Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D5.1 Global Model Based on Shared Anonymized Data, First Version (M21)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file:	Deliverable 5.1 Global Model Based on Shared Anonymized Data, First Version
Document version:	1.0
Document owner:	Sebastian Schäfer (RWTH)
Work package:	WP5
Task:	T5.1 Distributed Learning of a Global Model Based on Shared Anonymized Data
Deliverable type:	Report
Delivery month:	M21
Document status:	☒ approved by the document owner for internal review ☒ approved for submission to the EC

Document History:

Version	Author(s)	Date	Summary of changes made
0.1	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH), Alexey Kirichenko (F-Secure), Sivam Pasupathipillai (F-Secure), Tomas Jirsik (MU)	2020-12-18	First version of outline including first bullet points.
0.2	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH), Alexey Kirichenko (F-Secure), Sivam Pasupathipillai (F-Secure), Tomas Jirsik (MU)	2021-01-08	Finalized outline and structure.
0.3	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH), Alexey Kirichenko (F-Secure), Sivam Pasupathipillai (F-Secure), Tomas Jirsik (MU)	2021-01-25	Most sections complete and ready for review.
0.4	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH), Alexey Kirichenko (F-Secure), Sivam Pasupathipillai (F-Secure), Tomas Jirsik (MU)	2021-01-29	Incorporated feedback.
1.0	Sebastian Schäfer (RWTH)	2021-01-29	Final version ready to submit.

Internal review history:

Reviewed by	Date	Summary of comments
Martin Zadnik (CESNET)	2021-01-28	Technical and grammar check.

Executive Summary

This initial deliverable is one of the two deliverables for the task T5.1 Distributed Learning of a global model based on shared anonymized data. Its goal is to learn a global model, similarly as of the tasks T5.2 and T5.3, but using a different sharing approach. Therefore, the deliverables D5.1, D5.3, and D5.3 have some overlaps, especially with respect to background, motivation, and the context within SAPPAN. In general, WP5 focuses on sharing and federation for cyber threat detection and response and this task is one of the three tasks focusing on collaborative learning. In this Initial version of the deliverable, we present the showcases that we work on in the context of building global detection models. We present several showcases and sharing scenarios, including planned experiments and initial results. Most showcases are similar to the ones developed in WP3, namely DGA Detection, application and host profiling, and anomaly detection. In addition to the showcases in WP3, this deliverable also makes use of the anonymization techniques developed in T3.4, which will be used to share the data required to train the global models. However, since this task is still running, additional details regarding anonymization will be included in the deliverable of the task T3.4 as well as in the second version of this deliverable for the task T5.1.

Contents

1 Introduction

This deliverable for the Task T5.1 has similar goals as the tasks T5.2 and T5.3, hence, a large part of its motivation and context is similar to the other tasks. In SAPPAN, one of the innovations is to enable privacy preserving sharing of intrusion detection data, detection models, and response handling information. The objective of sharing this information is to improve the local capabilities of each participating organization by collaboration. Another flavor of sharing in the scope of SAPPAN is sharing among a cybersecurity service provider or vendor and various user groups of its customer organizations. In WP3, one of the tasks is to develop local detection and response mechanisms. In WP5, we want to utilize these mechanisms on the global level to improve them using different sharing mechanisms.

This deliverable focuses on building global detection models based on shared anonymized data. The idea is to collect data, prepare training data for machine learning models on the local level, anonymize it, and share it to the global level. If this is done by multiple parties, this global dataset can be used to build global detection models with increased accuracy or robustness compared to the local models. Throughout the first three tasks of WP5 we focus on the showcases for which we developed local detection mechanisms, as well as some additional ones. These mechanisms include machine learning models, process mining models, as well as statistical models. In the initial version of this deliverable, we describe the experiments and approaches to build global models based on shared local data. For some of the experiments we already have initial results. The final results will be presented in the second version of this Deliverable.

This document is structured as follows. First, we briefly outline the context of the task T5.1 in the overall scheme of the SAPPAN project. Next, we describe the general idea of this task in more detail, discuss privacy for machine learning models, and afterwards describe the different showcases including different approaches for building global detection models. First, we present the showcase of DGA detection including three sharing scenarios and initial results of our experiments. Next, we present our planned experiments for application and host profiling, followed by the showcase of anomalous behavior detection. Finally, we conclude this deliverable with a summary and a plan of our future work.

2 SAPPAN Context

The context for building a global model in SAPPAN is similar for the first three tasks in WP5. Hence, the following paragraph can also be found in the Deliverables D5.3 and D5.5.

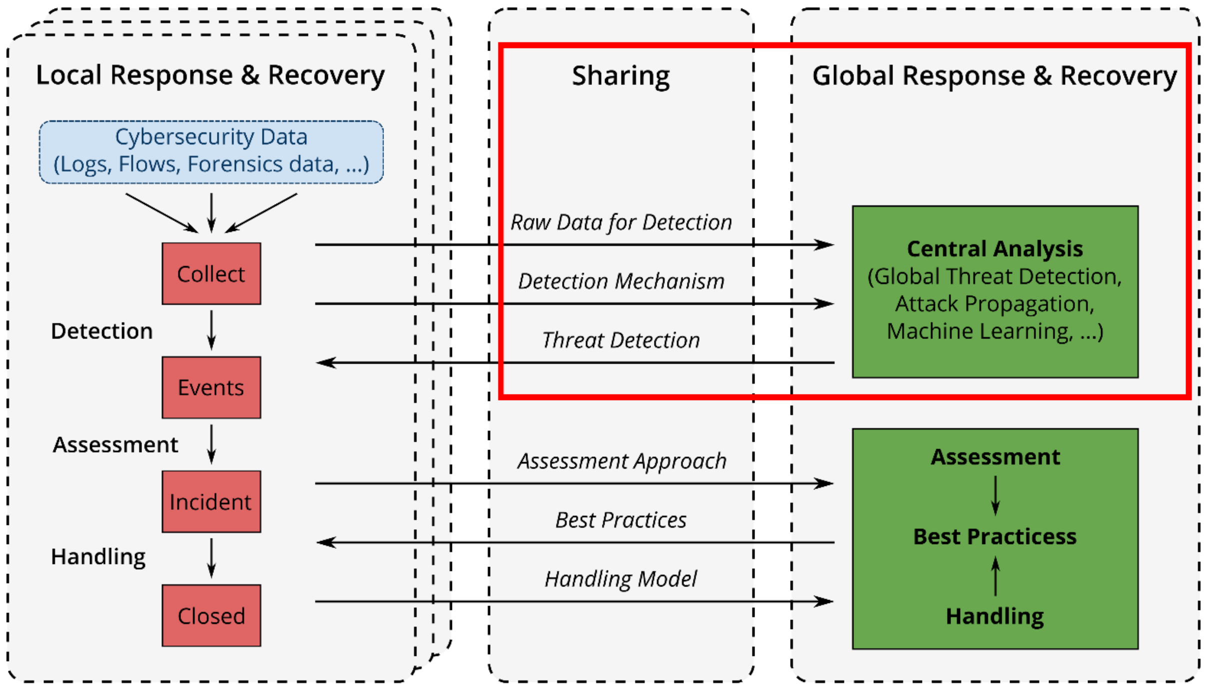


Figure 1: SAPPAN scheme regarding local and global response and recovery.

The overall scheme for sharing, detection, and response in SAPPAN is shown in Figure 1. The top half of the scheme describes the detection components, and the bottom half - the response components, while the left half corresponds to the local level, and the right half - to the global level. The goal of WP5 is to implement the global level with respect to sharing of data and models, building global models for detection, sharing of response and recovery information, as well as supporting visualization. The tasks T5.1, T5.2 and T5.3 include the development of global models for detection, based on several approaches. The general idea is to utilize the data and models developed in the task T3.3 on the local level by sharing them among multiple organizations to build global detection models. The goal is to end up with global detection mechanisms that are superior to the local ones. Another flavor of sharing in the scope of SAPPAN is sharing among a cybersecurity service provider or vendor and various user groups of its customer organizations. In such cases, we aggregate data or local attack detection models built in individual endpoints. Key problems with respect to sharing are, of course, privacy and efficiency, which are tackled by an assortment of approaches investigated in T5.1, T5.2 and T5.3. The approaches range from sharing of anonymized data, to sharing of only pre-trained models, to replacing sharing by other techniques. We apply these techniques to several showcases similar to those described in WP3 in order to build global detection models with adequate recall and precision and providing certain levels of privacy and efficiency, including cost-efficiency. For that, we will use anonymization techniques developed in the task T3.4 based on the privacy requirements described in the task T2.2.

For task T5.1, we develop approaches to building global models based on shared anonymized data. The first challenge is to anonymize the data in a way that eliminates all privacy risks but still preserves enough information to build a useful global model. The second goal is to use the shared data from multiple organizations to create a model superior to the local ones, e.g. with increased accuracy.

3 Distributed Learning of a Global Model Based on Shared Anonymized Data

This deliverable focuses on the creation of a global detection model based on shared anonymized data in order to centralize the detection that is applied at the local level in WP3 to the global level (WP5). To this end, data from several organizations is combined to detect attacks that are not detectable in a single organization alone, while respecting the privacy requirements of individuals and organizations. Leveraging shared local knowledge enables the identification of common attack patterns and properties on a global level. This gained knowledge enables us to reduce the false positive ratio by creating fitting detection models.

In D5.1, we share anonymized data which is the most simple and general form of knowledge distribution compared to the deliverables D5.3 (global model based on shared local models) and D5.5 (global model without sharing local models) which require trained machine learning models for either sharing the model itself or for sharing intelligence derived from a model (e.g. model predictions in the teacher-student setting). Thus, in this deliverable we are able to leverage anonymized data sharing in order to increase the performance of various detection types not just those based on traditional machine learning.

In order to investigate the benefit of private information sharing we make use of the following three use cases which were already defined in D3.4 (Algorithms for Analysis of Cybersecurity Data):

- Domain Generation Algorithm (DGA) Detection
- Application and Host Profiling
- Anomalous Behavior Detection

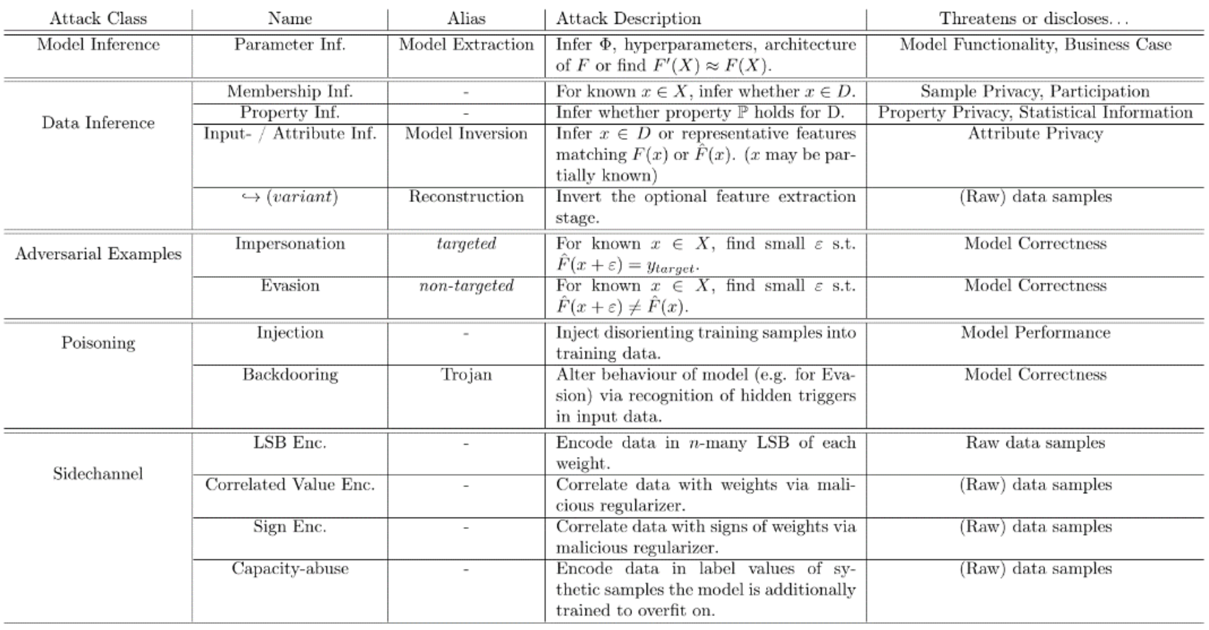
3.1 Privacy

The following briefly describes the landscape of privacy attacks against machine learning, followed by a short discussion about which of these attacks are relevant to this deliverable. The former part will be the same in all three deliverables D5.1, D5.3 and D5.5. Unwillingly disclosing private or sensitive information is in most cases inherent to the useful distribution of knowledge, independent of whether the knowledge is shared in the form of a data set or a decision model. Depending on the sharing scenario and on the form of knowledge, different attacks become feasible. The so-called inference attacks attempt to deduce sensitive information about data sets or models from their statistical characteristics and how the sets and models are processed. These inference attacks are, among others, listed in Figure 2 which describes the attack landscape in the context of machine learning classifiers. We consider the following notation: The parameters Φ of a K-discriminative classifier F with domain X and codomain Y={1,...,K} are trained on a labeled data set D={$(x_i,y_i)_{i=1,...,d}$} as a subset of X × Y drawn from unknown distribution D ~ $\mathcal{D}$. Trained model F represents a function that computes probabilities of class membership as follows:

$$F_\Phi : X \to \Delta^K, x \mapsto y = F_\Phi(x) \in \Delta^K = \{x \in [0,1]^K \mid \mathbf{1}^T x = 1\}$$

or just the predicted class as such:

$$\hat{F}_\Phi : X \to Y, x \mapsto y = \underset{i}{\mathrm{argmax}}\ F_\Phi(x)_i$$



Attack Class	Name	Alias	Attack Description	Threatens or discloses...
Model Inference	Parameter Inf.	Model Extraction	Infer Φ, hyperparameters, architecture of F or find $F'(X) \approx F(X)$.	Model Functionality, Business Case
Data Inference	Membership Inf.	-	For known $x \in X$, infer whether $x \in D$.	Sample Privacy, Participation
	Property Inf.	-	Infer whether property $\mathbb{P}$ holds for D.	Property Privacy, Statistical Information
	Input- / Attribute Inf.	Model Inversion	Infer $x \in D$ or representative features matching $F(x)$ or $\hat{F}(x)$. (x may be partially known)	Attribute Privacy
	$\hookrightarrow$ *(variant)*	Reconstruction	Invert the optional feature extraction stage.	(Raw) data samples
Adversarial Examples	Impersonation	*targeted*	For known $x \in X$, find small ε s.t. $\hat{F}(x+\varepsilon) = y_{target}$.	Model Correctness
	Evasion	*non-targeted*	For known $x \in X$, find small ε s.t. $\hat{F}(x+\varepsilon) \neq \hat{F}(x)$.	Model Correctness
Poisoning	Injection	-	Inject disorienting training samples into training data.	Model Performance
	Backdooring	Trojan	Alter behaviour of model (e.g. for Evasion) via recognition of hidden triggers in input data.	Model Correctness
Sidechannel	LSB Enc.	-	Encode data in n-many LSB of each weight.	Raw data samples
	Correlated Value Enc.	-	Correlate data with weights via malicious regularizer.	(Raw) data samples
	Sign Enc.	-	Correlate data with signs of weights via malicious regularizer.	(Raw) data samples
	Capacity-abuse	-	Encode data in label values of sythetic samples the model is additionally trained to overfit on.	(Raw) data samples

Figure 2: Privacy attack landscape in machine learning.

