Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D5.6 Global model without sharing local models, final version (M30)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file: Deliverable 5.6
Document version: 1.0
Document owner: Arthur Drichel (RWTH)

Work package: WP5
Task: T5.3 Federated learning of a global model without sharing local models
Deliverable type: Report
Delivery month: M30
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)	2021-09-15	First version of outline including first bullet points.
0.2	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH)	2021-10-01	Finalised outline and structure.
0.3	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH)	2021-10-27	First complete version ready for review.
0.4	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Sebastian Schäfer (RWTH), Mehdi Akbari Gurabi (FIT), Martin Zadnik (CESNET)	2021-10-29	Updated version with incorporated feedback.
1.0	Arthur Drichel (RWTH)	2021-10-29	Final version ready to submit.

Internal review history:

Reviewed by	Date	Summary of comments
Mehdi Akbari Gurabi (FIT)	2021-10-28	Comments regarding technical details and wording
Martin Zadnik (CESNET)	2021-10-29	Minor comments regarding grammar and spelling

Executive Summary

This deliverable is the final version of the deliverable D5.5 which relates to the task T5.3 "Federated Learning of a global model without sharing local models". Its goal is to learn a global model, similar to the tasks T5.1 and T5.2 but using a different approach to intelligence sharing. Therefore, the deliverables D5.2, D5.4, and D5.6 have some overlaps, especially with respect to the background, motivation, and 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. This deliverable focuses on different collaborative machine learning approaches to the Domain Generation Algorithm (DGA) detection methods developed in WP3. In detail, we conduct a comprehensive collaborative learning study for DGA) detection, including a total of 13,440 evaluation runs. In two real-world scenarios, we evaluate a total of eleven different variations of collaborative learning using three different state-of-the-art classifiers. We show that collaborative machine learning can reduce the false-positive rate (FPR) by up to 51.7%. However, while DGA detection benefits from collaborative machine learning, not all approaches and classifier types profit equally. We conclude our comprehensive study with a discussion of the privacy threats implicated by the different collaborative machine learning approaches.

Contents

1 Introduction

In the deliverable D5.6, we report the continuation of our work from the initial version of this deliverable D5.5. In D5.5, we described the experiments and approaches to build a global model without sharing local models or anonymised data. For some of the experiments, we already presented initial results. The final results are now described in this document.

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 organisation by collaboration. In WP3, one of the tasks is to develop local detection and response mechanisms. In WP5, we want to utilise 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, these approaches are the most privacy-preserving ones. However, because no data or complete models can be shared, approaches to learning global models are severely limited. Still, the goal is to 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 focus on the use case of DGA detection because the collaborative machine learning 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.

This document is structured as follows. First, we briefly outline the context of this task in the overall scheme of SAPPAN project. Second, we briefly describe the general idea of this task and introduce the showcase of DGA detection on which this deliverable focuses. We present the evaluation setup used for a comparative evaluation of different collaborative machine learning approaches to DGA detection. This setup includes the selected state-of-the-art machine learning classifiers, the different data sources used, the different sharing approaches examined, as well as a description of the methodology and the evaluation scenarios. Next, we present the evaluation results and additionally compare the approaches developed in this deliverable with the approaches of the deliverable D5.4 (global model based on shared local models). Afterwards, we discuss the privacy aspects of beneficial sharing approaches. Finally, we briefly comment on the showcase of application profiling before we conclude this deliverable with a summary of the results.

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 D5.2 and D5.4.

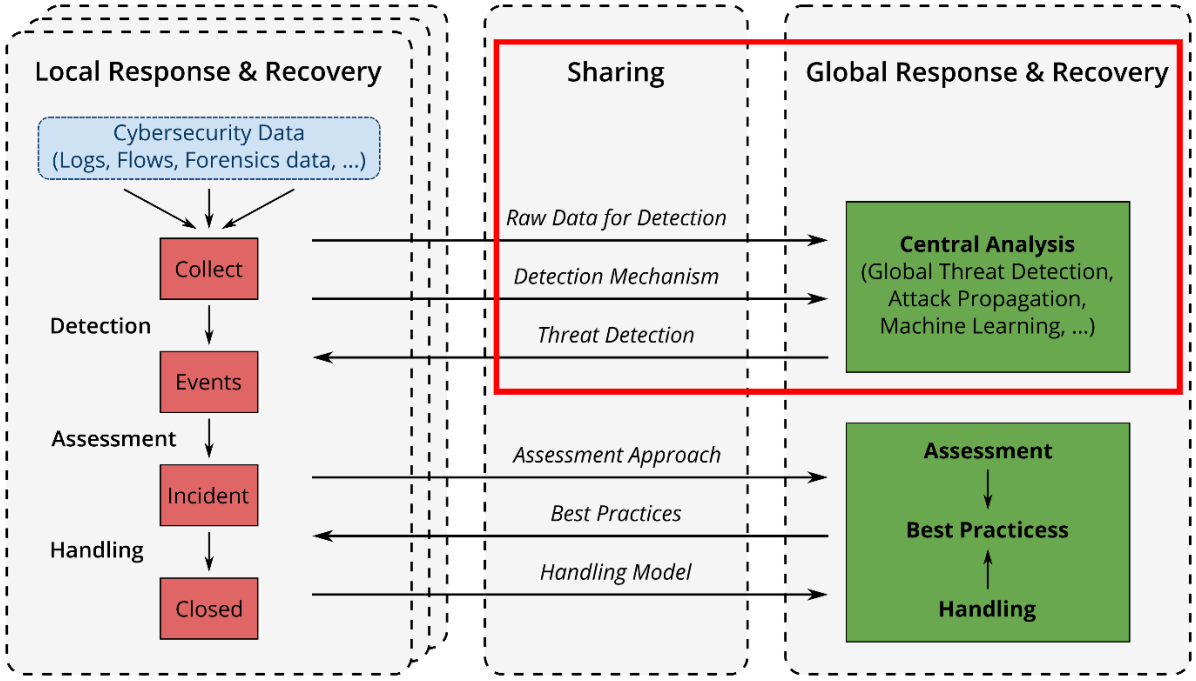

Fig. 1: SAPPAN scheme regarding local and global response and recovery.

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 visualisation. 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 utilise the data and models developed in the Task T3.3 on the local level by sharing them among multiple organisations to build global detection models. The goal is to end up with global detection mechanisms that are superior to the local ones. Another flavour of sharing in the scope of SAPPAN is sharing among a cybersecurity service provider or vendor and various user groups of its customer organisations. 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 anonymised 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 provide certain levels of privacy and efficiency, including cost-efficiency.

For task T5.3, we develop approaches to building global models without sharing local models or anonymised data. The aim is to use the shared intelligence of several organisations to create detection models that are superior to local ones, e.g., with increased accuracy and reduced false positive rate.

3 Federated Learning of a Global Model without Sharing Local Models

The goal of this task is to create a global model without sharing either anonymised 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, in this task the least amount of information is shared which might lead to the most privacy-preserving approaches. The two principles for creating a global model for this task are Federated Learning, in which only updates to a model are shared, and the Teacher-Student approach, in which the local models are queried using a local dataset.

3.1 Domain Generation Algorithm (DGA) Detection

We use the Domain Generation Algorithm (DGA) detection use case to analyse and compare the benefit of private data sharing within the deliverables D5.2 (global model based on shared anonymised data), D5.4 (global model based on shared local models), and D5.6 (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.

We presented DGA detection in D3.4 (Algorithms for Analysis of Cybersecurity Data) in detail. Additionally, we discussed the most important aspects in the initial version of this deliverable. However, since the following information is essential to understand the evaluation and the privacy analysis, we shortly repeat the most important parts.

Note, we have published the results of our comprehensive study on collaborative machine learning for DGA detection in the research paper "The More, the Better? A Study on Collaborative Machine Learning for DGA Detection" [22]. Therefore, parts of the following sections were previously published in and adapted from [22].

3.1.1 Background

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 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 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 of the C2 server.

The occurring NXDs within a network that are caused by the non-resolvable queries can be analysed 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. [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 the 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. In SAPPAN, we counteracted the explainability problem of deep learning classifiers by the development of a visual analytics system [17], which tries to bridge the gap between the predictions of deep neural networks and human-understandable features. The results were reported in the deliverables D3.8 and D3.9. Additionally, we reported the devel-

oped feature-based DGA multiclass classifier EXPLAIN [23] in the SAPPAN deliverable D5.2. This classifier is explainable by design, as its predictions are easier to trace back to features.

3.1.2 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) analyse the provided level of privacy of the collaboratively trained classifier.

First, we present the currently best contextless feature-based approach to 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 optimisation 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 a 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 the 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 ".Ml".

3.1.3 Data Sources

The main goals of our evaluation are (1) to determine whether we can improve the classification performance by leveraging different approaches to 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 D5.2, D5.4, and D5.6 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.

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 95 different known DGAs. We make use of all available data up to 2020-09-01.

Benign data

We obtain benign labelled NXDs from four different sources. Three of the sources are networks of project partners namely CESNET, Masaryk University, and RWTH Aachen University. It is crucial for a comprehensive study on collaborative machine learning to obtain real-world data from multiple parties. Fortunately, Siemens AG, a partner in another research project, provided us with additional data for our analysis. Due to this rich data, we are able to conduct collaborative machine learning experiments that are similar to a real-world setting. Moreover, the different benign data sources enable us to investigate whether collaboratively trained classifiers generalise well to different networks.

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 the classification performance of a 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 number of unique samples obtained from each source for benign data.

RWTH Aachen University (RWTH): We obtained a one-month recording of September 2019 from the central DNS resolver of RWTH Aachen University which is located in Germany. This recording comprises approximately 26 million unique benign NXDs that originate from academic and administrative networks, student residences' networks, and networks of the university hospital of RWTH Aachen.

Masaryk University (MU): We obtained a one-month recording from mid-May 2020 until mid-June 2020 from the networks of Masaryk University which is located in the Czech Republic. This recording contains approximately 8 million unique benign samples.

CESNET: We received additional benign samples from CESNET: an association of universities of the Czech Republic and the Czech Academy of Sciences consisting of 27 members in total. CESNET operates and develops the national e-infrastructure for science, research, and education. From this data source, we obtained a subset of occurred NXDs from the day recording of 2020-06-15. In total, we obtained approximately 362k unique samples.

Siemens: We obtained a one-month recording of July 2019 that comprises approximately 21 million unique NXDs from several DNS resolvers of Siemens AG which is a large company that operates in Asia, Europe, and the USA.

3.1.4 Sharing Approaches

In each deliverable (D5.2, D5.4, D5.6), we investigate different sharing approaches to the use case of DGA detection. In this deliverable, we investigate two collaborative learning approaches - Federated Learning and the Teacher-Student approach.

1. Federated Learning (FL)

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 randomly initialised model at the beginning. Then, each party individually improves the current model using own private data and subsequently summarises 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 summarise, 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 to 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 initialised 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 publicly available data (e.g., Alexa or Tranco top domain names [25] 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 Model Epoch

This approach is similar compared to the previous one but differs in the type and number 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 to federated learning might have an influence on the global model's classification performance as well as on the provided level of privacy.

2. Teacher-Student (T/S) 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 unlabelled 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.

3.1.5 Comprehensive Collaborative Machine Learning Study

In the following, we present a comprehensive study on collaborative machine learning approaches to deriving a global model without sharing local models or anonymised data. First, we provide an overview of our evaluation setup, including our dataset generation scheme and the evaluation methodology used. Subsequently, we present different sharing scenarios derived from research questions on possible real-world application environments for trained classifiers. These sharing scenarios are used to assess the performance of the different sharing approaches. Finally, we present the results of the study including a direct comparison to the sharing approaches developed in the deliverable D5.4.

