbf{G}_{m,i} - \mathbf{G}_{l,j}||^2\right\}$ for all $i, j \in \{1, 2, \ldots, N_m\}$ and $m, l \in 1, 2, \ldots, M$ where $\mathbf{G}_{m,i}$ represents the ith random vector in the mth group.

Deep generative models (VAE and AAE). VAE[38] and AAE [14] are generative variants of autoencoder. VAE utilizes reconstruction probabilities [41] or reconstruction error to compute anomaly scores imposing constraint on hidden latent codes produced by encoder f_ϕ to follow prior data distribution $P(G_m)$. AAE relaxes this constraint by using a GAN as described in Section III. Both AAE and VAE used four layers of (conv-batch-normalization-elu) in the encoder part and four layers of (conv-batch-normalization-elu) in the decoder network. The AAE network parameters such as (number of filters, filter size, strides) are chosen to be (340,3,1) for first and second layers and (120,3,1) for third and fourth layers of

both encoder and decoder layers. The middle hidden layer size is set to be same as rank $K = 60$ and the model is trained using Adam [2]. The decoding layer uses sigmoid function in order to capture the nonlinearity characteristics from latent representations produced by the hidden layer.

The proposed method AAE-α uses similar architecture as standard AAE but instead of drawing samples from the real distribution of data we filter 10th percentile to remove outliers. Figure 4 shows the sensitivity of AUC with percentile value. In our experiments, we used α=10.

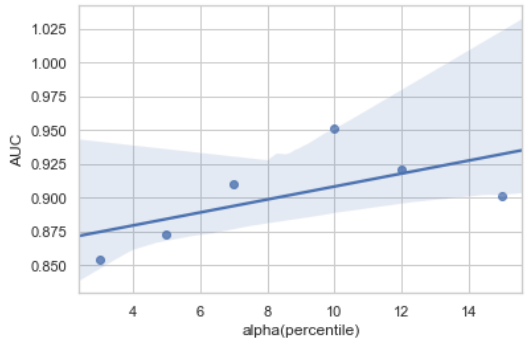


Figure 4. Model AUC sensitivity with α.

The dataset provided to us was not labeled, but a small set of sessions which had one attacker group activity identified by analyst in the past were shared with us. We used these sessions as validation dataset, machine identifiers found in these sessions were also filtered from the dataset to make sure we evaluate the models on completely unseen data.

As anomaly detection is an unsupervised learning problem, model evaluation is highly challenging. The performance of the model is evaluated using the area under the precision-recall curve (AUPRC) and area under the ROC curve (AUROC) metrics. AUPRC metric is more appropriate underclass imbalanced datasets [12]. Table IV summarizes the AUROC and AUPRC values of different methods. Interestingly, we observe that AAE and VAE methods achieve similar results when compared with each other, sampling data filtered by α percentile increases performance for the given dataset.

Table IV
RESULTS FOR DIFFERENT METHODS ON THE DATASET. THE HIGHEST PERFORMANCES ARE IN GRAY.

Method	AUPRC	AUROC
OCSVM	0.6212	0.4508
DAGMM	0.8422	0.5100
IF	0.6977	0.4833
AAE	0.9080	0.5130
VAE	0.9001	0.5007
MGM	0.8584	0.4809
OCSMM	0.8233	0.5097
AAE-α (Ours)	0.9526	0.6022

B. Discussion

New attacks discovered. We filtered all known malicious sessions and corresponding machine identifiers from the dataset. Next, We run the model to assign anomaly scores to all sessions in the dataset and sort the sessions by scores and choose top 0.001% (around 6000 sessions which were seen in 4000 enterprises). We presented top 0.001% anomalous sessions scored by model to threat analysts for review. They observed (a) 1260 sessions with red teaming and, pen testing activity which have similar behavior of an attacker; (b) 330 sessions have the activity of malicious banking trojan which steals sensitive information using off the shelf tools; (c) 40 sessions with 2 new targeted attack groups tool activity; (d) 2833 sessions with malicious executions of tools like PowerShell, WMIC, FTP, and client tools; (e) 620 sessions with security tool scans for checking malicious activity; (f) 417 sessions with benign activity like logging remotely and installing updates.