When restricting the view to the sharing scenarios in this deliverable, which is the sharing of (anonymized) data for classification models, the relevant attack class is the Data Inference class. This especially includes Membership Inference [34] and Property Inference [33] as well as Input Inference attacks [32]. In the former, the participation of an individual sample, or a group of samples, in a data set is to be determined while in the latter either the input into the feature extractor of a feature-based or a deep learning classifier or the input to a classifier model is inverted with the goal to reconstruct samples from shared output data. The other attack classes are not relevant to this deliverable. Especially Poisoning attacks [38] will not be addressed as we consider the honest-but-curious attacker model. Similarly, Model Inference [35], Adversarial Examples [37] and Side channel attacks [36] are not relevant to this deliverable. Anonymization techniques, privacy-preserving training algorithms and other measures allow to shrink the attack surface or the feasibility of an attack. A privacy evaluation is demonstrated on the DGA showcase: For each DGA sharing scenario, we evaluate the privacy leakage by measuring the success of the relevant attacks.

3.2 Domain Generation Algorithm (DGA) Detection

We use the Domain Generation Algorithm (DGA) detection use case to analyze and compare the benefit of private data sharing within the deliverables D5.1 (global model based on shared anonymized data), D5.3 (global model based on shared local models), and D5.5 (global model without sharing local models). In addition, we investigate the privacy implications caused by data sharing for this use case in all three deliverables. As a consequence, the three deliverables share the same texts for sections that include general information such as DGA detection background, state-of-the-art classifiers, or parts of the evaluation setup. However, sections that depend on the different sharing scenarios, such as the actual evaluation and the privacy study, are listed individually for each deliverable.

3.2.1 Background

We presented DGA detection in D3.4 (Algorithms for Analysis of Cybersecurity Data) in detail. Therefore, we only briefly discuss the most important aspects in this deliverable.

Modern botnets rely on DGAs to establish a connection to their command and control (C2) server. In contrast to using individual fixed IP-addresses or fixed domain names, the communication attempt of DGA-based malware is harder to block as such a malware generates a vast amount of algorithmically generated domains (AGDs). The botnet herder is aware of the generation scheme and thus is able to register a small subset of the generated domains in advance. The bots, however, query all generated AGDs, trying to obtain the valid IP-address for their C2 server. As most of the queried domains is not registered, the queries result in non-existent domain (NXD) responses. Only the domains that are registered by the botnet herder in advance resolve successfully to a valid IP-address of the C2 server.

The occurring NXDs within a network that are caused by the non-resolvable queries can be analyzed in order to detect DGA activities and thereby to take appropriate countermeasures even before the bots can be commanded to participate in any malicious action. This detection is, however, not trivial, since NXDs can also be the product of typing errors, misconfigured or outdated software, or the intentional misuse of the DNS e.g. by antivirus software. In the following, we refer to this detection in which we separate benign from malicious domain names as the DGA binary classification task.

In addition to this binary classification task, it is useful to not only detect malicious network activities but also to attribute the malicious AGDs to the specific DGAs that generated the domain names. This enables the malware family used to be narrowed down and targeted remediation measures to be taken. In the following, we refer to this classification as the DGA multiclass classification task.

In the past, several approaches have been proposed to detect DGA activities within networks. These approaches can be split into two groups: contextless and context-aware approaches. In SAPPAN, we focus on contextless approaches (e.g. [2, 3, 4, 5, 6, 7]), as they entirely rely on information that can be extracted from a single domain name for classification. Thereby, they are less resource intensive and less privacy invasive than context-aware approaches (e.g. [8, 9, 10, 11, 12, 13]) that depend on the extensive tracking of DNS traffic. Even though the classification of the contextless approaches relies solely on the domain name, they are able to compete with the context-aware approaches and achieve state-of-the-art performance [2, 3, 5, 6, 7].

A variety of different types of machine learning techniques have been proposed for the classification of domain names which can be divided into two groups: feature-based classifiers (e.g. [3, 8]) and deep learning (featureless) classifiers (e.g. [2, 5, 6, 7]). While the deep learning classifiers outperform the feature-based approaches in terms of classification performance [5, 6, 14, 15, 16], their predictions cannot be explained easily. For example, the predictions of a decision tree can easily be traced back to the individual features used to classify a domain name. Such a simple explanation is not possible for the predictions of a deep learning model. However, feature-based approaches rely on specific features that are hand-crafted using domain knowledge. The engineering of these features requires much more effort compared to the usage of deep learning classifiers where all important information has to be encoded and provided to the model. Moreover, after the feature engineering the best combination of features has to be selected which is not a trivial task.

While the feature-based and deep learning based approaches differ in their classification capabilities, they might also provide different privacy guarantees when trained on shared private data. Thus, we evaluate and compare feature-based as well as deep learning based approaches.

In our evaluation, we include classifiers which were developed within the SAPPAN project. In detail, we include the two ResNet-based classifiers [17] that we introduced in deliverable D3.4 (Algorithms for Analysis of Cybersecurity Data). There, we demonstrated that our classifiers achieve better classification scores (f1-score/false positive rate) than the state-of-the-art classifiers described in related work. Note, to counteract the explainability problem of deep learning classifiers we developed a visual analytics system [18] in SAPPAN which tries to bridge the gap between the predictions of deep neural networks and human understandable features.

3.2.2 Selected State-of-the-Art Classifiers

In the following, we present several state-of-the-art classifiers which we use in different sharing scenarios to (1) measure the benefit of private data sharing in terms of classification performance and (2) analyze the provided level of privacy of the collaboratively trained classifier.

First, we present the currently best contextless feature-based approach for DGA binary classification. We then continue with different types of deep learning classifiers including convolutional (CNNs), recurrent (RNNs), and residual neural networks (ResNets).

FANCI

Schüppen et al. [3] proposed a system called Feature-based Automated NXDomain Classification and Intelligence (FANCI). It is capable of separating benign from malicious domain names. FANCI implements an SVM and an RF-based classifier and makes use of 12 structural, 7 linguistic, and 22 statistical features for DGA binary classification. The authors of FANCI state that it uses 21 features, but feature #20 is a vector of 21 values, resulting in 41 values in total. The 41 features are extracted solely from the domain name that is to be classified. Thus, FANCI works completely context-less. FANCI does not incorporate DGA multiclass classification support.

Endgame

Woodbridge et al. [6] proposed two RNN-based classifiers for the DGA binary and multiclass classification. Both classifiers incorporate an embedding layer, a long short-term memory (LSTM) layer consisting of 128 hidden units with hyperbolic tangent activation, and a final output layer. The last layer of the binary classifier is composed of a single output node with sigmoid activation while the last layer of the multiclass classifier consists of as many nodes as DGA families are present. We denote the binary classifier by B-Endgame and the multiclass classifier by M-Endgame in the following.

NYU

Yu et al. [7] proposed a DGA binary classifier that is based on two stacked one-dimensional convolutional layers with 128 filters for DGA binary classification. We refer to this model as B-NYU in the following. We additionally adapted the binary model to a multiclass classifier by interchanging the last layer similarly to the M-Endgame model. Additionally, we use Adam [19] as optimization algorithm and the categorical cross-entropy for computing the loss during training. We refer to the multiclass enabled model as M-NYU in the following.

ResNet

In the context of SAPPAN we developed binary and a multiclass DGA classifier based on ResNets [17]. We presented all details as well as a comparative evaluation with the state-of-the-art in deliverable D3.4 (Algorithms for Analysis of Cybersecurity Data). ResNets make use of so-called skip connections between convolutional layers which build up residual blocks. These blocks allow the gradient to bypass layers unaltered during the training of a classifier and thereby effectively mitigate the vanishing gradient problem [20, 21]. Our proposed binary classifier, B-ResNet, consists of a single residual block with 128 filters per convolutional layer while our proposed multiclass classifier M-ResNet has a more complex architecture of eleven residual blocks and 256 filters per layer.

Class weighting

Tran et al. [5] showed that the model of Woodbridge et al. [6] is prone to class imbalances which reduce the overall classification performance of the DGA multiclass classifier. The authors mitigate the effect of class imbalances by using the proposed class weighting:

$$C_i = \left(\frac{total\ number\ of\ samples}{number\ of\ samples\ in\ class\ i}\right)^{\gamma}$$

The class weights control the magnitude of the weight updates during the training of a classifier. The rebalancing parameter γ denotes how much the dataset should be rebalanced. Setting γ = 0 makes the model behave cost-insensitive, setting γ = 1 makes the classifier treat every class equally regardless of the actual amount of samples per class included in the training set. Tran et al. empirically determined that γ = 0.3 works well for DGA multiclass classification. In all our experiments we thus use γ = 0.3 when working with cost-sensitive models. We denote deep learning models which incorporate class weighting with the suffix ".MI".

3.2.3 Evaluation Setup

The main goals of our evaluation are (1) to determine whether we can improve the classification performance by leveraging different approaches for private information sharing, (2) to quantify the level of privacy after enabling privacy-preserving techniques, and (3) to quantify the loss in utility after enabling privacy-preserving techniques.

For the deliverables D5.1, D5.3, and D5.5 we use the same evaluation setup (i.e. the same classifiers and datasets) in order to guarantee comparability of different information sharing scenarios for the use case of DGA detection.

Data Sources

In total, we use five different data sources, four for obtaining benign data and one for malicious data.

Malicious data

We obtain malicious domains from the open-source intelligence feed of DGArchive [22] which contains more than 126 million unique domains generated by 94 different known DGAs. We make use of all available data up to 2020-09-01.

Benign data

We obtain benign labeled NXDs from three different sources, namely from networks of CESNET, Masaryk University, and RWTH Aachen University. For data obtained from each of these sources we perform a simple pre-processing step in which we remove all duplicates, cast every domain name to lowercase (as the DNS operates case-insensitive), and filter against our malicious data obtained from DGArchive to clean the data as far as possible.

Additionally, we remove the intersection of all obtained samples from two of our benign data sources, namely from CESNET and Masaryk University. The reason for this is that the networks of both parties are interconnected and the recording period for data collection overlaps. Note, thereby we are also removing samples from both data sources which would naturally be present in both networks even when they were not interconnected. Such samples could be common typos of popular websites. This issue could have an effect on classification performance of classifier when samples of these networks are used for training or classification. However, since we record NXDs, we filter significantly fewer samples than if we were to record resolving DNS traffic. Thus, this effect could only have a negligible influence on the classification performance, but this has yet to be investigated. In the following we list the recording periods and the amount of unique samples obtained from each source for benign data.

- **CESNET** – Recording from 2020-06-15, 361995 samples
- **Masaryk University** - One-month recording (2020-05-15 - 2020-06-15), 7973807 samples
- **RWTH Aachen University** – One-month recording of September 2019, 26008295 samples

3.2.4 Sharing Scenarios

In each deliverable (D5.1, D5.3, D5.5), we investigate different sharing scenarios for the use case of DGA detection. In D3.6 (Cybersecurity Data Abstraction), the set of benign training samples has been identified as the main privacy-critical aspect of this use case. Thus, we focus in the following on the sharing of private benign labeled data.

In context of WP5, we started evaluating collaboratively trained global models without sharing anonymized data or local models (D5.5). Thus, we are able to present initial evaluations in D5.5. For this Deliverable D5.1 we developed three different sharing scenarios which we will evaluate in the final version of this deliverable. We present the three different sharing scenarios that we plan to evaluate in the following:

1. Sharing of anonymized domain names using the URL generalization technique developed in D3.6

As a part of the deliverable D3.6 (Cybersecurity Data Abstractions - Initial Version), a Python commandline tool has been developed to transform URLs into abstracted pseudo-URLs. These resulting abstractions of URLs do not contain privacy-critical information anymore, but are still suitable to be used as training data for machine learning, as shown by the results of experiments that have been presented in D3.6. The scope of D3.6, however, was set to sharing of trained classifiers. For an attacker, this means that there are two complicating factors for the extraction of privacy-critical information: The abstraction of the URLs and the extraction of training data from the classifier. Since this deliverable considers sharing of training data, the second factor does not apply anymore. We consequently solely rely on the abstraction of training data.

