Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D5.3 Global model based on shared local models, first version (M21)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file:	Deliverable D5.3
Document version:	1.0
Document owner:	Alexey Kirichenko (F-Secure)
Work package:	WP5
Task:	T5.2 Federated learning of a global model based on shared locally trained models
Deliverable type:	Report
Delivery month:	M21
Document status:	☒ approved by the document owner for internal review ☒ approved for submission to the EC

Document History:

Version	Author(s)	Date	Summary of changes made
0.1	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Alexey Kirichenko (FSC), Sebastian Schäfer (RWTH)	2021-01-08	Finalized outline and structure
0.2	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Tomas Jirsik (MU), David Karpuk (FSC), Alexey Kirichenko (FSC), Dmitriy Komashinskiy (FSC), Sebastian Schäfer (RWTH)	2021-01-27	Most Sections complete and ready for review
0.3	Arthur Drichel (RWTH), Benedikt Holmes (RWTH), Tomas Jirsik (MU), David Karpuk (FSC), Alexey Kirichenko (Editor, FSC), Dmitriy Komashinskiy (FSC), Sebastian Schäfer (RWTH)	2021-01-29	First complete version of the document converted from the D5.3 SAPPAN Confluence page
1.0	Alexey Kirichenko (FSC)	2021-01-31	Final version

Internal review history:

Reviewed by	Date	Summary of comments
Sebastian Schäfer (RWTH)	2021-01-29	Comments on spelling, wording, and content on the sections I didn't work on myself.
Mehdi Akbari G. (FIT)	2021-01-31	Editorial

Executive Summary

This document is the first of the two deliverables (an intermediate report) produced by Task 5.2, which is "Federated learning of a global model based on shared locally trained models". Similar to Tasks 5.1 and 5.3, the main goal of this line of work in SAPPAN is to train a global model based on contributions from multiple parties, while the differences are in the types of contributions and in the approaches to their sharing and aggregation. Therefore, the deliverables D5.1, D5.3, and D5.5 have certain common parts, especially with respect to the background and motivation for specific models (e.g., for DGA detection and application profiling) and the context within the SAPPAN project. In general, WP5 focuses on sharing and federation for cyber threat detection and response, and Task 5.2 is one of the three tasks investigating collaborative learning. In this initial deliverable of the task, we present the notes on the state-of-the-art in combining machine learning models and the showcases that we are working on in the context of building global detection models. The presented showcases include sharing scenarios, built prototypes and models, first validation results, and planned experiments. The showcases are similar to the ones developed in WP3, namely DGA detection, application and host profiling, and detection of anomalous operations with processes. We note that they demonstrate the two main types of threat detection techniques: detection of malicious objects (DGA detection) and malicious activities (anomalous process operations detection).

Table of Contents

1 Introduction

This deliverable is an intermediate report produced by Task 5.2: "Federated learning of a global model based on shared locally trained models". The high-level goals of Tasks 5.1, 5.2 and 5.3 are essentially the same (while the approaches to achieving the goals differ), hence, the deliverables D5.1, D5.3, and D5.5 have certain common parts, including the overall SAPPAN project context. One of the key SAPPAN objectives is to build a sharing platform that can be used for privacy-preserving sharing of intrusion detection data, detection models and response handling information. Sharing enabled by the platform is expected to improve the local detection capabilities of each participating organization. Examples of participating organizations are cybersecurity vendors, their partners (especially Managed Security Service Providers), their customers (especially those having internal security teams or experts), law enforcement agencies, and CERTs. Another flavour of sharing in the scope of SAPPAN is sharing among a cybersecurity vendor and various user groups of its customer organizations. In such cases, local detection capabilities can be, for instance, attack detection models built in individual endpoints (which is similar to the original use case of Federated Learning by Google - collaboratively learning Gboard’s query suggestion model on Android mobile phones, [1]).

The work in WP5 is closely connected to WP3 and WP4, where one of the tasks is to develop local detection and response mechanisms. In WP5, we combine - in various ways - some of those local mechanisms in order to obtain aggregated ("more global") models with high detection power. These mechanisms include machine learning models, process mining models, as well as statistical models. In this intermediate report of Task 5.2, we describe the approaches to building global models based on shared - in the Federated Learning fashion - local models and experiments with those. For some of the experiments we already have initial results. Further results will be presented in the second deliverable of Task 5.2.

This document is structured as follows. First, we present the notes on the state-of-the-art in combining machine learning models for cybersecurity applications, covering ensembling, distributed learning and federated learning and including several publicly known use cases in deployed solutions. Then we briefly outline the context of this task in the overall scope of the SAPPAN project and explain the general concept of the task. Privacy for machine learning models is discussed afterwards. Then we proceed to describe several showcases illustrating different approaches to building global detection models and first results of our experiments. The showcases include DGA detection, application profiling, and anomalous behaviour detection in endpoints. Finally, we briefly conclude this deliverable.

