# Tracing Privilege Misuse Through Behavioral Anomaly Detection in Geometric Spaces

Patrick Duessel
*Rheinische Friedrich-Wilhelms Universität*
Bonn, Germany
p7duessel@uni-bonn.de

Shoufu Luo
*CUNY Graduate Center*
New York City, United States
slou2@gradcentercuny.org

Ulrich Flegel
*Infineon Technologies AG*
Neubiberg, Germany
ulrich.flegel@udo.edu

Sven Dietrich
*John Jay College of Criminal Justice*
New York City, United States
spock@ieee.org

Michael Meier
*Rheinische Friedrich-Wilhelms Universität*
Bonn, Germany
mm@cs.uni-bonn.de

***Abstract*—Privilege misuse is a common technique used by insiders to ex-filtrate proprietary information or sabotage organizations. Although operating systems provide means to log security-related activities indicators of compromise are often difficult to detect due to the often proprietary nature of logging mechanisms in place - rendering the analysis of log files a daunting task. In this contribution we present a format-agnostic approach to detect privilege misuse based on rule-free user activity models learned over security audit logs typically provided by servers. We investigate language model based feature types (i.e. token grams, temporal token grams and attributed token grams) using One-Class Support Vector Machines (OC-SVM). We conduct experiments on synthetic as well as real-world data collected on Microsoft Windows 2008 servers to investigate the effect of feature types and similarity measures and demonstrate usability of this approach for privilege misuse detection as part of an insider threat detection program.**

***Index Terms*—Machine learning, privilege misuse, log analysis, attributed language models**

## I. Introduction

Theft of proprietary information is a growing concern to many organizations. Victimized organizations suffer data leakage, data destruction or even extortion schemes involving current or former employees primarily motivated by financial gains, espionage, or sabotage. The proliferation of social media, cloud-based storage, mobile applications and personal email accounts paired with inconsistent, or weak preventative and detective controls, open a multitude of attack channels. A recent survey [20] that shows that 34% of data breaches in 2019 involved some form of internal threat actors. Typical scenarios for privilege misuse include abuse of privileges granted to entrusted users, unauthorized use of privileges granted to other users, escalation of privileges, or simply human error. A wide range of solutions can be deployed to mitigate the risk of privilege misuse. The majority of these technologies can be broadly categorized into asset-centric and user-centric solutions. While asset-centric controls focuses on the protection of organizational assets (e.g. information, applications, systems, networks) from unauthorized disclosure, modification or destruction, user-centric safeguards focus on the prevention or detection of unauthorized user activities related to those assets. For example, malicious user activity could involve an user, who performs tasks that are unusual to his or her regular user activities - both from an activity or time perspective. A major benefit of user-centric approaches is the ability to detect suspicious activity on hosts or in the network independent from knowledge of high-value information assets and pre-defined content rules in addition to the advantage of being able to identify unknown malicious behavioral patterns.

Overall objective of this paper is to investigate to what extent "suspicious" user sessions can be detected based on learned user behavior models. More specifically, we will make the following contributions:

- We propose two novel feature types (i.e. temporal token grams and attributed token grams) that allow for encoding of temporal and event attribute information within regular $n$-gram language model features in a unified feature space.
- We measure the performance of the proposed feature types on both synthetic and real-world data in comparison to token gram based features, and discuss suitable model parameters obtained from comprehensive cross validation.

The paper is structured as follows: Related work is presented in Section II which outlines the current approaches towards malicious user behavior detection. The main contribution of this paper is presented in Section III which provides details on our proposed solution. A comprehensive experimental evaluation of the proposed methods on synthetic data as well as real security audit log data is presented in Section IV. Finally, conclusions can be found in Section V.

## II. Related Work

An interesting unsupervised approach is proposed in [21]. Unlike our approach which focuses on modeling user behavior on IT systems, authors introduce a clustering-based method over features extracted from network traffic (e.g. IP addresses, domains, connections, user-agent strings) and meta-data (e.g.

connection spikes) to detect suspicious connection behavior over correlated system logs (e.g. VPN, DHCP, web proxies etc.) and demonstrate that their approach is able to identify malicious events and policy violations not detected by rule-based systems.

In the realm of insider threat detection, numerous anomaly-based approaches are proposed [2], [3], [9], [10], [15]. Grier et al. [10] uses statistical analysis over extracted file meta-data features (e.g. file access timestamps) to detect emerging patterns of file copying. Another approach [3] aims at detecting anomalies using $k$-nearest neighbors over user generated actions. Hidden Markov Models in [15] are used to predict the probability of activity sequences. Chung et al. [4] present a method to detect misuse on relational databases employing working scopes of users (including schemes, tables and attributes), which are trained using frequent item set mining techniques.

A multitude of different feature types are investigated in the realm of anomaly-based insider threat detection [1], [8], [11]. Liu et al. [11] suggest to use $n$-gram and histogram models over operating system calls for insider threat detection, which are generated by higher-level user activities, e.g. db admin activity, email, open office applications, or software development. Anomalies are determined based on the Hamming distance between a syscall record and records in the training set for a given user. The authors also suggest an extension of the syscall $n$-gram model to include parameters in order to improve detection accuracy. Eberle et al. [8] suggest a graph-based approach to uncover structural differences in domains such as email correspondences and business processes. Although specific to the fields of network security Duessel et al. [6], [7] propose an attributed token based comparison of semi-structured byte sequences (e.g. HTTP) by incorporating network protocol analysis into byte-level language modeling. Their approach allows for learning language models (e.g. $n$-grams) over parsed attributes extracted from concrete syntax trees (CST) - a syntactic representation of application-level messages - using One-Class Support Vector Machines. Similarity between sequences is determined by convolution of kernels of matching attributes.

## III. Methodology

In this section we provide a brief introduction on content-based behavioral anomaly detection. This approach focuses on modeling privileged user activity based on the analysis of security audit logs. The following four stages outline the essential building blocks of the approach and will be explained in detail.

