Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D5.5 Global Model without Sharing Local Models, First Version (M21)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file: Deliverable 5.5 Global Model without Sharing Local Models, First Version
Document version: 1.0
Document owner: Benedikt Holmes (RWTH)

Work package: WP5
Task: T5.3 Federated Learning of a Global Model without Sharing Local Models
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)	2020-12-18	First version of outline including first bullet points.
0.2	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH)	2021-01-08	Finalized outline and structure.
0.3	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH)	2021-01-27	Most sections complete and ready for review.
0.4	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH), Mischa Obrecht (DL)	2021-01-28	Incorporate comments and suggestions from review.
1.0	Benedikt Holmes (RWTH)	2021-01-29	Final version ready to submit.

Internal review history:

Reviewed by	Date	Summary of comments
Mischa Obrecht (DL)	2021-01-28	Grammar, spelling, content, structure.

Executive Summary

This initial deliverable is one of two deliverables for the Task T5.3 Federated Learning of a global model without sharing local models. Its goal is to learn a global model, similarly as of the tasks T5.1 and T5.2, but using a different sharing approach. Therefore, the deliverables D5.1, D5.3, and D5.5 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 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 the different showcases and sharing scenarios, including planned experiments and initial results. The showcases are similar to the ones developed in WP3, namely Domain Generation Algorithm (DGA) detection, application and host profiling, and anomaly detection. This task does not require anonymization of data or models, because none of this is shared. Instead, we make use of federated learning and a teacher-student approach.

Contents

1 Introduction

This deliverable for Task T5.3 has similar goals as the tasks T5.1 and T5.2, hence, a large part of its motivation and context is similar to the other tasks. In SAPPAN, one goal is to build a sharing platform that can be used for privacy-preserving sharing of intrusion detection data, detection models, and response handling information. This sharing mechanism is intended to improve the local capabilities of each participating organization by collaboration. 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 without sharing complete local models or private data. This limits the approaches to either share potentially less privacy-critical model updates (federated learning), or to provide classification as-a-service of a global dataset without revealing the local models (teacher-student approach). Compared to the Tasks T5.1 and T5.2, this approach is the most privacy preserving one. However, when no data or complete models can be shared, this heavily limits the approaches for learning global models. The goal is, to still make use of collaboration to improve the local detection mechanisms. Throughout the first three tasks of WP5 we focus on the showcases which we developed local detection mechanisms for, as well as some additional ones. In this task, we mainly look at the showcases for DGA detection and application profiling, because the approaches are not applicable to all showcases. We will make use of federated learning, which only requires to share model updates instead of complete models, as well as a teacher-student approach, which only queries local models without revealing them or the used training 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 this task in the overall scheme of this 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 two sharing scenarios and first results of our experiments. Next, we briefly outline how the same approaches can be applied to application profiling. Finally, we briefly conclude the first version of this deliverable.

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 D5.1 and D5.3.

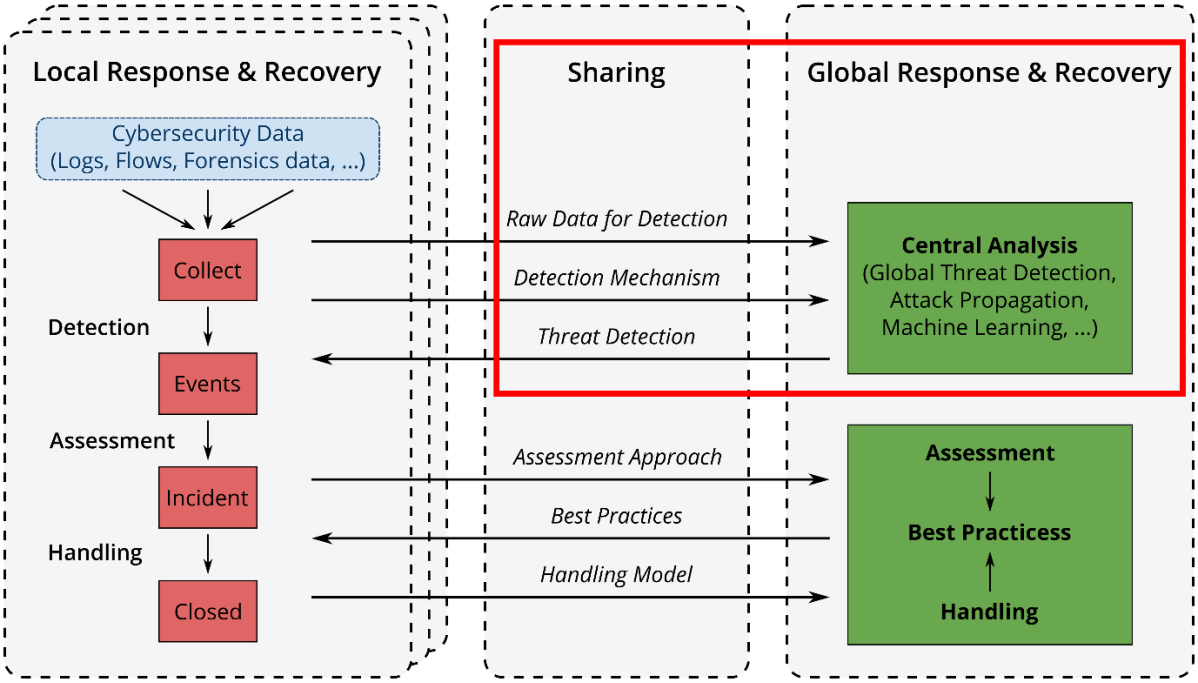

Figure 1: SAPPAN scheme regarding local and global response and recovery.

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 visualization. The tasks T5.1, T5.2, and T5.3 include the development of global models for detection based on different approaches. The general idea is to utilize data and models which are developed in 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. One problem with respect to sharing is of course privacy, which is tackled by the different approaches that are developed in the tasks T5.1, T5.2, and T5.3. The approaches range from sharing of anonymized data, over sharing only pre-trained models, to sharing neither of them. The goal is to apply these techniques to similar showcases as described in WP3 to build global detection models utilizing different levels of privacy. For that, we will use anonymization techniques developed in Task T3.4 based on the privacy requirements described in Task T2.2.

3 Federated Learning of a Global Model without Sharing Local Models

The goal of this task is to build a global model without sharing either anonymized data or local models. This is the most challenging task compared to T5.1 and T5.2 because less information is available on the global view. On the other hand, this task includes the most privacy preserving approaches because the least amount of information is shared. The two principles to still create a global model in this task are federated learning, where only updates to a model are shared, and the teacher-student approach, where the local models are queried using a public dataset.

