Simulated User Bots: Real Time Testing of Insider Threat Detection Systems

No Author Given

No Institute Given

Abstract. The insider threat is one of the most serious security problems faced by modern organizations. High profile cases demonstrate the serious consequences of successful attacks. The problem has been studied for many years leading to a number of technologies and products that have been widely deployed in government and commercial enterprises. A fundamental question is how well do these systems work? How may they be tested and how computationally expensive a widely deployed monitoring infrastructure cost? Measuring real systems that are dynamic in nature, encounter unknown configuration bugs and have sensitivities to the vagaries of human nature and adversarial behavior require a formal means to continuously test and evaluate deployed detection systems. We present a framework to deploy *in situ* simulated user bots (SUBs) that can emulate the actions of real users. By creating a user account and running a host in the enterprise network, a SUB can be introduced into an enterprise system that runs at a realistic pace and does not interfere with normal operations. Infusing malicious behavior into the SUB should be detected by the insider threat monitoring infrastructure. The SUB framework can be controlled to explore the limits of deployed systems and to test the effectiveness of insider evasion tactics, especially low and slow behaviors. We demonstrate our framework by generating user data to test the detection of malicious users and our ability to produce variable ground truths through intrusion detection testing using several commonly used machine learning techniques.

Keywords: Metrics, Simulated User Bots, Intrusion Detection Algorithms, Measuring Security

1 Introduction

The insider threat is among the most severe security threats in modern organizations. A number of technologies and products are being deployed broadly [1–3]. The User Behavior Analytics (UBA) market is predicted to approach $1 billion by 2021 [9]. The current generation of available commercial products are based upon network monitoring sensors and large scale data analytics that compute user models to detect abnormal behaviors indicative of malicious acts (eg., [19, 22]). A fundamental question is how well do they actually work and at what computational cost?

In this paper, we introduce simulated user bots (SUBs), a novel system designed to create realistic user behaviors in an enterprise computing environment that can be programmatically designed to simulate normal or malicious users. Similar to BotSwindler [16], we employ an endpoint agent to run host applications to mimic a real user without interfering with the normal operations on the enterprise system. The SUBs generate network and host data in the same manner as ordinary users, but can be controlled to purposely inject any behavior we wish. There are no traces that would clearly indicate that the SUBs are not real users in the system logs. To generate simulations, SUBs are dependent on a database, a corresponding translator and actions files. The SUBs are identical to users in almost every facet from having to login to the speed at which a user edits a document.

For the purposes of training our models, we used a detailed dataset consisting of 63 West Point cadets, containing up to two months of real usage data. Despite being from a homogeneous organization, users were unique in their behaviors. Statistics were gathered on the users in terms of the average frequency of visits to particular types of websites, the types of activities on files, etc.

The SUB behaviors introduced in this paper are designed to generate trace data typically used as indicators and detectors in identifying malicious users by insider threat systems. We enumerate a sample of typical indicators used in deployed systems in Section 3.1. These indicators are temporal statistics derived from an analysis of monitored network logs. These statistics derive group norms from which abnormal users are identified. A clever insider adversary should be able to easily avoid detection. Controlling the pace and frequency of these trace indicators generated by a SUB provides the means of testing the detection system at its margins, and hence can provide a detailed analysis of the ease or evading detection. Errors in the deployed monitoring infrastructure, either due to bugs in configurations or noise introduced by faulty sensors, may also be revealed if a SUB is undetected although it is directed to purposely exhibit the indicator.

Of particular interest is whether SUBs may be used to measure the computational costs of maintaining long term temporal statistics. Low and slow behaviors would cause an insider threat monitoring system to keep long term state information for many users, which would cause an increasing cost in terms of storage and computation. Hence, the SUB testing framework may also provide a means of evaluating not only the accuracy of a system, but its computational costs.

We have implemented our SUB framework using QEMU [13], running a Linux host. The SUB user actions are learned from a database of real user behavior previously acquired in prior work, that are translated to Python action files, which make use of vncdotool [4]. The SUB framework can be applied to any host operating system, but our work presently focuses on Linux. We present tests off of these action files for the SUBs to determine whether they pass or fail various commonly used algorithms for insider threat detection such as gaussian mixture models, support vector machines and bayesian networks. The results from our experiments validate our ability to modify the behavior of our SUBs. The ability to programmatically create dynamic users demonstrates the ease of

deployment of SUBs in real systems at little to no cost other than creating SUB user accounts within an enterprise system.

The remainder of the paper will be organized as follows. In Section 2, we describe the design of the simulated user bot development framework. We present both our experiments and results in Section 3. In Section 4, we detailed related work. The final two parts, Sections 5 and 6, are dedicated to future work and the conclusions of our work.

