Dynamic Traffic Generation with Containerization for Machine Learning

ABSTRACT

KEYWORDS

Network security, datasets, machine learning, intrusion detection

ACM Reference Format:
. 2019. Dynamic Traffic Generation with Containerization for Machine Learning. In *DYNAMICS '19: DYnamic and Novel Advances in Machine Learning and Intelligent Cyber Security Workshop, December 09–10, 2019, San Juan, PR.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

. . . . . .

. . . . . .

. . . . . .

This work provides the following contributions:

(1) We present a novel network traffic generation framework that is designed to improve several shortcomings of current datasets for NIDS evaluation in the following aspects:
   (a) Ground truth labels documenting conducted activities
   (b) Increased richness of data
   (c) Tunable topology and data composition
   (d) Additional capture of program logs and system calls

   This framework is openly accessible for researchers and allows for straightforward customization.
(2) We perform a number of experiments to demonstrate the fidelity to realism of the generated data.
(3) We present a number of use-cases to demonstrate how the design of our framework can boost performance for ML-based network intrusion detection systems, in particular for false-positive analysis and effective model training.

1.1 Outline

Outline of the coming sections.

. . . . . .

. . . . . .

. . . . . .

2 BACKGROUND

2.1 Data formats

A brief description on network flow and packets, as well as system and program logs since we also capture those.

. . . . . .

. . . . . .

. . . . . .

2.2 Related work and existing datasets

A similar description on existing frameworks and datasets as in the existing paper, with a slightly higher focus on existing testbeds and existing container tools. . . . . . .

. . . . . .

. . . . . .

2.2.1 Problems in modern datasets. We can import here a lot from the existing paper, but add the following issues:

(1) Invalid/Inconsistent traffic
(2) Extensibility of the attacks
(3) Lack of open-source implementation of attack traffic generation, which makes difficult to understand what exactly was detected
(4) Lack of related data sources

2.3 Containerization with Docker and Mininet

A similar description on Containers and Docker as before, extended by a description of Mininet . . . . . .

. . . . . .

. . . . . .

2.3.1 Metasploit and Metasploitable containers. A description of the capabilities and limitations of both metasploit and the metasploitable VM and the corresponding containers, which we will use.

2.4 Dataset Requirements and benefits of our framework

Here, we can point to a set of requirements by Cordero et al. (https://arxiv.org/pdf/1905.00304.pdf) for generating synthetic datasets.

(1) Fidelity to real traffic
   - Real traffic, consistent (not invalid after Cordero et al.)
   - Structural richness on packet level (in contrast to) Induced due to the different levels at which traffic variation is introduced

- Temporal activity levels? (actually not something we improve) We can look at test for realism of distributions (IP discovery, etc)

(2) Ground truth labels through containerisation
- Ground truth for attack behaviour, able to label 100
- Labels for different types of behaviour, reproducable useful for evaluation of model failures, what kind of behaviours cause failure applies to a large range of models also useful for evaluation of privacy infiltration methods, more niche
- Ground truth for label matching between traffic and program logs/sys logs useful for models that try to correlate events for detection this is more niche, but potentially because of the lack of data

(3) Extensive capture
- Packet availability
- Syslogs and for multiple scenarios program logs
- Potentially host logs? Depends if we want to cater to cloud computing applicability

(4) Better for ML-based methods
- Flexibility "The models should allow researchers to generate different classes of data, such as augmenting the amount of data representing sparse events, or choose different topology"
- Automisation of variable datasets through randomisation, automatically create structurally different datasets, but faithful to realism Especially novel in terms of network topologies, should emphasise this in use-cases
- Structural richness allows for learning deeper and more generalisable knowledge in models, less prone to overfitting
- Scalability "Train on as much data as necessary"

3 DESIGN

Explain the general idea and benefit of using containers for traffic generation in comparison to regular testbeds, similar to the existing paper.

. . . . . .
. . . . . .
. . . . . .