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 $D=\{(x_i,y_i)_{i=1,\dots,d}\}$ as a subset of $X \times Y$ drawn from unknown distribution $D \sim \mathcal{D}$. 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:

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

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 incremental sharing of weight updates (federated learning) or prediction APIs (teacher-student approach) for classification models, the relevant attack classes are both Data and Model Inference attacks [27, 26, 25, 24] as well as Poisoning [30]. The emphasis lies on Membership Inference and Model Extraction attacks. Attacks of the latter group aim at extracting the functionality of the target model through extensive querying, as in these sharing scenarios models are hidden behind prediction APIs. In the case of federated learning, a variant of the Model Inversion or Poisoning attack may be executed by a participant by him deviating from the federated learning protocol and releasing wrongful information for the iterative gradient updates. Thereby, the model learning process can be negatively influenced. The other attack classes, i.e., Adversarial Examples [29] and Sidechannel attacks [28], are not of relevance for 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.

Through this commonality, 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. As a reminder, we briefly discuss the most important aspects here.

Modern botnets rely on DGAs to establish a connection to their command and control (C2) server. In contrast to the usage of single fixed IP-addresses or fixed domain names, the communication attempt of DGA-based malware is harder to block as they generate a vast amount of algorithmically generated domains (AGDs). The botnet herder is aware of the generation scheme and thus 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 queried domains are 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 to their 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 partake 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. [1, 2, 3, 4, 5, 6]), 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. [7, 8, 9, 10, 11, 12]) 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 [1, 2, 4, 5, 6].

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. [2, 7]) and deep learning (featureless) classifiers (e.g. [1, 4, 5, 6]). While the deep learning classifiers outperform the feature-based approaches in terms of classification performance [4, 5, 13, 14, 15], 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 [16] 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 proposed in related work. Note, to counteract the explainability problem of deep learning classifiers we developed a visual analytics system [17] 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. [2] 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 contextless. FANCI does not incorporate DGA multiclass classification support.

Endgame

Woodbridge et al. [5] 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. [6] 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 [18] 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 [16]. 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 [19, 20]. 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. [4] showed that the model of Woodbridge et al. [5] 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 number 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 [21] 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 (MU), and RWTH Aachen University (RWTH). 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. The malicious data used in this use case is mostly publicly available and thus has no privacy constraints. In the following, we therefore focus 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 first evaluations in this deliverable. In total, we developed two different sharing scenarios which we will evaluate in the final version of this deliverable. We present the two sharing scenarios in the following:

1. Federated Learning

In federated learning, every party that is participating in the training of a global model first acquires the current model which can also be a random initialized model at the beginning. Then, each party individually improves the current model using own private data and subsequently summarizes the changes to the model as a small focused update. Every party distributes its model update and averages it together with the updates of all other participants. Using this federation step, every party is now able to apply the collaboratively computed update to the shared model in order to improve it. This is an iterative process and as many federation steps can be done until the global model reaches a maximum regarding classification performance on a certain validation set.

To summarize, all participating parties train a neural network classifier collaboratively by applying averaged model updates to a shared global model. In this deliverable we investigate two different approaches for federated learning that differ in their type and amount of federation steps. In the following, we describe both approaches.

a) Federation after model convergence

In this approach, initially either a randomly initialized or a pre-trained neural network classifier is distributed among all participating parties. This model is used as starting point for all collaborating parties. The pre-trained global model can be trained based on public available data (e.g. Alexa or Tranco top domain names [22] for benign training samples) such that there are no concerns regarding privacy. All parties then continue the training of the initial classifier using own private data. After each locally trained model converges (i.e. it reaches a maximum regarding classification performance on a validation set) all parties compute the updates in relation to the initial model. The updates equal the difference of the neural network classifiers' weights to the initial model. Subsequently, all updates are merged by averaging and applied to the initial model which yields the final global model.

b) Federation after each training epoch

This approach is similar compared to the previous one but differs in the type and amount of federation steps. Instead of training the initial model locally up to its convergence, each party shares the computed update after each training epoch. Then, similarly, all updates are merged by averaging and applied to the initial model which yields the global model for the next training epoch. All parties then continue the training of the updated global model for another epoch. This process is continued until the global model converges.

The chosen approach for federated learning might have an influence on the global model's classification performance as well as on the provided level of privacy which will be investigated in the final version of this deliverable.

2. Teacher-Student Approach

In this sharing scenario, we leverage the predictions of locally trained detection models (teachers). A party that wants to train a global model (student) has to be in the possession of a dataset for training. Samples of this potentially unlabeled dataset are used to query the teacher models of the other parties that are collaborating. The student model is then trained based on the combined predictions of the teachers. The predictions can be combined by the following two approaches:

a) Soft labels

The final label used for training the student equals the average of the teacher models' confidence scores for a given input sample.

b) Hard labels

The final label used for training the student equals the majority vote of the teacher models. A tie breaker is needed with an even number of teacher models. For instance, the tie breaker could be whether the average of the teacher models' confidence scores is above a certain threshold.

The type of labels used for training the student model might have an influence on the classification performance as well as on the provided level of privacy.

3.2.5 Evaluation: Federated Learning

We started our evaluation with the first scenario, i.e. federated learning with federation after model convergence, for which we already obtained results. We are thus able to present these results in the following. The evaluation and comparison with the other sharing scenarios will be available in the final version of this deliverable.

Experimental Setup

For our experiments, we make use of all three sources for benign labeled data (CESNET, Masaryk University, and RWTH Aachen University) and malicious labeled data from DGArchive as described in the evaluation setup. For training an initial pre-trained global model for federated learning, we make use of the Tranco top domains list that includes popular domains. In contrast to the often-used Alexa top domains list, Tranco is hardened against manipulation [22]. We create two pre-trained global models by either selecting the top most domains or by selecting a random subsample. We perform all federated learning experiments for all three deep learning based DGA binary detection classifiers (Endgame, NYU, and ResNet).

Experiments

In the following, we first describe the creation of appropriate datasets for our experiments. For convenience, we provide an illustration of this process in Figure 3.