3.1.5.1 Dataset Generation

We first describe our process of generating suitable datasets for our experiments using the above data sources. We provide an illustration of this process in Fig. 2 for convenience.

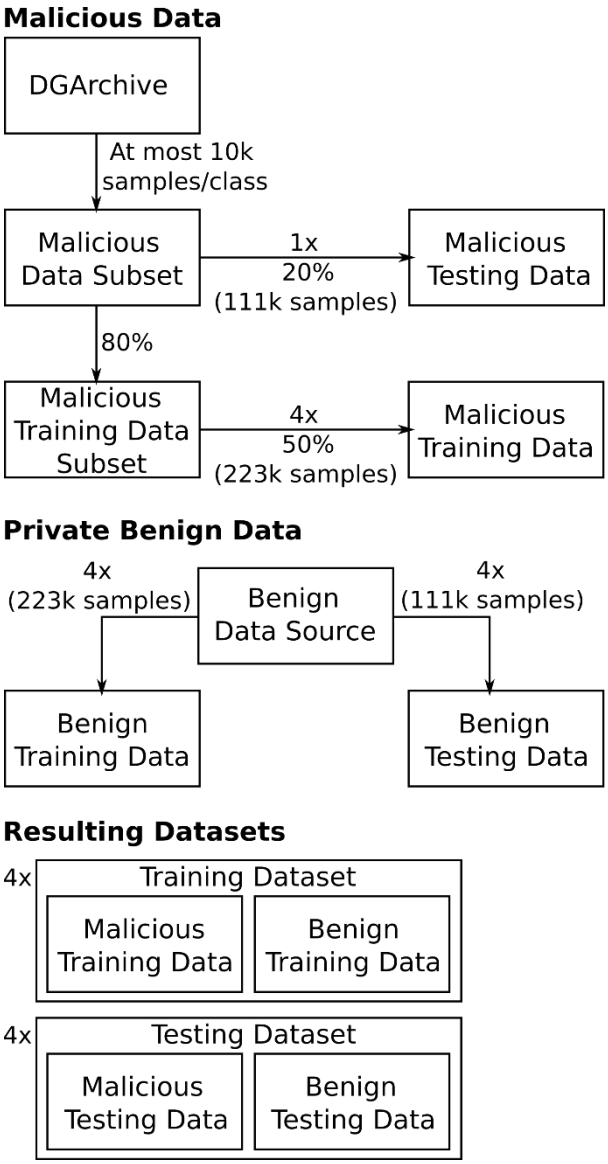

Fig. 2: Datasets generation scheme.

In order to create diverse datasets and to cope with the large number of available training samples, we first subsample our malicious labelled data into a smaller set that includes at most 10k samples per DGA family. We include all samples for DGA families for which less than 10k samples are available. Thereby, we also include samples from underrepresented DGA families. We do this because in [24] we showed that by including a few samples to the training of a classifier, its detection performance for underrepresented DGAs can be increased significantly without reducing its detection rates for well-represented DGAs.

From the selected subset we then split 20% (approximately 111k samples) stratified across all included DGA families for the test sets. For each of our benign data sources, we select individual malicious training data by subsampling 50% (approximately 223k samples) from the remaining malicious labelled samples. By subsampling from a larger common pool of malicious samples, we can create four training sets that contain both duplicate and unique malicious samples. We do this because, in a real-world scenario, it is very likely that the collaborating parties are using overlapping sets of malicious

samples. Note, in contrast to the benign labelled samples, the malicious samples are available in public repositories and are not privacy-sensitive.

The benign samples are not shared between different training sets as they are considered to be the main privacy-critical aspect of the collaborative DGA detection use case. We carry out a similar selection process for the benign training and testing samples. From each of the four benign data sources, we randomly subsample the same amount of benign training and testing samples as we did for malicious training and testing samples, respectively.

We use these data selections to create four training and testing dataset pairs, one for each benign data source. To this end, we combine the respective malicious and benign data selections to balanced training and testing datasets. Note that during this dataset generation we ensured that the samples included in the training and testing datasets are completely disjoint. Each of the four training and testing datasets includes approximately 446k and 223k samples, respectively.

Additionally, we create two balanced training datasets that include publicly available benign data using the same generation process. These datasets are used to train initial global models for our Federated Learning experiments. The public benign data originates from the Tranco list [25] which contains a ranking of the most popular domains that has been hardened against manipulation. Using this data we create two datasets, one contains the top entries of the list, while the benign data of the other dataset consists of random samples.

3.1.5.2 Methodology and Sharing Scenarios

Using these datasets, we are able to precisely measure the influence of collaborative machine learning on the classification performance of various classifiers. To obtain meaningful results, we repeat the whole dataset generation process five times and thereby create 20 individual training and testing dataset pairs which include malicious labelled samples from DGArchive and benign data from the four benign data sources. By this means, we also generate ten training datasets used in the Federated Learning experiments for training an initial global model using publicly available data. In the following, we repeat every experiment five times and present the averages of the individual results. Note, the datasets are generated similarly to a five-fold cross validation, i.e., the testing datasets are completely disjoint with both the training datasets and the testing datasets within the repetitions.

In the following, we exclude the feature-based approach FANCI from our study and concentrate on the three deep learning based approaches (B-Endgame, B-NYU, and B-ResNet) as neither Feature Extractor Sharing (which is examined in the deliverable D5.4) nor Federated Learning is possible using a feature-based approach. However, feature-based approaches to collaborative machine learning are analysed in detail in the deliverable D5.2. In this work, we train all deep learning classifiers using early stopping with a patience of three epochs to avoid overfitting and assess their performance during training on holdout sets that consist of random 5% splits of the used training data.

Additionally, in our comprehensive collaborative machine learning study, we focus on DGA binary detection. The reason for this is that 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. The difference between the training samples used in the binary and multiclass classification task is that the malicious samples of the multiclass task are additionally labelled as the DGA that generated a specific domain. However, as this information is not privacy-sensitive we focus on the binary classification task. For simplicity, in the following, we always refer to the binary versions of the three investigated deep learning classifiers as Endgame, NYU, and ResNet.

To measure the impact of collaborative machine learning on classification performance, we evaluate all possible combinations of participants for each examined approach. Overall, our comprehensive study consists of 13,440 evaluation passes, 9480 evaluations (including the baseline evaluations) are performed for this deliverable. The total number of individual experiments differs between the various collaborative machine learning approaches. In the following, we provide an overview of the experiments done per each investigated approach. We first present the baseline.

Baseline

All sharing approaches are compared against the baseline to evaluate their performance. The baseline evaluations are similar to traditional training and testing of a classifier using training data from a single organisation. Each organisation trains its own model using its private benign data and malicious training samples from DGArchive. No training data is shared among any organisations and also no global model is derived.

We train one classifier for each of the four organisations and evaluate them on every available test dataset. To this end, we perform five repetitions of training and testing for four organisations (RWTH, MU, CESNET, Siemens) and three classifier models (Endgame, NYU, ResNet), thus perform 5 * 4 * 4 * 3 = 240 baseline evaluations in total.

Federated Learning

Federated learning (FL) [26] is a technique to train a classifier collaboratively without sharing local data. First, a global model is initialised that is shared among all participants in the collaborative training. This global model can either be randomly initialised using standard initialisation methods or can be pre-trained using non-sensitive public data. In this work, we evaluate FL using three different initial global models, one that is randomly initialised (**Random Model**), and two pre-trained models. For pre-training a global model, we make use of malicious samples from DGArchive and benign domain names from the Tranco list [25]. This list contains a ranking of the most popular domain names, which is also protected against manipulation. One pre-trained model uses the top entries of the Tranco list for benign domain names, while the other model uses random samples. In the following, we refer to these models as **Tranco Top** and **Tranco Random**, respectively. After the global model is shared among all participants an iterative training procedure is performed. The global model is trained locally by each organisation using its private training data for each federation step. The weight updates to the global model in each federation step are then shared with all other participants,

such that everyone can now average the weight updates of the current step and add them to the global classifier's weights. Thereby, each party obtains the same global model that is then used within the next federation step. This iterative training process continues until the global model converges. The only data shared between the organisations are the model weight updates after each federation step and the initial global model. In this work, we investigate two federation approaches. In the first approach, we federate after each local model epoch (**Federation after Model Epoch**), while in the second approach we only federate once after all local models have converged (**Federation after Model Convergence**).

Here, we evaluate classifiers on all four different testing datasets, regardless of the combination of local classifiers used. Thereby, we are able to measure the generalisation capability of classifiers on benign data from unknown networks. Hence, we perform five repetitions using eleven organisation combinations (when using four benign data sources, there are only eleven possible combinations of organisations for building a combined model), three initial global models (Tranco Top, Tranco Random, Random Model), two possibilities for federation (after Model Epoch, after Model Convergence), four test datasets, and three classifier models (5 * 11 * 3 * 2 * 4 * 3 = 3960 evaluation passes).

Teacher-Student

The second examined sharing approach is based on a Teacher-Student (T/S) setup. Here, an organisation queries trained classifiers of other organisations (teachers) in order to obtain labels for its data. This labelled data is then used by the querying organisation to train an own classifier (student). Using this approach, the teacher classifiers are not exposed to the organisation that is training the student classifier. Thereby, white-box attacks against the privacy of an organisation that provides the labelling service are not applicable. Usually, more than one teacher is involved in the labelling process of a training sample, thus the individual labels or scores need to be combined. We examine two approaches: (1) majority voting on binary labels (Hard Labels) and (2) soft labelling by averaging confidence scores. A tie is possible when using the majority voting system with an even number of participants. In such a case, we resort to the soft labels approach, where we average all predictions and predict a domain name as malicious if the average is greater than 0.5 and benign otherwise. For this approach no global shared model is trained, instead, every party again derives an individual model.

We train classifiers that are similar to the baseline classifiers as teacher models for this approach. Here, we additionally need a training dataset that is labelled by the teachers and used for training a student classifier. Thus, in total, we perform five repetitions using eleven possible organisation combinations with four training datasets, two teacher result combination approaches (Hard and Soft labelling), four test datasets, and three classifier models (5 * 11 * 4 * 2 * 4 * 3 = 5280 evaluation passes).

Sharing Scenarios

The sharing approaches are evaluated in different scenarios which are derived from research questions on possible real-world application environments for trained classifiers.

Best Case: In this scenario, multiple network operators jointly train a classifier and are mostly interested in a good performance in their own networks. This is related to the following research question: is collaborative training beneficial for organisations that mostly classify data from their own distribution? In this best-case scenario, we provide averaged results for classifiers that are evaluated only on the test datasets containing samples coming from the organisations involved in the training.

Average Case: The average of all evaluations is used as a general performance indicator of the trained classifiers for each collaborative machine learning approach. We use this scenario for a comparative evaluation of the different sharing approaches.

Worst Case: The worst case scenario is contrasting the best case scenario. Here, classifiers are evaluated on all test datasets that contain samples from organisations that have not participated in the classifier training. Using this scenario, we examine the generalisation capability of collaboratively trained classifiers (i.e., whether the classifiers improve in their detection performance for samples originating from different networks).

3.1.5.3 Evaluation Results

In this subsection, we present the results of our comprehensive study. First, we highlight differences between the four organisations' data and provide an overview of the performance in the three sharing scenarios. Subsequently, we present the results of our comparative evaluation. Finally, we analyse the effect of the number of participants in collaborative machine learning.

Network Differences & Sharing Scenarios

To better explain the actual evaluation steps and to detail the calculations done for the different sharing scenarios we present the average scores for five repetitions of the baseline experiment using the Endgame classifier in Table 1. We provide the results for the Endgame classifier as an example. The results for the NYU and ResNet models are similar.

