Intelligent Data Analysis 27 (2023) 417–422
DOI 10.3233/IDA-216511
IOS Press

Synthetic dataset generator for anomaly detection in a university environment

Pavel Strnad, Lukáš Švarc and Petr Berka
Department of Information and Knowledge Engineering, Prague University of Economics and Business, Prague, Czech Republic

Abstract. This article introduces a recently developed synthetic dataset generator, which contains anonymised data from the Prague University of Economics and Business information system logs. The generator is opensource and is able to scale this data time-wise and also perform injection of the data with cyberattackers' behaviour patterns. The anonymised data still contains user behaviour patterns; therefore, individual anomalous behaviour can be detected. Different types of real attack behaviour patterns in the university environment have been selected; they are used to demonstrate attackers' behaviour in synthetically created system logs. The mentioned features allow other researchers to benchmark their anomaly detection algorithms with complex data.

Keywords: Anomaly detection, automated attacks, dataset, machine learning, synthetic generator, university environment

1. Introduction

The university environment is different from the usual business company environment. Users of university systems vary over time; new students come every year, while some others leave. There are many external experts who cooperate with the university but not in same way as casual employees. A specific user behaviour can be defined under such conditions, different from that observed in business environment. As Sosnovsky [1] suggests, some students use the system in an unintended way, exhibit long periods of off-task behaviour, try gaming the system, seek help of parents or peers, etc. Such usage patterns will manifest themselves in sequences of activity that do not represent student abilities and will result in student modelling anomalies causing subsequent suboptimal adaptive interventions from the system.

Just a few possibilities to train anomaly detection algorithms currently exist. Synthetic datasets are available for this task, but they usually come from business environments, in which users are trained for their jobs and so behaviour anomalies can easily be tracked. On the other hand, the university environment provides many cases in which a detected anomaly is just a behaviour of a new student as he struggles to work with the system. This data is hard to get to work with. There also are not many possibilities to obtain data including real attackers' behaviour since most of the common datasets have been manually developed. To provide another possibility for training the anomaly detection algorithms there have been an opensource synthetic log generator developed in the environment of Prague University of Economics and Business.

[*]Corresponding author: Pavel Strnad, Department of Information and Knowledge Engineering, Prague University of Economics and Business, W. Churchill Sq. 1938/4, 130 67 Prague 3 Žižkov, Czech Republic. Tel.: +420 731 135 806; E-mail: pavel.strnad@vse.cz.

Table 1
Original dataset description

Attribute	Description
Transaction ID	Unique number of every action
User ID	Numeric user identification
Date Time	Timestamp
URL String	Type of action
Source IP	User's IP address
Parameter String	User data in HTTP request
Session ID	Numeric session identification

2. University dataset

The dataset used in our synthetic log generator is built on real data that has been extracted from the information system of the Prague University of Economics and Business.

The original data used in this experiment is stored in a single database table containing all the activities performed in the university information system by both anonymous and authenticated users. Each activity is defined by the URL of the associated Perl script located on the server. User-supplied parameters are stored along with each action. A complete description of the data model is shown in Table 1.

For our purpose, i.e., that of creating a synthetic dataset generator, three different datasets have been extracted from the university system log. Each dataset contains one specific type of attackers' behaviour combined with everyday actions performed by users. All attacks have been labelled by a university security specialist according to previously resolved security incidents.

The simple automated attack dataset contains behaviour of a student who games the system by using his own script, supposed to help him achieve a better date of an exam as all the good ones were already full. The mentioned script repeatedly downloaded exam capacity in order to find a free spot on the preferred date. One simple action is repeated in the same time interval. This dataset contains data of 15 days. There were 4,071,172 actions performed, out of which 3,362 were malicious. The dataset size is 672 MB.

The advanced automated attack dataset contains behaviour of a compromised student account that has been downloading sensitive information about other users, also performed by a script with the same time intervals between subsequent actions. It has been classified as an advanced attack as the script performed different types of system actions and so it would be harder to find by machine learning algorithms that would be focused on repeating the same action multiple times. This dataset contains data of 15 days. There were 4,632.527 actions performed, out if which 21,548 were malicious. The dataset size is 748 MB.

The third dataset contains cyber-attack user behaviour. A user account has been compromised and the attacker performs malicious actions across the system. Each action is different and there is no automated behaviour pattern based on time intervals that would define this anomaly. This dataset contains data of 15 days. There were 5,695,202 actions performed, out of which 984 were malicious. The dataset size is 909 MB.