2 Approaches to Combining Machine Learning Models in Cybersecurity Applications

Traditional Machine Learning (ML) techniques have been used extensively in cybersecurity research for quite some time, including applications to intrusion detection, threat analysis, malware classification, and identification of malicious domain names [2, 3, 4]. To give a few examples of specific techniques and applications, Restricted Boltzmann Machines have been used for network anomaly detection [5], neural networks have been used for misuse detection [6] and malware detection [7, 8], and deep reinforcement learning has been used for intrusion detection in cyber-physical systems [9].

It has also become typical for cybersecurity vendors to power their technologies and services for detecting malicious activities and objects with machine learning. While the details of industrial ML-based engines and algorithms are usually not provided publicly, one can find a number of whitepapers, blog posts and articles claiming innovative cybersecurity applications of machine learning [10, 11, 12, 13, 14].

In the last several years, the machine learning techniques employed by the cybersecurity industry have had to evolve to accommodate an explosion in the pure volume of available data and its decentralized nature. Models constructed to serve the increasingly popular endpoint detection and response (EDR) products and more traditional endpoint protection (EPP) solutions offered by many cybersecurity firms face both of these challenges [15]. The cybersecurity realm naturally lends itself to methods of combining and distributing machine learning models and their training processes, and all such methods naturally have their benefits and drawbacks.

2.1 **Distributed Machine Learning**

By Distributed Machine Learning we mean a machine learning system which distributes the computational load of training a model over several nodes [16]. The distribution of computational tasks enables the training of more complex models that would otherwise be too burdensome for a single machine. For example, stochastic gradient descent is a distributed machine learning algorithm that allows a random subset of worker nodes to make gradient updates and transmit these updates back to a master node.

Distributed Machine Learning has found applications in cybersecurity thanks in part to the nature of the data, often collected from a disparate network of endpoints which themselves may be used as computational nodes for a distributed machine learning algorithm. In [17], the authors construct the SHIELD cybersecurity system which distributes the training and prediction of a number of machine learning algorithms, such as Latent Dirichlet Allocation, Support Vector Machines, and Recurrent Neural Networks, to detect anomalies and classify threats. The authors of [18] employ distributed machine learning techniques to classify traffic in edge nodes of IoT networks as malicious or benign. A machine learning technique whose architecture mimics that of blockchain and is trained using a distributed, decentralized version of stochastic gradient descent is introduced in [19].

Distributing the training of machine learning algorithms allows us to tackle enormous data sets and memory-hungry training algorithms more easily, but this typically comes at the cost of increased communication between a master node and a fleet of worker nodes. This communication cost can be a serious bottleneck if we have, for example, hundreds of worker nodes jointly performing gradient descent in a high-dimensional space. A second concern in distributed machine learning is that of security, as the more worker nodes we use the more likely it is that one or several of them may be compromised or controlled by an adversary. This opens up distributed machine learning algorithms to eavesdroppers and poisoning attacks if proper precautions are not taken.

2.2 **Ensemble Models**

An Ensemble Model is an aggregation of two or more classifiers that has the potential for performance superior to any of its individual components. A canonical example of ensemble model is random forest, which combines the results of many individual decision trees to perform classification tasks.

In the cyber security domain, ensemble methods have been used for intrusion detection in [20], wherein the authors use ensembles of voting classifiers, gradient boosting classifiers, random forests, and AdaBoost classifiers. Network flow classification was done in [21] using ensemble methods to better detect infected hosts.

An often encountered problem in cybersecurity is incomplete and unbalanced data; for example, almost all attack detection-related data collected by cybersecurity vendors comes from false alarms, as true security incidents are quite rare. Since this makes it difficult to train well-performing classifiers (achieving high recall and precision), the authors of [22] have used ensemble methods to obtain good performance on some of such problematic data sets. The same authors used ensemble methods which require training on only portions of the datasets in [23].
Ensemble models have the potential to outperform any individual classifier, but one pays the price of having to train multiple classifiers. When the underlying data sets are sufficiently large, this becomes infeasible or at least very expensive. Compared to canonical distributed machine learning algorithms, training ensemble models generally requires more computational power but less communication bandwidth.

2.3 **Federated Learning**

The Federated Learning paradigm allows one to construct a global model by aggregating models which are trained locally on end user devices [24]. To use Federated Learning to train, for example, a text predictor to be used on mobile devices, each end user updates their own model on their local device and then sends the local gradient update to a central aggregator. The central aggregator averages these gradient updates to construct a global gradient update, which is then pushed to the end users. This process is iterated until convergence.