1) **Data Extraction**. Security events are extracted from security audit logs, which are usually provided by on-board utilities of the operating system [12], [16]. In this contribution we will refer to the security audit logs provided by Microsoft Server 2008. To carry out behavioral analysis, single-host and cross-host user sessions are generated from security audit event streams. The analysis of cross-host user activity not only provides capabilities to detect nested sessions of privileged users (e.g. admin logs on to a local machine and remotely connects to a different machine using either his own, a shared, or compromised user id) but also detection of lateral movement. Details of the event extraction process can be found in Section III-A.
2) **Feature Extraction**. Upon session labeling, security events are mapped onto a metric space using data representations and features which reflect essential characteristics of each event. Details of the feature extraction process can be found in Section III-B.
3) **Similarity Computation**. The similarity computation between user sessions is crucial in order to carry out behavioral analysis in geometric spaces. With the utilization of vector data representations the similarity of security events can be determined by calculating their pairwise distance in the designated geometric space. Different similarity measures are considered in Section III-C.
4) **Anomaly Detection**. An anomaly detection algorithm learns a model of normality. Based on the model, malicious user sessions can be identified using similarity computation. Details on the anomaly detection process can be found in Section IV-C.

### A. Data Extraction

Operating systems allow for mechanisms to log user activity. The context of user activities is entailed by a user's session. In this subsection we will introduce the underlying user session model.

Let $E = (e_1, ..., e_i, ..., e_m)$ be a list of chronologically ordered events $e_i \in \Sigma$, as typically found in system audit logs. A user session $S = (e_1, ..., e_n) \subseteq E$ can be considered an ordered set of activities represented by logged events $e_i$ executed by a specific user on a specific host, where each event is logged at a point of time $i$ and $1 \leq i \leq n$ and $1 \leq i \leq m$. Each session is bounded by a session logon event and the session logoff event, respectively.

A user session model $G = (\Sigma, Q, \hat{q}, T, F)$ can be defined as a directed graph where $\Sigma$ refers to the universal set of audit log events, $Q$ denotes the set of states of a user session (represented as nodes), with $\hat{q}$ being the initial state, $T : Q \times \Sigma \rightarrow Q$ being the state transition function used to specify permissible state changes based on the observed log events and $F$ being the set of terminal states.

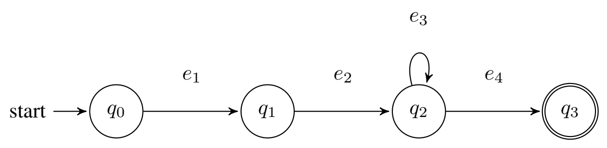


Fig. 1. Example of a user session graph with activity loop

The graph shows a user who logs onto a computer resulting in state $q_0$. After logon, the user carries out a series of activities logged by event $e_i$ which inhibits a state transition to the respective state $q_i$. Finally, the user logs out ($e_4$) resulting in terminal state $q_3$.

With respect to Microsoft's security logging format, a user session in our model consists of a time-ordered list of security events identified by a specific user (i.e. *SubjectUserName* or *TargetUserName*), host (i.e. *ComputerName*) and logging context on the local machine (i.e. *SubjectLogonId* or *TargetLogonId*). New sessions are recognized by a specific security event token (i.e. *4624*). A user session is closed, if the session is regularly ended by another specific security event token (e.g. *4634*).

The example below shows the format of a security event logged on a Microsoft Windows Server 2008.

```
<Event>
 <System>
  <Provider Name="Microsoft−Windows−Security−Auditing" Guid="
      {54849625−5478−4994−A5BA−3E3B0328C30D}" />
  <EventID>4634</EventID>
  <Version>0</Version>
  <Level>0</Level>
  <Task>12545</Task>
  <Opcode>0</Opcode>
  <Keywords>0x8020000000000000</Keywords>
  <TimeCreated SystemTime="2016−05−19T10:32:02.550644500Z" />
  <EventRecordID>1025108</EventRecordID>
  <Correlation />
  <Execution ProcessID="788" ThreadID="3144" />
  <Channel>Security</Channel>
  <Computer>76ffa691</Computer>
  <Security />
 </System>
 <EventData>
  <Data Name="TargetUserSid">8c0c6e34</Data>
  <Data Name="TargetUserName">3dee5118</Data>
  <Data Name="TargetDomainName">300fb30c</Data>
  <Data Name="TargetLogonId">0xf73f4b</Data>
  <Data Name="LogonType">3</Data>
 </EventData>
</Event>
```

Listing 1. Windows user session logon security audit event

Each event consists of two parts. The first part records system configuration information, including version, time, process, etc., while the second part records event-specific information such as user security ID, name, the domain name and event type. The computer name in the $System$ section as well as TargetUserId, TargetUserName, TargetDomainName and TargetLogonId in the $EventData$ section are sanitized for privacy reasons.

## B. Feature Extraction

In order to determine similarity between data points, at this stage user activity is mapped to a high-dimensional feature space using features which reflect essential characteristics of the user activity. In this contribution we focus on content-based features rather than typical meta-data features (e.g. session length, or the number of commands executed during the session).

In order to map a data point $x$ from the domain of user activities into an $N$-dimensional vector space over real numbers, a feature map $\phi : \mathcal{X} \mapsto \mathbb{R}^N$ is used. It defines extraction and mapping of features, where $\phi_i(x) \in \mathbb{R}_{\geq 0}$ refers to the value of the $i$-th dimension in $\mathbb{R}^N$:

$$x \longmapsto \phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_N(x)), \qquad (1)$$

Thereby, the sole choice of the mapping function $\phi(x)$ provides a powerful instrument to transform data into a representation that is suitable for a given problem.The following paragraphs describes different types of features that can be used to model user behavior:

*1) Token Features.:* Let $L \subseteq \Sigma^k$ be a language comprising state sub-sequences $w = (q_i, .., q_j) \in L$ of audit events of length $k$ with $i \geq 0$ and $i < j \leq n - k + 1$. Given the language $L$, a user session $S$ can be mapped to an $|L_k|$-dimensional feature space based on the following embedding function $\phi_w(S)$:

$$\phi_w(S) = \begin{cases} \sigma(w, G_S), & if\ w \in G_S \\ 0, & otherwise, \end{cases} \qquad (2)$$

where $\sigma(w, S)$ refers to a feature embedding function which returns either a binary value, a count or a relative frequency of $w$ observed in $S$. We will evaluate all three embedding functions during cross-validation.