Train Network	Test Network	ACC	TPR	FPR
RWTH Aachen University	**RWTH Aachen University**	0.99851	0.99968	0.00267
	Masaryk University	0.99304	0.99968	0.01359
	CESNET	0.96892	0.99968	0.06184
	Siemens	0.99834	0.99968	0.00300
Masaryk University	RWTH Aachen University	0.99717	0.99966	0.00532
	Masaryk University	0.99808	0.99966	0.00350
	CESNET	0.97180	0.99966	0.05606
	Siemens	0.99853	0.99966	0.00259
CESNET	RWTH Aachen University	0.99674	0.99739	0.00390
	Masaryk University	0.99444	0.99739	0.00852
	CESNET	0.99645	0.99739	0.00450
	Siemens	0.99771	0.99739	0.00196
Siemens	RWTH Aachen University	0.98923	0.99968	0.02122
	Masaryk University	0.99037	0.99968	0.01894
	CESNET	0.96833	0.99968	0.06302
	Siemens	0.99888	0.99968	0.00192
Best Case		0.99798	0.99910	0.00315
Average Case		0.99103	0.99910	0.01703
Worst Case		0.98872	0.99910	0.02166

Table 1: Averaged baseline results for endgame classifier.

Here we list the average scores for the five repetitions of Endgame classifiers per training dataset used and per test dataset separately. The true positive rates (TPRs) per training network are equal for all test networks as we use the same malicious samples within all four test datasets within a repetition. The best false positive rates (FPRs) on the individual test networks are always achieved by the classifiers that were trained using benign samples which originate from the same network as the testing samples. This is expected since those classifiers are specifically trained to extract and classify characteristics of the benign domain names from the respective network. For example, benign samples from different networks may miss certain features or exhibit other characteristics. The average of the table entries where the train network is equal to the test network represents the best-case scenario and is presented at the bottom of the table. The classifiers trained using benign samples from distinct networks achieve different results for the individual test datasets. Samples from CESNET are most commonly classified wrong. In some cases, the FPR for these evaluations is even greater than 6%. We reckon that this is due to the fact that the samples from this network are the most diverse as they originate from over 27 different organisations. Moreover, we filtered out the intersection of the samples from the Masaryk University network with the samples from the CESNET as both networks are interconnected. Thereby, we likely removed easily recognisable samples that are naturally occurring in both networks. This could be the reason for the larger FPRs for Masaryk University and CESNET compared to those for the other two networks. Similarly to the best case, we provide the results for the average and worst case in the lower part of Table 1. While the average case is calculated using the average of all table entries, the worst case only contains the results of the entries for which the train network differs from the test network. The results for the different sharing scenarios are not of interest for the baseline evaluation. As could be expected, the best case results are better than the average case results, which are better than the worst case results.

Sharing Approaches

In this section, we compare the different approaches to collaborative machine learning. First, for comparison and to show that training on publicly available data is not enough for DGA detection on private data, we display the averaged results achieved by the two different types of pre-trained models that we use within our Federated Learning experiments for all three classifier types over all test datasets in Table 2.

Classifier	Benign Data	ACC	TPR	FPR
Endgame	Tranco Top	0.68329	0.95492	0.58834
	Tranco Random	0.67730	0.95054	0.59594
NYU	Tranco Top	0.68612	0.94876	0.57652
	Tranco Random	0.71336	0.94310	0.51637
ResNet	Tranco Top	0.78667	0.94952	0.37617
	Tranco Random	0.79899	0.93619	0.33821

Table 2: Results of pre-trained models using public data.

All six trained classifiers yield high FPRs between 33% and 59%, indicating that training on publicly available data alone is insufficient for classifying privacy-sensitive domain names.

In Table 3, we present the results for the average case, for all classifiers, and all collaborative machine learning approaches examined in this deliverable.

Approach	Endgame			NYU			ResNet		
	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR
Baseline	0.99103	0.99910	0.01703	0.99151	0.99903	0.01600	0.99102	0.99844	0.01640
FL: Random Model - Model Convergence	0.76755	0.55131	0.01621	0.98300	0.98764	0.02163	0.98671	0.99278	0.01937
FL: Random Model - Model Epoch	0.99396	0.99968	0.01177	0.99392	0.99983	0.01199	0.99511	0.99935	0.00913
FL: Tranco Top - Model Convergence	0.99336	0.99958	0.01286	0.99325	0.99975	0.01326	0.99291	0.99974	0.01393
FL: Tranco Top - Model Epoch	0.99553	0.99956	0.00850	0.99400	0.99981	0.01181	0.99491	0.99922	0.00940
FL: Tranco Random - Model Convergence	0.99365	0.99946	0.01216	0.99360	0.99978	0.01257	0.99268	0.99967	0.01431
FL: Tranco Random - Model Epoch	0.99565	0.99952	0.00823	0.99413	0.99978	0.01153	0.99498	0.99929	0.00934
T/S: Soft Labels	0.98979	0.99963	0.02006	0.99029	0.99965	0.01908	0.99001	0.99949	0.01948
T/S: Hard Labels	0.98966	0.99968	0.02036	0.99017	0.99965	0.01930	0.99026	0.99950	0.01898

Table 3: Results of the average case including all classifier types for the collaborative machine learning approaches of D5.6.

In order to assess whether the collaborative training is beneficial, we additionally provide the baseline results at the top of the table. For convenience, we colour table entries red if the scores are worse than the ones of the baseline and green otherwise.

All approaches except for the FL setting Random Model - Model Convergence achieve better TPRs than the baseline. This is an expected outcome because the training datasets used by the individual organisations contain additional malicious labelled samples from which a collaboratively trained classifier can learn. Thus, due to collaboration, intelligence about additional malicious labelled training samples is combined in the jointly trained classifiers. The only exception is the FL setting Random Model - Model Convergence. In this setting, we use a randomly initialised model and federate the updates of the local models after they converge. While the NYU and the ResNet model are still functional and only achieve slightly worse classification scores, the TPR of the Endgame classifier falls from over 99.9% (baseline) to 55%. We reckon the reason to be that the model updates from a randomly initialised model to a fully converged model vary quite large and the individual organisations optimise their models to different local optima. Averaging and applying all model updates may result in a non-optimal global model. The Endgame model is far more affected by this compared to the CNN-based NYU and ResNet model. This is due to the fact that RNNs are processing inputs sequentially. Averaging the weight updates of fully converged models that are used to process sequential data can thus result in a non-functional global model.

In the following, we exclude the FL setting Random Model - Model Convergence from our study and mainly focus on the FPR for our assessment. All other FL scenarios lead to an improvement compared to the baseline results. Here, the Endgame model performs significantly better in scenarios where a pre-trained initial global model was used. We assume that this is due to the fact that when using a pre-trained model, there are significantly fewer gradients towards local optima for participants to optimise their models. Similar to the FL setting Random Model - Model Convergence, this is an important property, especially for RNN-based classifiers. In contrast, the ResNet model achieves the best results using the randomly initialised model. In all FL settings, federating after each model epoch achieves better results than federating after model convergence.

Ensemble classification leads to worse results than the baseline. Furthermore, it makes little difference whether soft or hard labels are used.

While the absolute improvement achieved by collaborative machine learning may seem rather small, the relative reduction in the FPR is significant and could be decisive for the real-world application of classifiers. Compared to the baseline classifiers, the best FL settings achieve on average a FPR reduction of 51.7%, 27.9%, and 44.3% for Endgame, NYU, and ResNet, respectively.

In summary, additional malicious samples in collaborative machine learning improve the TPRs for all sharing approaches. In the average-case scenario, only the collaborative machine learning approach FL is advantageous for the use case of DGA detection. Using federation after model epoch leads to better results than federation after model convergence in FL.

Collaboration Analysis

In this subsection, our goal is to determine whether an increasing number of participants positively or negatively affects the classification performance of jointly trained classifiers. To this end, we investigate two scenarios.

In the first scenario, we make use of the best-case scenario. Here, organisations want to use jointly trained classifiers to classify samples from their own networks most of the time. Thus, our goal is to determine whether the classification performance on those samples increases or decreases with an increasing number of participants. Thereby, organisations can decide whether or not it makes sense to use a collaboratively trained classifier for their own network.

In the second scenario, which represents the worst case, we want to determine whether an increasing collaboration improves the generalisation capabilities of the classifiers and thus the classification performance on samples from external networks.

We use the FPR as a proxy to determine the performance of the classifiers. Table 4 presents the achieved FPR scores for the different collaborative machine learning approaches and classifier types separated by the number of participants.

Approach	Parties	Best Case			Worst Case		
		Endgame	NYU	ResNet	Endgame	NYU	ResNet
Baseline	-	0.00315	0.00328	0.00326	0.02166	0.02024	0.02078
FL: Random Model - Model Epoch	2	0.00464	0.00620	0.00465	0.02032	0.01886	0.01668
	3	0.00738	0.00922	0.00516	0.02122	0.01786	0.01473
	4	0.01120	0.01120	0.00624	-	-	-
FL: Tranco Top - Model Convergence	2	0.00803	0.00783	0.00856	0.01769	0.01841	0.01924
	3	0.01147	0.01170	0.01230	0.01723	0.01834	0.01877
	4	0.01267	0.01365	0.01415	-	-	-
FL: Tranco Top - Model Epoch	2	0.00361	0.00535	0.00410	0.01571	0.01905	0.01655
	3	0.00491	0.00879	0.00597	0.01492	0.01830	0.01573
	4	0.00591	0.01205	0.00780	-	-	-
FL: Tranco Random - Model Convergence	2	0.00719	0.00745	0.00974	0.01763	0.01799	0.01889
	3	0.01022	0.01062	0.01297	0.01680	0.01773	0.01831
	4	0.01185	0.01235	0.01428	-	-	-
FL: Tranco Random - Model Epoch	2	0.00351	0.00520	0.00394	0.01551	0.01857	0.01593
	3	0.00458	0.00877	0.00599	0.01397	0.01775	0.01557
	4	0.00575	0.01146	0.00958	-	-	-
T/S: Soft Labels	2	0.00328	0.00339	0.00339	0.02264	0.02143	0.02198
	3	0.01735	0.01703	0.01646	0.02252	0.02171	0.02247
	4	0.01822	0.01776	0.01744	-	-	-
T/S: Hard Labels	2	0.00329	0.00331	0.00333	0.02386	0.02132	0.02103
	3	0.01725	0.01795	0.01650	0.02208	0.02246	0.02218
	4	0.01766	0.01761	0.01694	-	-	-

Table 4: FPRs of the best and worst case for all classifier types and collaborative machine learning approaches of D5.6 separated by the number of participants.

For visibility, we omit the ACC and the TPR metric, however, most of the time a better or worse FPR correlates with a better or worse ACC. In this evaluation, we are not primarily interested in comparing the achieved scores of the different approaches with the performance of the baseline that is presented at the top of the table. Rather, we are interested in whether an increasing cooperation improves or worsens the performance achieved. Thus, we colour code the entries different to Table 3. Here, we mark all table entries green if they are always improving with an increasing number of participants. When the approaches produce consecutive worse results, we colour them red. We do not colour any entries for approaches to which the increases or decreases in classification performance are not consecutive. In the following, we present the evaluation results for the best and worst case in detail.