2 Design and Architecture of SUB Framework

SUBs are created to replicate a real user with no hint of artificiality from the view of the system and its anomaly detection systems. To create users in this manner builds on both tools previously used for other purposes and those specially built for SUBs. The data generated from the SUB process is nearly identical to a real user in a system and can be fine-tuned to have either normal or malicious tendencies. The user behaviors are adapted from an existing database of users, but user behavior itself could be further modeled. The framework is easily modifiable and the results are not random as each user is reproducible given a specified set of actions and controls. The framework of our SUB setup consists of six main steps, which can further generalized into three layers: data, simulation and analysis. An overview of the process is shown in Figure 1.

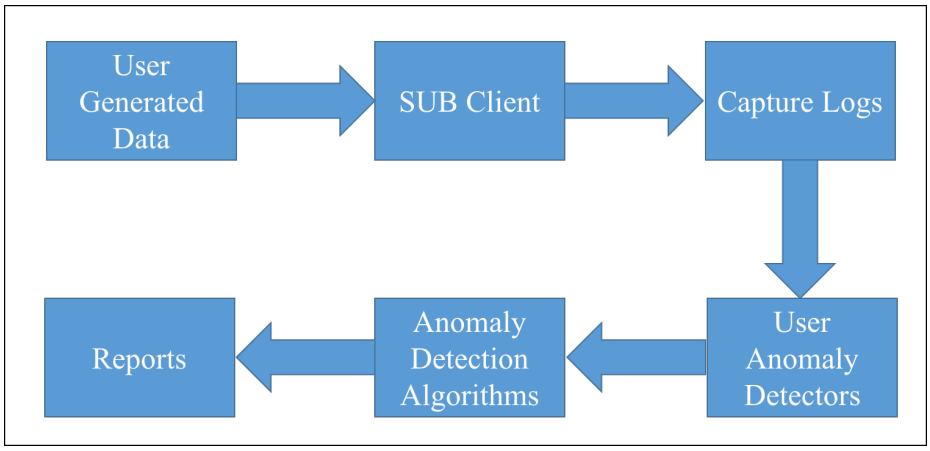


Fig. 1: Simulated User Bot System Overview

2.1 Data Layer

The data layer is composed of the first step in the diagram. Since we do not have a real enterprise in which our work has been deployed, we have employed a

detailed dataset as our surrogate environment or have tested by deriving general parameters from the dataset to architect more users. In either case, we employed a SUB database translator, written in Python. The translator takes a database stored in SQL with at least the following fields: user ID, event and timestamp. It then converts those fields into Python action files for users. We have defined various rules for our translator such that the translated actions can be performed on a SUB that does not have necessarily have access to existing files on a real user's local machine. We have created rules for all the various types of interactions a normal user may have on his or her local machine. Some of our rules for a user's web browsing are as follows:

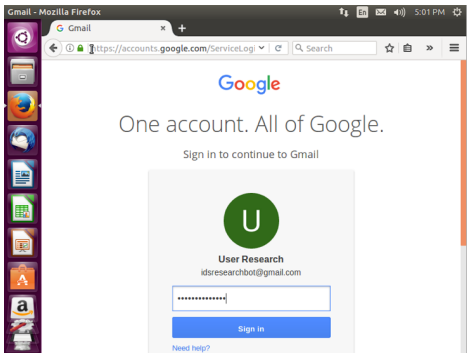
1. Derive the browser used by a particular user by searching through the information provided in a given browser tab if the information is not provided otherwise.
2. Perform the search with the same service as a given user by searching through the information contained in the browser tab unless the information is provided in another field.
3. Search through the browser tab for information regarding the email service used unless provided explicitly by the dataset.
4. Search social media site specified in the browser tab unless provided in the dataset.
5. Browse to the website referred to if the browser tab contains a full HTTP URL.

After the database has been translated, we have action files for each of the users. We can then view the records for each user and decide to remove particular users. By removing users, we have the baseline actions for those users and can add noise or change normal behavior to make a user appear malicious. These modified action files can be used to then proceed to the next steps of creating a SUB. Alternatively, we can derive general parameters for the users from the action files. This could also be done using the raw data, but for consistency, we conduct all exploratory analysis on our translated actions files. In the case in which general parameters are used, actions are selected using a random number generator from a predefined list of activities. For example, for a social media site visit, we have a specified list of the top ten social media sites and would arbitrary pick one for the user and then calculate how long a user should spend on that given site based on the history of how long other users spent on that particular site in the past. The random selection process we utilize to provide actions for the simulated users help guarantee that the users are unique from one another. The downside from the process of using parameters and random selection of actions is that we reduce the realism of the resulting user.