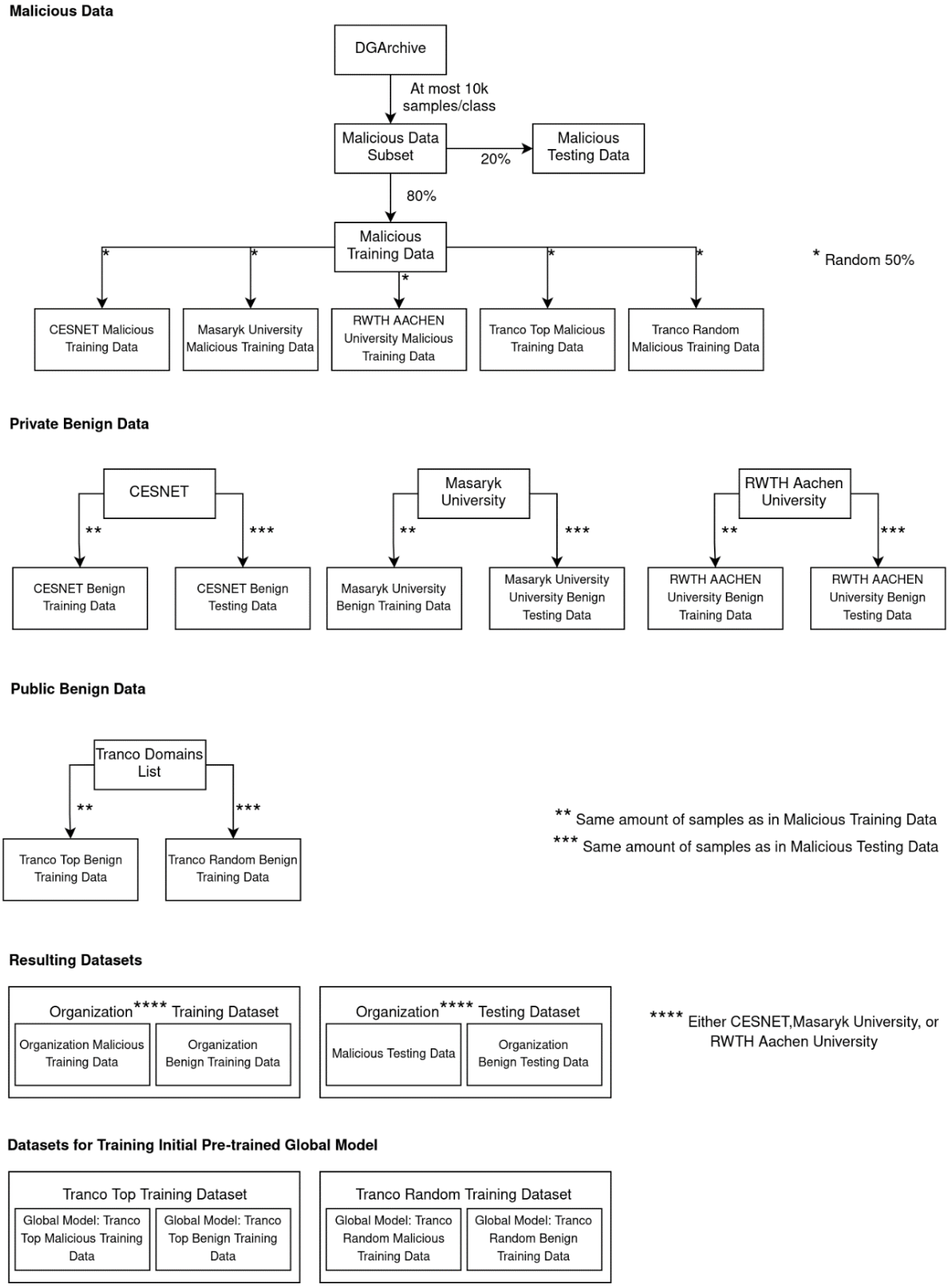

Figure 3: Datasets generation process for federated learning experiments.

We subsample our malicious labeled data into a smaller set that includes at most 10,000 samples per DGA family. If less than 10,000 samples per class are available we include all samples. We then split 20% of this data stratified over all included classes for the testing sets. From the remaining 80% we create malicious data used in

several training sets (for training the global models as well as for the federated learning itself) by randomly subsampling 50% of the remaining samples. The relational is that by subsampling from a bigger shared pool of malicious samples we create training sets which include both duplicated and unique samples. We do this as in a real-world scenario it is most likely that the parties which collaborate in training the global model share the same malicious samples. Moreover, as identified in deliverable D3.6 (Cybersecurity Data Abstraction), the set of benign training samples is the main privacy-critical aspect of this use case. Thus, benign samples are not shared between the different training sets. We create two balanced trainings sets for pre-trained global models, one by taking the top most entries of the Tranco list and one by randomly subsampling the list. Additionally, we create for each of our three benign data sources a balanced training set for federated learning, again, by randomly subsampling the respective data. For each benign source, we use the previously split 20% malicious samples and create appropriate balanced testing sets by adding random samples of the respective data (which are, however, disjoint to the benign data included in the training sets).

Using these datasets, we are able to precisely measure the influence of federated learning on the classification performance of the various classifiers. As the benign training samples are identified as the main privacy-critical aspect of this use case we mainly use the false positive rate (FPR) as a proxy to determine the possible gain or loss in classification performance. For the sake of completeness, we additionally provide the results for the accuracy (ACC), true positive rate (TPR), false negative rate (FNR), and the true negative rate (TNR). Note, by using the same malicious samples in all testing sets we reduce their influence on the accuracy metric and can thereby measure the classifiers' generalization capability on unseen benign samples that originate from different networks more easily.

Using this datasets generation process we perform five repetitions for all federated learning experiments and average the outcomes in order to obtain meaningful results.

In total, we evaluate two different global model (using top most and random samples of the Tranco list), three different classifiers (Endgame, NYU, and ResNet) and three participating parties (CESNET, Masaryk University, and RWTH Aachen University) for our federated learning experiments.

In the following we provide an outline of the results.

We first present baseline results in Table 1 which we use to assess the benefit of private information sharing in the context of federated learning. In 2, we show that models which are trained using public data only are not able to classify the network traffic obtained from all three of our benign data sources (false positive rates of 28% - 74%). In 3, we show that our approach of continuing the training of a pre-trained global model using private data (where the global model was initially trained using public data) has no negative influence on the classifier's classification performance compared to the baseline classifiers. In the Tables 4 – 6, we present the results of our federated learning experiments for each investigated classifier separately and compare the achieved classification scores to the baseline results. For convenience, we present the achieved classification results of the federated learning classifiers averaged over all testing datasets and for each classifier type in 7. In 8, we investigate the influence of the number of participating parties in federated learning on the classification performance for each

testing network separately. In 9, we present summarized results that are averaged over all testing datasets for convenience.