Best Case

Most of the collaborative approaches achieve (1) worse results compared to the baseline and (2) are decreasing in classification performance with the increasing number of participants. This behaviour can be explained by the fact that the baseline's best-case scenario is the ideal training and classification setting. I.e., the classifiers are assessed on data that comes from the same distribution as the samples used for training. No information about samples from other organisations is incorporated in these classifiers. Thereby, the baseline classifiers are specialised in classifying the samples that originate from the same network as the training samples used. Thus, it is not surprising that the baseline classifiers achieve almost the best results compared to the other approaches. The collaborative machine learning approaches, on the other hand, incorporate also information of samples from other networks. Thereby, they are less specialised in classifying samples from a single network but rather are more generalised and thus achieve worse results compared to the baseline. The fact that these approaches achieve worse results with an increasing number of participants can be explained similarly. The more participants, the less the classifiers are specialised on samples of a single network.

From these results, it can be seen that none of the investigated collaborative machine learning approaches of this deliverable is beneficial in the best-case scenario, where organisations want to use collaborative machine learning classifiers to classify samples from their own network most of the time. However, we also explored other collaborative machine learning approaches in the deliverable D5.4 which improve classification performance also in this particular scenario.

Worst Case

In this scenario, we evaluate whether an increasing number of participants improves the detection performance of jointly trained classifiers for samples that originate from external networks. Since we only have four different sources of benign data, the maximum number of participants in this scenario is three.

The results obtained for the T/S approach deteriorate as the number of participants increases for all classifiers except for Endgame. However, the achieved rates for Endgame are worse than those of the baseline.

For FL, the FPRs improve in all settings and for all classifiers except for Endgame when a randomly initialised model is used. We assume that this is due to the same reasons given in the average-case analysis section. The ResNet classifier, however, achieves the best results using a randomly initialised model. The achieved FPRs for Endgame and ResNet using federation after model epoch are significantly lower than for the approaches that make use of federation after model convergence. For the NYU classifier, no significant difference can be measured for the various models.

In summary, in the worst-case scenario, only the FL approaches improve performance with an increasing number of participants and achieve better scores than the baseline. FL with federation after model epoch achieves best results for Endgame and ResNet and thus generalises best to different networks. When comparing the different types of classifiers, Endgame and ResNet are better suited for FL than NYU. For RNN-based classifiers pre-trained initial models should be used.

3.1.5.4 Direct Comparison with Approaches Developed in D5.4 (Global Models based on Shared Local Models)

In this subsection, we compare the sharing approaches developed in this deliverable with the collaborative machine learning approaches investigated in the deliverable D5.4 (Global Models based on Shared Local Models). In the deliverable D5.2 (Global Model based on Shared Anonymised Data) we also proposed different sharing approaches. However, the focus of that deliverable was mainly on the comprehensive privacy study and the development of a context-less and feature-based approach to DGA multiclass classification that can be used as an anonymiser for domain names in a collaborative machine learning scenario. Hence, we only compare the approaches of this deliverable with the approaches investigated in D5.4. Future work could compare the approaches to intelligence sharing of D5.2 with the collaborative machine learning approaches of D5.4 and D5.6. However, the approaches of D5.2 are less advanced compared to the approaches of D5.4 and D5.6.

In addition to the 9480 evaluation passes done for the evaluation of the different sharing approaches of this deliverable, we now present the results of additional 3960 evaluation passes done for the deliverable D5.4. Our comprehensive study thus comprises a total of 13,440 evaluations.

Collaborative Machine Learning Approaches Developed in D5.4

In the following, we shortly repeat the sharing approaches which are taken from the deliverable D5.4 to which we compare the performance of the collaborative machine learning approaches investigated in this deliverable.

Ensemble Classification

In Ensemble classification a global classifier is built using the classifiers trained by each organisation. Similar to the baseline scenario each organisation first trains a classifier using their own private benign data. These classifiers are then shared with all participants. Each party now combines the individual classifiers to an Ensemble classifier. Similar to the T/S approach, the combination of the classifiers can be done (1) by using a majority voting system on the binary labels (**Hard Labels**) or (2) by averaging the results of the individual classifiers to a single confidence score (**Soft Labels**). When using the majority voting approach, a tie is resolved in the same way as in the T/S approach.

In total there are eleven possible combinations of organisations for building a combined model using four different parties. Thus, we perform five repetitions using eleven possible organisation combinations, two ensemble approaches (Hard and Soft Labels), four test datasets, and three classifier models (5 * 11 * 2 * 4 * 3 = 1320 evaluation passes).

Feature Extractor Sharing

This sharing approach is related to Transfer Learning. All deep learning classifiers under consideration use of a fully connected (dense) layer to output the final classification score. This layer can be viewed as a sort of classification layer that performs a logistic regression for binary classification. The output of this layer is a confidence score that indicates whether an input domain is benign (score < 0.5) or malicious (score ≥ 0.5). All layers before this classification layer can be treated as a feature extractor, which produces features used for classification by the final output layer. Instead of sharing the complete classifier as in the naive Ensemble approach, here, we hope to reduce the model's privacy leakage by sharing fewer layers. Thus in this approach, each organisation trains a model based on their own private training data. Subsequently, the trained feature extractors are shared among all participants. Each organisation now combines its own and received feature extractors to a new model. To this end, the feature extractors are applied in parallel and their outputs are concatenated and flattened. Additionally, a new dense classification layer is appended to the new model. This classification layer is not trained yet, thus the organisations freeze the weights of the feature extractors and use local training data to train the classification layer separately. In the end, each organisation obtains a model which incorporates information about samples occurring in the other organisations through the shared feature extractors. The resulting models are not identical, since the training of the classification layer is performed on private training samples.

For this approach, each organisation first trains a classifier using its own private benign data and derives an individual feature extractor. Then we combine the four feature extractors into eleven possible classifiers. In contrast to ensemble classification, here we require additional training to fit the final classification layer that combines the results of the shared feature extractors. Hence, in total we, perform five repetitions using eleven possible organisation combinations with four training datasets, four test datasets, and three classifier models (5 * 11 * 4 * 4 * 3 = 2640 evaluation passes).

Comparative Evaluation

In the following, we compare the sharing approaches developed in this deliverable with the collaborative machine learning approaches investigated in the deliverable D5.4. For convenience, we display the results of both deliverables in summarised tables.

Average Case

In Table 5, we display the results of the average case including all classifier types and all collaborative machine learning approaches.

Approach	Endgame			NYU			ResNet		
	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR
Baseline	0.99103	0.99910	0.01703	0.99151	0.99903	0.01600	0.99102	0.99844	0.01640
Ensemble: Soft Labels	0.98812	0.99975	0.02352	0.98870	0.99976	0.02236	0.98834	0.99946	0.02279
Ensemble: Hard Labels	0.98842	0.99981	0.02296	0.98901	0.99976	0.02174	0.98926	0.99909	0.02057
FL: Random Model - Model Convergence	0.76755	0.55131	0.01621	0.98300	0.98764	0.02163	0.98671	0.99278	0.01937
FL: Random Model - Model Epoch	0.99396	0.99968	0.01177	0.99392	0.99983	0.01199	0.99511	0.99935	0.00913
FL: Tranco Top - Model Convergence	0.99336	0.99958	0.01286	0.99325	0.99975	0.01326	0.99291	0.99974	0.01393
FL: Tranco Top - Model Epoch	0.99553	0.99956	0.00850	0.99400	0.99981	0.01181	0.99491	0.99922	0.00940
FL: Tranco Random - Model Convergence	0.99365	0.99946	0.01216	0.99360	0.99978	0.01257	0.99268	0.99967	0.01431
FL: Tranco Random - Model Epoch	0.99565	0.99952	0.00823	0.99413	0.99978	0.01153	0.99498	0.99929	0.00934
Feature Extractor Sharing	0.99394	0.99921	0.01133	0.99411	0.99913	0.01090	0.99307	0.99880	0.01266
T/S: Soft Labels	0.98979	0.99963	0.02006	0.99029	0.99965	0.01908	0.99001	0.99949	0.01948
T/S: Hard Labels	0.98966	0.99968	0.02036	0.99017	0.99965	0.01930	0.99026	0.99950	0.01898

Table 5: Results of the average case including all classifier types and all collaborative machine learning approaches.

From the approaches examined in D5.4, the only approach that leads to better classification results compared to the baseline is Feature Extractor Sharing which achieves results that are comparable to the ones achieved by FL.

Ensemble classification leads to worse results than the baseline. Comparing Ensemble classification to T/S, the T/S approach yields a lower FPR for all three classifier types. Furthermore, with either approach, it makes little difference whether soft or hard labels are used.

In summary, in the average-case scenario, only the collaborative machine learning approaches Feature Extractor Sharing and FL are advantageous for the use case of DGA detection while additional malicious samples in collaborative machine learning improve the TPRs for all sharing approaches.

Best Case

In Table 6, we display the FPRs of the best and worst case for all classifier types and all collaborative machine learning approaches separated by the number of participants.

Approach	Parties	Best Case			Worst Case		
		Endgame	NYU	ResNet	Endgame	NYU	ResNet
Baseline	-	0.00315	0.00328	0.00326	0.02166	0.02024	0.02078
Ensemble: Soft Labels	2	0.02149	0.02023	0.02081	0.02420	0.02306	0.02376
	3	0.02393	0.02266	0.02316	0.02493	0.02409	0.02387
	4	0.02489	0.02402	0.02366	-	-	-
Ensemble: Hard Labels	2	0.02068	0.01978	0.01887	0.02399	0.02242	0.02127
	3	0.02332	0.02207	0.02077	0.02462	0.02322	0.02186
	4	0.02404	0.02316	0.02167	-	-	-
FL: Random Model - Model Epoch	2	0.00464	0.00620	0.00465	0.02032	0.01886	0.01668
	3	0.00738	0.00922	0.00516	0.02122	0.01786	0.01473
	4	0.01120	0.01120	0.00624	-	-	-
FL: Tranco Top - Model Convergence	2	0.00803	0.00783	0.00856	0.01769	0.01841	0.01924
	3	0.01147	0.01170	0.01230	0.01723	0.01834	0.01877
	4	0.01267	0.01365	0.01415	-	-	-
FL: Tranco Top - Model Epoch	2	0.00361	0.00535	0.00410	0.01571	0.01905	0.01655
	3	0.00491	0.00879	0.00597	0.01492	0.01830	0.01573
	4	0.00591	0.01205	0.00780	-	-	-
FL: Tranco Random - Model Convergence	2	0.00719	0.00745	0.00974	0.01763	0.01799	0.01889
	3	0.01022	0.01062	0.01297	0.01680	0.01773	0.01831
	4	0.01185	0.01235	0.01428	-	-	-
FL: Tranco Random - Model Epoch	2	0.00351	0.00520	0.00394	0.01551	0.01857	0.01593
	3	0.00458	0.00877	0.00599	0.01397	0.01775	0.01557
	4	0.00575	0.01146	0.00958	-	-	-
Feature Extractor Sharing	2	0.00306	0.00303	0.00314	0.01871	0.01756	0.01804
	3	0.00295	0.00302	0.00307	0.01796	0.01722	0.01784
	4	0.00293	0.00298	0.00302	-	-	-
T/S: Soft Labels	2	0.00328	0.00339	0.00339	0.02264	0.02143	0.02198
	3	0.01735	0.01703	0.01646	0.02252	0.02171	0.02247
	4	0.01822	0.01776	0.01744	-	-	-
T/S: Hard Labels	2	0.00329	0.00331	0.00333	0.02386	0.02132	0.02103
	3	0.01725	0.01795	0.01650	0.02208	0.02246	0.02218
	4	0.01766	0.01761	0.01694	-	-	-

Table 6: FPRs of the best and worst case for all classifier types and all collaborative machine learning approaches separated by the number of participants.