2.2 Simulation Layer

The simulation layer has two components: the SUB client and the log capture steps. In the step consisting of the SUB client, we feed the actions for our predefined users into the automation framework built upon Quick Emulation (QEMU)

and connected to through virtual network computing (VNC), using vncdotool. Since we record the logs at the end of a run by our SUBs, it is paramount that there is no extraneous activity conducted locally, which is guaranteed by using a virtual connection. We additionally make use of Kernel Virtual Machine (KVM), which allows for hardware virtualization and increases the performance of the SUB client by over 40 percent. Our automation framework allows for a simulated user to perform any task a regular user would be able to do: login to an account, send an email, open a website, create and modify documents, etc. An example script and action is displayed in Figure 2.




```
move 0 0
rexpect vdo/firefox.png 0 100 0
move 25 175
click 1
rexpect vdo/firefox_search_bar.png 100 25 0
move 150 75
click 1
type "gmail.com”
key enter
rexpect vdo/gmail_loaded.png 100 0 0
type [redacted password]
key enter
```

Fig. 2: Example of Simulated Email Login

In the data layer, we noted the timestamps to be aware of the gap between activities. It is unrealistic to assume that a user proceeds directly from one task to the next without a pause. The delta in time between activities in the database allows us to train our users to have pauses between actions and helps improve our overall realism. We have additionally added a system of checks into our automation to guarantee that the prescribed action has completed execution and that the system is in the state we would expect it to be in before we continue to the next action. If it fails, we attempt to run the action again, before continuing onto the subsequent actions.

In some cases, we used an open source equivalent of a commercial software to avoid any potential issues with licensing. For example, we used Apache OpenOffice instead of the corresponding Microsoft Office suite. The open source software works nearly identically and handles all of the features that we observed in the database. Thus, no realism was compromised by using the open source alternative. The second part of the simulation layer is to capture both the system and web history logs. The logs allow us to understand the subtle underpinnings of the system when performing the various actions that a normal user would in his or her day to day activity. We originally experimented with several nonnative solutions, including the implementation of a feature to capture all actions on a machine and the corresponding logs. However, any external solution leaves

traces of a sensor in the logs themselves and we opted to instead use the logs produced by the host operating system.

2.3 Analysis Layer

After the data has been generated from the simulated user, we analyze the data that would trigger user anomaly detectors. From work declassified by an external third party operating and analyzing a deployed insider threat solution, we learned about the types of relationships between the threat type, behaviors, indicators and detectors from their redacted challenge problems as seen in Table 1. The insight into how real anomaly detection systems are constructed helped us define our own system for experiments using our SUBs.

Threat Type	Behavior	Indicator	Detector
Individuals with abnormal work habits	Uses a work-owned machine outside of normal work hours (i.e., 12AM-7AM EST) at work, at home, or at remote sites	Has anti-virus updates between 12AM and 7AM EST	At least 1 anti-virus log entry for definition updates between 12AM and 7AM EST
		In the top 5% of the frequency distribution of VPN activity between 12AM and 7AM EST	At least 28 VPN log records showing a connection that started between 12AM and 7AM EST
		In the top 5% of the frequency distribution of workstation logins between 12AM and 7AM EST	At least 128 ADDC log entries with timestamps between 12AM and 7AM EST
		In the top 5% of the frequency distribution of email activity between 12AM and 7AM EST	At least 24 email log records for outbound emails sent between 12AM and 7AM EST
		In the top 5% of the frequency distribution of website activity between 12AM and 7AM EST	At least 24,500 proxy log entries with a timestamp between 12AM and 7AM EST
Individuals with abnormally large data transfers that are not easily explained	Uses a work-owned machine for abnormally large inbound or outbound data transfers	In the top 5% of the frequency distribution of attempted access to prohibited file sharing websites	At least 2000 proxy log entries of attempted access to prohibited file sharing websites
		In the top 5% of the frequency distribution of large emails sent	At least 3 emails with attachments larger than 5 MB
		In the top 5% of the frequency distribution of VPN sessions that transfer large amounts of data	At least 7 VPN sessions where at least 210 MB are transferred
		In the top 5% of the frequency distribution of firewall log entries that transfer large amounts of data	At least 3 firewall log entries where at least 200 KB is transferred
		In the top 5% of the frequency distribution of web browsing sessions that transfer large amounts of data	At least 1 proxy log entry where at least 51 MB is transferred
		In the top 5% of the frequency distribution of USB attachments	At least 11 application white listing log entries of devices attached or mounted
Individuals that have unusual contact with foreign entities	Digitally communicates with non-US entities through a work-owned machine	In the top 5% of the frequency distribution of outbound emails to non-US email addresses	At least 1 outbound email to a non-US email address
		Have VPN sessions from non-US locations	At least 1 VPN session initiated from a non-US location
		In the top 5% of the frequency distribution of firewall log entries to non-US locations	At least 37 firewall log entries to non-US locations
		In the top 5% of the frequency distribution of web browsing sessions to non-US websites	At least 7100 proxy log entries to non-US owned websites