Based on these datasets, the patterns of these attacks were implemented into the generator.

As the generator is publicly presented, a data anonymisation of the original datasets is required. To keep its informational value while keeping sensitive data private, several new columns have been added to help the analysts track behaviour based on geological data that are connected with users' IP addresses.

The final anonymised and pre-processed datasets have a structure defined in Table 2, which contains a detailed description of the attributes and their semantics.

Table 2
Anonymised dataset description

Attribute	Type	Description
ID_NORM	Bigint	ID is renumbered so that each data set identifier starts with 0.
USER	Varchar	Anonymised user. e.g. user234
TIMESTAMP_NORM	Int	Unix timestamp renumbered so that each data set identifier starts with 0.
URL	Varchar	Anonymised URL. e.g. url234
IP	Varchar	Anonymised IP. e.g. ip789
PARAMETERS_HASH	Varchar	Hashed value of the original URL parameters, using SHA2-256 algorithm with salt.
ASN	Int	Autonomous system number to which the IP address belongs.
BGP_PREFIX	Varchar	Network prefix
AS_NAME	Varchar	Autonomous system name
NET_NAME	Varchar	Specified for IPs within the university network
COUNTRY_CODE	Char(2)	Country code of the IP

ATTACK	Char(3)	Label for the attack
		AAA = Advanced Automated Attack
		CA = Cyber-attack
		SAA = Simple Automated Attack
		normal = Normal traffic

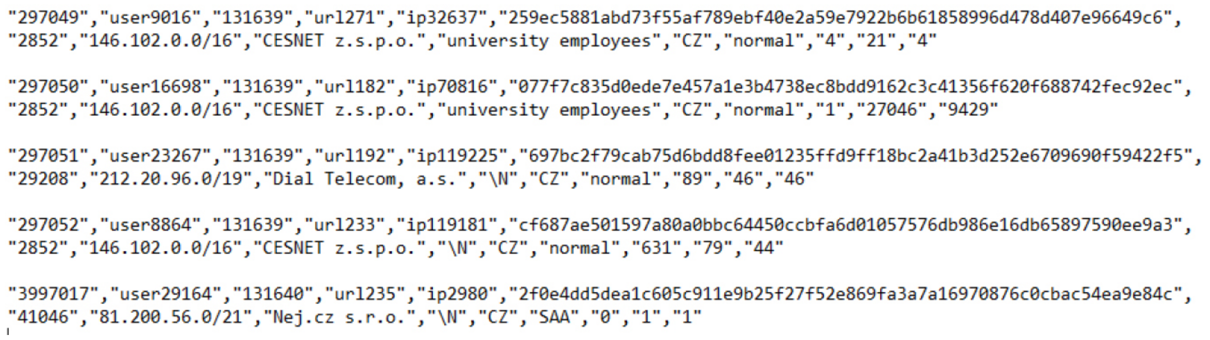
TIME_FROM_ LAST_ACTION	Int	This attribute contains a numeric value that represents the time passed between the last action of an individual user and the current one.
USER_PER_DAY	Int	This attribute contains a numeric value that represents a sum of all actions performed by a user until a current moment in a single day.
UNIQUE_ACTIONS_ PER_DAY	Int	This attribute contains a numeric value that represents a sum of unique actions performed by a user until a current moment in a single day.

3. Synthetic dataset generator

The generator contains anonymised data from the Prague University of Economics and Business information system logs; it is able to scale this data time-wise and also perform injection of the data with attack behaviour patterns. The anonymised data still contains user behaviour patterns; therefore individual anomalous behaviour can be detected. Different types of real attack behaviour patterns from a university environment have been selected and are used to demonstrate the attacker's behaviour in synthetically created system logs. The mentioned features allow other researchers to benchmark their security detection algorithms with complex data. The more complex a model is the more data is necessary. A simple automated attack dataset can be referred to as a perfect example, in which a single user performs a high volume of actions in a single day. This anomaly can very easily be detected by a selected security action threshold in user action counts per day. But for a user who would perform such an action every day, this threshold would need an additional level of classification. This is the reason why the generator has been developed to allow a possibility to benchmark anomaly detection algorithms on data with different synthetically manufactured scenarios.