*2) Temporal Token Features:* As an extension of the plain token features, time stamp information from security events can be incorporated into the user session model. Let $S = (e_1, ..., e_j, .., e_m)$ be a non-empty list of chronologically ordered events where each event $e_j$ is associated with a time stamp $u_j$. Furthermore, let $T = (t_1, t_2, ..., t_i, ..., t_r | t_{i-1} < t_i)$ be an ordered set of $r$ pre-defined time intervals, the following embedding function $\phi_{w_i}(S)$ can be defined for each token gram $w = (e_j, ..., e_{j+k-1} \mid 1 \leq j \leq m - k + 1)$ of event tokens contained in user session $S$ :

$$\phi_{w_i}(S) = \begin{cases} \sigma(w, G_S), & if w \in G_S \ \wedge \ t_{i-1} \leq u_1 \leq t_i \\ 0, & otherwise, \end{cases} \qquad (3)$$

where $u_1$ represents the time stamp of the first token of a particular token gram $w$. In our implementation, the function *time_index(T,w)* extracts and returns the index of the time stamp field of the first event of each extracted token gram $w$ according to the list of pre-defined time intervals $T$ (e.g. weekdays, hours of day etc.). Although this feature embedding also maps user activity to an $L_k$-dimensional feature space, two matching sub-sequences have non-zero values if and only if the start time of the execution of activities described by $w$ falls into the same pre-defined time interval.

This feature type can be particularly useful to model user activity based on the time of execution (e.g. regular administrative activities or service account activities).

*3) Attributed Token Features:* Attributed token features are another extension of activity features. Some events may have attributes (e.g. command parameters) which provide additional discriminating information. Let $S = (e_1, e_j, .., e_m)$ be a non-empty list of chronologically ordered events where each event $e_j = \{a_i, v_i\}^*$ is associated with a set of attribute-value pairs (e.g. argument strings of invoked commands). A feature mapping function $\phi_w(S)$ can be defined to map a sequence of events $w$ to an $n$-dimensional metric space $F \in \mathbb{R}^n$:

$$\phi_w(S) = \begin{cases} \sigma(w, G_S), & if\ w \in G_S \\ 0, & otherwise. \end{cases} \quad (4)$$

Unlike the extraction of user activity tokens, each feature is indexed by not only the event type but also by its attribute values. Thus, identical events (e.g. 4624) with different attribute-value sets are mapped to different regions in $F$. This feature set is most expressive, but also most expensive from a computational point of view. Attributed features have been introduced for the area of network intrusion detection [6], [7].

## C. Similarity Computation

The similarity computation between sequences is a crucial task for behavior-based anomaly detection. With the utilization of vectorial data representations, user activities can be compared by calculating their pairwise distance in a designated geometric space. Once a user session is mapped into a feature space $F$, a distance function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ can be applied to determine pairwise similarity between user sessions $\{S_1, ..., S_n\} \subset \mathcal{S}$.

There are several distance and kernel functions available to measure similarity between data points in vector spaces [17]. For the purpose of our contribution, we use the linear kernel and the Radial Basis Function (RBF) kernel.

The linear kernel is one of the most intuitive similarity measures and is defined by a dot product between two vectors $x$ and $y$:

$$\begin{aligned} k(x,y) &= \langle \phi(x), \phi(y) \rangle \\ &= \sum_{i=1}^{n} \phi_i(x)\phi_i(y). \end{aligned} \quad (5)$$

As opposed to linear kernels, non-linear kernels allow for an implicit non-linear mapping of data points in the feature space. A more complex similarity measure is provided by the Radial Basis Function Kernel (RBF) which is defined as follows:

$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right), \quad (6)$$

where $\sigma$ controls the width of the Gaussian distribution and directly affects the shape of the learner's decision surface. A large $\sigma$ results in a linear decision surface which indicates a linearly separable problem, while a small value of $\sigma$ generates a peaky surface which strongly adapts to the distribution of the data in the feature space.

## D. Anomaly Detection

For the purpose of this contribution, we will use the OC-SVM [18] as a means of geometric outlier detection for the identification of malicious user behavior. Based on known literature [13], [18] we provide a short introduction to One-class Support Vector Machines in this section as we will discuss data models in Section. IV as a result of model selection. The OC-SVM fits a minimal enclosing hyper-sphere to the data which is characterized by a center $\theta$ and a radius $R$. Mathematically, this can be formulated as a quadratic programming optimization problem:

$$\begin{aligned} \min_{\substack{R \in \mathbb{R} \\ \xi \in \mathbb{R}^n}} \quad & R^2 + C\sum_{i=1}^{n} \xi_i \\ \text{subject to:} \quad & ||\phi(x_i) - \theta||^2 \leq R^2 + \xi_i, \\ & \xi_i \geq 0. \end{aligned} \quad (7)$$

Given the constraint that the training objects are still contained in the sphere, expressed by the constraint in Eq.(7), minimizing $R^2$ will minimize the volume of the hypersphere.

A major benefit of this approach is the ability to control the generalization ability of the algorithm [14], which enables one to cope with noise in the training data and thus dispense with laborious sanitization, as proposed by Cretu et al. [5]. By introducing slack variables, such as $\xi_i$ and penalizing the cost function, we allow the constraint to be softened. The regularization parameter, $C = \frac{1}{N\nu}$, controls the trade-off between radius and errors (the number of training points that violate the constraint), where $\nu$ can be interpreted as the permissible fraction of outliers in the training data.