Federated Learning presents its practitioner with two immediate and substantial advantages over a traditional system in which end-users simply send all of their user data to the central aggregator. Firstly, the communication cost is reduced substantially, since the local models (or model updates) are typically much smaller than the data on which they are trained. Secondly, the end user's privacy and data confidentiality are preserved as they only transmit a model (or a model update) trained on their potentially sensitive data and not the data itself. The model may still leak sensitive data about the end-user, but Federated Learning can be done with a secure aggregator approach [25], which ameliorates this problem using techniques from cryptography.

Federated Learning avoids the high communication costs of traditional distributed machine learning algorithms, as only models are transmitted between the master and worker nodes as opposed to entire end user data sets. Moreover, the individual local models are often much smaller and faster to train than individual global models one might use in a comparable ensemble modelling approach.

While Federated Learning is a fairly new machine learning paradigm, it has already proved effective in constructing global models, e.g., for text prediction. Its relevance and value in the cybersecurity domain are not as well-understood, though there are a few existing works that showcase its potential. For instance, small, potentially vulnerable IoT devices, growing in popularity in the modern world, which are unable to train good defensive models on their own, are natural candidates for beneficiaries of Federated Learning approaches in the realm of cybersecurity.

Recently, cybersecurity vendor Kaspersky published a blog entry [26] explaining how they have implemented models using Federated Learning to filter out spam email. Similarly, Federated Learning has been used to detect phishing emails in [27], where data is aggregated across multiple organizations while preserving user privacy. Federated Learning has been applied to intrusion detection in [28], where the authors integrate Federated Learning with blockchain technology to detect local models which have been poisoned with adversarial examples. In [29], the authors apply Federated Learning to malware classification to achieve high accuracy on the VirusTotal data set. The authors of [30] exploit the privacy-preserving nature of Federated Learning and combine it with secure multi-party computation to construct support vector machine classifiers for Android malware detection. A deep neural network for network anomaly detection and traffic recognition and classification is trained in [31] using Federated Learning, outperforming centralized baseline methods.

IoT devices are natural candidates for end-user nodes in Federated Learning systems, and the authors of [32] exploit this observation to train an intrusion detection model using Federated Learning. However, similar systems were attacked using poisoning methods in [33], showing that Federated Learning systems designed for cybersecurity are themselves potentially vulnerable to attacks. Another application of Federated Learning to intrusion detection was done in [34], where a long short-term memory (LSTM) model was trained using Federated Learning to take advantage of the limited computing power on end-user devices. In [35], Federated Learning has been applied to malicious URL detection in order to increase the accuracy of some detection models.

Although the literature on applications of Federated Learning to cybersecurity is somewhat disparate and seems to lack any truly revolutionary work, the progress made so far is promising.

2.4 Model Combining in Deployed Cybersecurity Solutions

While many cybersecurity vendors emphasize the use of Machine Learning methods in their solutions, there are very few claims of applying model combining approaches to detecting cyberattacks and malicious objects. Besides, some of those claims are quite vague, making readers guess what the implemented approaches are.

Symantec seems to use a technique resembling Federated Learning for malware detection. In [36], we find the following description: "Moreover, Symantec Endpoint Protection can train and test the engine of Advanced Machine Learning (AML) by following certain processes. First, the AML model will be downloaded to the client's device and will let it run for a couple of days. During this period, AML's engine will determine which of the client's applications are exploiting its data. The information gathered will be forwarded to Symantec which allows them to adjust their AML model. Then, this will be modified to block and remove the applications that are exploiting the client's data."

Darktrace apparently uses ensembling in network intrusion detection. In [37], they say: "Employing multiple unsupervised, supervised, and deep learning techniques in a Bayesian framework, the Enterprise Immune System can integrate a vast number of weak indicators of anomalous behaviour to produce a single clear measure of threat probabilities."

Avira clearly claims the use of ensembling for malware and phishing detection in [38]: "Avira does not rely on a single approach to the problem, but uses an ensemble of different Machine Learning techniques, ranging from linear models such as logistic regression to nonlinear models such as kernelized support vector machines, random forests and, for problems where it is the best choice, Deep Learning techniques such as convolutional neural networks. Those techniques are applied for different detection tasks including malware detection and phishing detection, depending on the needs of the user and the capabilities of the underlying platform."

In their whitepaper [39], F-Secure explains in detail one of their approaches to ML model training which exhibits key features of Federated Learning, such as offloading parts of the model training computations to 3rd party machines and providing higher confidentiality guarantees for 3rd party data due to sharing of local models instead of raw data. In fact, some of the models trained in this fashion were developed in SAPPAN, including those covered later in this report.