The only approach that improves with an increasing number of participants is Feature Extractor Sharing. Moreover, for all three classifier types, Feature Extractor Sharing achieves better FPRs than the baseline. This is because this approach creates models that are similar to the baseline but also incorporates additional information about samples from other networks via feature extractors that are applied in parallel. Since the shared feature extractors are not retrained, information about samples from individual networks is very well preserved with this approach. In addition, the intelligence incorporated in the shared feature extractors is harnessed by this approach and leads to improvement even beyond baseline. Note, although the differences in FPRs are rather small, we argue that our results are significant because of the large number of evaluations done and the fact that this behaviour is observable for all three types of classifiers.

From these results, it can be seen that only Feature Extractor Sharing is beneficial in the best-case scenario, where organisations want to use collaborative machine learning classifiers to classify samples from their own network most of the time.

Worst Case

The results obtained for the Ensemble classification deteriorate as the number of participants increases for all three classifiers.

For Feature Extractor Sharing, the FPRs achieved improvement with an increased number of participants for all three classifier types. However, the rates are significantly worse than the ones achieved by Endgame and ResNet using FL with federation after model epoch. For NYU, the rates are comparable to those seen in the FL settings.

In summary, in the worst-case scenario, only the Feature Extractor Sharing and FL approaches improve in performance with an increasing number of participants and achieve better scores than the baseline. FL with federation after model epoch achieves best results for Endgame and ResNet and thus generalises best to different networks.

3.1.5.5 Conclusion

In this work, we performed a comprehensive study of collaborative machine learning for the real-world use case of DGA detection. Thereby, we identified advantageous and disadvantageous approaches to different types of classifiers and showed that collaborative machine learning can lead to a reduction in FPR by up to 51.7%. Additionally, we showed that the usage of publicly available data is insufficient for DGA detection on private data. This shows the need for privacy-preserving collaborative machine learning approaches. In two real-world cases, we have shown that greater participation in collaborative machine learning does not always lead to better classification results. In fact, we only assess Feature Extractor Sharing and FL of the four examined collaborative machine learning approaches as beneficial for DGA detection. Feature Extractor Sharing should be used if a party wants to classify samples that come from its own network most of the time. On the other hand, FL generalises best to unknown networks. The four examined collaborative machine learning variants based on T/S learning and Ensemble classification lead to worse results than the baseline.

3.1.6 Privacy Analysis

In the deliverable 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 disclosure of benign NXDs (bNXDs) may reveal end-user browsing history and behaviour, the usage of out-dated or misconfigured software. Further, knowledge of frequently occurring bNXDs originating from typing errors may be used to register typo-domains for a personalised phishing attack. The malicious data used in this use case is mostly publicly available and thus has no privacy constraints.

Any kind of sharing activity may generate an attack surface that threatens the privacy of the shared object. Thus, we investigate the privacy implications of the sharing approaches in the DGA detection use case. At this point, a thorough discussion on relevant privacy-threatening inference attacks against the beneficial sharing approaches shall complement our sharing scenario evaluation. In the deliverable D5.5 the Machine Learning privacy attack landscape was introduced in a brief but formal overview. The greatest privacy threat to our sharing approaches is the Membership Inference attack which aims at inferring whether or not a certain data sample was used to train the model and thus we focus on that attack in our defence evaluation.

Specifically, in Federated Learning, a Membership Inference (MemI) attack against the globally trained model can threaten the disclosure of participation on sample level (i.e., the classic MemI attack [27]) as well as on client level [28]. On a high level, a privacy-preserving model should not memorise training data while it learns from it. Assessing

the privacy leakage of our models is analogous to quantifying their potential memorisation. Although such attack is performed on a shared trained model, it still has relevance in this deliverable that handles sharing scenarios in which local models are not directly shared.

3.1.6.1 Evaluation: Membership Inference against Federated Learning

The Federated Learning paradigm was proposed to enhance the data privacy for participants by reducing the exposure of their data. This alone is not sufficient, as attacks still threaten data privacy in Federated Learning [29, 30]. Hence, in one of the beneficial sharing scenarios for DGA detection, Federated Learning, Membership Inference may still be deployed against the shared global model.

Defence Mechanism: Differential Privacy

Differential Privacy (DP) [31] is a privacy mechanism that introduces noise on data release to hide participation of an individual. In the original definition [31] the mechanism is designed for privacy-preserving data-release: any accumulative measure taken over a data base of individual records should not reveal that a certain individual's data was used to compute this measure. Masking participation by additive noise on the measure causes the measure computed over the entire data base and a measure computed over all entries except one to be indistinguishable for an attacker. The concept of ε-DP describes the above formally as such:

$$Pr(A(D_1) \in S) \leq e^{\varepsilon} \cdot Pr(A(D_2) \in S)$$

where ε (privacy budget) bounds the closeness of output distributions of any algorithm A for neighbouring datasets D_1 and D_2. A lower ε implies a better privacy guarantee. The concept of DP can be transferred to Machine Learning [32], where the goal is to hide the use of a single data sample during training. Although model training already is a stochastic process, Membership Inference attacks can be successfully launched against trained models. The data release mechanism in Machine Learning is the gradient computation based on the current model function's weight parameters and a batch of data. DP has successfully been integrated into existing gradient descent optimisers [32].

Obviously, this gradient will be computed multiple times during training and repeated data releases on the same input data will weaken the privacy guarantee as the noise cancels out over time. (ε, δ)-DP for repeating releases therefore uses a different formal notion with two parameters ε and δ:

$$Pr(A(D_1) \in S) \leq e^{\varepsilon} \cdot Pr(A(D_2) \in S) + \delta$$

where parameter ε is again the privacy budget while δ is the probability that the privacy guarantee does not hold. In DP-SGD [32] all gradient updates are clipped to a fixed range and subsequently, random noise is applied to each gradient update. Since, model training involves randomness, e.g., through initialisation and batch forming, it is impossible to predict the training steps and therefore the amount of data releases that will be done. Given the clipping and noise parameters used during training, ε/δ pairs for DP can only be estimated retrospectively.

A multitude of DP-driven research in FL exists (e.g., [33, 34, 35, 36, 37, 38]), that propose or investigate DP-based defences and on occasion examine the inherent privacy-utility trade-off. Improvements of or alternatives to the classic DP-SGD for the FL setting are proposed in [39, 40], any of which can be applied to the local training of each party. DP yields sound privacy guarantees w.r.t. the bounds (ε, δ), the quality of which are however influenced by the individual use case and its available data and hence, application of DP requires a further assessment of the resulting privacy/utility trade-off which we present here.

Evaluation Cases

In our assessment, we aim to measure privacy leakage as the inverse of the classification performance of a Membership Inference attack. We would like to compare the cases named in the following to (1) analyse the federated models' sensitivity to such an attack as well as to (2) compare the latter to the baseline models (local models trained only by one party's data).

In detail, the Membership Inference attack is evaluated against the following models:

1. Baseline models trained with non-DP optimiser (undefended)
2. Baseline models trained with DP optimiser (defensive training)
3. Federated model trained with non-DP optimiser (undefended)
4. Federated model trained with DP optimiser (defensive training)

For both the baseline and the federated models, the evaluation is run against versions of the model with and without privacy protection, i.e., once trained with a non-DP optimiser and once with a DP optimiser (the DP version of the Adam optimiser [18] is used). Thereby, we would like to assess the utility-privacy trade-off inherent to using DP-SGD. For federated models, we focus on the better performing training mode which is federation after each local training epoch.

Since a Membership Inference attack is a binary classification problem by itself, different ML approaches can be chosen for the attack model, as for instance Random Forest, Logistic Regression or K-Nearest-Neighbours. We evaluate the classifiers' performance with the advantage metric [41] defined by:

$$\text{advantage} = \max\{|\text{FPR} - \text{TPR}|\}$$

which is closely related to the AUC-ROC metric, i.e., a higher value describes a better performance of the attack model and therefore implies a higher data memorisation in the model and lower privacy guarantees. Concretely, we analyse the Membership Inference attack over a selected range for the clipping and noise parameters of DP-Adam and different Machine Learning approaches to the attack's binary classification problem.

Results

Results of the privacy-utility trade-off study for Membership Inference defences in Federated Learning are presented in the following. Models are trained with a mini-batch size of 250 samples and for 100 epochs with an early-stop after 3 consecutive epochs without improvement of the validation accuracy. For the application of the DP optimiser,

we choose a micro-batch size of 25, i.e., per mini-batch 10 values are used to calculate the required noise for the gradient update computed on that mini-batch. Choosing the micro-batch size is a trade-off between training effort and effectiveness of the DP optimiser. To remain concise, we further restrict the analysis scope: (1) We only evaluate the ResNet architecture. (2) For clipping and noise parameters we evaluate combinations of values in {0.5, 1, 2} in the baseline cases. (3) To train and evaluate the attack models 30,000 samples are subsampled from each of the original models' training and test dataset.

Membership Inference against Baseline Models

First, we assess the use of DP-based optimisers for the baseline models. Table 7 and Table 8 display the model utility after training and attacker advantage for the Membership Inference attack deployed against the trained models. The baseline models reach conversion after only a few epochs of training, while achieving a considerable classification performance (see Table 7). With some minor exceptions, the attack advantage for all target models and attack approaches lies below 0.5 (see Table 8).

Training Data	Trained Epochs	Test Data	Accuracy	TPR	FNR	TNR	FPR
CESNET	3	CESNET	0.9958	0.9975	0.0025	0.9942	0.0058
		MU	0.9950	0.9975	0.0025	0.9925	0.0075
		RWTH	0.9961	0.9975	0.0025	0.9947	0.0053
		Siemens	0.9979	0.9975	0.0025	0.9983	0.0017
MU	5	CESNET	0.9738	0.9994	0.0006	0.9482	0.0518
		MU	0.9981	0.9994	0.0006	0.9969	0.0031
		RWTH	0.9964	0.9994	0.0006	0.9935	0.0065
		Siemens	0.9987	0.9994	0.0006	0.9981	0.0019
RWTH	3	CESNET	0.9661	0.9999	0.0001	0.9323	0.0677
		MU	0.9901	0.9999	0.0001	0.9804	0.0196
		RWTH	0.9983	0.9999	0.0001	0.9968	0.0032
		Siemens	0.9988	0.9999	0.0001	0.9978	0.0022
Siemens	5	CESNET	0.9697	0.9997	0.0003	0.9397	0.0603
		MU	0.9952	0.9997	0.0003	0.9907	0.0093
		RWTH	0.9963	0.9997	0.0003	0.9929	0.0071
		Siemens	0.9991	0.9997	0.0003	0.9985	0.0015

Table 7: Utility of undefended baseline models.

Training Data	Attack Type	Attacker Advantage
CESNET	Threshold Attack	0.4020
	K-Nearest-Neighbours	0.4775
	Logistic Regression	0.4099
	Random Forest	0.4511
	Multi-Layered-Perceptron	0.4031
MU	Threshold Attack	0.4495
	K-Nearest-Neighbours	0.6004
	Logistic Regression	0.4444
	Random Forest	0.5647
	Multi-Layered-Perceptron	0.4368
RWTH	Threshold Attack	0.4493
	K-Nearest-Neighbours	0.4808
	Logistic Regression	0.4455
	Random Forest	0.5028
	Multi-Layered-Perceptron	0.4359
Siemens	Threshold Attack	0.4779
	K-Nearest-Neighbours	0.6368
	Logistic Regression	0.4875
	Random Forest	0.6052
	Multi-Layered-Perceptron	0.4635

Table 8: Advantage of membership inference attack against undefended baseline models.