False Positives. We noticed 417 false alarms detecting security operation tasks and remote sessions as malicious. One of the reasons for higher scores for these sessions is because these sessions were shared across multiple organizations. Since all the data is anonymized, we speculate that this may be the case of a contractor or part-time employee working for multiple organizations or it may be a feature of a security tool. Another set of false positives comes from scanning activity of security tools. These may be reduced by whitelisting rules.

Anomaly Scores interpretation and Alerting. Although research on anomaly detection for cyber defense spans more than two decades, adoption of statistical methods is limited due to two main reasons: (a) high false positive rate; (b) uninterpretable alerts. Analysts are inundated with a large number of alerts and triaging them takes significant time and resources; this results in low tolerance for false alarms and alerts that provide no contextual information to guide the investigation. To address this issue, we map raw event data into a high-level abstraction of MITRE ATT&CK TTPs. This helps the analyst to interpret the score with the technique name. We also key all raw data with corresponding session ID, machine identifier, and timestamp. Sessions with high anomaly scores can be presented to an analyst for further review.

VI. CONCLUSIONS AND FUTURE WORK

We developed a group anomaly detection model using adversarial autoencoders to detect targeted attackers who hide their activity using dual-use tools and system installed applications. We build two deep learning models one for feature extraction and one for generating anomaly score. We map low-level behaviors to high-level MITRE ATT&CK classification, which helps human analysts to interpret anomaly scores.

We tested our method on the real world dataset collected for two month time period from 12k enterprises. We discovered around 5000 instances of malicious activity – 40 sessions with custom tool activity of 2 targeted attack groups, around 3000 sessions with malicious activity using system installed tools, and 1000 sessions with pen testing and red teaming activity without any labeled data and supervision. The results show

that our method successfully detects threats that hide in plain sight with high precision and low false alarm rates.

Despite some interesting results of our proposed method to discover advanced threats, we want to highlight its inherent limitations and subjects for future work. The visibility of activities occurring on the endpoints is limited by collector configuration and settings. We may have missed some activity due to agent throttling settings and reporting configurations. We plan to address this limitation by augmenting the dataset with other data sources like email data and network data. As part of future work, we also aim to test the model performance on a long time ranges, create a multi-class classifier to identify multiple attack groups and evaluate on audit logs from other operating systems like Mac and Linux.

REFERENCES

[1] M Hutchins, Eric & J Cloppert, Michael & M Amin, Rohan Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Leading Issues in Information Warfare & Security Research. 1.

[2] Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[3] Mohiuddin Ahmed & Abdun Naser Mahmood &Jiankun Hu A Survey of Network Anomaly Detection techniques Journal of Network and Computer Applications 60 (2016) 19–31

[4] Akoglu, Leman and Tong, Hanghang and Koutra, Danai, Graph based anomaly detection and description: a survey Data Mining and Knowledge Discovery, May 01 2015, volume 29

[5] Edward Toth and Sanjay Chawla. 2018. Group Deviation Detection Methods: A Survey. ACM Comput. Surv. 51, 4, Article 77 (July 2018), 38 pages.

[6] Dhamija, Rachna and Tygar, J. D. and Hearst, Marti Why Phishing Works Proceedings of the SIGCHI Conference on Human Factors in Computing SystemsCHI '06

[7] Zhang, Junjie and Seifert, Christian and Stokes, Jack W. and Lee, Wenke ARROW: GenerAting SignatuRes to Detect DRive-By DOWnloads Proceedings of the 20th International Conference on World Wide Web WWW '11

[8] Candid Wueest, Himanshu Anand Living off the land and fileless attack techniques - An ISTR Special Report https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-living-off-the-land-and-fileless-attack-techniques-en.pdf

[9] Muandet, K., Schölkopf, B.: One-class support measure machines for group anomaly detection. Conference on Uncertainty in Artificial Intelligence (2013)

[10] Soleimani, H., Miller, D.J.: Atd: Anomalous topic discovery in high dimensional discrete data. IEEE Transactions on Knowledge and Data Engineering 28(9), 2267–2280 (Sept 2016)

[11] Yu, R., He, X., Liu, Y.: GLAD: Group anomaly detection in social media analysis. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 372–381. KDD '14, ACM, New York, NY, USA (2014)

[12] Davis, J., Goadrich, M.: The relationship between precision-recall and RoC curves. In: International Conference on Machine Learning (ICML) (2006)