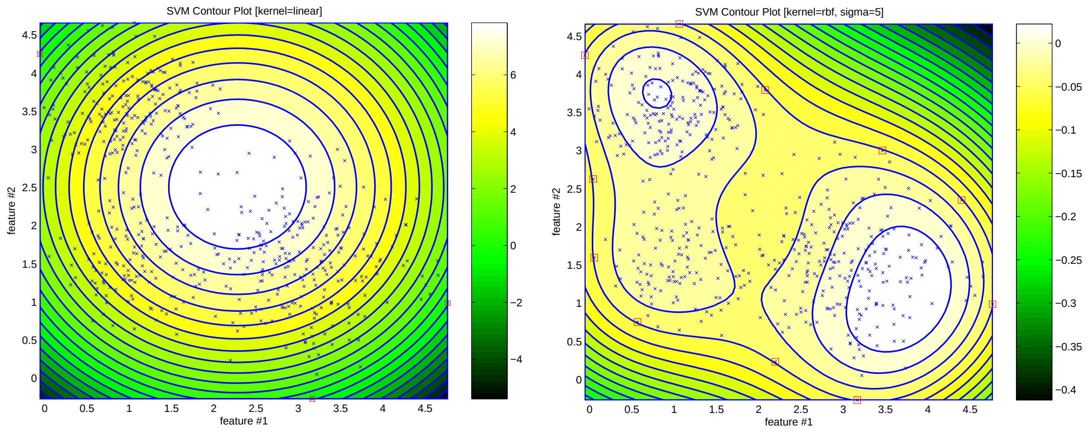
Having learned a model of normality learned, the anomaly score $S_z$ for an unknown data point $z$ can be defined as the distance from the center $\theta = \sum_i \alpha_i \phi(x_i)$ defined by support vectors in the feature space:

$$\begin{aligned} S_z &= ||\phi(z) - \theta||^2 \\ &= k(z,z) - 2\sum_i \alpha_i k(z, x_i) + \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j), \end{aligned} \quad (8)$$

where the similarity measure, $k(x, y)$, between two points $x$ and $y$, defines a kernel function as introduced in Section III-C. Depending on the similarity measure at hand, data models of different complexity can be learned. For example, as shown in Fig. 2(a) utilization of a linear kernel always results in an uniform hyper-sphere. Thus, the resulting model provides a rather general description of the data. However, if the data happens to follow a multi-modal distribution, the risk of absorbing outliers in low density regions of the hyper-sphere might increase. On the contrary, utilizing an RBF-kernel supports the distribution characteristics of the data, and therefore results in more complex data models, as shown in Fig. 2(b). The downside of these kinds of measures is their lack of interpretability, considering the data points are mapped onto an infinite dimension feature space.

## IV. Experiments

In this section we present the results of the evaluation of the proposed method. Experiments are conducted on two data sets. Preliminary experiments are performed on synthetic user session data. Finally the feature types are evaluated on real security audit log data. Characteristics of both data sets are outlined below.



(a) Linear Kernel

(b) RBF Kernel

Fig. 2. Learning and anomaly detection using OC-SVM - Example data models obtained over synthetic data (red squares represent support vectors constituting the data model by defining the decision surface for separation between normal data and outliers.))

### A. Data Set

In this section we will describe data sets used for experimental evaluation. Apart from conducting experiments on real-world security audit log data we have created a synthetic data set in order to evaluate the detection performance of our method for corner cases, i.e. detection of temporal anomalies and detection of semantic anomalies in user activity using the proposed novel feature types under the assumption that an attacker is able to mimic normal user behavior (i.e. masquerading).

*1) Synthetic Data Set:* Two synthetic data sets are created to investigate the performance and study the strengths and limitations of both temporal token and attributed token gram features.

*a) Temporal Data Set:* To this end, a base set of random event sequences is generated with a length in the range of $[2,\ 20]$. Each sequence consists of a series of events which are randomly drawn from a fixed set of 20 unique event identifiers. For the normal data set, sequence instances are randomly drawn from the base set of event sequences. In order to simulate the variance of user behavior, a fixed percentage of events in each sequence is permuted. The permutation factor is in the range of $[0,\ 0.5]$ with *0* being no permutation and *0.5* resulting in full permutation of the events recorded during a session. In order to show the benefits of the temporal feature type, each event is timestamped to *1200PM UTC*. For the attack partition, a percentage of event sequences are randomly drawn from the normal data set. In order to simulate the variance of the attacker's behavior, a fixed percentage of events in each sequence is permuted. Furthermore, permutation factors for normal and "attack" data sets are independent. Attack instances are timestamped to *1200PM+4h UTC*.

*b) Attribute Data Set:* Similarly to the temporal data set, a base set of random event sequences is generated with a length in the range of $[2,\ 20]$. Each sequence consists of a series of events which are randomly drawn from a fixed set of 20 unique event identifiers. For the normal data set, sequence instances are randomly drawn from the base set of event sequences. In order to simulate a variance of user behavior, a fixed percentage of events in each sequence is permuted. The permutation factor is in the range of $[0,\ 0.5]$. In order to show the benefits of the attributed feature type, each event is associated with a set of attribute-values. The number of attribute-values for each event is random, but limited to a maximum number of 5 in our experiments. For the attack partition, a percentage of event sequences are randomly drawn from the normal data set. In order to simulate the variance in the attacker's behavior, a fixed percentage of events in each sequence is permuted. The permutation factor is independent from the permutation factor used to modify normal sequence instances. Furthermore, an additional attribute mutation factor is introduced for the attack dataset which allows to replace either individual attributes or associated values for each event by distinctive random numbers, not seen before. The mutation factor is in the range of $[0,\ 1]$ where $0$ refers to no replacement of attributes or values at all and $1$ refers to full replacement of existing attribute or value identifiers. The mutation factor should simulate variation of properties of identical events.