2. Sharing of extracted features of feature-based approach FANCI

In this scenario, we utilize FANCI's feature extraction method as a form of data anonymization. This type of data anonymization will have no impact on the classification performance of feature-based classifiers compared to the models which were trained using non-anonymized data (i.e. domain names in cleartext). Other anonymization techniques might have a negative impact on the classification performance. However, it is unclear whether feature extraction is an appropriate approach for data anonymization. Thus, in this deliverable we perform an extensive study investigating the privacy implications of using this approach.

3. Sharing of first-n-layers (feature extractor) of deep learning classifiers

Based on the previous scenario, we designed an approach for sharing extracted features in the context of deep learning based classifiers. We assume that the first-n-layers of a deep neural network are used as a sort of feature extractor while the rear layers are used for the actual classification.

In Figure 3 we display such a layer partition.

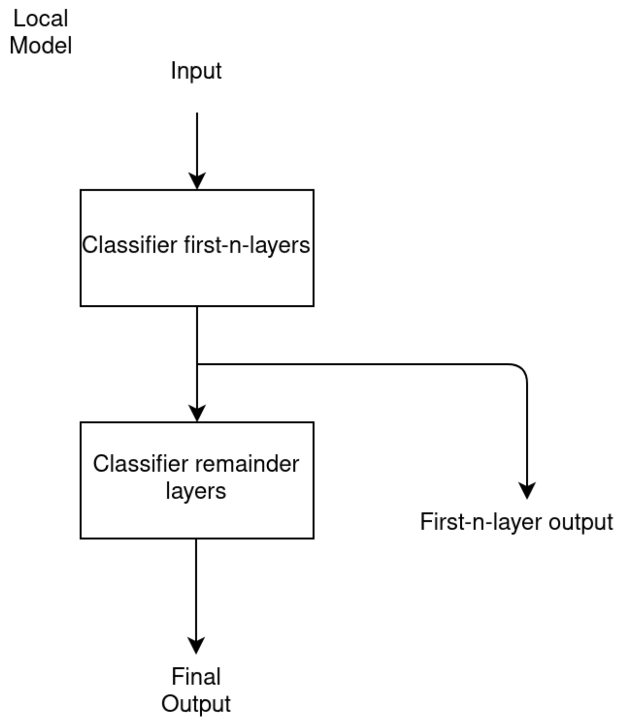


Figure 3: Partition of a neural network classifier in feature-extractor and classification layers.

Every party that is participating in the collaborative training of a classifier in this scenario trains a local DGA detection model using their own private data. Thereafter, each party is able to share either the feature extractor (i.e. the classifier's first-n-layers) or data that is anonymized by feeding domain names to the feature extractor.

Depending on which information is shared a different classification performance and privacy guarantees can be achieved. In this deliverable we will investigate which sharing scenario makes the most sense in terms of classification performance and privacy.

A possible approach for combining the feature extractors of different neural networks belonging to different parties can be seen in Figure 4.

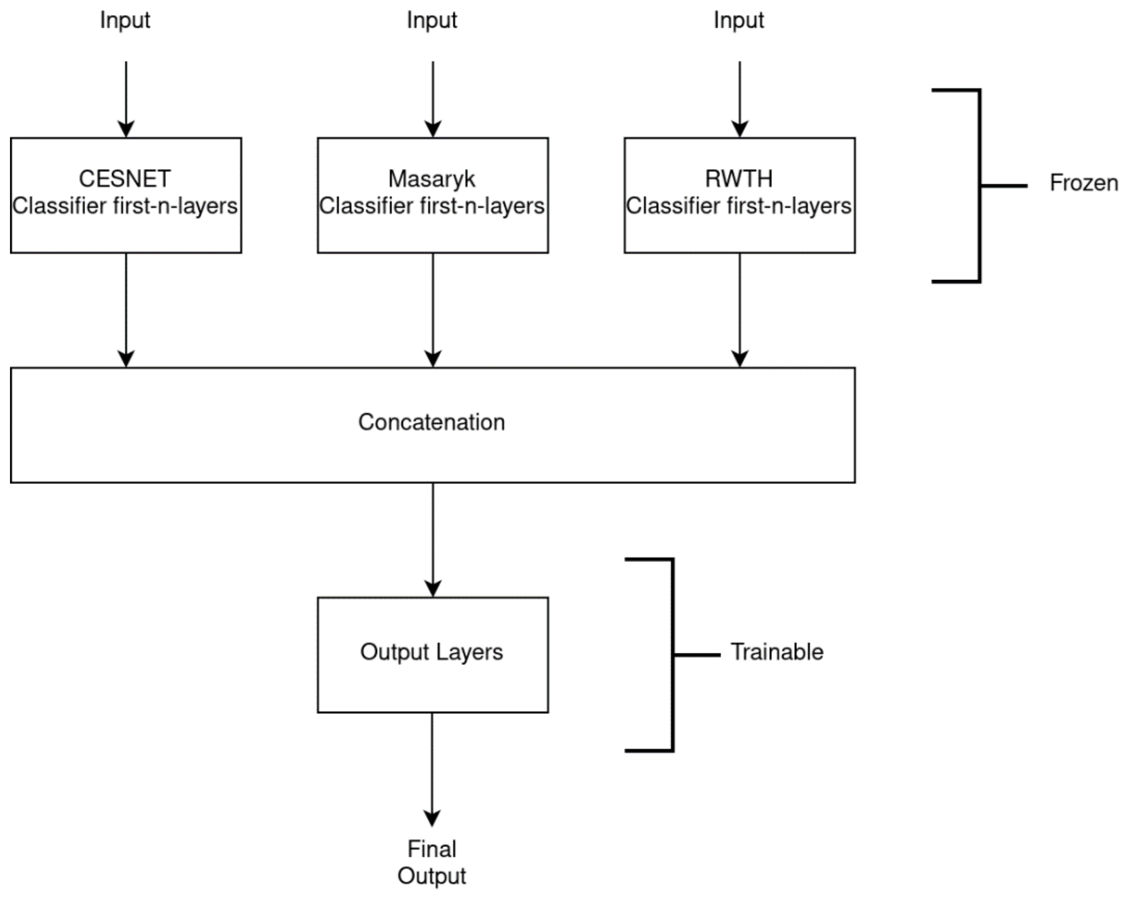

Input
Input
Input
CESNET Classifier first-n-layers
Masaryk Classifier first-n-layers
RWTH Classifier first-n-layers
Frozen
Concatenation
Output Layers
Trainable
Final Output

Figure 4: Approach for combining feature extractors of different neural networks and parties.

3.2.5 Privacy Analysis

Quantifying the success any of the Input Inference attacks requires a distance (or similarity) metric on the space of strings which is a superset of the domain space. Here, we restrict ourselves to the family of edit distances, that each compute a minimum-change distance between two strings by considering character insertions, deletions, substitutions and sometimes the special case of (adjacent) character transpositions. Table 1 lists common edit distances and gives an overview of their properties.

Edit Distance	**Substitution**	**Insertion**	**Deletion**	**Transposition**	**Drawbacks**
Hamming	Yes	No	No	No	Only for strings of equal length
Jaro-Winkler	No	No	No	Yes	Does not fulfil triangle inequality
Longest-Common-Subsequence	No	Yes	Yes	No	No substitution
Levenshtein	Yes	Yes	Yes	No	-
Damerau-Levenshtein	Yes	Yes	Yes	Yes (adjacent)	-

Table 1: Overview of edit distances.

As reconstructions may differ in length, the Hamming distance is not suitable here. The Jaro-Winkler distance does not fulfill the triangle-inequality, hence it is by definition not a mathematical metric. The (Damerau-)Levenshtein distance seems a better choice than the Longest-Common-Subsequence metric because the latter does not consider character substitutions. The Levenshtein distance computes the minimum amount of edit operations (insertion, deletion, substitution) to transform one string into the other. The continued Damerau-Levenshtein distance also respects the case in which two adjacent characters are transposed instead of accounting two character substitution edits.

Sharing of Anonymized Domain Names using the URL Generalization Technique Developed in D3.6

Since the privacy guarantee provided by the abstraction of URL elements except for the domain is very intuitive, it is not considered further. For the domain, however, the privacy-guarantee is highly dependent on the chosen parameters. To get a better impression of the suitability of chosen parameters, the commandline tool has been extended to include methods which perform attacks on abstracted domains. These methods allow to get an impression of the expected computation time and accuracy of attacks, and hence, provide a way to test the suitability of chosen parameters.

In the context of sharing anonymized domain names, the tool generates bitstrings for the domain names in a similarity-preserving way, utilizing an approach based on Bloom filters that originally comes from the area of privacy-preserving record linkage [1]**.** Details can be found in D3.6. For the transformation of domains into the Bloom filters, two parameters are of crucial importance: The size of the Bloom filter and the amount of hash functions used. The combination of both determines the proportion of bits that are set in the Bloom filter, sensitive to the length of the input domain. For the chosen approach, sparse Bloom filters (very low proportion of ones) do not provide good privacy-guarantees, whole too dense ones (very high proportion of ones) have a negative impact on the classifiers training on them. The tool allows to test such configurations and to find sweet-spots in the parameter assignment. For example, it can be used to identify suitable intervals of domain lengths, for which different parameter values are then chosen. It is to be noted that overlaps of ranges should be avoided to preserve utility of machine learning models trained on the abstracted domains, i.e., for each interval of domain lengths with different parameters, the length of the resulting Bloom filter should change as well.

Sharing of Extracted Features of Feature-based Approach FANCI

The FANCI classifier is trained only on labeled domain samples and is agnostic about the context of the real-world DNS connection being made. Additionally, it uses only a small amount of distinct features (45) to solve the binary classification problem of distinguishing benign and malicious domains. Thus, FANCI is highly capable while being frugal towards its required input, which immediately relates to FANCI's attack surface. Due to this small attack surface, FANCI (or its feature extractor respectively) is chosen to be evaluated here as a representative for feature-based approaches in the context of sharing data for DGA detection.

FANCI consists of (1) a feature extractor which computes the 45 relevant statistical and structural features from domain names and (2) the classifier which operates on these feature vectors. FANCI feature vectors offer a reduced/compressed representation of the domain samples and are, in the sharing scenario, accompanied by the respective binary labels of each sample. The main goal of this privacy analysis is to evaluate whether the FANCI feature extractor guarantees sufficient privacy, i.e., whether or not it is possible to draw inferences from a FANCI feature vector about the actual domain name, or from a batch of feature vectors about the membership of certain domain names respectively.

Threats

An adversary with access to the public FANCI feature extractor FE and a shared batch of FANCI feature vectors, including the corresponding labels, may be capable of executing the following attacks:

- Reconstruction: Determine which domain name sample x is likely represented by a certain feature vector FE(x)
- (Group) Membership Inference: There are different groups inside the set of benign samples for which membership could reveal information about software usage on the side of the owner of the shared data set.

For the Reconstruction attack, a (1) manual and a (2) deep learning reconstruction approach are investigated, that each attempt inverts the functionality of the FANCI feature extraction. This shall give insight whether sharing of FANCI features can be considered with sufficient privacy preservation.

FANCI Feature Extractor and Feature Vectors

The FANCI feature extractor is publicly available, since it is required for public use. The official implementation of the FANCI feature extractor, as given by the source in the respective paper, computes a vector with 45 statistical and structural features for each domain name. FANCI feature vectors offer a reduced/compressed representation of the domain samples. Table 2 gives a description of the 45 features on the toy-example domain "dga.b0tn3t.co.uk", or details refer to [3].

#	Feature Name	Example Value	Data Type	Choices	Normalization Factor
0	length	16.0	int	253	253
1	1_part	0.0	bool	2	1
2	2_part	1.0	bool	2	1
3	3_part	0.0	bool	2	1
4	4_part	0.0	bool	2	1
5	vowel_ratio	0.1428	float	DSF-LEN+1	1
6	digit_ratio	0.2222	float	DSF-LEN+1	1

7	contains_ipv4_addr	0.0	bool	2	1
8	contains_digits	1.0	bool	2	1
9	has_valid_tld	1.0	bool	2	1
10	contains_one_char_subdomains	0.0	bool	2	1
11	contains_wwwdot	0.0	bool	2	1
12	subdomain_lengths_mean	4.5	float	DSF-LEN	253
13	prefix_repetition	0.0	bool	2	1
14	char_diversity	0.8888	float	DSF-LEN	1
15	contains_subdomain_of_only_digits	0.0	bool	2	1
16	contains_tld_as_infix	0.0	bool	2	1
17	1_gram_std	0.3307	float	FIXED	1_gram_max
18	1_gram_median	1.0	float	A	1_gram_max
19	1_gram_mean	1.125	float	FIXED	1_gram_max
20	1_gram_min	1.0	float	DSF-LEN+1	1_gram_max
21	1_gram_max	2.0	float	DSF-LEN	1_gram_max
22	1_gram_perc_25	1.0	float	A	1_gram_max
23	1_gram_perc_75	1.0	float	A	1_gram_max
24	2_gram_std	0.0	float	FIXED	2_gram_max
25	2_gram_median	1.0	float	A	2_gram_max
26	2_gram_mean	1.0	float	FIXED	2_gram_max
27	2_gram_min	1.0	float	DSF-LEN+1	2_gram_max
28	2_gram_max	1.0	float	DSF-LEN	2_gram_max
29	2_gram_perc_25	1.0	float	A	2_gram_max
30	2_gram_perc_75	1.0	float	A	2_gram_max
31	3_gram_std	0.0	float	FIXED	3_gram_max
32	3_gram_median	1.0	float	A	3_gram_max
33	3_gram_mean	1.0	float	FIXED	3_gram_max

34	3_gram_min	1.0	float	DSF-LEN+1	3_gram_max
35	3_gram_max	1.0	float	DSF-LEN	3_gram_max
36	3_gram_perc_25	1.0	float	A	3_gram_max
37	3_gram_perc_75	1.0	float	A	3_gram_max
38	hex_part_ratio	0.0	float	DSF-LEN+1	1
39	underscore_ratio	0.0	float	DSF-LEN+1	1
40	alphabet_size	8.0	int	A=38	38
41	shannon_entropy	2.9477	float	~194	$\log_2$(alphabet_size)
42	ratio_of_repeated_chars	0.125	float	A+1	1
43	consecutive_consonant_ratio	0.4444	float	DSF-LEN+1	1
44	consecutive_digits_ratio	0.0	float	DSF-LEN+1	1

Table 2: Features extracted by the open source implementation of the FANCI feature ex-tractor. Example values are given for example domain 'dga.b0tn3t.co.uk'. Choices describes the amount of unique values a feature can take on. Normalization factor is the value used to normalize the feature value to the range [0, 1].