[13] Kiran, B., Thomas, D.M., Parakkal, R.: An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. ArXiv e-prints (Jan 2018)

[14] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. arXiv preprint arXiv:1511.05644 (2015)

[15] Raghavendra Chalapathy, Edward Toth, Sanjay Chawla Group Anomaly Detection using Deep Generative Models ArXiv e-prints (April 2018)

[16] Denning, Dorothy E. An Intrusion-Detection Model IEEE Trans. Softw. Eng.February 1987

[17] Du, Min and Li, Feifei and Zheng, Guineng and Srikumar, Vivek DeepLog: Anomaly Detection and Diagnosis from System Logs Through Deep Learning Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security

[18] Brown, Andy and Tuor, Aaron and Hutchinson, Brian and Nichols, Nicole Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection Proceedings of the First Workshop on Machine Learning for Computing Systems MLCS'18 2018

[19] Zhou, Chong and Paffenroth, Randy C. Anomaly Detection with Robust Deep Autoencoders Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD '17

[20] Peter J Huber and Elvezio M. Ronchetti Robust statistics, Willeym 2011

[21] Bo Yang and Xiao Fu and Nicholas D. Sidiropoulos and Mingyi Hong Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering Proceedings of the 34th International Conference on Machine Learning,ICML 2017, Sydney, NSW, Australia, 6-11 August 2017

[22] Xiong, Liang and P'oczos, Barnab'as and Schneider, Jeff Group Anomaly Detection Using Flexible Genre Models Proceedings of the 24th International Conference on Neural Information Processing Systems NIPS'11, 2011

[23] Yu, Rose and Qiu, Huida and Wen, Zhen and Lin, ChingYung and Liu, Yan A Survey on Social Media Anomaly Detection SIGKDD Explor. Newsl ,June 2016

[24] Zhai, Shuangfei and Cheng, Yu and Lu, Weining and Zhang, Zhongfei Deep Structured Energy Based Models for Anomaly Detection Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16

[25] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 1999, pp. 582–588.

[26] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

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

[28] Z. Bo, Q. Song, M. R. Min, W. C. C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," *International Conference on Learning Representations*, 2018.

[29] Xiong, L., Póczos, B., Schneider, J., Connolly, A., VanderPlas, J.: Hierarchical probabilistic models for group anomaly detection. In: AISTATS 2011 (2011)

[30] Steven Noel, Eric Robertson, and Sushil Jajodia. Correlating intrusion events and building attack scenarios through attack graph distances. In ACSAC. IEEE, 2004.

[31] Wei Wang and Thomas E Daniels. A graph based approach toward network forensics analysis. Transactions on Information and System Security (TISSEC), 2008

[32] Guofei Gu, Phillip Porras, Vinod Yegneswaran, and Martin Fong. Bothunter: Detecting malware infection through IDS-driven dialog correlation. In 16th USENIX Security Symposium (USENIX Security 07). USENIX Association, 2007.

[33] Kexin Pei, Zhongshu Gu, Brendan Saltaformaggio, Shiqing Ma, Fei Wang, Zhiwei Zhang, Luo Si, Xiangyu Zhang, and Dongyan Xu. Hercule Attack story reconstruction via community discovery on correlated log graph. In Proceedings of the 32Nd Annual Conference on Computer Security Applications, pages 583–595. ACM, 2016.

[34] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. 2013. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In Proc. International Conference on Dependable Systems and Networks (ACSAC). 199–208.

[35] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H Chin, and Sumayah Alrwais. Detection of early-stage enterprise infection by mining large-scale log data. In Proc. International Conference on Dependable Systems and Networks (DSN). 45–56

[36] Thomas Schlegl and Philipp Seeb"ock and Sebastian M. Waldstein and Ursula Schmidt-Erfurth and Georg Langs Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery Information Processing in Medical Imaging - 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings

[37] Pagliardini, M., Gupta, P., Jaggi, M. Unsupervised learning of sentence embeddings using compositional n-gram features. arXiv preprint arXiv:1703.02507 (2017)

[38] Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes (Ml), 1–14

[39] Adversarial tactics, techniques and common knowledge. https://attack.mitre.org.
[40] Hodge, Victoria and Austin, Jim A Survey of Outlier Detection Methodologies Artif. Intell. Rev. October 2004
[41] An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. SNU Data Mining Center, Tech. Rep. (2015)
[42] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.