*2) Real-world Data Set:* The Los Alamos National Laboratory (LANL) data set [19] is a well known among security practitioners for the analysis of host security events. However, limitations of this data set (e.g. anonymized time stamps and limited logging of event details) required us to record a separate, more detailed data set to support our experiments. The data set consists of 107,892 security audit events recorded on a Windows Server 2008 (Terminal Server) over a period of two months (May 18, 2016 - July 5, 2016). The server is located in a production environment of a company with more than 5.000 employees [1].The audit log data set contains 11,606

[1]Due to confidentiality constraints the name of the organization cannot be disclosed.

sanitized user sessions executed by 178 distinct users. Table I shows descriptive session statistics for two different users in the baseline data set. Criteria for the selection of users include the number of sessions available as well as average, minimum and maximum number of events across sessions.

| Id | User Id | #Sessions | Session Length | | |
|---|---|---|---|---|---|
| | | | $N_{min}$ | $N_{max}$ | $N_{avg}$ |
| 0 | 897ee68c4ce0900 | 5230 | 1 | 72 | 2 |
| 1 | 14a7a913da97fe3 | 531 | 1 | 990 | 38 |

TABLE I
SESSION STATISTICS FROM REAL-WORLD DATASET

Examples of user sessions for two users are presented in Table II.

| User | Sample User Sessions |
|---|---|
| 14a7a913 | (4624,4634), (4624 ,4647)<br>(4624,4690,4658,4656,4658,4690,4658,4656,4658,4690, 4658,4656,4658,4647)<br>(46254,4689,4689,4689,4689,4689,4689,4689,4689,4689,4689,4689,4689) |
| 897ee68c | (4624,4634), (4624,4647), (4624,4656)<br>(4624,4688,4888,4888) |

TABLE II
EXAMPLES OF USER SESSIONS PER USER

In order to investigate the detection of privilege misuse, 45 sessions from 15 pre-defined privilege misuse use cases were executed and recorded in a separate Windows 7 test environment. The logging environment was consistently configured before recording (i.e. identical local security policy settings) [2]. The use cases define key activities such that the execution of individual user sessions may vary between users. The specific activities that are preformed and the tools that are used, during the execution of individual use cases, are documented. Time stamps, users and hosts were adjusted to match the characteristics of the baseline data set before merging them with the baseline log data. Although some of the use cases (e.g. case #9-15) might also be detectable by policy monitoring or network traffic monitoring, objective in our experiments is to evaluate to what extent use case specific activity patterns are reflected by the Windows security logging and to what extent these patterns deviate from a learned normal user behavior model. A detailed description of privilege misuse use cases can be found in Table VI.

## B. Experiment Setup

Detection accuracy is measured in terms of the receiver operating curve (*ROC*) which integrates true positive value over a defined false positive range between zero and one percent (i.e.[0, 0.01]). In order to determine the optimal data model parameters a 10-fold cross validation is conducted. Training, validation and test data sets are distinct in order to avoid over-fitting effects. Privilege misuse sessions are specifically excluded from training and validation partitions during cross-validation and only included in distinct test partitions to enable unknown threat detection. Models are trained on the training partition using different model parameters and subsequently evaluated against the validation partition. Finally the optimal model is tested on a distinct test partition in order to determine the test error. For statistical reason experiments are repeated 10 times. False positives, false negatives and area-under-ROC (*AUC*) is averaged over experiment repetitions. Parameters and ranges evaluated during experiments are depicted in Table III.

[2]Logging parameters are configured to log successful audit events related to account logon, account log management, log events, object access, policy change, privilege use, process tracking and system events

| Parameter | Range | Description |
|---|---|---|
| kernel | 'linear', 'rbf' | Refers to choice of kernel function to be used bty the Support Vector Machine which creates a linear or non-linear decision surface. |
| ftype | 'token', 'attr-token', 'temp-token' | Refers to type of features to extract from user sessions (token: plain event ids, attr-token: event ids with parameters (e.g. command arguments), temp-token: event ids associated with timestamps. |
| embed | 'freq', 'binary', 'count' | Refers to embedding of extracted features in feature space (binary: 0/1 embedding, count: number of occurrences of a specific feature observed in data instance, freq: relative frequency of a feature specific feature observed in a data instance.) |
| klen | 1, 2, 3, 4 | Refers to length of extracted gram features. |
| nu | 0.001, 0.01, 0.1, 1 | *Nu* - Refers to regularization parameter used by the Support Vector Machine to learn a data model. Lower regularization increases the margin at the decision surface and prevents over-fitting by allowing a certain amount of constraint violation during optimization. |
| gamma | 0.01, 0.1, 0.5, 0.75, 1.0 | Refers to width of a non-linear Radial Basis Function (RBF) kernel and affects the shape of the decision surface. Gamma is important for linear non-separable data samples. A higher gamma value will result a more "flat" decision surface. Low gamma values are preferable for data samples which are not easily separable by a linear kernel function. |

TABLE III
MODEL PARAMETERS

## C. Experimental Results

In this section we will present the results of the experimental evaluation of anomaly detection over privileged user sessions.

*1) Experiments on synthetic data:* An artificial data set is generated to simulate user sessions and objectively test sensitivity and benefits of *temporal* and *attributed token* features for boundary use cases. However, both feature types are also investigated on realistic data in this section as well. The synthetic data set contains 10% attack sequences. For the experiments, event ids in user sessions are permuted by *10%* for both normal and attack sequence instances to simulate variation in user behavior during user sessions. The data Set also contains 10% attack sequences. An OC-SVM is used to train on 750 "normal" instances. Optimal parameters are determined following a *10*-fold cross validation approach. Unlike classification tasks labels of training data are not taken into account to learn a decision function. In the context of unsupervised learning, data labels are solely used for the purpose of evaluating the detection accuracy.

*a)* ***Temporal Token Anomaly Detection:*** In order to evaluate temporal behavior anomaly detection a synthetic dataset is created as described in Section IV-A. As shown in Fig. 3 temporal token gram features significantly outperform regular token gram features on the data set.

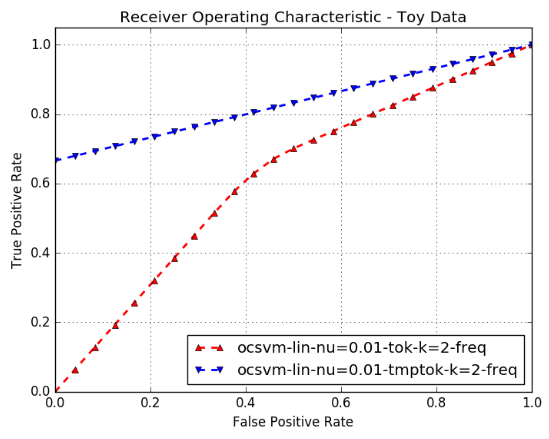


Fig. 3. Detection accuracy - temporal token features

The reason for the improved detection accuracy resides in the temporal annotation of individual events. Fig. 4 shows a comparison of regular token and temporal token features.