Results

In the following, we first present baseline results which we use to assess the benefit of private information sharing in the context of federated learning. To this end, we train for each classifier under investigation and for each party that will be collaborating in federated learning a distinct classifier. We then assess their performance on test sets which include benign samples that origin from the same network as the utilized training samples. We display the results in Table 1.

Classifier	Train Network	Test Network	ACC	TPR	FNR	TNR	FPR
Endgame	CESNET	CESNET	0.99645	0.99739	0.00261	0.99550	0.00450
NYU	CESNET	CESNET	0.99628	0.99748	0.00252	0.99509	0.00491
ResNet	CESNET	CESNET	0.99562	0.99643	0.00357	0.99480	0.00520
Endgame	MU	MU	0.99808	0.99966	0.00034	0.99650	0.00350
NYU	MU	MU	0.99817	0.99965	0.00035	0.99670	0.00330
ResNet	MU	MU	0.99787	0.99885	0.00115	0.99690	0.00310
Endgame	RWTH	RWTH	0.99851	0.99968	0.00032	0.99733	0.00267
NYU	RWTH	RWTH	0.99815	0.99916	0.00084	0.99713	0.00287
ResNet	RWTH	RWTH	0.99813	0.99900	0.00100	0.99726	0.00274

Table 1: Baseline results.

All classifiers perform similarly well using the different training/testing set combinations. In general, the classifiers are best in classifying samples from the RWTH University networks and worst in classifying samples from the CESNET networks.

In our next experiment, we determine the classification performance of the pre-trained global models that include public benign data (either top most domains or random domains from the Tranco list) in their training and are used as initial global models in our federated learning experiments. For each classifier we thus train two global models and classify each testing set. We present the results of this experiment in Table 2.

Classifier	Train Network	Test Network	ACC	TPR	FNR	TNR	FPR
Endgame	Tranco top	CESNET	0.82354	0.95492	0.04508	0.69217	0.30783
NYU	Tranco top	CESNET	0.77825	0.94876	0.05124	0.60775	0.39225
ResNet	Tranco top	CESNET	0.78243	0.94952	0.05048	0.61534	0.38466
Endgame	Tranco top	MU	0.69219	0.95492	0.04508	0.42946	0.57054
NYU	Tranco top	MU	0.66667	0.94876	0.05124	0.38457	0.61543
ResNet	Tranco top	MU	0.72029	0.94952	0.05048	0.49107	0.50893
Endgame	Tranco top	RWTH	0.60577	0.95492	0.04508	0.25663	0.74337
NYU	Tranco top	RWTH	0.62388	0.94876	0.05124	0.29901	0.70099
ResNet	Tranco top	RWTH	0.83266	0.94952	0.05048	0.71581	0.28419
Endgame	Tranco random	CESNET	0.80271	0.95054	0.04946	0.65488	0.34512
NYU	Tranco random	CESNET	0.78195	0.94310	0.05690	0.62080	0.37920
ResNet	Tranco random	CESNET	0.81406	0.93619	0.06381	0.69193	0.30807
Endgame	Tranco random	MU	0.69818	0.95054	0.04946	0.44583	0.55417
NYU	Tranco random	MU	0.69026	0.94310	0.05690	0.43742	0.56258
ResNet	Tranco random	MU	0.75457	0.93619	0.06381	0.57295	0.42705
Endgame	Tranco random	RWTH	0.61386	0.95054	0.04946	0.27719	0.72281
NYU	Tranco random	RWTH	0.68084	0.94310	0.05690	0.41858	0.58142
ResNet	Tranco random	RWTH	0.80826	0.93619	0.06381	0.68033	0.31967

Table 2: Pre-trained global model results.

Regarding the false positive rates which range from 28% to 74%, it can be seen that the global models that are trained using public data only are not able to classify the network traffic within any of networks from which we obtained benign data.

In the following, we investigate whether our approach of continuing the training of a pre-trained global model using private data (where the global model was initially trained using public data) has any negative influence on the classifier's classification performance compared to the baseline classifiers. To this end, we proceed with the training of the global models using the training sets of each party participating in federated learning and subsequently classify the respective testing sets. We present the results of this experiment in Table 3.

Classifier	Global Model	Train Network	Test Network	ACC	TPR	FNR	TNR	FPR
Endgame	Tranco top	CESNET	CESNET	0.99666	0.99758	0.00242	0.99574	0.00426
NYU	Tranco top	CESNET	CESNET	0.99654	0.99740	0.00260	0.99568	0.00432
ResNet	Tranco top	CESNET	CESNET	0.99573	0.99680	0.00320	0.99465	0.00535
Endgame	Tranco top	MU	MU	0.99804	0.99902	0.00098	0.99706	0.00294
NYU	Tranco top	MU	MU	0.99810	0.99938	0.00062	0.99681	0.00319
ResNet	Tranco top	MU	MU	0.99788	0.99908	0.00092	0.99667	0.00333
Endgame	Tranco top	RWTH	RWTH	0.99839	0.99931	0.00069	0.99748	0.00252
NYU	Tranco top	RWTH	RWTH	0.99840	0.99938	0.00062	0.99743	0.00257
ResNet	Tranco top	RWTH	RWTH	0.99809	0.99894	0.00106	0.99724	0.00276
Endgame	Tranco random	CESNET	CESNET	0.99675	0.99751	0.00249	0.99599	0.00401
NYU	Tranco random	CESNET	CESNET	0.99665	0.99815	0.00185	0.99515	0.00485
ResNet	Tranco random	CESNET	CESNET	0.99566	0.99621	0.00379	0.99511	0.00489
Endgame	Tranco random	MU	MU	0.99808	0.99913	0.00087	0.99702	0.00298
NYU	Tranco random	MU	MU	0.99811	0.99957	0.00043	0.99666	0.00334
ResNet	Tranco random	MU	MU	0.99786	0.99894	0.00106	0.99679	0.00321
Endgame	Tranco random	RWTH	RWTH	0.99850	0.99955	0.00045	0.99746	0.00254
NYU	Tranco random	RWTH	RWTH	0.99848	0.99954	0.00046	0.99741	0.00259
ResNet	Tranco random	RWTH	RWTH	0.99820	0.99920	0.00080	0.99721	0.00279

Table 3: Results of global models that are further trained using private data (no federation).