As was mentioned above, Kaspersky published a blog post [12] elaborating on their application of Federated Learning to spam detection: "E-mails can contain private data, so storing and processing them in their original form would be unacceptable. (...) We solve that problem by using the federated learning method, neatly eliminating the need to collect legitimate e-mails and instead training models in a decentralized way. Model training takes place directly on the client's mail servers, and the central server receives only the trained weights of the machine-learning models, not message text. At the central server, algorithms combine the data with the resulting version of the model, and then we send it back to client's solutions, where model again proceeds to analyse the stream of e-mails."

We expect to see more cases of combining machine learning models in cybersecurity solutions in the coming years, likely including hybrid approaches and mixing techniques from ensembling and distributed and federated learning.

3 SAPPAN Context

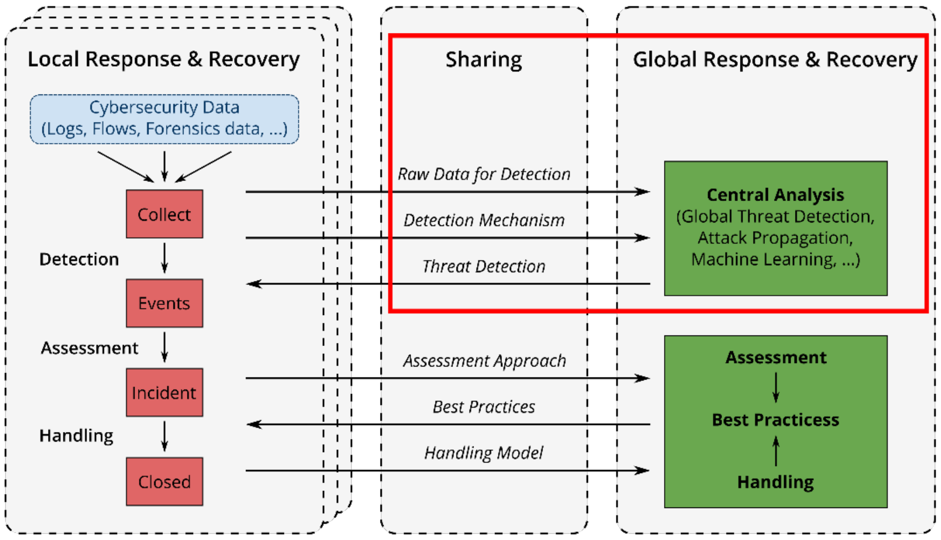


Figure 1. SAPPAN scheme regarding local and global response and recovery

The overall scheme for sharing, detection, and response in SAPPAN is shown in Figure 1. The top half of the scheme describes the detection components, and the bottom half - the response components, while the left half corresponds to the local level, and the right half - to the global level. The goal of WP5 is to implement the global level with respect to sharing of data and models, building global models for detection, sharing of response and recovery information, as well as visualization. The tasks 5.1, 5.2 and 5.3 include the development of global models for detection, based on several approaches. The general idea is to utilize the data and models developed in Task 3.3 on the local level by sharing them among multiple organizations to build global detection models. The goal is to end up with global detection mechanisms that are superior to the local ones. Another flavour of sharing in the scope of SAPPAN is sharing among a cybersecurity vendor and various user groups of its customer organizations. In such cases, we aggregate data or local attack detection models built in individual endpoints. Key problems with respect to sharing are, of course, privacy and efficiency, which are tackled by an assortment of approaches investigated in T5.1, T5.2 and T5.3. The approaches range from sharing of anonymized data, to sharing of only pre-trained models, to replacing sharing by other techniques. We apply these techniques to several showcases similar to those described in WP3 in order to build global detection models with adequate recall and precision and providing certain levels of privacy and efficiency, including cost-efficiency.

In Task 5.2 and in this deliverable, our focus is on the approaches to building global models based on shared - in the Federated Learning fashion - local models. The general Federated Learning paradigm and some of its characteristics and applications were presented in the "Approaches to Combining Machine Learning Models in Cybersecurity Applications" section above.
It is worth noting that an automated system that can recognize true alerts (that require response actions) is the first step towards a truly automated response system. Understanding the type and severity of a security incident that triggered an alert can then be used to choose appropriate response actions, which can be suggested to security personnel or, in certain cases, even carried out automatically.