The original work about FANCI considers domains with a maximum length of 253 characters (in accordance with RFC 1035 [23]) and a set of 39 valid characters ('a-z' + '0-9' + '.-_'). Although the character '_' is not legitimate according to RFC 1035, it appears in some malicious domains samples. FANCI extracts features from the D(ot-and)-S(uffix)-F(ree), short DSF, part of the domain name which excludes the top-level domain and all dots between subdomains. The length of the DSF (DSF-LEN) can easily be reconstructed by the following observation: `DSF-LEN = alphabet_size / char_diversity = alphabet_size / (alphabet_size / DSF-LEN)`. The alphabet size for the DSF is A=38, as the dot is already excluded.

To comprehend the complexity- and dimensionality reduction of the feature extractor, the following plot in Figure 5 displays slight overestimations of the in- and output space sizes of the feature extractor for increasing domain lengths. The input space is the space of domain names and the output space the of possible feature vectors for that length. The reduction factor (red line) displays the size relation between the two spaces, more specifically by how much the input space is larger than the output space. For numerical reasons the plot ends early at 160 characters. While the output space size saturates, the input space grows exponentially with increasing domain length, as does the reduction factor.

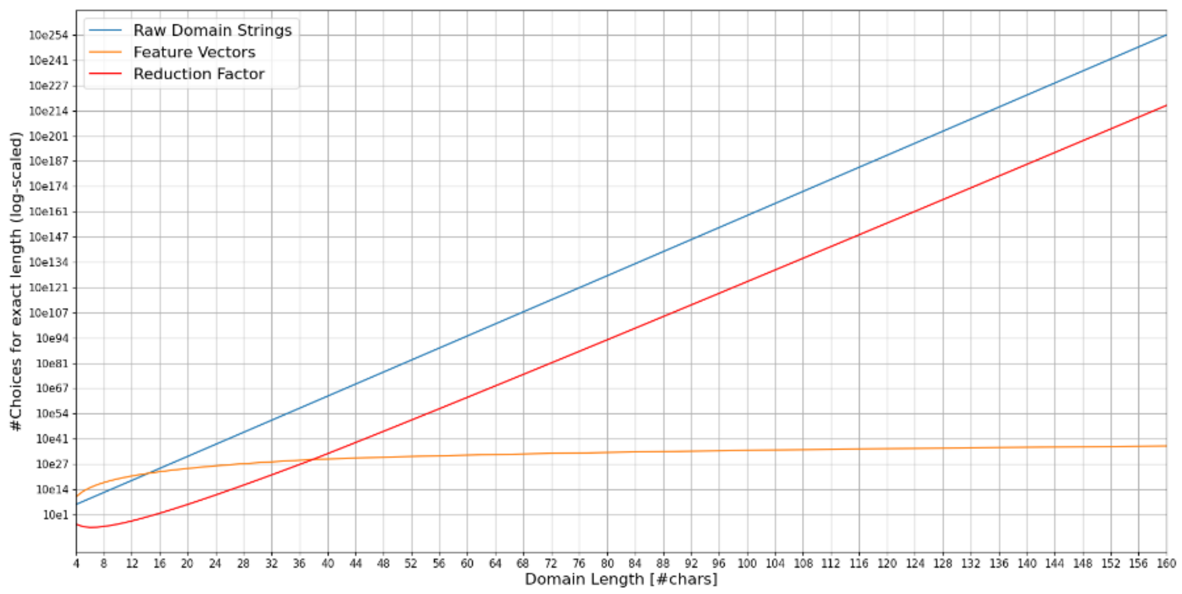


Figure 5: An overestimation of the sizes of both input and output spaces of the FANCI feature extractor over varying length of the original domain name. Red plot de-scribes the reduction, i.e., by how much the input space is larger than the out-put space.

The overestimation of both space sizes is performed as follows. Under the assumption of a minimum TLD of length 2 a separating dot and a minimum DSF-LEN of 1, the minimum domain length theoretically is 4 characters long. With 39 characters and a domain length 4 ≤ L ≤ 253, the domain name space size is given by: 39^L. The size of the feature vector space for a fixed length L can be computed as the multiplication of the amount of possible value choices for each feature. The maximum DSF-LEN can be overestimated as L-3 when ignoring the separating dots. Boolean and integer features have a fix amount of values they can take on. All other floating-point features are computed as fraction of the integer alphabet size A or DSF-LEN, sometimes including the value 0 in the numerator, thus these features can take on as many values as the value of A, or DSF-LEN respectively. For simplicity, A is always set to its maximum value of 39. The standard deviation and mean characteristics of the uni-, bi- and tri-grams distributions are fixed, as they are considered to be determined by the other statistical features: The standard deviation may follow the range rule estimation (std = (max-min)/4) and the mean value will be close to `DSF-LEN/A`. The choices for the Shannon entropy bases on the distribution of discrete probabilities for all characters, thus the amount of unique discrete distributions for k characters and a total of n > k occurrences determine the amount of values the Shannon entropy can take. For 0 ≤ n ≤ 253 and 0 ≤ k ≤ 38, the Shannon entropy can take 194 different values on average.

<u>Manual Reconstruction</u>

We first attempt to reconstruct a domain name from its feature vector by investigating whether the combination of available FANCI features leaks new information that can be used to determine the original domain with more certainty.

Inferred Features

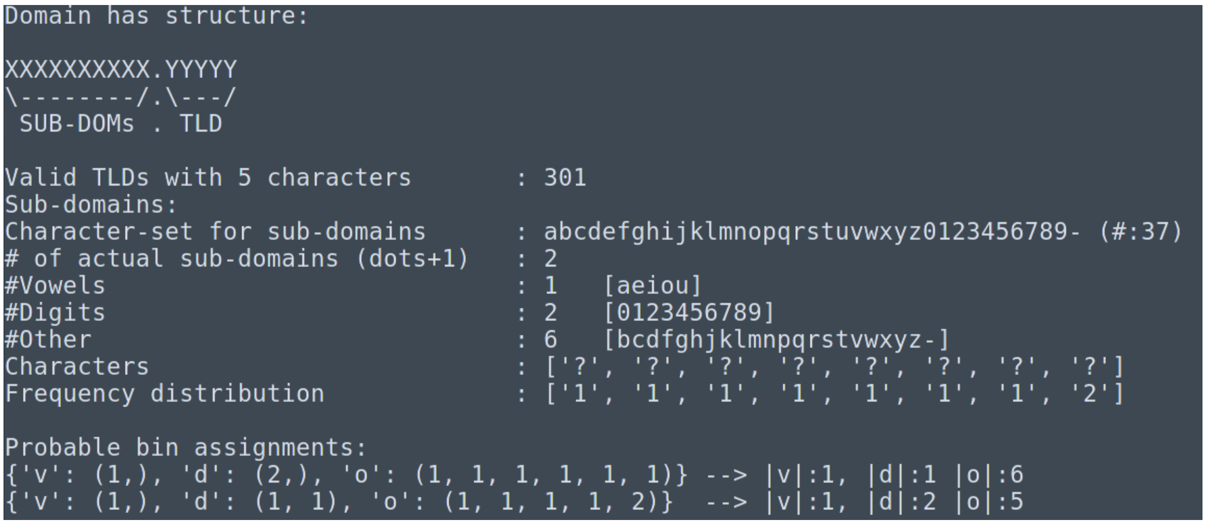
Table 3 demonstrates some new information can be inferred (or reconstructed) by combination of existing features. The information is new in the sense, that no feature in the original feature vector solely holds this information. The extraction of this new information is not bound to any special subspace of the domain name space.

#	Inferred Feature	Computation	Note
1	DSF length	DSF-LEN = *alphabet_size* / *char_diversity*	-
2	Actual amount of sub-domains	ACTUAL-PARTS = DSF-LEN / *subdo-main_lengths_mean*	Feature *X_part* caps the maximum amount of subdomain in the one-hot encoding at 4, this can be cor-rected.
3	TLD length	TLD-LEN = *length* - (DSF-LEN + ACTUAL-PARTS)	Amount of separating dots ~ AC-TUAL-PARTS.
4	Alphabet	Made up out of 'a-z' plus additional '0-9' or '_-' depending if features *contains_digits* > 0.0 or *underscore_ratio* > 0.0	-
5-1	Absolute digit count	DIGITS = *digit_ratio* * DSF-LEN	-
5-2	Absolute vowel count	VOWELS = *vowel_ratio* * (DSF-LEN - DIGITS)	Feature *vowel_ratio* is computed w.r.t. total alpha-chars in DSF that can be estimated by DSF-LEN - DIGITS.
5-3	Absolute other count	OTHER = DSF-LEN - (DIGITS + VOWELS)	-
6	Fre-quency distribu-tion	Brute-force discrete frequency distribution F over unknown $\{x_1,...,x_n\}$ with known *shan-non_entropy* and n = *alphabet_size*	Shannon entropy is computed as weighted sum of character fre-quencies. Restriction: $sum(\{x_1,...,x_n\})$ = DSF-LEN and $x_i \geq 1$ for all i in {1,...,n}.
7	Amount of unique charac-ters	Match vowel, digits and other amounts to bins in discrete frequency distribution F, i.e., find par-tition F1 v F2 v F3 = F with sum(F1) = VOWELS, sum(F2) = DIGITS and sum(F3) = OTHER	-

Table 3: List of new features inferred (in capital letters) from the available FANCI features (italics).

We briefly demonstrate the reconstruction of additional information of our short toy-example domain "dga.b0tn3t.co.uk" in Figure 6. The original feature vector for this do-main can be viewed in Table 2.

After reconstructing the character frequency distribution and the probably unique counts for each vowels, digits and other characters, there is at the best of our knowledge no more information that can further reveal structural information about the original domain name. Further, there is also no feature that gives information about which characters of the groups vowels, digits and other are actually included in the domain and at what character position they would occur. Without such information, no further manual inference can take place.



```
Domain has structure:

XXXXXXXXXX.YYYYY
\--------/.\---/
 SUB-DOMs . TLD

Valid TLDs with 5 characters       : 301
Sub-domains:
Character-set for sub-domains      : abcdefghijklmnopqrstuvwxyz0123456789- (#:37)
# of actual sub-domains (dots+1)   : 2
#Vowels                            : 1   [aeiou]
#Digits                            : 2   [0123456789]
#Other                             : 6   [bcdfghjklmnpqrstvwxyz-]
Characters                         : ['?', '?', '?', '?', '?', '?', '?', '?']
Frequency distribution             : ['1', '1', '1', '1', '1', '1', '1', '2']

Probable bin assignments:
{'v': (1,), 'd': (2,), 'o': (1, 1, 1, 1, 1, 1)} --> |v|:1, |d|:1 |o|:6
{'v': (1,), 'd': (1, 1), 'o': (1, 1, 1, 1, 2)}  --> |v|:1, |d|:2 |o|:5
```

Figure 6: Inferred features for example domain 'dga.b0tn3t.co.uk'.

Evaluation

Privacy leakage can be measured by how far a new inferred information decreases the amount of pre-images that map to this feature vector. The naive brute-forcing of all domains that map to a fixed feature vector would require iterating 39^{length} many domains, since there are 39 domain characters. recognized by the FANCI feature extractor. A cleverer approach would reduce the amount of domain pre-images by using the insights from the newly inferred features. From a combinatorial point of view, the following

1. With a fixed TLD length, the amount of choices *T* for the TLD can be estimated by the public valid-TLD list that is shipped with the feature extractor.
2. The character frequency distribution for the sub-domains is uniquely determined.
3. For every of the *U* settings of unique chars (per each of the groups vowels, digits and other):
    - The choice of *K* unique chars from a *N*-large set is (N choose K)
    - For each character-set of size *N*:
        - the possible amount of sub-domains of length *L* are N^L
        - and for each DSF there are (L-1 choose ACTUAL-PARTS-1) possibilities to insert the separating dots.

In conclusion, the remaining amount of possible pre-image can be calculated by:

$$T \cdot \sum_{U} \binom{5}{\text{UNIQUE-VOWELS}} \cdot \binom{10}{\text{UNIQUE-DIGITS}} \cdot \binom{23}{\text{UNIQUE-OTHER}} \cdot N^{L} \cdot \binom{L-1}{\text{ACTUAL-PARTS}-1}$$

This formula still gives a large amount of valid pre-images and iterating over these pre-images is still very time consuming. Privacy holds in this case, as the amount of valid pre-images is large and the selecting the correct one is close to impossible. We demonstrate this on the short toy-example domain where the above formula evaluates to:

$$351 \cdot (\binom{5}{1} \cdot \binom{10}{1} \cdot \binom{23}{6} \cdot 8^{16} \cdot \binom{15}{1} + \binom{5}{1} \cdot \binom{10}{2} \cdot \binom{23}{5} \cdot 8^{16} \cdot \binom{15}{1}) = 1.8699\text{e}25$$

This only reduces the pre-image space by a factor of 0.6528 when comparing this to the naive approach with 39^{16} = 2.8644e25 many pre-images.