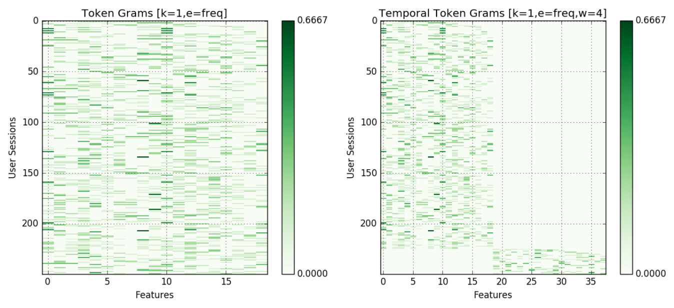


Fig. 4. 1-gram feature map - token grams (left) vs. temporal token grams (right)

The last 20 rows in the matrix represent synthetic attack instances (where similar activity is executed during different times of the day). As shown, it is almost impossible to differentiate normal from attack instances using regular token grams extracted from event id sequences (left). However, through incorporation of temporal information attacks can be perfectly separated from normal data as shown in the right figure.

*b)* ***Attribute Token Anomaly Detection:*** In this section we investigate the benefits *attributed token gram features* of model user activity based not only on the event type, but also considering the event properties. To this end we leverage a synthetic data set described in Section IV-A to test boundary use cases. As shown in Fig. IV-C1b attributed token gram features outperform regular token gram features on the data set due to the encoding of meta-data features in the extracted features.

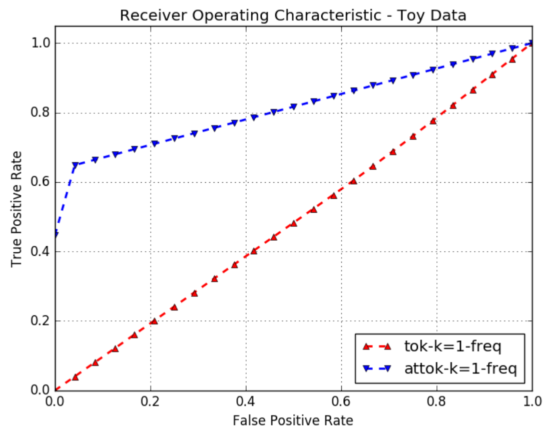


Fig. 5. Detection accuracy - attributed token features

*2) Experiments on real-world data:* In this section we will investigate the performance of proposed feature types on real data. To this end, we extract user sessions of a non-technical privileged user ("14a7a913") as well as a technical privileged user ("897ee68c"), in order to evaluate detection accuracy for different user types.

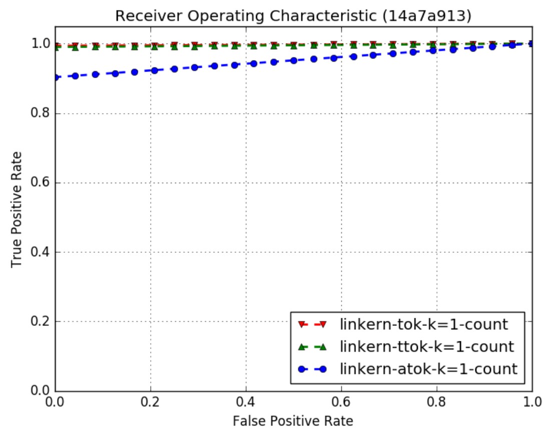


Fig. 6. Detection accuracy - User "14a7a913' (non-technical user)

As shown in Fig 6 and Fig. 7 all feature types demonstrate similar detection capabilities. The results on real-world data show that regular token grams seem to be the preferred feature type as results suggest in Table IV. This is mainly due to the

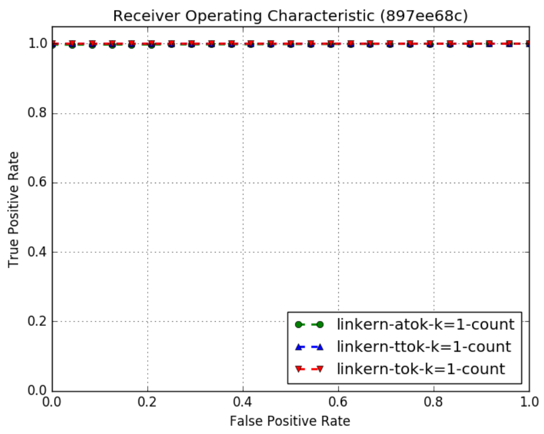


Fig. 7. Detection accuracy - User ”897ee68c” (technical user)

fact that the real user sessions are not as difficult to delineate as user sessions used during the preliminary experiments on synthetic data. Based on investigated misuse cases and event types, the sequence of events in misuse sessions significantly differ from the normal user sessions. Thus, in the context of this data set, token gram features seem to be sufficient to detect anomalous behavior of both technical and non-technical users.

| User | Feature | Parameters | AUC | Freq. |
|---|---|---|---|---|
| 897ee68c | token | k=linear, nu=0.001, n=1,e=cnt | 1.0 | 100% |
| | temporal | k=linear, nu=0.001, n=1,e=cnt, w=24 | 1.0 | 100% |
| | attributes | k=linear, nu=0.001, klen=1,e=cnt | 1.0 | 100% |
| 14a7a913 | token | k=linear, nu=0.001, n=1,e=cnt | 1.0 | 75% |
| | | k=linear, nu=0.001, n=3,e=freq | 1.0 | 20% |
| | | k=linear, nu=0.001, n=2,e=cnt | 1.0 | 5% |
| | temporal | k=linear, nu=0.001, n=1,e=cnt,w=24 | 1.0 | 80% |
| | | k=linear, nu=0.001, n=3,e=freq ,w=24 | 1.0 | 15% |
| | | k=linear, nu=0.001, n=2,e=cnt ,w=24 | 1.0 | 5% |
| | attributes | k=linear, nu=0.001, klen=1,e=cnt,max_attr=5 | 0.95 | 60% |
| | | k=linear, nu=0.1, klen=1,e=cnt,max_attr=5 | 0.84 | 30% |
| | | k=rbf, nu=0.01, klen=1,e=cnt,max_attr=5 | 0.53 | 10% |