4 Distributed Learning of a Global Model Based on Shared Local Models

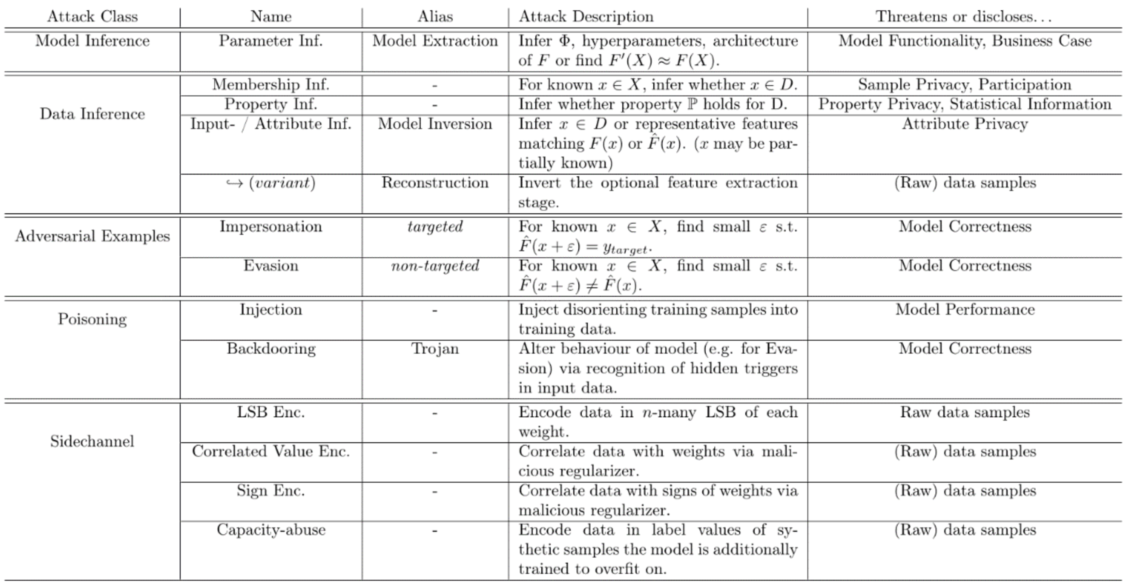
4.1 Privacy

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

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

or just the predicted class as such:

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



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

Figure 2. Attack landscape in machine learning.

When restricting the view to the sharing scenarios in this deliverable, which is about building an ensemble (or aggregate) classifier out of locally trained models, the relevant attack class is the Data Inference attacks [40, 41, 42] with an emphasis on Membership Inference and Model Inversion attacks. As trained models are willingly shared, these attacks can be performed in the white-box scenario, when an adversary has full access to the model and its weight. This enables Model Inversion attacks which might require access to the gradient computation of a shared deep learning model. The other attack classes, i.e., Model Extraction [43], Adversarial Examples [44], Poisoning [45] and Side-channel attacks [46], are not of relevance for this deliverable. Anonymization techniques, privacy-preserving training algorithms and other measures allow to shrink the attack surface or the feasibility of an attack. A privacy evaluation is demonstrated in the DGA showcase: For each DGA sharing scenario, we evaluate the privacy leakage by measuring the success of the relevant attacks.

4.2 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 D5.1 (global model based on shared anonymized data), D5.3 (global model based on shared local models), and D5.5 (global model without sharing local models). In addition, we investigate the privacy implications caused by data sharing for this use case in all the three deliverables.

Due to this commonality, the three deliverables share the same text for the sections that include general information such as DGA detection background, state-of-the-art classifiers, and parts of the evaluation setup. However, the sections that depend on specific sharing scenarios, such as the actual evaluation and the privacy study, are deliverable-specific.

4.2.1 Background

We presented DGA detection in D3.4 (Algorithms for Analysis of Cybersecurity Data) in detail. As a reminder, we briefly discuss the most important aspects here.

Modern botnets rely on DGAs to establish a connection to their command and control (C2) servers. In contrast to the usage of single fixed IP-addresses or fixed domain names, communication attempts of DGA-based malware are harder to block as they utilise a large number of algorithmically generated domains (AGDs). The botnet herder is aware of the generation scheme and thus able to register a small subset of the generated domains in advance. The bots, however, query all generated AGDs, trying to obtain the valid IP-address for their C2 server. As most 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 to their 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 partake in any malicious action. This detection is, however, not trivial, since NXDs can also be the product of typing errors, misconfigured or outdated software, or the intentional misuse of the DNS e.g., by antivirus software. In the following, we refer to this detection in which we separate benign from malicious domain names as the DGA binary classification task.

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

In the past, several approaches have been proposed to detect DGA activities within networks. These approaches can be split into two groups: contextless and context-aware approaches. In SAPPAN, we focus on contextless approaches (e.g. [47, 48, 49, 50, 51, 52]), as they rely entirely 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. [53, 54, 55, 56, 57, 58]) 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 [47, 48, 50, 51, 52].

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. [48, 53]) and deep learning (featureless) classifiers (e.g. [47, 50, 51, 52]). While the deep learning classifiers outperform the feature-based approaches in terms of classification performance [50, 51, 59, 60, 61], 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 greater 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 the 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 may 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 [62] that we introduced in deliverable D3.4 (Algorithms for Analysis of Cybersecurity Data). There, we demonstrated that our classifiers achieve better classification scores (f1-score / false positive rate) than the state-of-the-art classifiers proposed in related work. We note that in order to address the explainability problem of deep learning classifiers, we developed in SAPPAN a visual analytics system [63] which tries to bridge the gap between the predictions of deep neural networks and human understandable features.