<u>Deep Learning Reconstruction</u>

To avoid manual reconstruction pitfalls and to detach from viewing linear relationships only, a deep learning based generative decoder model is trained with the task of mapping feature vectors back to domain names. In this case, a secondary data set is used, of which the samples are fed through the FANCI feature extractor and subsequently, batches of features vectors and original domains are used to train the decoder network. The goal is that the decoder network is trained to learn the inverse mapping that in reality likely acts on a small subspace of the domain and feature space. However, the model has to be trained with a different data set and with the assumption that the data's characteristics are transferable to another set of benign NXdomain samples.

Architecture and Training

To reconstruct domains as character sequences of variables length from vectors of fixed length, an approach based on Sequence to Sequence (Seq2Seq) autoencoders is constructed [25]. For training the model, feature vectors are normalized (according to the normalization factors in Table X) and fed into multiple parallel layers of same-size dense layers to enable the network to learn a better suited representation. The output of the two parallel networks is interpreted as initial state inputs into an LSTM layer. The LSTM is trained with teacher forcing [26, 27], to predict the next character based on the current string and the provided state. Teacher forcing is an effective method of training a recurrent neural network model that (re-)uses the output from prior time steps as input to the model. Inference is started by feeding in the normalized feature vector into the dense layers and a string with a start marker into the LSTM. During inference, each character is predicted by a single inference run of the model on the current partial string, starting with an empty string. Between each run, the new character is appended to the current input. Similarly, the state of the LSTM layer is preserved in each of the inference runs. Inference stops when a string of maximum length (253) is generated or whenever the end marker is predicted. Including the start and end markers, the available characters can be encoded as 41 integers. In this study, the architecture of the Seq2Seq decoder takes the form depicted in Figure 7.

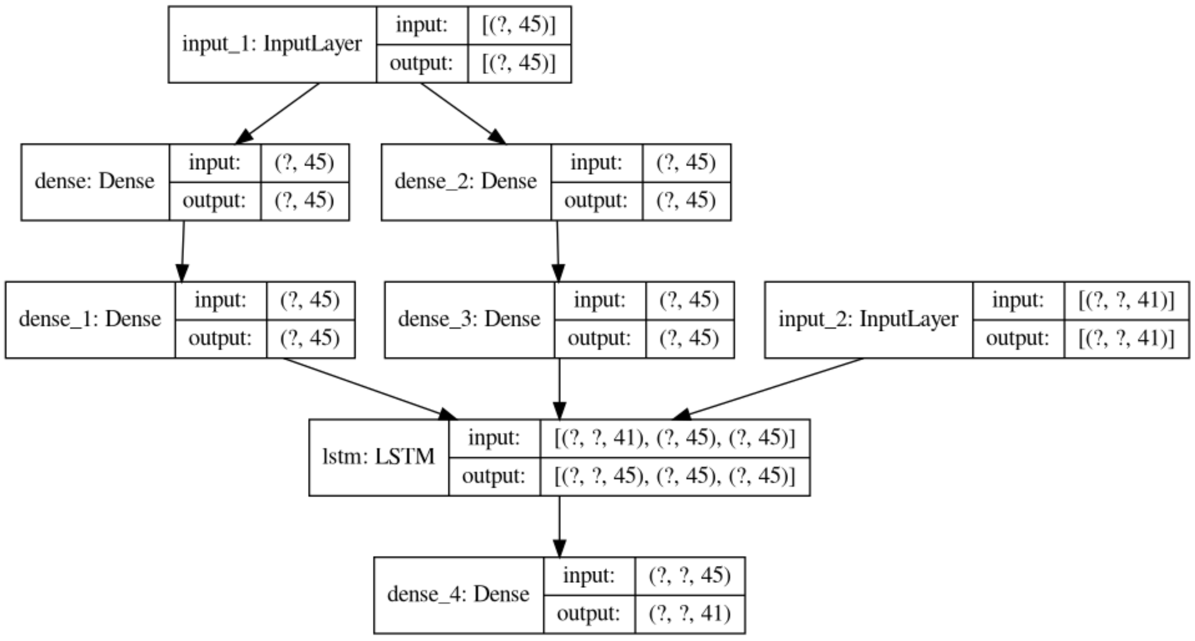

input_1: InputLayer | input: [(?, 45)] | output: [(?, 45)]
dense: Dense | input: (?, 45) | output: (?, 45)
dense_2: Dense | input: (?, 45) | output: (?, 45)
dense_1: Dense | input: (?, 45) | output: (?, 45)
dense_3: Dense | input: (?, 45) | output: (?, 45)
input_2: InputLayer | input: [(?, ?, 41)] | output: [(?, ?, 41)]
lstm: LSTM | input: [(?, ?, 41), (?, 45), (?, 45)] | output: [(?, ?, 45), (?, 45), (?, 45)]
dense_4: Dense | input: (?, ?, 45) | output: (?, ?, 41)

Figure 7: The sequence to sequence based architecture for the decoder model that learns to map fixed-length feature vectors to the space of variable-length domain names.

To train the decoder, a data set different to the attacked set is used. The Tranco data set [24] which captures a current ranking of valid domains is publicly available and can be utilized by any attacker. The Seq2Seq decoder is trained on a random 10% sample of the full 8PPV Tranco data set (6.516.002 samples from the timespan: 11th August 2020 to 9th September 2020 (30 days)) with the categorical cross-entropy loss using the RMSprop optimizer with the following settings: (learning_rate=0.001, rho=0.9, momentum=0.0, epsilon=1e-07). The model is trained for at most 100 epochs with a batch size of 512.

Evaluation

We evaluate the success of this attack by measuring the performance of the decoder network on a private data set of NXdomains with help of the Levenshtein distance. As representative private data set we use the above mentioned RWTH, Masaryk and CESNET benign data sets and the (Damerau-)Levenshtein distances as well as normalized versions of both distance metrics. Normalization is performed by dividing the integer edit distance by the longer length of the compared strings, which preserves the triangle-inequality for the metric. For each of the large RWTH and Masaryk data sets a random sample of 1,000,000 benign samples is taken as representative set. The reconstruction capability of the trained decoder can be tested by comparing the original domain string with the output of the decoder model when it is given the respective feature vector of that domain. Due to the timely effort of the evaluation, its results will be presented in the next version of this deliverable. We suspect the case, that the underlying distribution of the Tranco data set is too far away of those of the mentioned benign data sets, as the samples from the Tranco data set only occupy exactly one subdomain each and are valid domains nevertheless. Should the reconstruction capability be negatively impacted by this bias, one could repeat the experiment by training the decoder on each one of the benign data sets and choose the other two remaining data sets as the sets that are attacked. This is a valid scenario within this deliverable, as a sharing participant would have access to his own benign data and not just a publicly available list of benign domains.

Group Membership Inference

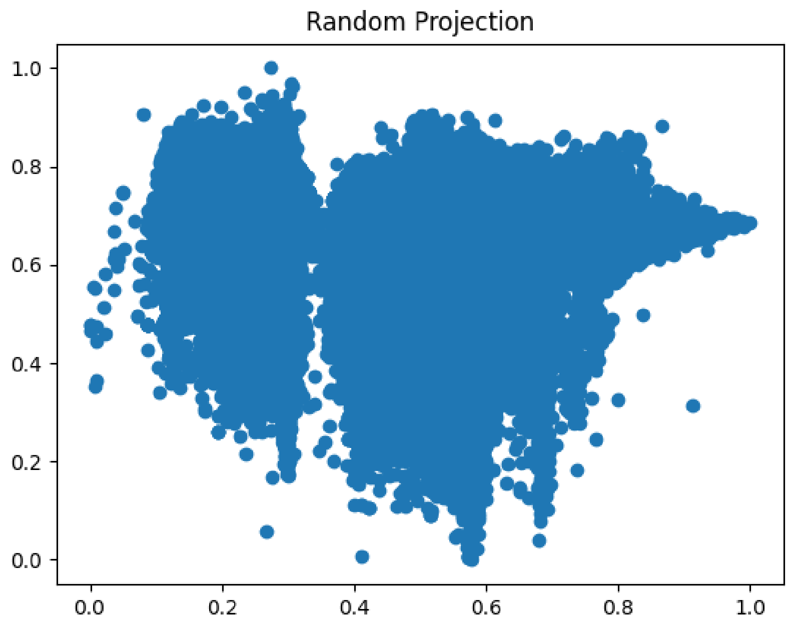
Inside a set of benign samples, one may suspect the presence of multiple sources generating NX domain samples. The most obvious ones are listed below:

- Typos by the end-user
- Misconfigured Software
- Benign DGAs

If an attacker can successfully determine that a shared batch of benign samples originates from one of these sources, it might leak private insight about user behavior and software usage. Under the assumption that domains from these sources exhibit unique characteristics, identifying these groups is an unsupervised learning problem. Thus, clustering or at least manifolds embeddings in combination with dimensionality reductions can be applied. This helps to visualize the benign data in a low-dimensional representation in which the classes can be visually identified and separated. If an attacker learns an embedding or about clusters of data and can successfully link them to one of the above sources, new feature vectors can be matched to these groups or cluster via a nearest neighbor classification. Unfortunately, none of the above-mentioned benign data sets are labelled with an appropriate source identifier, such that resulting clusters from the following evaluation are difficult to verify.

Evaluation

As a first measure we apply a random projection [28] and a PC (principal component) projection [29] on a two-dimensional plane normalized to the ranges [0, 1] on each axis. Results are given below for the benign data sets CESNET (see Figure 10), Masaryk (see Figure 9), and RWTH (see Figure 8), while there are only 1,000,000 random samples each taken from the two latter sets.



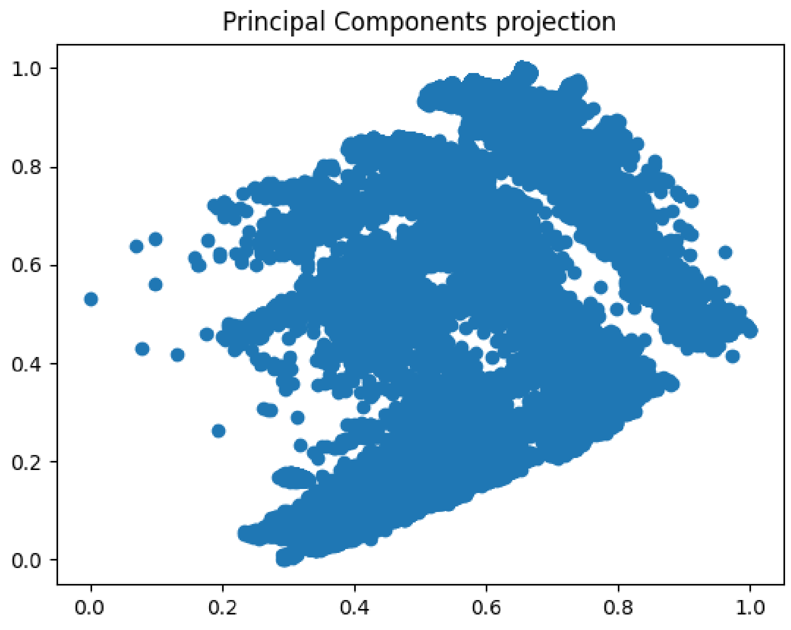


Figure 8: Results of a random projection and a PC projection of the RWTH benign NXdomains data set onto a two-dimensional plane with both axes normalized to range [0, 1].

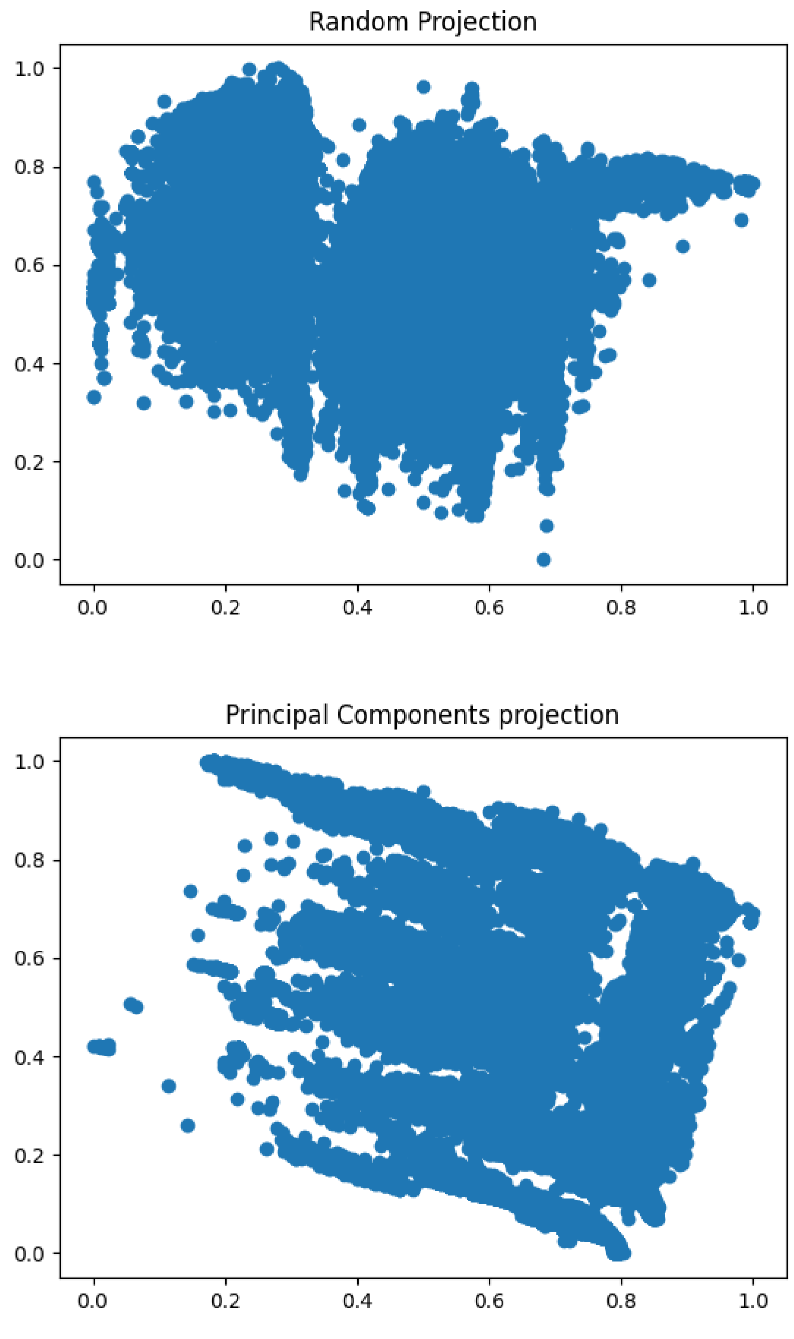


Figure 9: Results of a random projection and a PC projection of the Masaryk benign NXdomains data set onto a two-dimensional plane with both axes normalized to range [0, 1].