Results for the same setting where training is guided by a DP-based optimiser are held in Table 9 and Table 10. For each data source, nine models are trained for the different combinations of the clipping and noise parameters. Note, that the DP-trained models take up to 3-4 times longer to train (in terms of training epochs) yet lose up to 2% classification accuracy (see Table 9). Table 10 lists the (ε, δ) parameters with respect to the achievable DP guarantee. With increasing noise multiplier, we receive better privacy budgets (ε) for smaller δ. Compared to the undefended models, the attack approaches K-Nearest-Neighbour and Random Forest achieve higher confidence while the other three approaches perform worse. Higher clipping and noise values also naturally reduce the attack's performance. The case with the highest clipping and noise

parameters yields the worst attack classifier performance but also comes with the greatest loss of the target model's classification performance.

Training Data	L2 Clipping	Noise Multiplier	Trained Epochs	Accuracy	TPR	FNR	TNR	FPR
CESNET	0.5	0.5	10	0.9874	0.9912	0.0088	0.9837	0.0163
	0.5	1	14	0.9843	0.9875	0.0125	0.9812	0.0188
	0.5	2	15	0.9713	0.9814	0.0186	0.9613	0.0387
	1	0.5	10	0.9811	0.9859	0.0141	0.9764	0.0236
	1	1	12	0.9824	0.9859	0.0141	0.9790	0.0210
	1	2	12	0.9732	0.9824	0.0176	0.9639	0.0361
	2	0.5	9	0.9830	0.9876	0.0124	0.9783	0.0217
	2	1	12	0.9809	0.9877	0.0123	0.9742	0.0258
	2	2	11	0.9706	0.9810	0.0190	0.9602	0.0398
MU	0.5	0.5	10	0.9780	0.9934	0.0066	0.9626	0.0374
	0.5	1	8	0.9843	0.9956	0.0044	0.9730	0.0270
	0.5	2	15	0.9766	0.9936	0.0064	0.9595	0.0405
	1	0.5	11	0.9844	0.9958	0.0042	0.9731	0.0269
	1	1	9	0.9844	0.9957	0.0043	0.9731	0.0269
	1	2	14	0.9765	0.9907	0.0093	0.9623	0.0377
	2	0.5	8	0.9836	0.9947	0.0053	0.9725	0.0275
	2	1	12	0.9828	0.9952	0.0048	0.9704	0.0296
	2	2	16	0.9723	0.9917	0.0083	0.9530	0.0470
RWTH	0.5	0.5	11	0.9639	0.9965	0.0035	0.9314	0.0686
	0.5	1	11	0.9617	0.9981	0.0019	0.9254	0.0746
	0.5	2	4	0.9435	0.9943	0.0057	0.8927	0.1073
	1	0.5	12	0.9542	0.9968	0.0032	0.9115	0.0885
	1	1	9	0.9582	0.9974	0.0026	0.9190	0.0810
	1	2	8	0.9447	0.9980	0.0020	0.8915	0.1085

	2	0.5	7	0.9647	0.9977	0.0023	0.9318	0.0682
	2	1	13	0.9545	0.9971	0.0029	0.9119	0.0881
	2	2	11	0.9461	0.9960	0.0040	0.8961	0.1039
Siemens	0.5	0.5	7	0.9721	0.9968	0.0032	0.9474	0.0526
	0.5	1	11	0.9672	0.9971	0.0029	0.9373	0.0627
	0.5	2	15	0.9568	0.9954	0.0046	0.9183	0.0817
	1	0.5	6	0.9744	0.9971	0.0029	0.9518	0.0482
	1	1	9	0.9732	0.9976	0.0024	0.9487	0.0513
	1	2	10	0.9679	0.9961	0.0039	0.9398	0.0602
	2	0.5	9	0.9739	0.9976	0.0024	0.9503	0.0497
	2	1	9	0.9706	0.9962	0.0038	0.9450	0.0550
	2	2	8	0.9659	0.9935	0.0065	0.9383	0.0617

Table 9: Utility of DP-trained baseline models. Performance averaged over all four test datasets.

Training Data	L2 Clipping	Noise Multiplier	Delta	Epsilon	Attack Type	Attacker Advantage
CESNET	0.5	0.5	0.10000	0.85464	Threshold Attack	0.3847
			0.01000	2.10457	K-Nearest-Neighbours	0.7016
			0.00100	3.23114	Logistic Regression	0.3915
			0.00010	4.30759	Random Forest	0.6369
			0.00001	5.35422	Multi-Layered-Perceptron	0.3831
	0.5	1	0.10000	0.00000	Threshold Attack	0.3921
			0.01000	0.18235	K-Nearest-Neighbours	0.7064
			0.00100	0.35947	Logistic Regression	0.3948
			0.00010	0.53659	Random Forest	0.6840
			0.00001	0.71372	Multi-Layered-Perceptron	0.3809
			0.10000	0.00000	Threshold Attack	0.3656

	0.5	2	0.01000	0.04540	K-Nearest-Neighbours	0.6248
			0.00100	0.10430	Logistic Regression	0.3620
			0.00010	0.14658	Random Forest	0.5917
			0.00001	0.18698	Multi-Layered-Perceptron	0.3727
	1	0.5	0.10000	0.85464	Threshold Attack	0.3868
			0.01000	2.10457	K-Nearest-Neighbours	0.6003
			0.00100	3.23114	Logistic Regression	0.3828
			0.00010	4.30759	Random Forest	0.5705
			0.00001	5.35422	Multi-Layered-Perceptron	0.3924
	1	1	0.10000	0.00000	Threshold Attack	0.3719
			0.01000	0.16732	K-Nearest-Neighbours	0.5951
			0.00100	0.34444	Logistic Regression	0.3744
			0.00010	0.52156	Random Forest	0.5689
			0.00001	0.69868	Multi-Layered-Perceptron	0.3716
	1	2	0.10000	0.00000	Threshold Attack	0.3570
			0.01000	0.03721	K-Nearest-Neighbours	0.5351
			0.00100	0.09117	Logistic Regression	0.3607
			0.00010	0.13186	Random Forest	0.5237
			0.00001	0.17225	Multi-Layered-Perceptron	0.3609
	2	0.5	0.10000	0.79996	Threshold Attack	0.3619
			0.01000	2.02893	K-Nearest-Neighbours	0.4927
			0.00100	3.14419	Logistic Regression	0.3680
			0.00010	4.20519	Random Forest	0.4785

			0.00001	5.24529	Multi-Layered-Perceptron	0.3619
	2	1	0.10000	0.00000	Threshold Attack	0.3590
			0.01000	0.16732	K-Nearest-Neighbours	0.5588
			0.00100	0.34444	Logistic Regression	0.3608
			0.00010	0.52156	Random Forest	0.5427
			0.00001	0.69868	Multi-Layered-Perceptron	0.3587
	2	2	0.10000	0.00000	Threshold Attack	0.3421
			0.01000	0.03433	K-Nearest-Neighbours	0.4455
			0.00100	0.08649	Logistic Regression	0.3468
			0.00010	0.12695	Random Forest	0.4147
			0.00001	0.16734	Multi-Layered-Perceptron	0.3485
MU	0.5	0.5	0.10000	0.85464	Threshold Attack	0.3774
			0.01000	2.10457	K-Nearest-Neighbours	0.7076
			0.00100	3.23114	Logistic Regression	0.3863
			0.00010	4.30759	Random Forest	0.6861
			0.00001	5.35422	Multi-Layered-Perceptron	0.3776
	0.5	1	0.10000	0.00000	Threshold Attack	0.3319
			0.01000	0.13726	K-Nearest-Neighbours	0.5784
			0.00100	0.31438	Logistic Regression	0.3369
			0.00010	0.49150	Random Forest	0.5476
			0.00001	0.66862	Multi-Layered-Perceptron	0.3291
	0.5	2	0.10000	0.00000	Threshold Attack	0.3414
			0.01000	0.04540	K-Nearest-Neighbours	0.6616

			0.00100	0.10430	Logistic Regression	0.3365
			0.00010	0.14658	Random Forest	0.6019
			0.00001	0.18698	Multi-Layered-Perceptron	0.3385
	1	0.5	0.10000	0.90707	Threshold Attack	0.3599
			0.01000	2.17816	K-Nearest-Neighbours	0.6148
			0.00100	3.31815	Logistic Regression	0.3523
			0.00010	4.41005	Random Forest	0.5928
			0.00001	5.45668	Multi-Layered-Perceptron	0.3467
	1	1	0.10000	0.00000	Threshold Attack	0.3161
			0.01000	0.14477	K-Nearest-Neighbours	0.5064
			0.00100	0.32190	Logistic Regression	0.3195
			0.00010	0.49902	Random Forest	0.4909
			0.00001	0.67614	Multi-Layered-Perceptron	0.3229
	1	2	0.10000	0.00000	Threshold Attack	0.3644
			0.01000	0.04273	K-Nearest-Neighbours	0.6305
			0.00100	0.10005	Logistic Regression	0.3621
			0.00010	0.14167	Random Forest	0.6011
			0.00001	0.18207	Multi-Layered-Perceptron	0.3619
	2	0.5	0.10000	0.74525	Threshold Attack	0.3258
			0.01000	1.95325	K-Nearest-Neighbours	0.4992
			0.00100	3.05610	Logistic Regression	0.3227
			0.00010	4.10273	Random Forest	0.4847
			0.00001	5.12087	Multi-Layered-Perceptron	0.3349

	2	1	0.10000	0.00000	Threshold Attack	0.3513
			0.01000	0.16732	K-Nearest-Neighbours	0.5849
			0.00100	0.34444	Logistic Regression	0.3467
			0.00010	0.52156	Random Forest	0.5503
			0.00001	0.69868	Multi-Layered-Perceptron	0.3385
	2	2	0.10000	0.00000	Threshold Attack	0.3476
			0.01000	0.04801	K-Nearest-Neighbours	0.6608
			0.00100	0.10842	Logistic Regression	0.3468
			0.00010	0.15149	Random Forest	0.6469
			0.00001	0.19189	Multi-Layered-Perceptron	0.3445
RWTH	0.5	0.5	0.10000	0.90707	Threshold Attack	0.4473
			0.01000	2.17816	K-Nearest-Neighbours	0.7775
			0.00100	3.31815	Logistic Regression	0.4440
			0.00010	4.41005	Random Forest	0.7391
			0.00001	5.45668	Multi-Layered-Perceptron	0.4537
	0.5	1	0.10000	0.00000	Threshold Attack	0.4481
			0.01000	0.15980	K-Nearest-Neighbours	0.7700
			0.00100	0.33692	Logistic Regression	0.4357
			0.00010	0.51405	Random Forest	0.7469
			0.00001	0.69117	Multi-Layered-Perceptron	0.4465
	0.5	2	0.10000	0.00000	Threshold Attack	0.3486
			0.01000	0.01078	K-Nearest-Neighbours	0.3571
			0.00100	0.05220	Logistic Regression	0.3443

			0.00010	0.09259	Random Forest	0.3809
			0.00001	0.13299	Multi-Layered-Perceptron	0.3496
	1	0.5	0.10000	0.95714	Threshold Attack	0.4495
			0.01000	2.24522	K-Nearest-Neighbours	0.7704
			0.00100	3.40516	Logistic Regression	0.4457
			0.00010	4.50163	Random Forest	0.7541
			0.00001	5.55913	Multi-Layered-Perceptron	0.4549
	1	1	0.10000	0.00000	Threshold Attack	0.4412
			0.01000	0.14477	K-Nearest-Neighbours	0.5872
			0.00100	0.32190	Logistic Regression	0.4419
			0.00010	0.49902	Random Forest	0.5712
			0.00001	0.67614	Multi-Layered-Perceptron	0.4360
	1	2	0.10000	0.00000	Threshold Attack	0.3844
			0.01000	0.02511	K-Nearest-Neighbours	0.4576
			0.00100	0.07183	Logistic Regression	0.3308
			0.00010	0.11222	Random Forest	0.4623
			0.00001	0.15262	Multi-Layered-Perceptron	0.3776
	2	0.5	0.10000	0.68714	Threshold Attack	0.4364
			0.01000	1.87371	K-Nearest-Neighbours	0.5383
			0.00100	2.95364	Logistic Regression	0.4293
			0.00010	3.99531	Random Forest	0.5287
			0.00001	4.99644	Multi-Layered-Perceptron	0.4479
			0.10000	0.00000	Threshold Attack	0.4440