4.2.2 Selected State-of-the-Art Classifiers

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

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

FANCI

Schüppen et al. [48] 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 to be classified. Thus, FANCI works completely contextless. FANCI does not incorporate DGA multiclass classification support.

Endgame

Woodbridge et al. [51] 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. [52] 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 [64] as optimization algorithm and the categorical cross-entropy for computing the loss during training. We refer to the multiclass enabled model as M-NYU in the following.

ResNet

In the context of SAPPAN, we developed a binary and a multiclass DGA classifier based on ResNets [62]. We presented all the details as well as a comparative evaluation with the state-of-the-art in deliverable D3.4 (Algorithms for Analysis of Cybersecurity Data). ResNets make use of so called skip connections between convolutional layers which build up residual blocks. These blocks allow the gradient to bypass layers unaltered during the training of a classifier and thereby effectively mitigate the vanishing gradient problem [65, 66]. 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. [50] showed that the model of Woodbridge et al. [51] 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 re-balanced. 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 incorpo-rate class weighting with the suffix ".MI".

4.2.3 Evaluation Setup

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

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

Data Sources

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

Malicious data

We obtain malicious domains from the open-source intelligence feed of DGArchive [67] which contains more than 126 million unique domains generated by 94 different known DGAs. We make use of all the available data up to 2020-09-01.

Benign data

We obtain benign labelled NXDs from three different sources, namely from the net-works of CESNET, Masaryk University, and RWTH Aachen University. For the 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 well as possible.

Additionally, we remove the intersection of all the samples obtained 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 periods for data collection overlap. Note, thereby we are also removing samples from both data sources which would naturally be present in both networks even when they were not interconnected. Such samples could be common typos of popular websites. This issue could have an effect on classification performance of classifiers 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 numbers of unique samples obtained from each source for benign data:

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

4.2.4 Sharing Scenario

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

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

Sharing Scenario - Ensemble Classification

In this scenario, we build a global model based on shared local models. Each collaborating party trains a local DGA detection model using their own private data. These models are then shared, enabling every party to create the global model. When a party wants to evaluate new domain names they feed them into all the local models. The final classification results can then be obtained by two approaches:

1. Averaging the local models' confidence scores

The final classification result equals the average of the local models' confidence scores. For instance, an average score above a certain threshold (e.g., 0.5) would indicate that the input domain is classified as malicious.

2. Majority vote

The final classification result equals the majority vote of the local models. A tie breaker is needed for an even number of local models. For instance, the tie breaker could be whether the average of the local models' confidence scores is above a certain threshold.

We also plan to quantify the loss in utility after applying different types of privacy-enhancing techniques. In this scenario, the overall privacy could be enhanced by using differentially private stochastic gradient descent to train the local models. However, this might have a negative impact on the classification performance.

4.2.5 Planned Privacy Analysis

In the next iteration of this deliverable, we plan to present an evaluation of privacy in the DGA sharing scenarios. This includes an assessment of the existing threats against these scenarios which are Model Inversion and Membership Inference. As the local models in this sharing scenario are sharded freely, Model Extraction attacks would not be of relevance here. Also, as the ensemble classifier is built via a collection of locally trained models, there is no global training procedure that an adversary can utilize as an attack surface. However, as the plain local models are accessible, the Input Inference attacks possibly pose a threat to privacy. This assessment will be quantified as objectively as possible, for instance by the success ratio of Membership Inference attacks, or by a sample distance metric as it is currently done in deliverable 5.1 with the Levenshtein distance. For attacks that pose a severe threat towards privacy in sharing, sophisticated defence methods shall be evaluated, which can include training the local models with a privacy-preserving training algorithm (e.g., with differential privacy [68]) before sharing a model or utilizing the PATE approach [69]. Privacy-preserving techniques must be evaluated and compared regarding the negative performance impact (e.g., loss in accuracy) they likely carry with them.

4.3 Application Profiling

4.3.1 Background

The background for application profiling in the context of sharing is similar in all the tasks of WP5. Hence, the following paragraph can also be found in the Deliverables D5.1 and D5.5.

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

4.3.2 Sharing Scenario

In the following, we will describe the sharing scenarios we plan to evaluate for the application profiling showcase. The scenarios are similar to the ones described in Deliverable D5.1. However, instead of sharing datasets to compute global models, the local models are shared and combined to a global model. This has the advantage that no datasets need to be anonymized because they stay local. On the other hand, less information is shared and hence less information from the local level is available to build the global models. This is a trade-off between privacy and utility. At the moment of writing this deliverable, we have not yet conducted the experiments to measure this. In the following, we will outline the general idea of the three approaches.