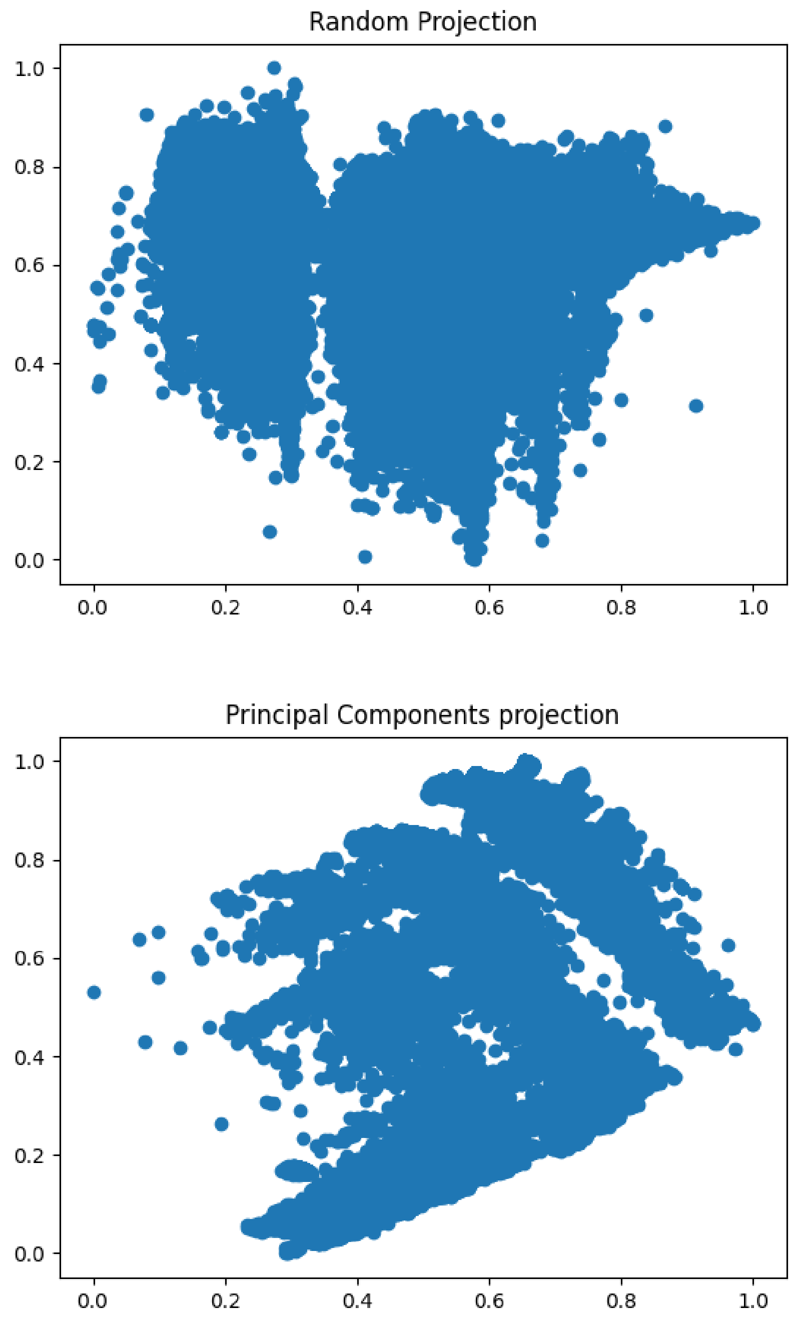


Figure 10: Results of a random projection and a PC projection of the CESNET benign NXdomains data set onto a two-dimensional plane with both axes normalized to range [0, 1].

First insights show that for each projection method the resulting embedding looks very similar to the of the other benign data sets. However, none of the embeddings show any significant clusters in any of the benign data sets although there exists some indication to separable groups, e.g., a split a x=0.3 for all random projections or the top right part in the PC projections. This could imply that an attacker could also not distinguish between feature vectors coming from different sources, which is still to be investigated. In the future, group membership inference can also be evaluated with other embedding (or projection) methods of which the results are likely more expressive but also require more computational cost. These include the locally linear embedding [30] method and its variants as well as t-SNE [31] initialized with a PCA embedding.

Sharing of First-n-layers (Feature Extractor) of Deep Learning Classifiers

Similar to sharing the feature extractor of a classic machine learning approach, the feature extractor of a deep learning model can also be used for sharing data. Depending on the architecture of a deep learning model, the first few layers of the trained model are thought of as an equivalent to a hand-selected feature extractor. Partially sharing a trained model may results in a new attack surface. In the particular case in which the first part of a horizontally split model is shared, the following privacy attacks become relevant:

- Model Inversion
- (Group) Membership Inference

The goals of both the previous Reconstruction attack and the Model Inversion attack are very similar; they differ, however, in their operating principle. Here, the published feature extractor is a neural network model specifically representing a differentiable function. Hence, an inversion may directly utilize the weights in the shared model to invert that function instead of just learning an inversion mapping from samples. Model Inversion tries to recreate an input from a known output (in this case the features) by an iterative gradient optimization of the input values on a suitable loss that compares the model output to the known output. A privacy evaluation of the Model Inversion attack on a deep learning feature extractor can again be quantified with the Levenshtein distance.

<u>Evaluation</u>

For the next version of this deliverable we plan to conduct the threat assessment of Model Inversion and Membership Inference attacks against this sharing scenario. This assessment can be performed over the following combination of parameters:

- Vary the deep learning architecture used for the DGA classifier, in this case the available NYU, Endgame or ResNet architectures.
- Vary the point at which the model is split into feature extractor and classifier.

3.2.6 Development of a Feature-based DGA Multiclass Classifier

Since our preliminary privacy analysis of the feature-based approach FANCI yielded promising results, we are developing a contextless and feature-based approach for DGA multiclass classification. Such an approach would enable us to compare the provided privacy guarantees of feature-based and deep learning approaches in the context of DGA multiclass classification. To the best of our knowledge there is currently no contextless and feature-based multiclass classifier for DGA attribution.

Naïve approach: adaption of FANCI to a multiclass classifier

Since, there currently is no contextless and feature-based multiclass classifier for DGA attribution we adapt FANCI to a multiclass classifier in order to allow for comparisons with featureless approaches. We construct multiclass classifiers from the existing binary classifiers by either using multiple one-vs.-one (OvO) or one-vs.-rest (OvR) classifiers. Thereby, we can reduce the problem of multiclass classification to multiple binary classification problems. We refer to the multiclass enabled SVM approaches as M-FANCI-SVM-OvO and M-FANCI-SVM-OvR, and to the multiclass enabled RF approaches as M-FANCI-RF-OvO and M-FANCI-RF-OvR in the following. Further, in contrast to the SVM implementation, the utilized RF-based implementation is inherently capable of multiclass classification. Thus, we additionally evaluate FANCI's RF implementation which we enabled to a multiclass classifier and refer to it as M-FANCI-RF in the following.

Preliminary Evaluation

In a preliminary evaluation using the datasets described in deliverable D3.4 (Algorithms for Analysis of Cybersecurity Data) for DGA multiclass classification we obtained not promising results. Compared to our developed ResNet.MI model, the adapted FANCI multiclass classifier achieve f1-scores which are 23% - 45% worse. This is due to the fact that the features used in FANCI are specifically created to distinguish between benign and malicious NXDs and not to distinguish between domains generated by different DGA families. Thus, we reckon that for a promising multiclass classifier it is necessary to craft new features.

Development of a Novel Feature-based DGA Multiclass Classifier

Feature-based approaches rely on specific features that are hand-crafted using domain knowledge. The engineering of such features requires much more effort compared to the usage of deep learning classifiers where all important information has to be encoded and provided to the model. The deep learning model subsequently learns the relevant features on its own in an end-to-end fashion during training. Additionally, after the feature engineering the best combination of features has to be selected which is not a trivial task. The combination of several engineered features could contain mutual information which could render single features useless for classification. Such features have to be removed as their extraction from raw data might require significant processing time which could have a negative impact on the real-time capability of a classifier. Additionally, as we have seen in the preliminary privacy analysis of FANCI, different features have different impacts on the privacy. Thus, an appropriate feature selection could help in improving the provided level of privacy. Lastly, in the development process of a feature-based classifier a huge amount of hyperparameters have to be optimized.

Currently, we are still in the development of such a classifier and we plan to present our feature engineering, feature selection, hyperparameter optimization, as well as an comparative evaluation between our novel classifier with deep learning models as well as with all adapted FANCI approaches in the final version of this deliverable.

3.3 Application and Host Profiling

The background for application profiling in the context of sharing is similar in all work packages of WP5. Hence, the following paragraph can also be found in the Deliverables D5.3 and D5.5.

The general idea of application and host profiling is to model the behavior of hosts and applications based on network data as well as system events, which was described in more detail in Deliverable D3.4. Based on the profiles, the idea is to detect anomalies, i.e. when a host or application behaves not as expected. Another use-case is to use the profiles while investigating incidents, e.g. to classify the type of host before executing recovery steps. The application profiling can be further divided into identification and classification. For identification, the goal is to simply detect the operating system and list of applications on a host. This already works well by just monitoring DNS traffic. The goal of the classification task is to not only identify an application, but to compare the behavior of a monitored application with a reference model. This can either provide more detailed information, like the application version, or information whether the application behaves as expected. The classification task relies more on system event data, e.g. monitored by the F-Secure Sensor or software like Sysmon, instead of network traffic.

The goal of this work package is to make the profiles more robust by computing the profiles based on data from multiple organizations. For application profiles, this means to include edge-case behavior of applications in the profiles which might not be observed by all collaborating organizations. We hope, that the sharing will result in more accurate profiles. Analogously, the representative host behavior profiles can be shared to achieve a more robust profile of behavior. Moreover, sharing a host profile with anomaly description could verify, whether this anomaly is only local or is observed at a large scale (and hence is not so anomalous).

3.3.1 Sharing Scenarios

In the following, we will describe the different scenarios that we want to evaluate for sharing and collaboration in the context of application profiling. They are based on the three scenarios described in Deliverable D3.4 for application fingerprinting, with the goal of increasing the robustness using collaboration and sharing of data between multiple organizations. Robustness in this case means, that the different approaches model the behavior of an application more precisely. Another advantage of a global model is, that it can be used by organizations that are not able to compute a local model by themselves, e.g., because they don't have access to sufficient amounts of data. The scenarios describe collaborative rule extraction, process mining, as well as machine learning based on shared anonymized data. For system event data, we will use the F-Secure Sensor (described in Deliverable DX.X) as well as System Monitor (Sysmon), a Windows system service for logging of system activities. For network data, we will use a tool that we developed for this project, which is able to label network packets with the application that produced them with ground truth accuracy.

First, we will give more details about the labeling of datasets, i.e., the tool we developed for labeling of network data, as well as the F-Secure Sensor and Sysmon for monitoring system events. Next, we describe the experiments and approaches for collaborative application profiling. At the time of writing this deliverable, we are still in the planning phase of the experiments for collaboration in the context of application and host profiling. The focus for now was to develop and integrate the tools to generate labeled datasets with ground-truth accuracy, as well as to finish the local approaches in WP3 first. This is because we will use similar techniques for profiling on the global level compared to the local level, just on different data. Hence, we want to finish the development of the local approaches first.

Labeling Datasets

To be able to share anonymized data for a global model, the (labeled) datasets need to be created in the first place. For system events, we already have tools, e.g. the F-Secure Sensor, to collect such data. In the context of application profiling (without considering anomaly detection), this data is already labeled because all events include a reference to a specific process. For network data, the labeling by process or application requires a combination of network monitoring and system monitoring. For the labeling of network traffic, we developed a tool, which already works for Windows and will be extended for Linux as well. The tool, in the following referred to ad GALF (Ground Truth Application Labeling Framework), currently is able to label network traffic with ground truth accuracy.

The Windows version consists of mainly three components: event logging, packet capturing and packet labeling. The event logging component makes use of the Windows monitoring tool Procmon (Process Monitor). Procmon logs every event occurring on a Windows machine together with the application causing the event. Those events are filtered to only get network events. The XML output of Procmon is then parsed and the information of individual events is forwarded to the labeling component via a queue.

Simultaneously, the capturing component runs Tshark and parses its output in real-time. By this, one gets a stream of packets that is passed to the labeling thread via another queue. Network events and network packets have the same order and are both timestamped. GALF makes use of these properties and matches an event to a packet by first comparing source IP address, source port, destination IP address and destination port. Then, only if those are the same in the event and in the packet, GALF considers the timestamps and assigns the packet with the smallest time-delta the application specified in the event.

Another special case arises if one considers DNS network packets. Applications usually use the system DNS resolver if they need to resolve a URL. Since Procmon reports which application is triggering the network event this would lead to having every DNS packet labeled as svchost.exe instead of the original application requesting the resolver. To take care of this GALF considers the DNS client events log of Windows. Once activated, this log saves all the DNS queries together with the process ID of the original requesting application. To finally label DNS packets, GALF first compares the query logged in the events with the query written in the packet and uses matches again the event to the packet with the smallest time-delta. To map process IDs to their applications, GALF additionally polls tasklist.exe periodically. In the end, GALF can match packets with the application that was causing it - even for short-lived connections such as DNS.