	2	1	0.01000	0.17483	K-Nearest-Neighbours	0.7432
			0.00100	0.35196	Logistic Regression	0.4335
			0.00010	0.52908	Random Forest	0.7175
			0.00001	0.70620	Multi-Layered-Perceptron	0.4519
	2	2	0.10000	0.00000	Threshold Attack	0.3935
			0.01000	0.03433	K-Nearest-Neighbours	0.4635
			0.00100	0.08649	Logistic Regression	0.3945
			0.00010	0.12695	Random Forest	0.4656
			0.00001	0.16734	Multi-Layered-Perceptron	0.3793
Siemens	0.5	0.5	0.10000	0.68714	Threshold Attack	0.4124
			0.01000	1.87371	K-Nearest-Neighbours	0.6847
			0.00100	2.95364	Logistic Regression	0.4100
			0.00010	3.99531	Random Forest	0.6455
			0.00001	4.99644	Multi-Layered-Perceptron	0.4103
	0.5	1	0.10000	0.00000	Threshold Attack	0.3907
			0.01000	0.15980	K-Nearest-Neighbours	0.7475
			0.00100	0.33692	Logistic Regression	0.3796
			0.00010	0.51405	Random Forest	0.7257
			0.00001	0.69117	Multi-Layered-Perceptron	0.3832
	0.5	2	0.10000	0.00000	Threshold Attack	0.3825
			0.01000	0.04540	K-Nearest-Neighbours	0.7461
			0.00100	0.10430	Logistic Regression	0.3820
			0.00010	0.14658	Random Forest	0.7315

			0.00001	0.18698	Multi-Layered-Perceptron	0.3931
	1	0.5	0.10000	0.62689	Threshold Attack	0.3454
			0.01000	1.78670	K-Nearest-Neighbours	0.5312
			0.00100	2.85118	Logistic Regression	0.3469
			0.00010	3.87088	Random Forest	0.5161
			0.00001	4.87201	Multi-Layered-Perceptron	0.3341
	1	1	0.10000	0.00000	Threshold Attack	0.3464
			0.01000	0.14477	K-Nearest-Neighbours	0.5707
			0.00100	0.32190	Logistic Regression	0.3552
			0.00010	0.49902	Random Forest	0.5565
			0.00001	0.67614	Multi-Layered-Perceptron	0.3297
	1	2	0.10000	0.00000	Threshold Attack	0.3534
			0.01000	0.03136	K-Nearest-Neighbours	0.5444
			0.00100	0.08164	Logistic Regression	0.3572
			0.00010	0.12204	Random Forest	0.5253
			0.00001	0.16244	Multi-Layered-Perceptron	0.3432
	2	0.5	0.10000	0.79996	Threshold Attack	0.3516
			0.01000	2.02893	K-Nearest-Neighbours	0.5723
			0.00100	3.14419	Logistic Regression	0.3592
			0.00010	4.20519	Random Forest	0.5757
			0.00001	5.24529	Multi-Layered-Perceptron	0.3529
	2	1	0.10000	0.00000	Threshold Attack	0.3462
			0.01000	0.14477	K-Nearest-Neighbours	0.5337

			0.00100	0.32190	Logistic Regression	0.3223
			0.00010	0.49902	Random Forest	0.4941
			0.00001	0.67614	Multi-Layered-Perceptron	0.3449
	2	2	0.10000	0.00000	Threshold Attack	0.3284
			0.01000	0.02511	K-Nearest-Neighbours	0.3833
			0.00100	0.07183	Logistic Regression	0.3191
			0.00010	0.11222	Random Forest	0.3661
			0.00001	0.15262	Multi-Layered-Perceptron	0.2997

Table 10: Achievable ε for selected values of δ for DP-trained baseline models and advantage of membership inference attacks.

Membership Inference against Federated Models

We repeat the evaluations with and without a DP-based optimiser as a defence mechanism for the federated models where the weights are aggregated after each local training epoch. We focus on the case where all parties participate in the training protocol and hence provide results in a five-fold manner for more descriptiveness. In all the following tables, the results do not significantly differ between the different folds. Table 11 and Table 12 display training and attack evaluation results for the undefended scenario. Again, training takes more epochs to converge, in fact, each of the four parties performs 8-20 local epochs. Tested on each participant's local data, the final models achieve a classification performance better than the baseline models. Table 12 lists the Membership Inference attack classifier performances and surprisingly these are higher than 0.7 with the Random Forest approach achieving up to 0.99 advantage. To recite, this implies that an attack is possible that can accurately distinguish training data from test data and therefore disclose the data that was used to train the target model.

Fold	Trained Epochs	Test Data	Accuracy	TPR	FNR	TNR	FPR
1	8	CESNET	0.9822	0.9997	0.0003	0.9648	0.0352
		MU	0.9982	0.9997	0.0003	0.9966	0.0034
		RWTH	0.9986	0.9997	0.0003	0.9974	0.0026
		Siemens	0.9991	0.9997	0.0003	0.9986	0.0014
2	20	CESNET	0.9911	0.9995	0.0005	0.9826	0.0174
		MU	0.9983	0.9995	0.0005	0.9971	0.0029
		RWTH	0.9986	0.9995	0.0005	0.9977	0.0023
		Siemens	0.9990	0.9995	0.0005	0.9986	0.0014
3	17	CESNET	0.9909	0.9996	0.0004	0.9822	0.0178
		MU	0.9983	0.9996	0.0004	0.9970	0.0030
		RWTH	0.9986	0.9996	0.0004	0.9977	0.0023
		Siemens	0.9990	0.9996	0.0004	0.9985	0.0015
4	16	CESNET	0.9891	0.9996	0.0004	0.9786	0.0214
		MU	0.9983	0.9996	0.0004	0.9970	0.0030
		RWTH	0.9986	0.9996	0.0004	0.9977	0.0023
		Siemens	0.9991	0.9996	0.0004	0.9985	0.0015
5	19	CESNET	0.9896	0.9994	0.0006	0.9798	0.0202
		MU	0.9982	0.9994	0.0006	0.9971	0.0029
		RWTH	0.9984	0.9994	0.0006	0.9975	0.0025
		Siemens	0.9990	0.9994	0.0006	0.9987	0.0013

Table 11: Utility of undefended federated models (5-fold).

Fold	Attack Type	Attacker Advantage
1	Threshold Attack	0.7616
	K-Nearest-Neighbours	0.9023
	Logistic Regression	0.7581
	Random Forest	0.9989
	Multi-Layered-Perceptron	0.8069
2	Threshold Attack	0.7858
	K-Nearest-Neighbours	0.9217
	Logistic Regression	0.7800
	Random Forest	0.9995
	Multi-Layered-Perceptron	0.8585
3	Threshold Attack	0.7806
	K-Nearest-Neighbours	0.9019
	Logistic Regression	0.7816
	Random Forest	0.9997
	Multi-Layered-Perceptron	0.8311
4	Threshold Attack	0.7812
	K-Nearest-Neighbours	0.9100
	Logistic Regression	0.7735
	Random Forest	0.9991
	Multi-Layered-Perceptron	0.8393
5	Threshold Attack	0.7820
	K-Nearest-Neighbours	0.9088
	Logistic Regression	0.7744
	Random Forest	0.9987
	Multi-Layered-Perceptron	0.8371

Table 12: Advantage of membership inference attack against undefended federated models (5-fold).

Table 13 and Table 14 hold results for the same federated training setup but defended with the DP-based optimiser. Due to the longer training (in wall clock time), exploring the whole space for the clipping and noise parameters is too extensive for this scope. Hence, the results for the DP-trained federated models are performed only for one case: with the clipping value of 1 and the noise multiplier of 5. On average, training with the DP-based optimiser does not introduce any significant increase in epochs required for training convergence (see Table 13). The influence of DP decreases the classification accuracy of the final model by approximately 1%. Also, the FPR reduction gained by the federation in the first place is lost.

Compared to the attack against the undefended models, the usage of the DP-optimiser reduces the attacker's advantage by 13% on average (see Table 14). With advantages higher than 0.5, most attacks still perform quite well. Despite the losses in target model classification accuracy and FPR, the Random Forest attack models seem to be completely insensitive to the defence mechanism and remain with an advantage of > 0.99.

Fold	Trained Epochs	Test Data	Accuracy	AUC-ROC	TPR	FNR	TNR	FPR
1	22	CESNET	0.9520	0.9520	0.9890	0.0110	0.9150	0.0850
		MU	0.9751	0.9751	0.9890	0.0110	0.9612	0.0388
		RWTH	0.9863	0.9863	0.9890	0.0110	0.9836	0.0164
		Siemens	0.9880	0.9880	0.9890	0.0110	0.9869	0.0131
2	14	CESNET	0.9491	0.9491	0.9867	0.0133	0.9115	0.0885
		MU	0.9683	0.9683	0.9867	0.0133	0.9500	0.0500
		RWTH	0.9843	0.9843	0.9867	0.0133	0.9819	0.0181
		Siemens	0.9843	0.9843	0.9867	0.0133	0.9819	0.0181
3	15	CESNET	0.9439	0.9439	0.9858	0.0142	0.9020	0.0980
		MU	0.9662	0.9662	0.9858	0.0142	0.9466	0.0534
		RWTH	0.9830	0.9830	0.9858	0.0142	0.9801	0.0199
		Siemens	0.9832	0.9832	0.9858	0.0142	0.9806	0.0194
4	15	CESNET	0.9501	0.9501	0.9867	0.0133	0.9135	0.0865
		MU	0.9743	0.9743	0.9867	0.0133	0.9619	0.0381
		RWTH	0.9851	0.9851	0.9867	0.0133	0.9835	0.0165
		Siemens	0.9856	0.9856	0.9867	0.0133	0.9846	0.0154
5	16	CESNET	0.9520	0.9520	0.9847	0.0153	0.9193	0.0807
		MU	0.9735	0.9735	0.9847	0.0153	0.9624	0.0376
		RWTH	0.9836	0.9836	0.9847	0.0153	0.9825	0.0175
		Siemens	0.9858	0.9858	0.9847	0.0153	0.9870	0.0130

Table 13: Utility of DP-trained federated models (5-fold).

Fold	Attack Type	Attacker Advantage	Relative Improvement Compared to Undefended Federation
1	Threshold Attack	0.5762	-0.1854
	K-Nearest-Neighbours	0.8645	-0.0378
	Logistic Regression	0.5873	-0.1708
	Random Forest	0.9993	+0.0004
	Multi-Layered-Perceptron	0.7796	-0.0273
2	Threshold Attack	0.5511	-0.2347
	K-Nearest-Neighbours	0.8844	-0.0373
	Logistic Regression	0.5560	-0.224
	Random Forest	0.9997	+0.0002
	Multi-Layered-Perceptron	0.8676	+0.0091
3	Threshold Attack	0.5463	-0.2343
	K-Nearest-Neighbours	0.8833	-0.0186
	Logistic Regression	0.5540	-0.2276
	Random Forest	0.9995	-0.0002
	Multi-Layered-Perceptron	0.7677	-0.0634
4	Threshold Attack	0.5561	-0.2251
	K-Nearest-Neighbours	0.8813	-0.0287
	Logistic Regression	0.5685	-0.205
	Random Forest	0.9993	+0.0002
	Multi-Layered-Perceptron	0.7999	-0.0394
5	Threshold Attack	0.5517	-0.2303
	K-Nearest-Neighbours	0.8800	-0.0288
	Logistic Regression	0.5501	-0.2243
	Random Forest	0.9993	+0.0006
	Multi-Layered-Perceptron	0.7729	-0.0642

Table 14: Advantage of membership inference attack against DP-trained federated models (5-fold).