Sharing of rule sets to build a global model

Instead of sharing datasets to extract rules on the global level, as described in Deliverable D5.1, we directly share locally extracted rules to the global level. Ideally, the rule extraction process already takes care of filtering out personal information to remove it from such rules, which should only describe the behaviour of an application independent of specific users. The global model can be built by combining the local rules. For this merging process, we plan to evaluate multiple options. The base approaches are building a union of all rules, or building an intersection. The motivation for union is that it allows for integration of edge-case behaviour into the global rule set, which was only contained in one or a few of the shared local rules. However, besides benign edge case behaviour, such rules can also be a result from misconfigured or even infected applications, which leads to the motivation of building the intersection of all rules. By doing that, we end up with a global rule set that only contains rules that are assumed to be correct by all participating organizations.

For the identification task, we expect that the union approach leads to a higher false positive rate but lower false negative rate, while the intersection approach leads to a lower false positive rate but higher false negative rate. Depending on the use-case, either approach or a hybrid one can be useful. For the classification task, we expect that a hybrid approach is necessary, because edge-cases are relevant for the detection of anomalies. On the other hand, such edge-cases are already anomalies, e.g. if they only occur in one rule set, and including them as expected behaviour to the global rule set might be misleading.

Sharing of process mining models to build a global model

The general idea of sharing process mining models is similar to the sharing of rule sets as described above. We compute process mining models in the form of petri-nets (as described in Deliverable D3.4). Petri-nets basically model all possible sequences of events, in our case either DNS events for the identification task or system events for the classification task. With a fitting syntax, these petri-nets could also be converted to a set of rules, by translating all transitions accordingly. The approaches using union or intersection described above can be applied in the same way to these models, by adding or removing the corresponding transition. Hence, the same implications with respect to utility apply.

We plan to evaluate the described approaches for the rule-based and process mining approaches for the final version of this deliverable.

Sharing of machine learning models to build a global model

As described in Deliverable D5.1, we focus on the rule-based and process mining approaches, especially for the classification task. However, in case we are able to develop machine learning models for the identification task that perform similar to the other approaches, these models can also be shared to the global level. Similar to the

DGA detection showcase, these models can be combined in form of an ensemble-classifier. However, because the other approaches seem more promising, it is not yet clear whether we will also evaluate the performance of such a machine learning based model on the global level.

4.4 Detection of Anomalous Process Launch and Process Access Operations in Endpoints

The Process Launch Distribution (PLD) model for detecting anomalous process launch events was described in D3.4. Here, we elaborate on the local PLD models aggregation approach, show some experimental results, and introduce a similar but somewhat more advanced model for detecting anomalous process access events.

4.4.1 Background

As explained in D3.4, PLD models the distribution of process launch events in endpoints and detects events anomalous for this distribution. It belongs with the class of statistical distribution models with thresholds. The underlying assumptions in PLD are (i) that benign events in a training set are much more frequent than events connected with attacks, and (ii) that process launch events where common processes are used in anomalous ways are sufficiently reliable signs of attacks. PLD defines a score function that reflects how anomalous a given process launch event is: the lower the PLD score is, i.e., the closer it is to zero, the more anomalous the process launch event is from the PLD model point of view, thus, the more suspicious it is from our security intuition point of view. Given a training set of process launches observed in selected endpoints, we can build a PLD score distribution model and define several threshold values corresponding to exponentially lower percentiles of the PLD score cumulative distribution in the training set. Those thresholds naturally define anomaly categories (an example is provided in Figure 3 below), and the higher the anomaly category index is, the more suspicious a process launch event is.

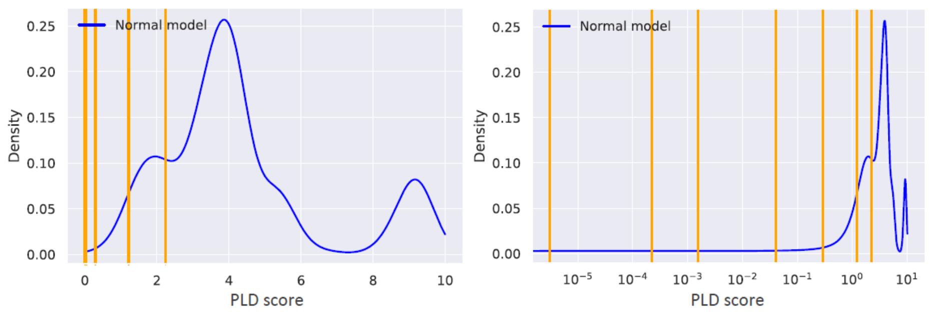


Figure 3. Density distribution of PLD score (blue curve) and thresholds inferred from it (orange lines). Left: PLD score linear scale. Right: PLD score log scale. In both graphs, from right to left: 7 thresholds for categories.