The developed generator is published in the GitHub[1] repository, available to anyone who wants to experiment with machine learning techniques on similar types of data. Example of the final dataset is shown in Fig. 1.

[1]https://github.com/HellhoundAI/Synthetic-Dataset-Generator.



```
"297049","user9016","131639","url271","ip32637","259ec5881abd73f55af789ebf40e2a59e7922b6b61858996d478d407e96649c6",
"2852","146.102.0.0/16","CESNET z.s.p.o.","university employees","CZ","normal","4","21","4"

"297050","user16698","131639","url182","ip70816","077f7c835d0ede7e457a1e3b4738ec8bdd9162c3c41356f620f688742fec92ec",
"2852","146.102.0.0/16","CESNET z.s.p.o.","university employees","CZ","normal","1","27046","9429"

"297051","user23267","131639","url192","ip119225","697bc2f79cab75d6bdd8fee01235ffd9ff18bc2a41b3d252e6709690f59422f5",
"29208","212.20.96.0/19","Dial Telecom, a.s.","\N","CZ","normal","89","46","46"

"297052","user8864","131639","url233","ip119181","cf687ae501597a80a0bbc64450ccbfa6d01057576db986e16db65897590ee9a3",
"2852","146.102.0.0/16","CESNET z.s.p.o.","\N","CZ","normal","631","79","44"

"3997017","user29164","131640","url235","ip2980","2f0e4dd5dea1c605c911e9b25f27f52e869fa3a7a16970876c0cbac54ea9e84c",
"41046","81.200.56.0/21","Nej.cz s.r.o.","\N","CZ","SAA","0","1","1"
```

Fig. 1. Example of final dataset.

3.1. Generator architecture

As a platform, Python 3 has been selected for its flexibility, speed (relative to other interpreted languages) and ease of implementation, especially when working with text files. Despite initial attempts with the Pandas library, no external libraries are used in the final versions – only internal Python libraries. Therefore, the generator can be run with clean Python 3. The architecture is split into two main parts – inserting (generating) attacks into a file with network logs; and counting the delays between user actions (or user actions per day) in the resulting file (into which the attacks have been generated).

3.2. Attack insertion algorithm

The program itself always (with one exception – the transform mode described in the Delay Counting Algorithm section) requires two text files as an input – one file with regular network logs into which attack (or attacks) will be randomly inserted (log file); and another file with the network logs of this specific attack (attack file). These text files are always assumed to be in a standard CSV format. The time intervals between subsequent actions of the attacker in the attack file (i.e., intervals between subsequent rows in the file) are crucial for identification of the attack; these intervals are therefore calculated at the beginning. The user can then choose how many attacks to insert. The log file is assumed to represent a 15-day period, but ultimately a specific number of days is not crucial for the program. The user can choose to scale (upward) the resulting output file up to n periods and thus also spread the inserted attacks over these n periods. In other words, if the user chooses to insert 10 attacks in total, they can then also choose what period these 10 attacks will cover – 15, 30, 45, etc. days. Furthermore, the distribution of attacks over each period is random; so each period will have a different number of attacks, adding up to the total number of 10 attacks.

The insertion of an attack into one single period is also done at random, i.e., the row in the log file where the attack will be inserted is randomly chosen. The key condition is keeping the intervals the same between the attacker's subsequent actions (mentioned in the first paragraph of this chapter). Because of the randomised insertion, it is well possible that with, for example, four periods and 10 attacks, all the attacks will be inserted into one period and none into the remaining ones. Nevertheless, as one single attack is always thought of as a single entity, it can never be split between two periods.

To summarise, the algorithm makes $n + m - 1$ (n being the number of attacks chosen by user, m being the number of periods chosen by user) random choices – first, how many attacks (0 up to n) will be inserted into one given period (this choice occurs $m - 1$ times). And second, at what row (1 up to the very last row) in the log file a single attack will be inserted (this choice occurs n times). So, if $n = 10$