These results show that there are no significant differences between the classification performance of the further trained models compared to the baseline classifiers independent of the used global model.

In the following, we present the results of our federated learning experiments. For every combination of participating parties (four combinations for three parties) and for every global model and classifier combination we train a distinct model using federated learning which we assess based on the three different testing datasets. In total, we use five different classification scenarios to assess the classification performance of the models trained using federated learning and to compare them with the baseline classifiers. In the following, we describe the five scenarios:

1. **Baseline - Average:** averaged results of all baseline classifiers for a specific testing dataset.
2. **Baseline – Generalization:** averaged results of all baseline classifiers except of the baseline classifier that is trained on benign samples that origin from the same network as the benign samples included the testing dataset.
3. **Federated Learning – Average:** averaged results of all federated learning classifiers for a specific testing dataset, including every combination of participating parties.
4. **Federated Learning - Test Network in Federation:** averaged results of federated learning classifiers for a specific testing dataset, excluding all combinations of participating parties that do not include the party from which the benign testing samples originate from.
5. **Federated Learning – Generalization:** averaged results of federated learning classifiers for a specific testing dataset, excluding all combinations of participating parties that do include the party from which the benign testing samples originate from.

In Table 4, 5, and 6 we display the individual results for the Endgame, NYU, and ResNet classifiers, respectively. Additionally, we present the achieved classification results of the federated learning classifiers averaged over all testing datasets and for each classifier in 7 for convenience.

Classifier	Scenario	Global Model	Test Network	ACC	TPR	FNR	TNR	FPR
Endgame	Baseline - Average	-	CESNET	0.97906	0.99891	0.00109	0.95920	0.04080
Endgame	Baseline – Generalization	-	CESNET	0.97036	0.99967	0.00033	0.94105	0.05895
Endgame	Federated Learning - Average	Tranco top	CESNET	0.98644	0.99913	0.00087	0.97376	0.02624
Endgame	Federated Learning – Test Network in Federation	Tranco top	CESNET	0.98885	0.99908	0.00092	0.97861	0.02139
Endgame	Federated Learning – Generalization	Tranco top	CESNET	0.97924	0.99928	0.00072	0.95920	0.04080
Endgame	Federated Learning – Average	Tranco random	CESNET	0.98657	0.99906	0.00094	0.97407	0.02593
Endgame	Federated Learning – Test Network in Federation	Tranco random	CESNET	0.98897	0.99895	0.00105	0.97900	0.02100
Endgame	Federated Learning – Generalization	Tranco random	CESNET	0.97935	0.99942	0.00058	0.95927	0.04073

Endgame	Baseline – Average	-	MU	0.99519	0.99891	0.00109	0.99146	0.00854
Endgame	Baseline – Generalization	-	MU	0.99374	0.99853	0.00147	0.98895	0.01105
Endgame	Federated Learning – Average	Tranco top	MU	0.99654	0.99913	0.00087	0.99394	0.00606
Endgame	Federated Learning – Test Network in Federation	Tranco top	MU	0.99737	0.99917	0.00083	0.99556	0.00444
Endgame	Federated Learning – Generalization	Tranco top	MU	0.99404	0.99902	0.00098	0.98907	0.01093
Endgame	Federated Learning – Average	Tranco random	MU	0.99627	0.99906	0.00094	0.99348	0.00652
Endgame	Federated Learning – Test Network in Federation	Tranco random	MU	0.99724	0.99913	0.00087	0.99536	0.00464
Endgame	Federated Learning – Generalization	Tranco random	MU	0.99336	0.99888	0.00112	0.98783	0.01217
Endgame	Baseline – Average	-	RWTH	0.99747	0.99891	0.00109	0.99604	0.00396
Endgame	Baseline – Generalization	-	RWTH	0.99696	0.99852	0.00148	0.99539	0.00461
Endgame	Federated Learning – Average	Tranco top	RWTH	0.99815	0.99913	0.00087	0.99717	0.00283
Endgame	Federated Learning – Test Network in Federation	Tranco top	RWTH	0.99829	0.99916	0.00084	0.99743	0.00257
Endgame	Federated Learning – Generalization	Tranco top	RWTH	0.99772	0.99905	0.00095	0.99638	0.00362
Endgame	Federated Learning – Average	Tranco random	RWTH	0.99806	0.99906	0.00094	0.99705	0.00295
Endgame	Federated Learning – Test Network in Federation	Tranco random	RWTH	0.99825	0.99912	0.00088	0.99737	0.00263
Endgame	Federated Learning – Generalization	Tranco random	RWTH	0.99749	0.99889	0.00111	0.99610	0.00390

Table 4: Endgame federated learning results.

Classifier	Scenario	Global Model	Test Network	ACC	TPR	FNR	TNR	FPR
NYU	Baseline – Average	-	CESNET	0.97822	0.99876	0.00124	0.95768	0.04232
NYU	Baseline – Generalization	-	CESNET	0.96919	0.99941	0.00059	0.93898	0.06102
NYU	Federated Learning – Average	Tranco top	CESNET	0.98246	0.99968	0.00032	0.96525	0.03475
NYU	Federated Learning – Test Network in Federation	Tranco top	CESNET	0.98548	0.99961	0.00039	0.97135	0.02865
NYU	Federated Learning – Generalization	Tranco top	CESNET	0.97341	0.99989	0.00011	0.94694	0.05306
NYU	Federated Learning – Average	Tranco random	CESNET	0.98243	0.99974	0.00026	0.96513	0.03487
NYU	Federated Learning – Test Network in Federation	Tranco random	CESNET	0.98560	0.99970	0.00030	0.97150	0.02850
NYU	Federated Learning – Generalization	Tranco random	CESNET	0.97292	0.99986	0.00014	0.94599	0.05401
NYU	Baseline – Average	-	MU	0.99585	0.99876	0.00124	0.99293	0.00707
NYU	Baseline – Generalization	-	MU	0.99469	0.99832	0.00168	0.99105	0.00895
NYU	Federated Learning – Average	Tranco top	MU	0.99657	0.99968	0.00032	0.99346	0.00654
NYU	Federated Learning – Test Network in Federation	Tranco top	MU	0.99720	0.99972	0.00028	0.99468	0.00532
NYU	Federated Learning – Generalization	Tranco top	MU	0.99469	0.99957	0.00043	0.98982	0.01018
NYU	Federated Learning – Average	Tranco random	MU	0.99717	0.99974	0.00026	0.99460	0.00540