Table 1: Example of Threat Types, Behaviors, Indicators and Detectors

For the purposes of illustrating the effectiveness of our SUB framework, we then consider three commonly used algorithms to detect anomalous behavior: Gaussian Mixture Models, Support Vector Machines and Bayesian Networks. The final part of our analysis is to create a method to produce reports and to organize all of the data into easier to understand visualizations. For the three algorithms described below and any corresponding visualizations, we used im-

plementations in Python due to ease in which existing packages can be used and modified to suit our needs.

Gaussian Mixture Model A Gaussian Mixture Model is a probabilistic model that has gained popularity in intrusion detection systems. Promising results have been found using Gaussian Mixture Models in well known datasets such as the KDD99 data set from Lincoln Labs of MIT [11]. The model itself can be represented as shown in Equation 1.

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i g(\mathbf{x}|\mu_\mathbf{i}, \mathbf{\Sigma_i}), \tag{1}$$

where $\mathbf{x}$ is a D-dimensional continuous value data vector, w_i are mixture weights and $g(\mathbf{x}|\mu_\mathbf{i}, \mathbf{\Sigma_i})$ are component Gaussian densities. A key differentiating factor between Gaussian Mixture Models and k-means clustering is the inclusion of the covariance structure of the data. The EM algorithm is run to estimate the parameters of the model. The process is done in a way such that there is a guaranteed monotonic increase in the model's likelihood value. Further details on Gaussian Mixture Models and the EM algorithm can be found in the reference by Douglas Reynolds [28].

Support Vector Machine Support vector machines are normally supervised classifiers that attempt to map input vectors into a high dimensional feature space using optimal hyperplanes, those with the maximal margin between the vectors of the two classes, for separable classes. The details of the algorithm are described in detail by Cortes and Vapnik [18]. However, researchers a few years later pushed the concept further to tackle unlabelled data, better known as unsupervised learning. The method attempts to find a function that is positive on a subset of the input space and negative on the complement by mapping the input data into a higher dimensional space and using the origin as a negative training point. The objective function is given by

$$\begin{aligned} \min_{w \in F, \xi \in R^l, \rho \in R} \frac{1}{2}||w||^2 + \frac{1}{vl}\sum_i \xi_i - \rho \\ \text{s.t. } (w \cdot \Phi(x_i)) \geq \rho - \xi_i, \xi_i \geq 0 \end{aligned} \tag{2}$$

where ν is a parameter between 0 and 1 that controls how tightly the support vector machine fits the data. Complete details on the algorithm can be found in their seminal work [30].

Bayesian Network Bayesian Networks make use of probabilistic relationships among variables of interest in an acyclic graphical model. Figure 3 shows an example of how such a network can be used to model a threat type, behavior, indicators and detectors. The flow of the diagram was adapted from challenge problems released by an external third party operating and analyzing a deployed

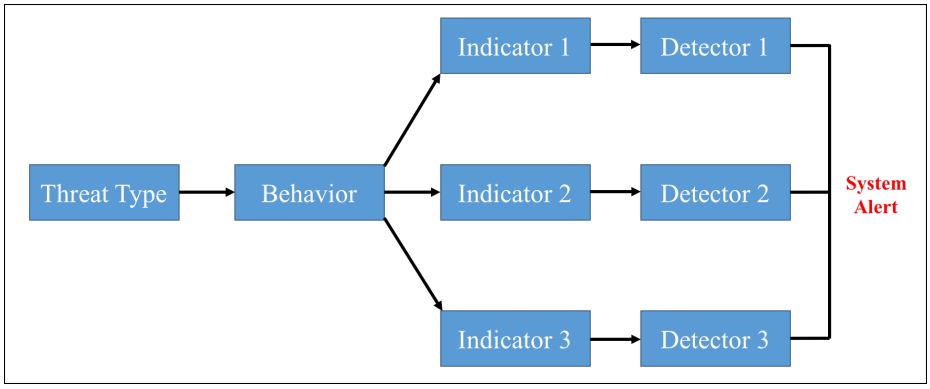


Fig. 3: Bayesian Network Representing Third Party Challenge Problem

insider threat solution and is designed to be representative of a real system. Formulaically, a bayesian network can be represented as follows:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\pi_i) \tag{3}$$

where $X_1, X_2, \ldots, X_n$ represents random variables and π_i is the set of parents of X_i. In many regards, it is more difficult to learn the structure of a bayesian network than it is to learn the parameters. The EM algorithm can again be used to learn the parameters when the structure of the network is known. It often takes expert knowledge to suggest a structure for a given problem. Further details on the algorithm are contained in [14].