Conclusion on Membership Inference Threat Assessment

Results from Table 14 display that the defence mechanism based on the DP-Adam optimiser is not sufficient to defend against the Membership inversion attack as at least one attack type's advantage is not reduced significantly or at all.

As the evaluation was performed on the baseline as well as the federated models, a comparison with respect to the Membership Inference attack success can be drawn: Higher attacker advantage primarily correlates with more training epochs required for conversion, regardless of whether they are caused by the DP-defence or the federation itself. As the goal of this evaluation is to assess the federated model's relative training data memorisation, we quantify this by the attacker's advantage: a success metric of the Membership Inference attack. High advantage implies high memorisation in the target model and, in parallel, high memorisation correlates with the extent to which training data directly influence the model's weights in the training process. Hence, it is not surprising that the Membership Inference attack performs so badly on the undefended baseline models, as these already converge after only 3-5 epochs of training. The defended baseline models train 3-4 times that many epochs and, therefore, the weights in the model are more specialised. Hence, the Membership Inference attack performs better and for the federated models, we observe the same. Note, however, that the evaluation is potentially biased by the following aspects:

1. The evaluation set is only a medium sized subset (30,000 samples) of the available training data.
2. Whether DP noise cancels out as an effect of weight aggregation in Federated Learning is not analysed here.
3. We assumed that all parties use the same hyperparameters for local training. Especially for the DP-optimiser's parameters, this must not hold true in practice.
4. Due to the extensive effort, a grid search on the parameters space of the clipping and noise parameters was not performed.

When neglecting the potential biases to the evaluation and focusing on the results at hand, we can conclude for the privacy-utility trade-off of the DP-based defence that a sacrifice of the target model's classification gain by the federation is not met by an equal reduction in attack performance. A local DP-defence that operates with higher clipping and noise parameters will most likely decrease the attacker's advantage but at the cost of sacrificing more utility. Due to the peak performance of the Random Forest attack models, Membership Inference remains a privacy threat to federated DGA detection models. However, in the future a more extensive evaluation is to be performed for a more unbiased verdict or a different defence approach must be examined.

3.1.6.2 Other Attacks (Privacy & Security)

In one of our disseminated works [22], we discussed further privacy and security implications, that we want to repeat here for completeness.

Gradient Leakage

To retain valuable contribution from parties in FL, it is crucial to ensure the non-disclosure of a participant's local data. However, retaining data locality in FL is not sufficient,

as [42, 43, 44] demonstrate the necessity to shield gradient updates from inspection by the aggregating instance(s) that may reconstruct or infer sensitive data.

Inference attacks during the execution of the FL protocol can be rendered infeasible via a secure aggregation protocol [45], which computes the global average gradient in a secure multiparty computation (SMC) protocol that completely obstructs the inspection of local updates, thereby providing the best possible privacy. Multiple improvements have been proposed (e.g., [46, 47]): With the currently best performance overhead of O(N log(N)) per FL round [47], deploying secure aggregation is easily applicable in our use case with $N \leq 4$ parties.

Byzantine Attackers

Our work only considers the presence of trusted parties, yet for completeness we also give a quick view on Byzantine parties in FL [48], that are defined by arbitrary or faulty behaviour, including intentional misbehaviour such as privacy-threatening Free-riding [49] or sabotage (as in Poisoning [50] or Backdooring [51, 52]). Distributed learning in the Byzantine setting has been studied in [48, 53] (and larger literature bodies referenced in [54, 55]).

In [48], the authors argue that a single Byzantine user can influence any linear aggregation mechanism, and therefore also model conversion, to an arbitrarily large extent and present their first Byzantine-tolerant defence termed Krum. Other effective defences have been proposed, which are based on Krum or utilise similar insights that updates from malicious FL parties are separable from benign ones and can be filtered out [48, 50, 56, 44]. The authors of [53] propose a Byzantine-robust and Sybil-resistant defence. Additionally, integration of their defence mechanism was presented in [44]. The defence mechanism was integrated into the secure aggregation protocol by [45] utilising secure distance computation via homomorphic encryption (HE).

Unfortunately, [55] provides the first insights into the incompatibility of DP-based and Byzantine defences.

3.2 Application Profiling

We developed multiple approaches to application profiling which are described in the deliverable D3.5. We distinguish between two use cases, namely the identification and classification case. For the identification case, the goal is to identify which applications are running on a host based on network traffic. For this, we developed a rule-based as well as a process mining-based approach. The goal of the classification case is to determine whether an application behaves as expected, i.e., to detect anomalous behaviour which could be caused by malicious activity. However, for application profiling, we focused on the local use cases considered for WP3. Classical collaboration approaches such as the Teacher-Student approach or Federated Learning are applicable for machine learning, but not so much for our rule-based and process mining based approaches. We also experimented with deep learning for application profiling (D3.4), however, the other approaches showed more promising results. Hence, for WP5, we only continued with experiments based on shared anonymised data in the deliverable D5.2 when it comes to application profiling.

4 Conclusion

In this deliverable, we presented the results of task T5.3: "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 to the 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 performed a comprehensive study of collaborative machine learning for the real-world use case of DGA detection and discussed the privacy implications caused by beneficial sharing approaches. Thereby, we identified advantageous and disadvantageous approaches to different types of classifiers and showed that collaborative machine learning can lead to a reduction in FPR by up to 51.7%. Additionally, we showed that the usage of publicly available data is insufficient for DGA detection on private data. This shows the need for privacy-preserving collaborative machine learning approaches to DGA detection. In two real-world cases, we have shown that greater participation in collaborative training does not always lead to better classification results. Moreover, we comparatively evaluated multiple collaborative machine learning approaches to DGA detection that we examined in the deliverable D5.4. In fact, we only assess Feature Extractor Sharing and FL of the four examined collaborative machine learning approaches as beneficial for DGA detection. Feature Extractor Sharing should be used if a party wants to classify samples that come from their own network most of the time. On the other hand, FL generalises best to unknown networks. The four examined collaborative machine learning variants based on T/S learning and Ensemble classification lead to worse results than the baseline. In terms of privacy, we have thoroughly discussed the applicability of inference attacks for FL.

5 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] A. Drichel, B. Holmes, J. von Brandt, and U. Meyer. The More, the Better? A Study on Collaborative Machine Learning for DGA Detection. In Workshop on Cyber-Security Arms Race. 2021.

[23] A. Drichel, N. Faerber, and U. Meyer. First Step Towards EXPLAINable DGA Multiclass Classification. In International Conference on Availability, Reliability and Security. ACM, 2021.

[24] A. Drichel, U. Meyer, S. Schüppen, and D. Teubert. 2020. Making Use of NXt to Nothing: Effect of Class Imbalances on DGA Detection Classifiers. In Conference on Availability, Reliability and Security. ACM.

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

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Artificial Intelligence and Statistics. PMLR.

[27] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership Inference Attacks Against Machine Learning Models. Security and Privacy. IEEE. 2017.

[28] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. Computer Communications. IEEE. 2019.

[29] L. Lyu, H. Yu, X. Ma, L. Sun, J. Zhao, Q. Yang, and P. S. Yu. Privacy and Robustness in Federated Learning: Attacks and Defenses. arXiv:2012.06337. 2020.

[30] S. Shen, T. Zhu, D. Wu, W. Wang, W. Zhou. From distributed machine learning to federated learning: In the view of data privacy and security. Concurrency and Computation: Practice and Experience. 2020.

[31] C. Dwork. Differential Privacy. Automata, Languages and Programming. Springer Berlin Heidelberg. 2006.

[32] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with Differential Privacy. ACM. 2016.

[33] B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. Computer and Communications Security. ACM. 2017.

[34] R. C Geyer, T. Klein, and M. Nabi. Differentially Private Federated Learning: A Client Level Perspective. arXiv:1712.07557. 2017.

[35] Q. Zheng, S. Chen, Q. Long, and W. Su. Federated f-Differential Privacy. Artificial Intelligence and Statistics. PMLR. 2021.

[36] R. Kerkouche, G. Ács, C. Castelluccia, and P. Genevès. Constrained Differentially Private Federated Learning for Low-bandwidth Devices. arXiv:2103.00342. 2021.

[37] M. Kim, and O. Gunlu, and R. F. Schaefer. Federated Learning with Local Differential Privacy: Trade-offs between Privacy, Utility, and Communication. Cryptology ePrint Archive, Report 2021/142. 2021.

[38] P. Kairouz, and H. B. McMahan. Advances and Open Problems in Federated Learning. Foundations and Trends in Machine Learning. 2021.

[39] P. Kairouz, B. McMahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu. Practical and Private (Deep) Learning without Sampling or Shuffling. arXiv:2103.00039. 2021.

[40] C. Ye, and Y. Cui. Sample-based Federated Learning via Mini-batch SSCA. arXiv:2103.09506. 2021.

[41] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. 31st Computer Security Foundations Symposium (CSF). IEEE. 2018.

[42] L. Zhu, Z. Liu, and S. Han. Deep Leakage from Gradients. Advances in Neural Information Processing Systems. 2019.

[43] T. Orekondy, S. J. Oh, Y. Zhang, B. Schiele, and M. Fritz. Gradient-Leaks: Understanding and Controlling Deanonymization in Federated Learning. arXiv:1805.05838. 2020.

[44] J. Zhu, and M. Blaschko. R-GAP: Recursive Gradient Attack on Privacy. arXiv:2010.07733. 2021.

[45] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. ACM. 2017.

[46] J. Guo, Z. Liu, K. Lam, J. Zhao, Y. Chen, and C. Xing. Secure Weighted Aggregation in Federated Learning. arXiv:2010.08730. 2020.

[47] J. So, B. Güler, and A. S. Avestimehr. Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning. Journal on Selected Areas in Information Theory. IEEE. 2021.

[48] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. Neural Information Processing Systems. Curran Associates Inc. 2017.

[49] Y. Fraboni, R. Vidal, and M. Lorenzi. Free-rider Attacks on Model Aggregation in Federated Learning. Artificial Intelligence and Statistics. PMLR. 2021.

[50] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu. Data Poisoning Attacks Against Federated Learning Systems. Computer Security -- ESORICS 2020. Springer. 2020.

[51] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan. Can You Really Backdoor Federated Learning?. arXiv:1911.07963. 2019.

[52] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to Backdoor Federated Learning. Artificial Intelligence and Statistics. PMLR. 2020.

[53] N. Malecki, H. Paik, A. Ignjatovic, A. Blair, and E. Bertino. Simeon -- Secure Federated Machine Learning Through Iterative Filtering. arXiv:2103.07704. 2021.

[54] J. So, B. Güler, and A. S. Avestimehr. Byzantine-Resilient Secure Federated Learning. Journal on Selected Areas in Communications. IEEE. 2020.

[55] R. Guerraoui, N. Gupta, R. Pinot, S. Rouault, and J. Stephan. Differential Privacy and Byzantine Resilience in SGD: Do They Add Up?. arXiv:2102.08166. 2021.

[56] K. Yadav, and B. B. Gupta. Clustering Algorithm to Detect Adversaries in Federated Learning. arXiv:2102.10799. 2021.