NYU	Federated Learning – Test Network in Federation	Tranco random	MU	0.99776	0.99974	0.00026	0.99578	0.00422
NYU	Federated Learning – Generalization	Tranco random	MU	0.99539	0.99974	0.00026	0.99105	0.00895
NYU	Baseline – Average	-	RWTH	0.99719	0.99876	0.00124	0.99561	0.00439
NYU	Baseline – Generalization	-	RWTH	0.99671	0.99856	0.00144	0.99485	0.00515
NYU	Federated Learning – Average	Tranco top	RWTH	0.99803	0.99968	0.00032	0.99638	0.00362
NYU	Federated Learning – Test Network in Federation	Tranco top	RWTH	0.99835	0.99974	0.00026	0.99697	0.00303
NYU	Federated Learning – Generalization	Tranco top	RWTH	0.99705	0.99950	0.00050	0.99460	0.00540
NYU	Federated Learning – Average	Tranco random	RWTH	0.99783	0.99974	0.00026	0.99592	0.00408
NYU	Federated Learning – Test Network in Federation	Tranco random	RWTH	0.99833	0.99980	0.00020	0.99686	0.00314
NYU	Federated Learning – Generalization	Tranco random	RWTH	0.99633	0.99953	0.00047	0.99312	0.00688

Table 5: NYU federated learning results.

Classifier	Scenario	Global Model	Test Network	ACC	TPR	FNR	TNR	FPR
ResNet	Baseline – Average	-	CESNET	0.97865	0.99809	0.00191	0.95920	0.04080
ResNet	Baseline - Generalization	-	CESNET	0.97017	0.99893	0.00107	0.94140	0.05860
ResNet	Federated Learning – Average	Tranco top	CESNET	0.98101	0.99944	0.00056	0.96259	0.03741
ResNet	Federated Learning – Test Network in Federation	Tranco top	CESNET	0.98349	0.99938	0.00062	0.96761	0.03239

ResNet	Federated Learning – Generalization	Tranco top	CESNET	0.97356	0.99961	0.00039	0.94751	0.05249
ResNet	Federated Learning – Average	Tranco random	CESNET	0.98232	0.99947	0.00053	0.96517	0.03483
ResNet	Federated Learning – Test Network in Federation	Tranco random	CESNET	0.98545	0.99937	0.00063	0.97152	0.02848
ResNet	Federated Learning – Generalization	Tranco random	CESNET	0.97295	0.99978	0.00022	0.94611	0.05389
ResNet	Baseline – Average	-	MU	0.99506	0.99809	0.00191	0.99202	0.00798
ResNet	Baseline – Generalization	-	MU	0.99365	0.99772	0.00228	0.98959	0.01041
ResNet	Federated Learning – Average	Tranco top	MU	0.99721	0.99944	0.00056	0.99499	0.00501
ResNet	Federated Learning – Test Network in Federation	Tranco top	MU	0.99785	0.99948	0.00052	0.99621	0.00379
ResNet	Federated Learning – Generalization	Tranco top	MU	0.99530	0.99930	0.00070	0.99130	0.00870
ResNet	Federated Learning – Average	Tranco random	MU	0.99737	0.99947	0.00053	0.99526	0.00474
ResNet	Federated Learning – Test Network in Federation	Tranco random	MU	0.99793	0.99954	0.00046	0.99631	0.00369
ResNet	Federated Learning – Generalization	Tranco random	MU	0.99569	0.99926	0.00074	0.99212	0.00788
ResNet	Baseline – Average	-	RWTH	0.99681	0.99809	0.00191	0.99552	0.00448
ResNet	Baseline – Generalization	-	RWTH	0.99615	0.99764	0.00236	0.99465	0.00535
ResNet	Federated Learning – Average	Tranco top	RWTH	0.99798	0.99944	0.00056	0.99653	0.00347
ResNet	Federated Learning – Test Network in Federation	Tranco top	RWTH	0.99833	0.99950	0.00050	0.99717	0.00283

ResNet	Federated Learning – Generalization	Tranco top	RWTH	0.99694	0.99925	0.00075	0.99463	0.00537
ResNet	Federated Learning – Average	Tranco random	RWTH	0.99773	0.99947	0.00053	0.99598	0.00402
ResNet	Federated Learning – Test Network in Federation	Tranco random	RWTH	0.99821	0.99957	0.00043	0.99684	0.00316
ResNet	Federated Learning – Generalization	Tranco random	RWTH	0.99628	0.99916	0.00084	0.99340	0.00660

Table 6: ResNet federated learning results.

Classifier	Scenario	Global Model	ACC	TPR	FNR	TNR	FPR
Endgame	Baseline - Average	-	0.99057	0.99891	0.00109	0.98223	0.01777
Endgame	Baseline - Generalization	-	0.98702	0.99891	0.00109	0.97513	0.02487
Endgame	Federated Learning – Average	Tranco top	0.99371	0.99913	0.00087	0.98829	0.01171
Endgame	Federated Learning – Test Network in Federation	Tranco top	0.99483	0.99914	0.00086	0.99053	0.00947
Endgame	Federated Learning – Generalization	Tranco top	0.99033	0.99911	0.00089	0.98155	0.01845
Endgame	Federated Learning – Average	Tranco random	0.99363	0.99906	0.00094	0.98820	0.01180
Endgame	Federated Learning – Test Network in Federation	Tranco random	0.99482	0.99907	0.00093	0.99058	0.00942
Endgame	Federated Learning – Generalization	Tranco random	0.99007	0.99906	0.00094	0.98107	0.01893
NYU	Baseline - Average	-	0.99042	0.99876	0.00124	0.98208	0.01792
NYU	Baseline - Generalization	-	0.98686	0.99876	0.00124	0.97496	0.02504
NYU	Federated Learning – Average	Tranco top	0.99235	0.99968	0.00032	0.98503	0.01497
NYU	Federated Learning – Test Network in Federation	Tranco top	0.99368	0.99969	0.00031	0.98767	0.01233
NYU	Federated Learning – Generalization	Tranco top	0.98838	0.99965	0.00035	0.97712	0.02288