3.1 Modes of Operation

(1) Stand-alone scenarios for traffic generation
- Straightforward generation of traffic for the selected service
- Easy to costumize and can be used as prior testing for modifying services in the other two options

(2) Network-wide activity emulation
- A topology is generated randomly with a set of hosts and corresponding services running on them
- Scenarios are started and hooked to corresponding host network interfaces according to a launch script that emulates empirical activity measurements

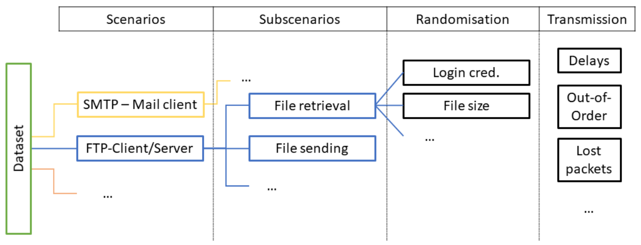

Figure 1: Visualization of the different levels at which traffic variation is introduced in DetGen.

- These activity measurements come in the form of a timeline of events that can be drawn from a set of basic distributions that we provide or can be generated by third party such as the GAN in the Doppelganger paper (this way we can avoid being scrutinised for a lack of realism in this field).

(3) Microservice activity emulation
- In this mode, we set up a VM that hosts a number of service containers and emulates the situation of a typical microservice host
- We run the same scenarios, but the client containers are located on another machine.
- We collect system call logs from both the containers and the host
- This is the least developed mode, but the additionally collected system calls give the most realistic picture in this state
- It might be good to define and describe this setting already if we want to implement any container-specific attacks and models in the future

3.2 Scenarios and subscenarios

Here we describe the design process of the scenarios, just like in the previous paper.

3.2.1 Randomization. . . . . . .

. . . . . .
. . . . . .

3.2.2 Network transmission. . . . . . .

. . . . . .
. . . . . .

3.2.3 Implementation Process. . . . . . .

. . . . . .
. . . . . .

3.2.4 Attack generation with Metasploit/Metasploitable. . . . . . .

. . . . . .
. . . . . .

3.3 Network-emulation

Describe the design of how the topology is generated before the data capture is started, and how the scenarios are launched and stopped during the data capture.

3.3.1 *Topology generation.* ......

......

......

3.3.2 *Launch-script.* ......

......

......

3.3.3 *Dataset coalescence.* ......

......

......

3.3.4 Activity timeline input. Here, we describe how the activity timeline in the launch script can come from unsophisticated distributions that we include, or from sophisticated 3rd party models such as Doppelganger. Since the modelling of computer activity is a field of ongoing research, it seems best to shift the responsibility for realistic models away by allowing this input.

3.4 Microservice-emulation mode

......

......

......

4 FIDELITY CONFIRMATION EXPERIMENTS

This section is important to demonstrate that our data is valid and overcomes the difficulties entailed with synthetic data generation. Cordero et al. have proposed some more simple test that we can refer to first

Question to be answered: What requirements are there for the additional data, program logs and system logs, that we collect? Should we put less emphasise on these data sources in general if we are not able to perform these tests, and refer to them in future work? I am not aware of any papers that discuss these requirements in a similar way.

4.1 Data correctness tests

This section is concerned with dataset defects, artifacts, or invalid data (inconsistent MTU etc.). These are very straightforward to test and should not take up much space.

4.2 Diversity tests

These tests, also from Cordero et al. quantify diversity via the entropy of different quantities such as IP diversity, Time-to-Live, Maximum-segment-size, Window size, ToS. I think we should keep this relatively short and omit comparison to other datasets since this is already done by Cordero et al.

4.3 Structural dataset dimensionality

Autoencoders are often used to compress non-deterministic, noisy data. Bahadur et al. have developed a procedure to estimate the ,,dimensionality" (to be understood as the complexity) of a dataset using variational autoencoders. I believe we can transfer this concept to sequence compression and estimate the overall complexity of connection sequences in our framework with both real traffic captures and existing network intrusion datasets.

Showing that our data is closer to real-world data would be a good test for "artificially predictable patterns", as described by Cordero et al., and go hand-in-hand with demonstrating the benefits of our framework for the training of deep-learning models. The importance of data that is less artificially predictable and closer to real-life traffic in terms of statistical variations lies in its suitability as a benchmark for detection rates, since less complex data is easier to train on and yields unrealistically high detection rates.