The Linux version is planned to use Systemtap as a means to intercept the kernel functions sending IP packets. By this, it is possible to learn the process IDs of the processes responsible for the according packet together with source IP address, source port, destination IP address and destination port. Then, it is again possible to match network packets and processes based on this 4-tuple. To address DNS packets, it is planned to use Systemtap to also intercept calls to the internal DNS resolver which can then be mapped with the process ID, the DNS query, the IP addresses and the ports again.

Finally, we want to design a sharing interface such that other parties can share their labeled network traces generated with GALF. This will make use of MISP, which is the sharing platform that we use for SAPPAN. As defined in our architecture, the platform will be used to share the information that a dataset is available, including meta information about the data and where to receive it from. Hence, we will implement a data model for MISP that allows for the sharing of datasets used for application and host profiling.

Before such datasets are shared, they need to be anonymized and sanitized. Anonymization includes steps like obfuscation of IP addresses, while the sanitization step implements measures to enforce organization specific policies, e.g., the exclusion of security-critical hosts from the dataset. In general, user specific data should be anonymized before sharing. In the case of datasets produced by GALF, this can for example include DNS queries initiated by the browser. Generally speaking, when looking at DNS, most of the user-initiated queries will come from the browser. Other applications mostly use the DNS protocol in an automated fashion and independent of the user, with exceptions of course. At the time of writing this Deliverable, techniques for anonymization and sanitization are still in an early stage of development.

Sharing of Labeled Data for Collaborative Rule Extraction

In Deliverable D3.4, we described the approach of statistical or rule-based modeling of application behavior. The idea for collaboration is to have more data from different environments available for each application that we want to profile. As of now, we use the rule-based approach only for the identification of applications based on DNS data. This allows to create a list of active applications for each host by monitoring their network activity. This information can be used to determine the type of the host (e.g. server, developer machine, office machine, etc.). However, this generally doesn't allow to determine whether the applications behave as expected. For this, we will use process mining and machine learning as described below, which allow to model more complex behavior in a more automated fashion compared to the rule-based approach.

The first option for collaboration is to extract rules from each shared dataset on the global level separately, which is similar to sharing of each party's rules in the first place (as described in Deliverable D5.3). The different rule sets can then be merged to a global rule set. For merging, we also plan to evaluate multiple approaches. One option is to build the union of all rules. Under the assumption that all shared rules are correct, this results in a more precise ruleset. However, this doesn't allow to filter or correct rules that are flawed. Another option is to combine all rules by computing the intersection, which has the potential to filter false rules. However, this approach might rules describing edge-case behavior which was not observed by all organizations. Hence, a hybrid approach will hopefully combine the best of both worlds. One way could be to include a rule to the global ruleset, if at least two organizations can assert the correctness of the specific rule.

The second option is to first merge the shared datasets into one global dataset, and compute the rules and statistics based on that. For merging, we have similar options as for the merging of rulesets. However, because we merge the available data on a lower level, it is possible that more information is preserved this way. For example, a rule might be too complex or precise to compare it to similar rules of other organizations. However, if the datasets are merged, it might be easier to extract rules that fit to the global datasets as well as the local ones.

At the time of writing this deliverable, we did not yet conduct the experiments to evaluate the different approaches of extracting rules in a collaborative manner, because it requires access to labeled datasets as well as proper anonymization. Using the developed labeling tool as well as the sensors for monitoring system events, we are now able to create such datasets.

Sharing of Labeled Data for Collaborative Process Mining

The second approach for application profiling that we described in Deliverable D3.4 is process mining. The basic idea is to generate process model for each application which models its behavior based on the monitored data. The output of the process mining techniques consists of Petri Nets containing the system events, or in case of network data, DNS events (i.e. queries). We are currently working on process mining techniques based on DNS data (similar to the rule-based approach), as well as system events collected by the F-Secure Sensor as well as Sysmon. The first will primarily be used for application identification, and the latter for application classification, and later for anomalous behavior detection. For collaboration, the idea is similar to the rule-based approach. The goal is to make each application model more robust, e.g., by including edge-case behavior that was not monitored by each organization.

This will be done by merging the datasets first, and apply the process mining approaches to the global dataset to compute a global model for each application. Creating models for each dataset first and then merge the models instead is not as intuitive as for the rule-based approach, because of the more complex structure of the process mining models. How such models can be merged will be investigated for Deliverable D5.3, which covers the computation of a global model based on locally shared models, instead of shared anonymized data.

For the merging of datasets, we will investigate the same options as for the rule-based approach. This means, we will experiment with simply computing the union of each local dataset, the intersection, or something in between. The goal is to end up with a process mining model that describes the behavior of an application more precise. This is under the assumption that a specific application generally behaves similar for different organizations. Since the users are different, the applications might be used in a different way, resulting in more diverse data. However, some application might behave drastically differently when used in a different environment, which could result in a global model that performs worse compared to the local models for each organization. Hence, this approach can only be used for applications that don't fall into that category.

Sharing of Labeled Data for Collaborative Machine Learning

The third approach we described for application profiling in Deliverable D3.4 is machine learning, i.e. deep learning. For DGA detection as well as phishing detection, the machine learning approach works very well. However, for the application profiling case, the results so far indicate that the rule-based and process mining based approaches are better suited.

For the identification task, the rule-based approach on DNS traffic only so far works pretty well. For identification, it does not make sense to use system events, since then the task becomes trivial. The advantage of using network traffic is that it is easy to monitor and does not require to install a sensor on each host. As far as application identification based on network traffic goes, one advantage to use machine learning is that the process of creating the model is more straight forward compared to the extraction of different rules. However, the rules can be comprehended, and to some extend be verified, by humans, while deep learning models are a black box.

For the classification task, it is not only important to determine whether the behavior of a monitored application is conforming to the reference model or not, but also how it differs in case it is not entirely conform, which is likely. With a process mining model, the exact events that either were different, unexpected, or missing, can be visualized easily which allows to further analyze the anomaly. A deep learning model on the other hand simply outputs a confidence score, which can indicate how well the data fits to the model, but not where it differs. This could partially be solved by using machine learning approaches like random forests which allow more inside into the decision process, however, this would require more manual effort for feature engineering compared to process mining.

The goal is to at least evaluate the deep learning approach for the identification case, especially with federated learning as described in Deliverable D5.5. Because we already investigate the federated learning approach for other showcases, this can be applied more easily without building everything from ground up.

Sharing and Collaboration for Host Profiling

Sharing and collaboration for the host profiling are in many ways similar to application profiling. The main difference is that there is no given host that should share some basic properties as it is in the case of the applications. In other words, we expect that an application should show similar behavioral features across all installations even across different environments/companies/networks. The name of the application is a unique identifier of the behavior. Such a unique identifier that would mark a specific behavior and that would be interpretable across different environments/companies/networks does not exist in the case of the host behavioral profiling. We can share only descriptions of the common host behavior enriched with labels that we assign to the behavior. There is no generally accepted set of labels, the main key to look at when leveraging the shared information is the description of the behavior itself.

On the global level, sharing and collaboration enable us to get the bigger picture, in general. On the global level, we can adjust the description of the common host behavior to be more robust and represent a wider range of the hosts. Once the robust host behavior profiles are available, the improved anomaly detections can be deployed. We can share the anomaly detection linked with the typical behavior description, and observe, how many other entities also observe this combination of the anomaly and behavior. If such a combination is observed over a large scale of the entities contributing to the sharing, we can set this anomaly as a global problem which could lead us to look for more system changes that we would look at in the case that the anomaly is observed only at one entity.

Last, but not least, the sharing of the typical behaviors with labels can improve the overall visibility of the network. The asset management in the SME is usually done manually and it is hard to keep it updated. Using the shared behaviors observable from the network traffic, we can label the hosts and make at least some asset management updates without the need for direct access to the individual machines and manual work.

3.4 Anomalous Behavior Detection

This Section describes how we plan to use shared anonymized data in order to detect anomalous behavior that can be relevant for detecting cyberattacks. First, we present a model for detecting potentially malicious user behavior by analyzing the login activity of users within an organization (focusing on Linux and Windows machines). Next, we outline the benefit of collaboration for detecting anomalous behavior based on application profiles, which will focus on Windows applications.

3.4.1 Sharing Scenarios

In the following, we describe the different sharing scenarios for anomalous behavior detection.

Detecting Anomalous Login Activity of Users in Linux and Windows Machines

Anomalous login activity can often indicate malicious operations. We started from building organization-wide models, based on the data collected in end-user machines by FSC security monitoring sensors (sharing on the end-user level), for identifying anomalous login behavior. While those organization-specific models are used at the moment separately, the next step would naturally be to explore potential benefits and confidentiality requirements of sharing models, or their appropriate derivatives, within certain groups of organizations.

Detecting Anomalous Behavior Based on Profiles

The idea of anomalous behavior detection based on profiles is to detect deviations from the expected behavior. As teased in Deliverable D3.4, we want to approach this for the host profiles as well as application profiles. In the context of this task, the focus will be on detecting anomalies based on the application profiles, because host profiles differ too much among organizations, such that they don't benefit much from collaboration.

Anomalous behavior detection based on application profiles in WP5 solely relies on increasing robustness off the behavioral models resulting from collaboration. We expect that this allow for more accurate prediction of the expected behavior, resulting in more accurate detection of unexpected behavior. However, the focus will not be to automatically classify such anomalies as malicious or benign, but to increase the accuracy in a sense that detected anomalies actually correspond to unexpected behavior. For process mining models, the exact deviations can be computed, e.g., if a system event happened with higher privileges than usual. We want to use visualization techniques to highlight these deviations from the reference models, such that they can be classified by an analyst more easily.

3.4.2 Models for Detecting Anomalous Login Activity

The basic user login activity can be defined as the following event:

- User U successfully logged in to endpoint H

Data of the user login activity are collected by F-Secure's endpoint sensors in the machines of the customer organizations. Machine learning is used to identify potentially malicious login attempts by formulating our task as a multiclass classification problem. The focus is on detecting two types of behavior, which might go unnoticed by other detection components:

- **initial access -** an attacker succeeds in logging in to an organization's endpoint from the Internet (the login's source IP is outside of the organization's network)
- **lateral movement -** an attacker, already in control of an organization's endpoint, logs in to another endpoint within the organization's network (the login's source IP is within the organization's network)

These two types of behavior correspond to the two types of alerts produced by our detection models.

Data and Approach

At the high level, for a given organization, we train a model where one class corresponds to one endpoint. At the inference time, if data coming from endpoint A are not assigned by the model to the class of A, it is considered an anomaly and triggers an alert. If data from a specific endpoint were not included in the model training set, that endpoint will obviously be skipped by the detection flow.

Login attempts are represented by these two platform-dependent event types:

- **linux_audit events** (Linux endpoints) **-** events with *event.data.message_type='USER_AUTH'* represent login attempts.
- **windows_security_event** (Windows endpoints) **-** events with *event.data.event_id=4624* represent login attempts.

In data, only successful logins are considered. To address sharing-related confidentiality concerns, we follow the principle of data minimization and handle only the following information of successful login events:

- **user id** (a randomly generated value identifying the user)
- **sensor id** (F-Secure's security monitoring sensor ID in the endpoint)
- **source IP** address
- **authentication protocol** used

To improve the accuracy of the model, we implemented login deduplication: we consider only unique logins across a given organization. We call a login unique if it occurs on exactly one endpoint. Examples of non-unique logins include: system users logging in to multiple endpoints to perform maintenance tasks; application users - for cluster management software, etc. Filtering out non-unique logins improved the accuracy of the model by an order of magnitude (on average, for the organizations from the model piloting group).

The bag of words approach is used to model the login activity of an endpoint. We consider user IDs, IP addresses and authentication protocol identifiers as words in our dictionary and, for each endpoint, we create a *document* from the set of successful logins observed by the security monitoring sensor in a particular time period. To produce a vector representation of the login activity, we use a count vectorizer on this bag-of-words document.

If a login event is identified as anomalous, the features of the event are enriched - in order to provide relevant information to security analysts - with additional information:

- **process details** of the process performing the login
- **endpoint sensor version** and **operating system**

The model training flow considers a large amount of historical data (e.g., 2 months). The model updating (re-training) procedure is therefore supposed to be carried out periodically (e.g., monthly).

Data Analysis Pipeline

The pipeline is composed of two distinct flows: **model training** and **detection**. The pipeline assumes the following:

- malicious login attempts can be identified by considering the historical login activity within the organization
- the training data do not contain login attempts that are both malicious and successful

The **model training flow** (Figure 11) produces a multiclass classification model for each organization. Each sensor / endpoint within the organization is represented in the model as a class, and at the inference time the model uses the login activity on an endpoint to predict its sensor id.

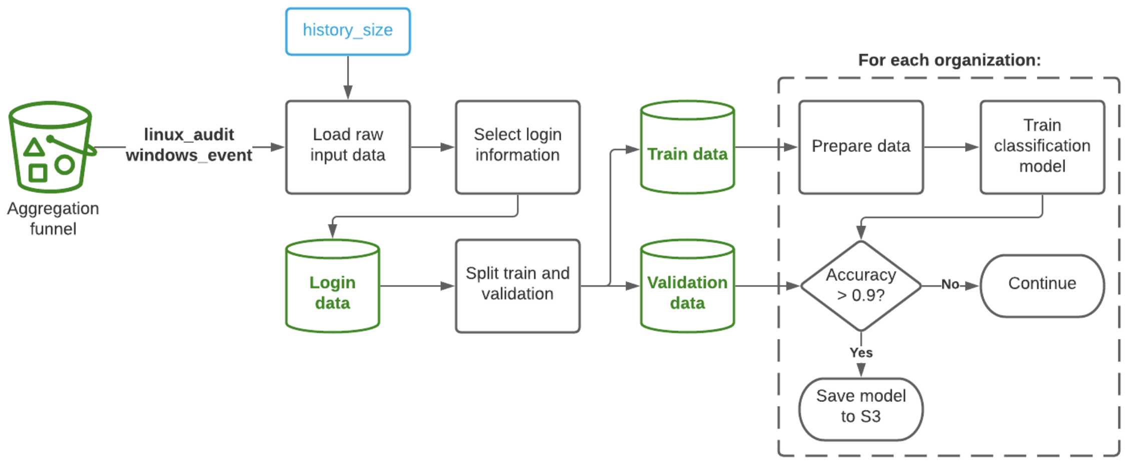


Figure 11: The model training flow in the data analysis pipeline.