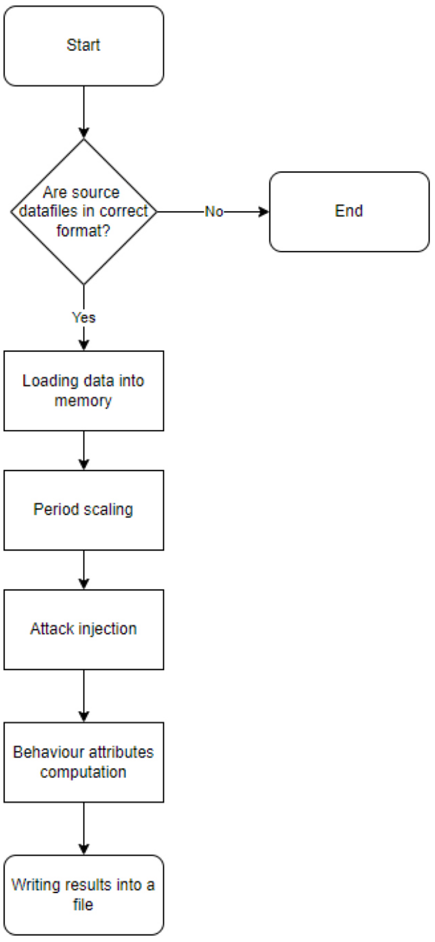

Start
Are source datafiles in correct format?
No
End
Yes
Loading data into memory
Period scaling
Attack injection
Behaviour attributes computation
Writing results into a file

Fig. 2. Dataset creation scheme.

and $m = 4$, the program makes 13 random choices. The '-1' in '$n + m - 1$' means that when inserting into the last period, no random choice needs to be made – the program must simply insert the remaining attacks into the last period.

The scheme of dataset creation is shown in Fig. 2.

3.3. Delay counting algorithm

Compared to the attack insertion algorithm, the delay counting is fairly straightforward. The program takes as an input the resulting output file of attack insertion and adds three columns to it:

1. Time from last action, which counts, for each user in the file, the interval between their actions,
2. User per day, which simply counts, for each user, how many times he made an action (i.e., how many times he is in the network logs) per one day,
3. Unique actions per day, which counts, for each user and each URL, the number of times it has been accessed per day.

The program can also be run in the transform mode, which applies all of the above to an already existing file. This can be useful for a transformation (hence the name) of an existing reference dataset.

4. Conclusions and future work

Due to the increase of cyber-attacks in recent years, the importance of misuse-detection-based anomaly detection algorithms has drastically risen. Both supervised and unsupervised machine learning techniques must be improved and developed in order to become more successful against attackers. Synthetic dataset generators provide one the most common ways to benchmark these algorithms and improve them. Both supervised an unsupervised machine learning algorithm benchmarking can benefit from using the synthetic dataset generator that has been presented in this paper.

A synthetic dataset generator for anomaly detection in a university environment has been developed based on real data and attacks from the information system of the Prague University of Economics and Business. In total, three datasets with three different types of attacks have been prepared – Simple Automated Attack, Advanced Automated Attack and Cyber-attack. Based on these datasets, the patterns of these attacks have been implemented into the generator.

The opensource generator built on Python 3 contains anonymised data and is able to scale this time-wise and also perform injection of data with the attack behaviour patterns. The anonymised data still contains user behaviour patterns; individual anomalous behaviour types can therefore be detected. The algorithm is highly scalable and features using user-defined log and attack files. It can be tuned up with the aid of two basic parameters that control the size of the generated data and the number of the injected attacks. Each dataset represents a period of 15 days containing about half a million user actions; this scope can be enlarged, while the attack injection is based on a random pattern. Users can select how many attacks the final dataset should contain, and the generator will randomly insert the attack pattern into the dataset.

Different scenarios for the defined datasets from the university environment can be deveoped in order to simulate complex situations in advance. The scalability and data injection features allow users to prepare datasets with malicious activities in order to complete complex benchmarks. As the generator provides three different types of attackers' behaviour, all kinds of machine learning algorithm features from signature-based classification to deep learning neural networks can be tested. The provided benchmark results can define the complexity of the selected machine learning algorithm and the number of different scenarios to be positively covered.

Future work on the synthetic dataset generator will include adding more behaviour attributes to provide easier access for data analytics. More attack types will possibly be added in order to extend the behaviour pattern database. Community based development is also considered for future advancement. We are aware that there is no attack modification function implemented, but we plan to add this feature in the future when a larger community gathers around this project.

Acknowledgments

This paper has been financially supported by Internal Grant Agency of Prague University of Economics and Business by Project F4/37/2021.

Reference

[1] S. Sosnovsky et al., Detection of Student Modelling Anomalies, Lifelong Technology-Enhanced Learning, Springer International Publishing, 2018, 531–536.