3 Experiments and Results

3.1 Data and Preparation

Data We used a dataset collected from a group of 63 West Point cadets [1]. Each cadet had software installed on his or her machine to track usage. The earliest installations were January 15, 2015, and the latest installation was on February 13, 20015. Each user had a participant/device Windows System ID and unique ID number. The cadets had up to three extraction dates for the data from their machines, ranging from February 10, 2015, for the first pull to March 12, 2015, for the last data collection. The participants were assigned different labs to simulate masquerader data within the dataset. Participants were given different labs to prevent adjacent sitting cadets from influencing each other's work. Time stamps were taken from the desktop of the device as soon as the masquerader turned the platform on or off. There were two masquerader labs in total. For the purposes of our work, we opt to not use the imposter data explicitly and remove

[1] Data from the West Point cadets was gathered under an IRB approved protocol

the data points from the dataset since our primary objective is to replicate how a normal user acts on a machine without being directed to perform a certain set of actions.

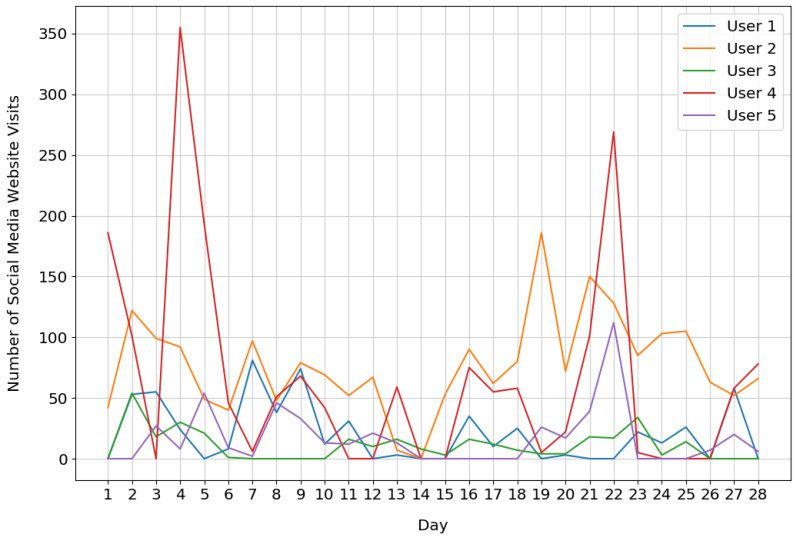


Fig. 4: Social Media Website Visits from West Point Cadets

Information contained within the dataset includes, but is not limited to, a unique ID for each user, a time stamp for a given action, an action column to describe the event that takes place and a details field that provides more information on a given action. The details field contains specific information such as the exact terms searched for in a search, the title of a page visited, etc. The West Point dataset is a nearly homogeneous population of cadets. However, as can be seen in Figure 2, the number of visits to social media websites vary widely from one cadet to the next. The same observation can be made for any variable behavior and such diversity of behavior is useful for machine learning algorithms.

Hardware We ran all of our experiments on a MacPro with sixteen gigabytes of RAM (DDR3, 1066MHz) and four physical cores (Intel(R) Xeon (R) CPU E5520, 2.27 GHz). The operating system on the host machine has been changed from the default OS X to Ubuntu 16.04 Desktop. QEMU has been developed for both OS X and Linux distros, but performance is much better on Linux. Our simulations are run on a image running Ubuntu 16.04 Desktop as well. However, our framework is adapted to work on Windows-based SUBs as well.

Threat, Behavior, Indicators and Detectors We will use the challenge problems from an external third party operating and analyzing a deployed insider threat solutionas a guide to set our threat, behavior, indicators and detectors. We define our threat to be individuals who use their machines with abnormal work habits. The behavior associated with this threat is a user who uses their machine outside normal work hours (between 5:00:01 PM and 6:59:59 AM EST). We will consider three indicators of this behavior:

1. In the top 5 percent of the daily frequency average distribution of Google or Bing searches between 5:00:01 PM and 6:59:59 AM EST
2. In the top 5 percent of the daily frequency average distribution of social media website visits between 5:00:01 PM and 6:59:59 AM EST
3. In the top 5 percent of the daily frequency average distribution of actions on files and documents between 5:00:01 PM and 6:59:59 AM EST

This corresponds to the following three detectors:

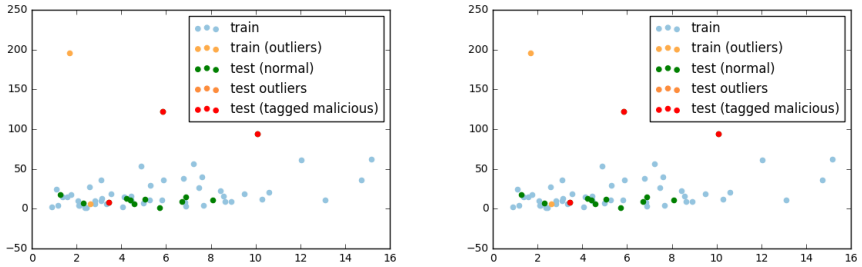
1. At least 13 log entries for a Google or Bing search between 5:00:01 PM and 6:59:59 AM EST.
2. At least 61 log entries for a social media website visit between 5:00:01 PM and 6:59:59 AM EST.
3. At least 90 log entries for actions on files and documents between 5:00:01 PM and 6:59:59 AM EST.

3.2 Experimental Setup, Results and Discussion

For our experiments, we used a shuffling algorithm to split the 63 West Point cadets into a training group of 50 users and the remainder were placed in the testing group. We selected Gaussian Mixture Models as the first method to run on the data. After running the algorithm, we determined anomalous users in the test data composed of 13 users. We then randomly selected three users in the test data who were not originally marked by the algorithm and added behaviors to their actions to make them appear anomalous to test our ability to create anomalous SUBs. We then ran the Gaussian Mixture Models, Support Vector Machines and Bayesian Network algorithms.

Gaussian Mixture Models In our experiments, we used the detectors listed above as the three features of our users and set the number of components in our Gaussian Mixture Models correspondingly. We also set the false positive rate to be 5 percent since we assume that all of our users are normal. The results are shown in Figure 5a. Gaussian Mixture Models were able to detect all three of the injected malicious users.

Support Vector Machines Similarly, using Support Vector Machines, we were able to detect all three of our injected malicious users as shown in Figure 5b. We used a grid search for hyperparameter tuning for the algorithm since poor



(a) Gaussian Mixture Model (b) Support Vector Machine

Fig. 5: Algorithms Run on Injected, Anomalous SUBs

parameters resulted in poor performance. After searching, we found the optimal values for the parameters with $\nu = 0.264$ and $\gamma = 1.0$. We additionally used the gaussian kernel, which performed better than sigmoid kernel.

Bayesian Networks Given the importance of expert knowledge, we were not surprised to find that the thresholds had to be adjusted to detect any of the users as malicious. This is given the simple network we considered. We adjusted the thresholds for the detectors as follows: Detector 1 to 8, Detector 2 to 40, and Detector 3 to 50. We then assumed that the probability of an intruder given any single detector is $\frac{1}{3}$ and that two detectors being set off meant that the probability of an intruder increased to $\frac{2}{3}$. Using the probabilities listed, we were able to detect all three of our injected malicious users. The difficulty to define conditional probabilities a priori is a major challenge with bayesian networks and explains the use of either expert knowledge or significant data analysis from an enterprise environment. Without access to either, one can only make basic assumptions that are unlikely to be reflective of reality. It is unrealistic to assume that all detectors are equally as predictive of an insider threat. For problems posed by an external third party operating and analyzing a deployed insider threat solution this is a major challenge that evaluators face as well and is often approximated from existing datasets albeit with different users since direct access to enterprise data is not always available. Thus, it validates the use of SUBs in such a network to create different users of differing levels of maliciousness to generate enough data to determine those probabilities for the development of meaningful intrusion detection systems.

4 Related Work

Testing intrusion detection systems has been a long standing problem and one that has been identified almost since the inception of the systems themselves.

Lincoln Laboratory of MIT performed comparative evaluation of various systems developed under DARPA funding in both 1998 and 1999 [25, 24]. However, the methods used by Lincoln Laboratory came under scrutiny for various reasons such as a failure to explain the validation of their test data [26]. Despite the identification of this issue regarding a lack of testing tools as early as the late 1990s, the 2015 report on Cybersecurity Experimentation identifies the lack of common tools for testing systems as one of the major challenges in current cybersecurity research [12]. Related work in this area can be categorized into synthetic data generation approaches, network based approaches, and integrated test bed approaches.