A training dataset consists of all the successful login events on the endpoints in an organization, in a specified training period (which is configurable). The last day of the training period is used as a validation set, that is, the model is trained on the training period data with the last day's data excluded. Then, the data from the last day is used to check if the model can consistently predict the sensor identities from the login activity on the sensors. If the accuracy of the model on the validation set is above a specified threshold (e.g., 90%) the model is kept, otherwise it is discarded. The accuracy is understood here as the percentage of correctly predicted endpoints for the login events in the validation set.

The **detection flow** (Figure 12) uses the models generated by the model training flow to detect anomalous login behavior.

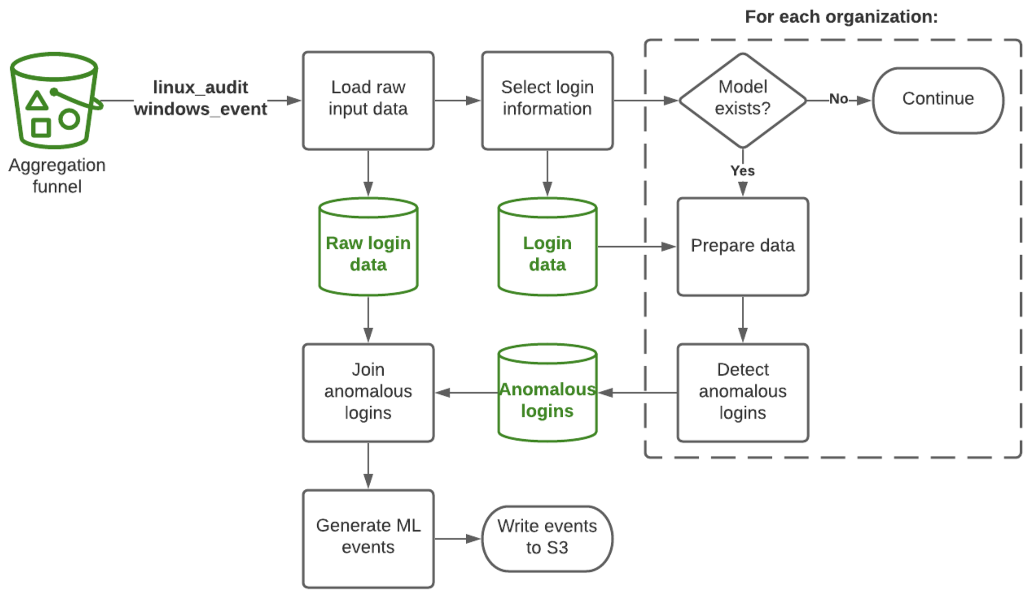


Figure 12: The model detection flow in the data analysis pipeline.

As mentioned, to avoid sharing-related data confidentiality concerns, the information used by the model is a small subset of the information observed by the security monitoring sensors in the endpoints. At the same time, the unused part of the information can be important for security analysts for investigating detected anomalous behavior. For this reason, the alerts on identified anomalous login attempts are augmented with certain parts of the security monitoring data to provide a context for the potentially malicious logins.

The model for a given organization is used to assign logins to the endpoints in that organization. If the model classification is correct, the login activity is considered normal. Otherwise, the activity is considered anomalous. The currently deployed model is a Naive Bayes classifier, and all the model parameters are set to their default values. Naive Bayes was selected since it provides good accuracy in the validation phase. More advanced models may be considered in the future.

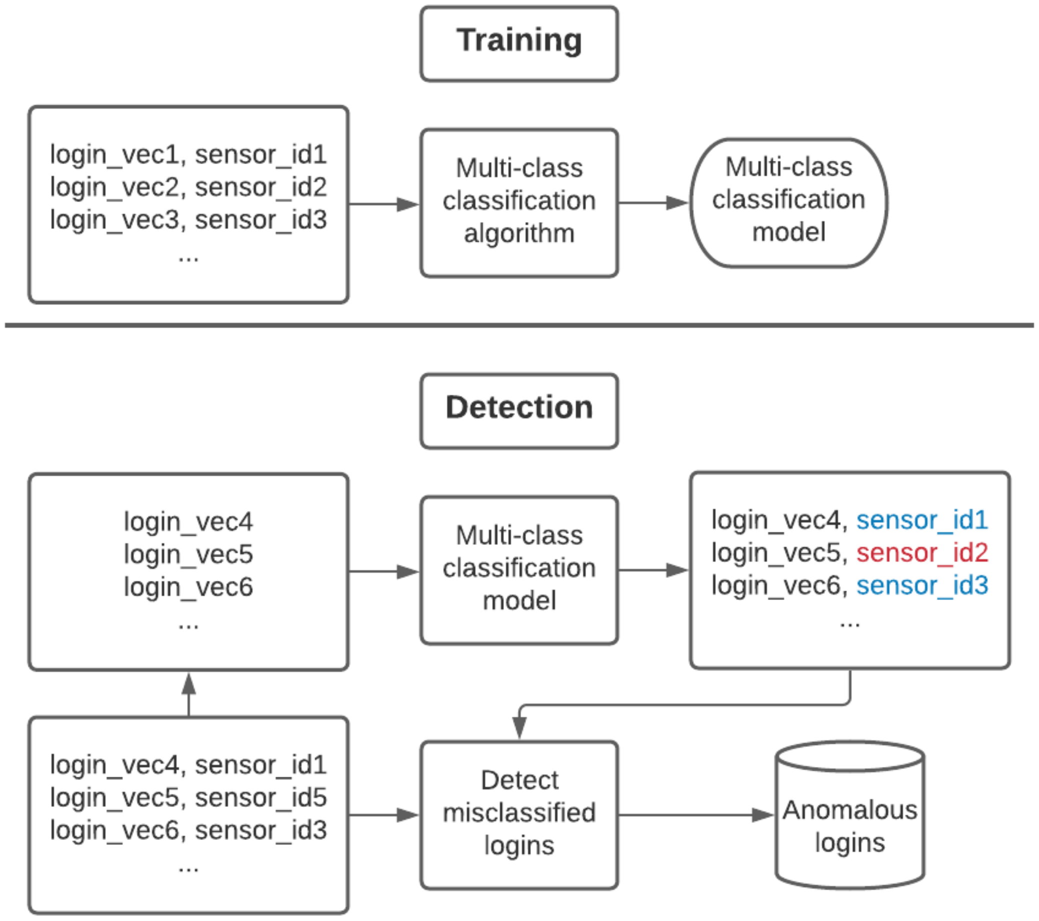


Figure 13: Training and detection flow in the data analysis pipeline.

As a note, the output of the model in the recent experiments - unsurprisingly - contains non-negligible numbers of false positives. The current approach to addressing this issue is to filter the anomalous logins identified by the model with a post-hoc filter incorporating expert knowledge from security analysts. At the same time, the experts involved in the model piloting are positive about the model's value for the overall attack detection logic, which indicates that the chosen approach can be considered promising.

4 Conclusion

In this deliverable, we presented the current state of our approaches and experiments regarding distributed learning on a global model based on shared anonymized data. We briefly discussed the context of this task within SAPPAN, and presented our showcases. Similar as in Deliverable D3.4, this deliverable focuses on the showcase of DGA detection, because it is the most mature both in WP3 as well as WP5. In detail, we defined three different scenarios for creating a global model based on shared anonymized data for DGA detection. In the final version of this deliverable, we will evaluate these sharing scenarios and compare the gain in classification performance as well as the provided privacy guarantees with each other. A preliminary privacy analysis of the feature-based DGA detection approach FANCI showed that sets of feature vectors seem to be resilient against a reconstruction attack and can therefore be used as sort of anonymization process in this sharing scenario. Since we cover the use case of DGA detection in all three deliverable (D5.1, D5.3, and D5.5) that are focusing on the creation of global models based on knowledge distribution, we will be able to compare the different approaches with each other. For the application and host profiling, we focused on describing the general ideas for our different sharing scenarios. We are still finalizing our approaches on the local level for WP3, which are required to build models on the global level, which will be included in the second version of this deliverable. For detecting anomalous login activity, we presented the model and training details and mentioned the positive opinion of security experts on the first piloting results.

For future work and the second version of this deliverable, we plan to further conduct the experiments for DGA detection and extend the privacy analysis of machine learning models. For application profiling, we so far only designed the scenarios and experiments. For the next deliverable, we will build global models for application profiling accordingly and measure how well they generalize compared to the local models. For anomaly detection, we will evaluate how well suited the application profiles are for this task on the global level. Additionally, we will conduct further experiments with the models for detecting anomalous login activity.

References

[1] R. Schnell, T. Bachteler & J. Reiher. Privacy-preserving record linkage using Bloom filters. BMC Medical Informatics and Decision Making 9, 41 (2009). https://doi.org/10.1186/1472-6947-9-41

[2] J. Saxe and K. Berlin. eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. arXiv:1702.08568. 2017.

[3] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer. FANCI: Feature-Based Automated NXDomain Classification and Intelligence. In USENIX Security Symposium. 2018.

[4] R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. De Cock. An Evaluation of DGA Classifiers. In IEEE International Conference on Big Data. 2018

[5] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen. A LSTM Based Framework for Handling Multiclass Imbalance in DGA Botnet Detection. Neurocomputing 275. 2018.

[6] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. arXiv:1611.00791. 2016

[7] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock. Character Level Based Detection of DGA Domain Names. In International Joint Conference on Neural Networks. IEEE. 2018.

[8] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S.Abu-Nimeh, W. Lee, and D. Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In USENIX Security Symposium. 2012.

[9] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel. Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains. ACM Transactions on Information and System Security 16, 4. 2014.

[10] M. Grill, I. Nikolaev, V. Valeros, and M. Rehak. Detecting DGA Malware Using NetFlow. In IFIP/IEEE International Symposium on Integrated Network Management. 2015.

[11] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero. Phoenix: DGA-Based Botnet Tracking and Intelligence. In Detection of Intrusions and Malware, and Vulnerability Assessment. Springer. 2014

[12] Y. Shi, G. Chen, and J. Li. Malicious Domain Name Detection Based on Extreme Machine Learning. Neural Processing Letters 48, 3. 2018.

[13] S. Yadav and A. L. N. Reddy. Winning with DNS Failures: Strategies for Faster Botnet Detection. In International Conference on Security and Privacy in Communication Systems. Springer. 2011.

[14] J. Peck, C. Nie, R. Sivaguru, C. Grumer, F. Olumofin, B. Yu, A. Nascimento, and M. De Cock. CharBot: A Simple and Effective Method for Evading DGA Classifiers. ArXiv:1905.01078. 2019

[15] J. Spooren, D. Preuveneers, L. Desmet, P. Janssen, and W. Joosen. Detection of algorithmically generated domain names used by botnets: A dual arms race. In Proceedings of the 34rd ACM/SIGAPP Symposium On Applied Computing. ACM. 2019.

[16] P. Lison and V. Mavroeidis. Automatic Detection of Malware-Generated Domains with Recurrent Neural Models. In Norwegian Information Security Conference. 2017.

[17] A. Drichel, U. Meyer, S. Schüppen, and D. Teubert. Analyzing the real-world applicability of DGA classifiers. in International Conference on Availability, Reliability and Security. ACM, 2020.

[18] F. Becker, A. Drichel, C. Müller, and T. Ertl. Interpretable visualizations of deep neural networks for domain generation algorithm detection. In Symposium on Visualization for Cyber Security. IEEE, 2020.

[19] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In International Conference on Learning Representations. 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. In Computer Vision – ECCV. Springer. 2016.

[22] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla. A Comprehensive Measurement Study of Domain Generating Malware. In USENIX Security Symposium. 2016.

[23] P. Mockapetris. Domain Names - Implementation and Specification. Technical Report. RFC1035. 1987. https://tools.ietf.org/html/rfc1035

[24] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019).

[25] Sutskever, Ilya, Oriol Vinyals and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. NIPS (2014).

[26] R. J. Williams and D. Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. Neural Computation. 1989.

[27] I. J. Goodfellow and Y. Bengio and A. Courville. Deep Learning. MIT press. 2016.

[28] P. Li, T. Hastie and K. W. Church. Very Sparse Random Projections. 2006.

[29] N. Halko and P. Martinsson and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. 2010

[30] S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. American Association for the Advancement of Science. 2000.

[31] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. Journal of Machine Learning Research. 2008.

[32] M. Fredrikson, S. Jha and T. Ristenpart. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015.

[33] G. Ateniese, G. Felici, L. V. Mancini, A. Spognardi, A. Villani and D. Vitali. Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers. CoRR. 2013.

[34] R. Shokri, M. Stronati, C. Song and V. Shmatikov. Membership Inference Attacks against Machine Learning Models. CoRR. 2016.

[35] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter and T. Ristenpart. Stealing machine learning models via prediction apis. 25th USENIX Security Symposium (USENIX Security 16).

[36] C. Song, T. Ristenpart and V. Shmatikov. Machine learning models that remember too much. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.

[37] I. J. Goodfellow and J. Shlens and C. Szegedy. Explaining and Harnessing Adversarial Examples. 2014.

[38] H. Rehman, A. Ekelhart and R. Mayer. Backdoor Attacks in Neural Networks - A Systematic Evaluation on Multiple Traffic Sign Datasets. 2019.