NYU	Federated Learning – Average	Tranco random	0.99248	0.99974	0.00026	0.98522	0.01478
NYU	Federated Learning – Test Network in Federation	Tranco random	0.99390	0.99975	0.00025	0.98805	0.01195
NYU	Federated Learning – Generalization	Tranco random	0.98822	0.99971	0.00029	0.97672	0.02328
ResNet	Baseline - Average	-	0.99017	0.99809	0.00191	0.98225	0.01775
ResNet	Baseline - Generalization	-	0.98665	0.99809	0.00191	0.97522	0.02478
ResNet	Federated Learning – Average	Tranco top	0.99207	0.99944	0.00056	0.98470	0.01530
ResNet	Federated Learning – Test Network in Federation	Tranco top	0.99322	0.99945	0.00055	0.98700	0.01300
ResNet	Federated Learning – Generalization	Tranco top	0.98860	0.99939	0.00061	0.97781	0.02219
ResNet	Federated Learning – Average	Tranco random	0.99247	0.99947	0.00053	0.98547	0.01453
ResNet	Federated Learning – Test Network in Federation	Tranco random	0.99386	0.99949	0.00051	0.98823	0.01177
ResNet	Federated Learning – Generalization	Tranco random	0.98831	0.99940	0.00060	0.97721	0.02279

Table 7: Summary of federated learning results averaged over all testing datasets.

We can measure a performance gain on every testing dataset and for each classifier type when federated learning is used regardless of the used global model compared to the baseline classifiers. Federated learning decreases the false positive rate and improves the true positive rate for every classifier in all scenarios. Thus, private information sharing in the context of federated learning is beneficial as it clearly improves the classification performance.

In detail, comparing the results of the baseline evaluation scenarios with the ones which make use of federated learning (regardless of the used classifier and global model), it can be seen that both, the false positive rate as well as the true positive rate improve which is also reflected in the accuracy metric. We reckon the improvement of the true positive rate to be caused by the additional samples per DGA family included in the federated learning process. These samples help in the detection of samples which were not seen by a classifier during training as the classifier has more data to learn from and thus generalizes better on new input data.

The general improvements can be directly seen by comparing the results of the “Baseline – Average” with the “Federated Learning – Average” scenario.

To demonstrate the maximal possible gain in classification performance, we defined the “Federated Learning - Test Network in Federation” scenario. Here we present the

averaged results of the federated learning classifiers that were collaboratively trained including the party that provides the benign testing samples. We can observe the best classification results here as this is the best-case scenario for federated learning.

The worst case for the baseline classifiers as well as for the federated learning classifier is the “Generalization” scenario. Here, we present the averaged results of those classifiers which have no access to any samples of the testing networks. Comparing the baseline scenario with the federated learning one we can observe again an improvement in true positive rates but also in false positive rates. Thus, we reckon that by using benign samples of different sources the classifiers trained collaboratively using federated learning generalize better on unseen input data.

Lastly, we investigate the influence of the number of participating parties in federated learning on the classification performance for each testing network separately. Here, we only investigate the “Federated Learning - Test Network in Federation” scenario in order to evaluate the influence of the parties’ individual model updates in federated learning on the classification performance. We display the results of this experiment in in 8. In 9, we present summarized results that are averaged over all testing datasets for convenience.

Classifier	#Organizations	Global Model	Test Network	ACC	TPR	FNR	TNR	FPR
Endgame	2	Tranco top	CESNET	0.98998	0.99903	0.00097	0.98092	0.01908
Endgame	3	Tranco top	CESNET	0.98658	0.99918	0.00082	0.97398	0.02602
Endgame	2	Tranco top	MU	0.99761	0.99916	0.00084	0.99607	0.00393
Endgame	3	Tranco top	MU	0.99687	0.99918	0.00082	0.99455	0.00545
Endgame	2	Tranco top	RWTH	0.99830	0.99915	0.00085	0.99744	0.00256
Endgame	3	Tranco top	RWTH	0.99829	0.99918	0.00082	0.99739	0.00261
Endgame	2	Tranco random	CESNET	0.99007	0.99888	0.00112	0.98126	0.01874
Endgame	3	Tranco random	CESNET	0.98677	0.99907	0.00093	0.97447	0.02553
Endgame	2	Tranco random	MU	0.99738	0.99915	0.00085	0.99560	0.00440
Endgame	3	Tranco random	MU	0.99697	0.99907	0.00093	0.99487	0.00513
Endgame	2	Tranco random	RWTH	0.99827	0.99915	0.00085	0.99739	0.00261
Endgame	3	Tranco random	RWTH	0.99820	0.99907	0.00093	0.99733	0.00267
NYU	2	Tranco top	CESNET	0.98751	0.99953	0.00047	0.97548	0.02452
NYU	3	Tranco top	CESNET	0.98143	0.99976	0.00024	0.96309	0.03691
NYU	2	Tranco top	MU	0.99751	0.99969	0.00031	0.99532	0.00468

NYU	3	Tranco top	MU	0.99657	0.99976	0.00024	0.99339	0.00661
NYU	2	Tranco top	RWTH	0.99839	0.99973	0.00027	0.99706	0.00294
NYU	3	Tranco top	RWTH	0.99828	0.99976	0.00024	0.99679	0.00321
NYU	2	Tranco random	CESNET	0.98786	0.99964	0.00036	0.97609	0.02391
NYU	3	Tranco random	CESNET	0.98108	0.99982	0.00018	0.96234	0.03766
NYU	2	Tranco random	MU	0.99789	0.99970	0.00030	0.99608	0.00392
NYU	3	Tranco random	MU	0.99750	0.99982	0.00018	0.99518	0.00482
NYU	2	Tranco random	RWTH	0.99841	0.99980	0.00020	0.99703	0.00297
NYU	3	Tranco random	RWTH	0.99817	0.99982	0.00018	0.99652	0.00348
ResNet	2	Tranco top	CESNET	0.98542	0.99928	0.00072	0.97157	0.02843
ResNet	3	Tranco top	CESNET	0.97964	0.99958	0.00042	0.95969	0.04031
ResNet	2	Tranco top	MU	0.99788	0.99943	0.00057	0.99633	0.00367
ResNet	3	Tranco top	MU	0.99778	0.99958	0.00042	0.99598	0.00402
ResNet	2	Tranco top	RWTH	0.99833	0.99946	0.00054	0.99721	0.00279
ResNet	3	Tranco top	RWTH	0.99833	0.99958	0.00042	0.99708	0.00292
ResNet	2	Tranco random	CESNET	0.98774	0.99921	0.00079	0.97627	0.02373
ResNet	3	Tranco random	CESNET	0.98086	0.99968	0.00032	0.96203	0.03797
ResNet	2	Tranco random	MU	0.99791	0.99947	0.00053	0.99634	0.00366
ResNet	3	Tranco random	MU	0.99797	0.99968	0.00032	0.99626	0.00374
ResNet	2	Tranco random	RWTH	0.99823	0.99952	0.00048	0.99694	0.00306
ResNet	3	Tranco random	RWTH	0.99816	0.99968	0.00032	0.99664	0.00336

Table 8: Federated learning results split by number of participating parties.