4.1 Synthetic Data Generation

At a high level, many have embarked on the task of generating realistic data sets, which is a direct by product of our work with SUBs. Data generation has been done at all levels for various purposes such as the work done by Boggs et. al. for the purposes of defense in depth measurement of web applications [15]. Generation of synthetic data was used successfully during the DARPA ADAMS project to test Insider Threat Detection models offline. The synthetic data generation system developed for ADAMS used a set of interconnected models individually trained on real world data sets to model a large organization and directly generate the data that would be collected by an Insider Threat Detection System [21]. However, while synthetic data generation systems are evidently effective at conducting offline tests, they cannot be used to test live systems *in situ* unlike our SUBs. Given the recent advent of deep neural network based techniques, researchers have started using deep learning based architectures for generating synthetic data sets. Alzantot et. al. use a dataset of accelerometer traces to illustrate that their synthetic generator, a combination of Long Short Term Memory networks and Mixture Density Network, can create believable data [10]. However, even their work acknowledges that discriminator models can distinguish between real and synthetic data with an accuracy of roughly 50 percent.

4.2 Network Based Approaches

Development of Network Traffic Modeling and Generation systems is a mature field with many open source tools available such as Ostinato, Iperf3, and Netperf [7, 5, 6]. These tools allow the user to configure a large number of parameters that may be measured by an Intrusion Detection System, such as flow volume distribution, burstiness, and packet rates. In addition extensive work has been done developing systems that both model and generate network traffic, such as Swing [31]. More recently, Spatio-Temporal Cluster Models have been used to model and generate traffic across a cluster [23]. All of these systems have the potential to be used in testing network based IDS in action. However, none of them can also drive simultaneous tests for host based IDS or Insider Threat Detection Systems.

4.3 Integrated Test Beds

Cyber test bed systems are the most similar to SUBs in that they frequently involve tests that drive both the host and network level events. The Lincoln Labs of MIT testbed LARIAT has a complete framework for specifying network configuration, driving user inputs, and modeling user behavior [32, 17]. Another tool, the Skaion Traffic Generation Tool, is used in the National Cyber Range and has the ability to simulate user actions within the test bed environment [8, 20]. Integrated test beds have strong advantages but are resource intensive. Integrated testbeds also do not test the vagaries and potential configuration bugs in a live system as SUBs provide.

5 Future Work

In future work, the immediate application of our work would be to find a suitable enterprise to test our framework to determine its true efficacy. A simulated environment offers the ability to prototype, but does not allow us to test the broader impact potential of our framework until adequate real world testing is conducted.

In the absence of the ability to apply our work to a real enterprise, we will test our framework with other datasets such as the Are You You? (RUU) dataset [29]. This will allow us to generalize our SUB database translator to handle a wider variety of datasets. More generally, we wish to create more intelligent users that do not necessarily emulate a real user to a large extent. Through the use of Hidden Markov Models with adequate memory, we could derive users and their paths, building on the work applied to Clickstream [27]. There is also the possibility of applying deep neural networks of various architectures to model user behavior and create more robust user action files than adding systematic noise to existing users.

In this paper, we enumerated a sample of indicators used in intrusion detection systems from a third party operating and analyzing a deployed insider threat solution. We are actively investigating other indicators to enhance SUB behaviors to ensure those indicators are fully tested and vetted. The SUB framework also provides a testing mechanism for newly created indicators that may be proposed by security researchers.

Another area of development for the SUB framework is to improve the performance albeit it is nearly native speed. We can continue work on passing the onboard video card to the instance instead of relying on the CPU, which is a performance constraint when using QEMU. Another area of improvement lies in our ability to increase the speed of running a simulation by speeding up the system clock for non-work hours. This would reduce the realism of the system, but vastly reduce simulation time for the purposes of creating a dataset. The feature would be optional since users in a real environment would need to emulate their real user counterparts and be idle during non-work hours.

The SUB framework has potential for intrusion detection algorithm development. One algorithm or method of significant interest is to detect slow insider

threats that often go unnoticed by traditional indicators and detectors such as the ones specified in our experimental section. Slow insider attacks are difficult to determine given the long time horizon over which they may take place. As such, there is significant memory overhead to keep track of the activities for each individual user. With the SUB framework, we can experiment with different algorithms or memory management techniques to see if a slow insider threat can be successfully detected consistently.

6 Conclusion

We presented the framework to design and to create SUBs. The simulated users have the ability to almost identically mimic the behaviors and actions of real users. All of this can be done within an enterprise network with no need to interfere with the normal operations of the system. We were able to design an experiment to show that we could inject malicious activities such that a normal user appears to act as a malicious user. In the case of the malicious users added, we were able to successfully detect them. The ability to do so provides optimism that the same will hold true in an enterprise level deployment to determine whether the anomaly detection system is indeed catching the intruders.