TABLE IV
AVERAGE ACCURACY (AUC) ON THE TEST DATA OF THE MOST FREQUENT MODELS PER FEATURE TYPE OVER VALIDATION DATA (20 REPETITIONS)

The example below shows the session of detected suspicious user behavior - a session in which a privileged user bypasses existing firewall configurations. The administrator logs in locally and deploys a port forwarding tool (ie. fpipe) which allows to accept inbound traffic at a specific port and forward to another local listening port which could not be accessed from outside due to firewall configuration. To this end, the administrator logs in with dedicated admin account, opens the File Explorer, accesses the USB drive and copies the “fpipe” tool to the local file system. The administrator then configure and start fpipe as persistent service and finally logs out. As shown below, several security events are generated for this user activity sequence. Most of the events are related to process creation (i.e. 4688) and differ from the sample sequence in Table II significantly. The highlighted security events represent the anatomy of the “firewall bypass” use case. The user logs on to the system locally (i.e. 4624), the system triggers successful logon with Local Security Authority (i.e. 4611), “dllhost.exe” process is created (i.e. 4688 - in “ProcessName” attribute) followed by installation and registration of “svchost” with the Service Control Manager (i.e. 4697) and creation of several key processes (i.e. 4688). The user then inserts the USB drive which is detected as an external device (i.e. 6416) triggering creation of several additional processes (i.e. 4688) and execution of “fpipe” tool (i.e. 4697) which results in the creation and installation of a corresponding process handling and re-directing inbound network packets.

| |
|---|
| **4624**, 4624, **4611**, 4688, **4697**, 4697, 4697, 4697, 4697, 4697, 4688, 4688, 4688, 4688, 4688,4688, 4688, 4688, 4688, 4688, 4688, **4611**, 4611, 4688, 4688, 4688, 4688) |
| (4624, 4688, 4688, 4688, 4688, **6416**, 6416, 6416, 4688, 6416, 4688, 4688, 4688, 4688, 4688, 4688, 4688, 4688, 4688, **4611**, 4688, 4611, 4688, 4688, 4688, 4688, **4697**, 4688, 4688, 4688) |
| (4624, 4688, 4888, **4847**) |

TABLE V
EXAMPLE OF DETECTED PRIVILEGE MISUSE USER SESSIONS FROM REAL-WORLD DATA SET (FIREWALL BYPASS)

As shown in the example above the meaning of logged security events cannot be determined easily. Specific details are logged as part of security event attributes (i.e. EventData) which provides additional context for user activity triggered events. An example of an‘fpipe” execution (i.e. 4697) log entry is depicted below.

```
    <Event>
     <System>
       <EventID>4697</EventID>
        ...
    </System>
    <EventData>
      <Data Name=”SubjectUserSid”>S
            −1−5−21−1505976851−2394108727−4248911256−1002</
            Data>
      <Data Name=”SubjectUserName”>admin1</Data>
      <Data Name=”SubjectDomainName”>DESKTOP−1BMDPVD</Data
            >
      <Data Name=”SubjectLogonId”>0x2f1c5d</Data>
      <Data Name=”ServiceName”>forwardMe</Data>
      <Data Name=”ServiceFileName”>”C:\Users\root\Downloads\fpipe\
            FPipe.exe” −−l=”8080” −−s=”8080” −−r=”80 10.x.x.x”</
            Data>
      <Data Name=”ServiceType”>0x10</Data>
      <Data Name=”ServiceStartType”>2</Data>
      <Data Name=”ServiceAccount”>LocalSystem</Data>
  </EventData>
</Event>
```

Listing 2. “New service” event with “Fpipe” specific event attributes

An analysis of false negatives revealed that instances of attack class#1 (i.e. ‘shutdown”) were most frequently missed during testing. Involved session instances are comparably short and do not differ significantly from the sessions that are considered normal. For instance, the session $S = (4624, 4624, 4624, 1100, 4647)$ is an example of an undetected session. Except for the event logging service shutdown event (i.e. 1100) itself, no discriminating activity is logged. Overall, the false positive rate involving experiments over real-world data does not exceed 1.5%.

## V. Conclusion

In this contribution we have proposed a method to detect privilege misuse based on the automated content-based analysis of security audit logs using unsupervised machine learning. To this end, we have defined three feature extraction methods (i.e. *token*, *temporal* token as well as *attributed* token grams) that can be used to embed semi-structured information contained in security audit log data into geometric spaces for geometric outlier detection. We have conducted experiments on both synthetic data as well as real-world security audit logs to investigate the effect of different types and similarity measures on the detection of privilege misuse. For the purpose of experiments we have defined and recorded a comprehensive set of 45 different privilege misuse sessions. The model selection was carried out through extensive cross validation. The best models were applied on distinct test data partitions. Throughout experiments, privilege misuse sessions were distinctively included in both validation as well as test data partitions whereas training data partitions did not include any privilege misuse sessions.

Experimental results confirmed suitability of unsupervised machine learning using language models extracted from security audit log data. Our results on real-world data suggest that all feature types show comparable performance based on models trained for individual users while attaining low false positive rates on real data of less than 1.5%. Based on an average of 232 sessions per days observed in the real-world data set, the number of false positives per days is therefore lower than four. Overall, low $n$-gram length and count embedding with low regularization chosen during cross validation indicates a low complex hyper-sphere enclosing normal data. While results on real-world data indicate comparable performance of all feature types, experiments on synthetic data - designed to test corner use cases - showed that there are scenarios where both *temporal* and *attributed* token grams significantly outperform plain token grams, such as in advanced scenarios where attackers may mimic normal user behavior in an attempt to bypass detection. Novel feature types boosted detection accuracy from 0 to 63% for *temporal* token grams and 0 to 40% at 0% false positives for *attributed* token grams. The reason for of both feature types outperforming plain token features resides in the incorporation of additional content and session meta-data information in tokens, thereby reducing the amount of ambiguity that could lead to false positives or false negatives.