Classifier	#Organizations	Global Model	ACC	TPR	FNR	TNR	FPR
Endgame	2	Tranco top	0.99530	0.99911	0.00089	0.99148	0.00852
Endgame	3	Tranco top	0.99391	0.99918	0.00082	0.98864	0.01136
Endgame	2	Tranco random	0.99524	0.99906	0.00094	0.99142	0.00858

Endgame	3	Tranco random	0.99398	0.99907	0.00093	0.98889	0.01111
NYU	2	Tranco top	0.99447	0.99965	0.00035	0.98929	0.01071
NYU	3	Tranco top	0.99209	0.99976	0.00024	0.98442	0.01558
NYU	2	Tranco random	0.99472	0.99971	0.00029	0.98973	0.01027
NYU	3	Tranco random	0.99225	0.99982	0.00018	0.98468	0.01532
ResNet	2	Tranco top	0.99388	0.99939	0.00061	0.98837	0.01163
ResNet	3	Tranco top	0.99192	0.99958	0.00042	0.98425	0.01575
ResNet	2	Tranco random	0.99463	0.99940	0.00060	0.98985	0.01015
ResNet	3	Tranco random	0.99233	0.99968	0.00032	0.98498	0.01502

Table 9: Summary of federated learning results split by number of participating parties.

Regardless of the used classifier and global model the averaged results of the classifiers which were trained collaboratively by only two parties achieve better classification scores than the federated learning classifier which was collaboratively trained by three parties. We reckon that this is caused by the fact that we federate the computed updates of the locally trained models after all local models have converged. In case of federated learning in a two-party setup, the influence of the weight updates of the two parties on the global model decreases by adding an additional party. This is an expected outcome since we evaluate the federated learning classifiers that were collaboratively trained including the party that provides the benign testing samples. In the final version of this deliverable we will investigate whether this holds also true for federated learning models that were trained through federation after each local training epoch. There, the results could be different as the classifiers trained in such a scenario could better generalize on unseen data.

The privacy implications caused by using private information sharing in the context of federated learning have to be investigated in the final version of this deliverable.

3.2.6 Planned Privacy Analysis

In the next iteration of this deliverable, we plan to present an evaluation of privacy in the DGA sharing scenarios. This includes an assessment of the existing threats against these scenarios which are Input Inference, more specifically Membership Inference and Model Inversion, Model Extraction and possibly also Poisoning attacks, as these mainly threaten only model correctness and not data privacy. This assessment will be quantified as objectively as possible, for instance by the success ratio of Membership Inference attacks, or by a sample distance metric as it is currently done in Deliverable D5.1 with the Levenshtein distance. For attacks that pose a sever threat towards privacy in sharing, sophisticated defense methods shall be evaluated, which can include training the model with a privacy-preserving training algorithm (e.g., with differential privacy [23]) before sharing model updates or hiding plain model updates with a bitmask computed via secure multiparty computation (SMPC). This bitmask would be applied before each party shares its updates and would cancel out during summation

of all model updates. Privacy-preserving techniques must be evaluated and compared regarding the negative performance impact (e.g., loss in accuracy) they likely carry with them. Using a SMPC bitmask in federated learning will likely have zero impact on the classification performance compared to a model obtained by regular federated learning.

3.3 Application Profiling

The background for application profiling in the context of sharing is similar in all deliverables of WP5. Hence, the following paragraph can also be found in the Deliverables D5.1 and D5.3.

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, such as 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.

3.3.1 Sharing Scenarios

In the following, we will describe the sharing scenarios that we want to evaluate for the application profiling showcase. Similar to the scenarios in Deliverable D5.1, our focus so far was to finalize the approaches for building local models, as well as preparing the scenarios in the sharing context by developing tools for data sharing. In this task, no data and no local models are shared to build a global model. Similar to the DGA detection approaches, this leaves us with two approaches: federated learning and the teacher-student approach.

Federated learning is an approach to collaboratively train machine learning models. As described in Deliverable D5.1, this is currently not our primary focus for application profiling. Nonetheless, since we will experiment with federated learning in the context of DGA detection, we will try to apply the same approaches on the machine learning models we develop for application profiling, in case they yield comparable results to process mining.

On the other hand, the teacher-student approach is not limited to machine learning. The basic idea is to label a public dataset by querying the local models (teachers) of organizations without sharing them. The global model (student) can then be trained with the labeled dataset. However, in the case of application profiling, this is only applicable if the global model is a machine learning model. Process mining does not need a labeled dataset as it only discovers the processes in the available data. This can be compared to unsupervised learning in the context of machine learning. However, when

feeding the data into the process mining model, we can compute how well it fits the model, i.e. using a value between 0 and 1. The idea is to learn this function with a neural network using the teacher-student approach. However, at the time of this deliverable, we did not conduct any experiments. Hence, it is unclear whether this approach works.

4 Conclusion

In this deliverable, we presented the first results of Task 5.3: “Federated learning of a global model without sharing local models”. We briefly discussed the context of this task within the overall scope of the SAPPAN project and explained its general concept. 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 first defined two different sharing scenarios (federated learning and the teacher-student approach) which each come with two different variations. We then presented our evaluation results for the sharing scenario “federated learning – federation after model convergence” and could demonstrate that private information sharing in the context of federated learning for DGA detection is beneficial as it clearly improves the classification performance. In all investigated cases, federated learning decreases the false positive rate and improves the true positive rate for all three considered classifiers.

In the final version of this deliverable, we plan to comparatively evaluate the remaining sharing scenarios for DGA detection and to conduct a privacy analysis. 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 use case of application profiling we plan to evaluate how well the federated learning and teacher-student approaches work for application profiling, by comparing them with the global models developed in T5.1 and T5.3.

References

[1] 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.

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

[3] 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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

[10] 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

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

[12] 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.

[13] 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

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

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

[21] 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.

[22] V. L. Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In Network and Distributed System Security Symposium. Internet Society. 2019.

[23] C. Dwork. Differential Privacy. 2006.

[24] 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.

[25] 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.

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

[27] 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).

[28] 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.

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

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