4.4 Explorating Artificial Delays

This section is already existing, we could potentially expand this. I think it is sufficient and analysing it more does not add much to the paper as the performance of TC netem is relatively well accepted. I think we could even move this section to the appendix.

5 USE-CASES

5.1 Benefits of ground-truth labels/dynamic dataset generation

Possible title: **Dataset tuning to decrease false-positives**

Extensive ground-truth labels for our activities are arguably the most important contribution of the DetGen framework, so we should highlight their benefit most. Since ground-truth labels on attack data are existing in other datasets, we should emphasise the benefit of having labels for different activities. In my eyes, the most striking benefit arises for false-positive analysis, which we could then combine with showcasing the benefit of being able to generate different amounts of traffic for different activities.

Plan. Implement the LSTM-model in the paper "An LSTM-Based Deep Learning Approach forClassifying Malicious Traffic at the Packet Level", train it on our data (both benign and attack traffic). Extract labels of traffic responsible for false-positives, show how much they are clustered around particular activities (potentially rare activities) compared to the overall traffic. Give potential reason for this. Generate a new dataset with increased amounts of the activities responsible for false positivies. Demonstrate that false-positives decrease.

5.1.1 Benefits of structural richness. Possible title: **Harder benchmarks**

As described above, the importance of data that is less artificially predictable and closer to real-life traffic in terms of statistical variations lies in its suitability as a benchmark for detection rates. In particular, we want to demonstrate that our data functions is a more difficult and realistic benchmark that is less prone to inflating detection rates than existing datasets, something that is often a point of criticism for models evaluated on synthetic data.

To show that the training and detection is harder on our data, we could generate a dataset with similar attacks and services as the CICIDS-17 dataset, and train the above described LSTM model on both datasets. We could show that the training loss goes down more slowly on our data, as well as other metrics (increased validation loss –¿ overfitting etc.). We could then go ahead and show that the same attacks are detected easier by the same model in the CICIDS-17 data than in our data, concluding that it is a less realistic benchmark.

It would be good to also include a comparison with actual real-world traffic here to bolster our conclusion, but due to the lack of structured real-life datasets it is difficult to create a fair and scientific comparison.

5.2 Show utility of flexible topology

This is another possibility to demonstrate that the flexibility provided through containerisation allows for better benchmarking and more in-depth evaluation.

Methods aiming at modelling network structures are an established method for botnet and pivoting detection. Even though the network topology is a crucial variable in their training, their evaluation is to my knowledge only done on single static datasets such as the LANL-15.

My idea is that we generate about 10 datasets with different topologies, (especially different numbers of subnets and servers), and highlight the variation in prediction accuracy, i.e. the accuracy on one dataset is significantly higher/lower, and the average across the different datasets is a better indicator that eliminates the topology as a variable. We do not necessarily need to implement any attack traffic here since the modelling accuracy on benign traffic should be sufficiently quantifiable.

A suitable candidate for the evaluation would be the paper ”Link prediction in dynamic networks using randomdot product graphs”, which comes from people at Imperial College that I know. The model tries to give a probability for each connection to appear in a network, in order to spot connections between unlikely pairs as anomalous behaviour. I could ask the people for the implemented model so we do not have to do it ourselves, or we could even have a chat with them.

5.3 Customisable attack traffic by using metasploitable

Rob and I discussed that by using a combination of a metasploit-attack container and a metasploitable-victim container, we could generate and embed a significantly larger number of attack traffic types more efficiently than we are currently. Since we can attach the metasploitable-container to the network interface of regular containers, we can embed the implemented vulnerabilities very easily in a given scenario. Furthermore, since both containers are well maintained, we can keep the attack catalogue up-to-date.

I think the best use-case is to showcase the process of adding a new type of attack to the dataset, embedding it in a proper way to a given scenario, generating data from it. We could additionally implement a corresponding detection method, but I think this would not demonstrate anything.

5.4 Simple use-case for the microservice mode?

6 CONCLUSIONS

6.1 Difficulties and limitations

6.2 Future work