References

1. https://www.crunchbase.com/organization/observeit#/entity
2. https://www.crunchbase.com/organization/niara-inc#/entity
3. https://www.crunchbase.com/organization/alphasoc#/entity
4. https://github.com/sibson/vncdotool
5. Iperf3. http://software.es.net/iperf/, accessed: 2017-03-17
6. Netperf. http://www.netperf.org/netperf/, accessed: 2017-03-17
7. Ostinato network traffic generator and analyzer. http://ostinato.org/, accessed: 2017-03-17
8. Skaion Corporation. http://www.skaion.com/, accessed: 2017-03-17
9. Market guide for user and entity behavior analytics (Sep 2015), https://www.gartner.com/doc/3134524/market-guide-user-entity-behavior
10. Alzantot, M., Chakraborty, S., Srivastava, M.B.: Sensegen: A deep learning architecture for synthetic sensor data generation. CoRR abs/1701.08886 (2017), http://arxiv.org/abs/1701.08886
11. Bahrololum, M., Khaleghi, M.: Anomaly intrusion detection system using gaussian mixture model 1, 1162–1167 (Nov 2008)
12. Balenson, D., Tinnel, L., Benzel, T.: Cybersecurity Experimentation of the Future. Tech. rep., University of Southern California (07 2015)
13. Bellard, F.: Qemu, a fast and portable dynamic translator pp. 41–41 (2005), http://dl.acm.org/citation.cfm?id=1247360.1247401
14. Ben-Gal, I.: Bayesian Networks. John Wiley & Sons, Ltd (2008), http://dx.doi.org/10.1002/9780470061572.eqr089
15. Boggs, N., Zhao, H., Du, S., Stolfo, S.J.: Synthetic data generation and defense in depth measurement of web applications. Research in Attacks, Intrusions and Defenses Lecture Notes in Computer Science pp. 234–254 (2014)

16. Bowen, B.M., Prabhu, P., Kemerlis, V.P., Sidiroglou, S., Keromytis, A.D., Stolfo, S.J.: Botswindler: Tamper resistant injection of believable decoys in vm-based hosts for crimeware detection. Lecture Notes in Computer Science Recent Advances in Intrusion Detection p. 118137 (2010)
17. Braje, T.: Cyber ranges. Lincoln Laboratory Journal 22(1), 24–32 (November 2016)
18. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995), http://dx.doi.org/10.1023/A:1022627411411
19. Eberle, W., Graves, J., Holder, L.: Insider threat detection using a graph-based approach. Journal of Applied Security Research 6(1), 32–81 (2010), http://dx.doi.org/10.1080/19361610.2011.529413
20. Ferguson, B., Tall, A., Olsen, D.: National cyber range overview. In: 2014 IEEE Military Communications Conference. pp. 123–128 (Oct 2014)
21. Glasser, J., Lindauer, B.: Bridging the gap: A pragmatic approach to generating insider threat data. 2013 IEEE Security and Privacy Workshops (2013)
22. Kandias, M., Mylonas, A., Virvilis, N., Theoharidou, M., Gritzalis, D.: An Insider Threat Prediction Model, pp. 26–37. Springer Berlin Heidelberg, Berlin, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-15152-1_3
23. Li, T., Liu, J.: Cluster-based spatiotemporal background traffic generation for network simulation. ACM Trans. Model. Comput. Simul. 25(1), 4:1–4:25 (Nov 2014), http://doi.acm.org/10.1145/2667222
24. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: Analysis and results of the 1999 darpa off-line intrusion detection evaluation. Lecture Notes in Computer Science Recent Advances in Intrusion Detection pp. 162–182 (2000)
25. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., Mcclung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., et al.: Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00
26. McHugh, J.: Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans. Inf. Syst. Secur. 3(4), 262–294 (Nov 2000), http://doi.acm.org/10.1145/382912.382923
27. Montgomery, A.L., Li, S., Srinivasan, K., Liechty, J.C.: Modeling online browsing and path analysis using clickstream data. Marketing Science 23, 579–595 (2004)
28. Reynolds, D.: Gaussian Mixture Models, pp. 659–663. Springer US, Boston, MA (2009), http://dx.doi.org/10.1007/978-0-387-73003-5_196
29. Salem, M.B., Stolfo, S.J.: Modeling User Search Behavior for Masquerade Detection, pp. 181–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-23644-0_10
30. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. 13(7), 1443–1471 (Jul 2001), https://doi.org/10.1162/089976601750264965
31. Vishwanath, K.V., Vahdat, A.: Swing: Realistic and responsive network traffic generation. IEEE/ACM Transactions on Networking 17(3), 712–725 (June 2009)
32. Wright, C.V., Connelly, C., Braje, T., Rabek, J.C., Rossey, L.M., Cunningham, R.K.: Generating Client Workloads and High-Fidelity Network Traffic for Controllable, Repeatable Experiments in Computer Security, pp. 218–237. Springer Berlin Heidelberg, Berlin, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-15512-3_12