## References

[1] A. Azaria, A. Richardson, S. Kraus, and V. S. Subrahmanian, "Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 2, pp. 135–155, June 2014.

[2] Balabit, "Shell Control Box - Privileged User Monitoring," Internet: https://www.balabit.com/network-security/scb, 2016.

[3] B. Böse, B. Avasarala, S. Tirthapura, Y. Y. Chung, and D. Steiner, "Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–12, 2017.

[4] C. Chung, M. Gertz, and K. Levitt, "Demids: A misuse detection system for database systems," in *Integrity and Internal Control in Information Systems*. Springer, 2000, pp. 159–178.

[5] G. Cretu, A. Stavrou, M. Locasto, S. Stolfo, and A. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, 2008.

[6] P. Düssel, C. Gehl, U. Flegel, S. Dietrich, and M. Meier, "Detecting zero-day attacks using context-aware anomaly detection at application-layer," in *International Journal of Information Security*, 2016, pp. 1–16.

[7] P. Düssel, C. Gehl, P. Laskov, and K. Rieck, "Incorporation of application layer protocol syntax into anomaly detection," in *International Conference on Information Systems Security*, 2008, pp. 188–202.

[8] W. Eberle, J. Graves, and L. Holder, "Insider threat detection using a graph-based approach," *Journal of Applied Security Research*, vol. 6, no. 1, pp. 32–81, 2010.

[9] E.Yuan and S. Malek, "Mining software component interactions to detect security threats at the architectural level," in *2016 13th Working IEEE/IFIP Conference on Software Architecture*, Apr. 2016, pp. 211–220.

[10] J. Grier, "Detecting data theft using stochastic forensics," *Digit. Investig.*, vol. 8, pp. S71–S77, Aug. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2011.05.009

[11] A. Liu, C. Martin, T. Hetherington, and S. Matzner, "A comparison of system call feature representations for insider threat detection," in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, June 2005, pp. 340–347.

[12] Microsoft, "Security Auditing Overview," Internet: https://technet.microsoft.com/en-us/library/dn319078%28v=ws.11%29.aspx, 2016.

[13] K.-R. Mueller, S. Mika, G. Raetsch, K. Tsuda, and B. Schoelkopf, "An introduction to kernel-based learning algorithms," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 12, no. 2, pp. 181–201, 2001.

[14] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Neural Networks*, vol. 12, no. 2, pp. 181–201, May 2001.

[15] T. Rashid, I. Agrafiotis, and J. Nurse, "A new take on detecting insider threats: Exploring the use of hidden markov models," in *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, ser. MIST '16. New York, NY, USA: ACM, 2016, pp. 47–56.

[16] Redhat, "Understanding Redhat Audit Log Files," Internet: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security_Guide/sec-Understanding_Audit_Log_Files.html, 2016.

[17] K. Rieck and P. Laskov, "Linear-time computation of similarity measures for sequential data," *Journal of Machine Learning Research*, vol. 9, pp. 23–48, 2008.

[18] D. Tax and R. Duin, "Data domain description by support vectors," in *Proc. ESANN*, M. Verleysen, Ed. Brussels: D. Facto Press, 1999, pp. 251–256.

[19] M. J. M. Turcotte, A. D. Kent, and C. Hash, *Unified Host and Network Data Set*. World Scientific, nov 2018, ch. Chapter 1, pp. 1–22. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9781786345646_001

[20] Verizon, "Verizon Data Breach Investigations Report," Internet: https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf, 2019.

[21] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proceedings of the 29th Annual Computer Security Applications Conference*, ser. ACSAC '13. New York, NY, USA: ACM, 2013, pp. 199–208.

## Appendix

### Privilege Misuse Use Cases

| Id | Use Case Title | Desription | #Sessions |
|---|---|---|---|
| 1 | Assign_Admin_Privileges | Admin assigns admin privileges to non-admin user during approved shutdown session. | 3 |
| 2 | Create_Batch_Job | Admin registers batch file with Windows scheduler during approved shutdown session. | 3 |
| 3 | Disable_Antivirus | Admin disables anti-virus during approved shutdown session. | 3 |
| 4 | Change_Policy | Admin performs one policy change during approved shutdown session. | 3 |
| 5 | Change_Multi_Policy | Admin performs five policy changes during approved shutdown session. | 3 |
| 6 | Change_Firewall | Admin performs one firewall change (e.g. disable firewall, add/change/delete rule) during approved restart session. | 3 |
| 7 | Data_Leakage_SAM | Admin copies local SAM file for offline password cracking to local USB drive. | 3 |
| 8 | Data_Leakage_Sensitive_File | Admin copies sensitive file from network drive to local USB drive. | 3 |
| 9 | Software_Install_Fpipe | Admin installs *fpipe* tool to accept and forward inbound traffic at specific port to another local port in order to bypass firewall ingress configuration. | 3 |
| 10 | Software_Install_Malware | Admin installs keylogger to record credentials of all users on a machine. Anti-malware is assumed to be disabled. | 3 |
| 11 | Delete_Audit_Log | Admin deletes audit log file using *wevtutil* utility tool | 3 |
| 12 | Network_Scanning | Admin performs network scanning using *nmap* tool. | 3 |
| 13 | Embedded_Admin_Session | Admin opens console and logs in as another admin (out of the three most active users in baseline data set) to perform some activities. | 3 |
| 14 | Network_Data_Exfiltration | Admin exfiltrates a file using on-board *TFTP*, *FTP* or *netcat* tool | 3 |
| 15 | Time_Stomping | Admin alters time stamps of file (i.e. modify,access,change) using anti-forensics tools to obfuscate tracks. | 3 |

TABLE VI
PRIVILEGE MISUSE USER SESSIONS