At the inference time, the PLD score is computed for each observed process launch event and alerts are raised for the events with high anomaly categories.

4.4.2 Distributed Online Training of PLD Model

The simplest way to train a PLD model would be to collect details of process launch events occurring in customer machines in a security monitoring service backend and compute their PLD scores distribution and the thresholds. Since this would expose unnecessary customer information to the security provider and entail significant data handling costs, we took a different approach in the project. Simple local PLD models are produced at every participating endpoint at regular time points, and only those local models are sent to the backend and aggregated into a global PLD model there. This approach dramatically reduces the amount of data transmitted to and processed at the backend, thereby easing the data confidentiality concerns as well. While it is not exactly federated learning in the original Google style, it exhibits clear similarities and brings the same key benefits.

4.4.3 Combining Local and Global PLD Models, Experimental Results

Since the chosen training approach requires local PLD models in every endpoint, we can compute for a new process launch event both its local and global PLD scores, and take both numbers into account when deciding whether an alert should be raised for the event. The availability of local PLD scores can help reduce the false positive rate by not alerting on events with high global but low local PLD anomaly categories.

A local PLD model measures anomalousness relative to a single endpoint, and the global PLD model measures anomalousness within the entire set of the monitored endpoints. The interplay between these two models allows us to better understand how anomalous a process launch event truly is. In Figure 4, we display the count of events as a function of both global (the horizontal axis) and local (the vertical axis) anomaly categories (the categories definition can be found in D3.4). This plot is based on one month of data from all the endpoints in a validation set. As expected, the most populous square is when both models predict the event is not anomalous at all, and the number of events steadily decreases as both categories increase. The jump occurring when the local PLD anomaly category = 90 is due to a smoothing factor introduced into the defining equation of the PLD score, intended to highlight locally anomalous events.

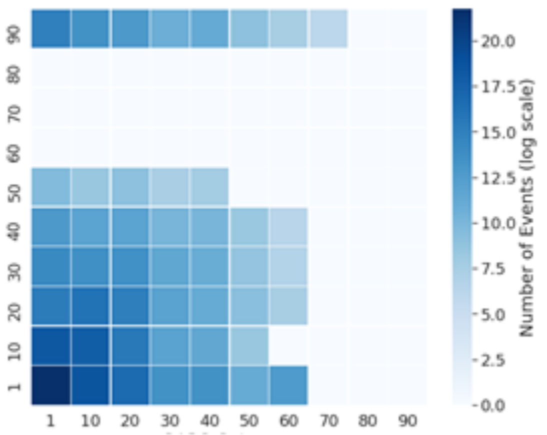


Figure 4. A heatmap displaying the number of events (log scale) contained in each (LPLD category, GPLD category) pair, for one month's worth of endpoint data.

To further study the effectiveness of our PLD models, we plot in Figure 5 the probability that a process launch event triggers a rule-based (expert-defined) detection as a function of both PLD scores. As one would expect, this probability is generally higher for events which have high local and global PLD anomaly categories. The lack of obvious direct correlation and the 'drop-offs' in probability on either side of the global PLD = 60 and 70 categories are likely consequences of various complexities of our detection rule engine (e.g., heuristic FP prevention rules) and the lack of events with the global PLD score higher than 70 in the validation data (see also Figure 4). Overall, the heatmap shows that our two models combined can serve as a decent predictor for how suspicious a certain process launch is, validating their usefulness.

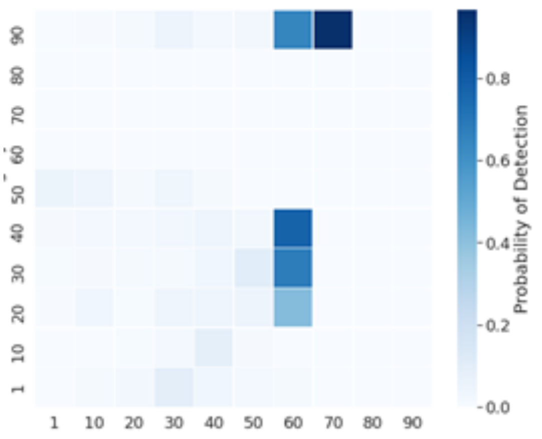


Figure 5. Probability of detection for events as a function of both PLD scores, for one month's worth of endpoint data.

4.4.4 Anomalous Process Access (APA) Model

The APA model is similar to PLD and can be considered an extension of the main ideas behind PLD.

An event of the *process access* (called sometimes *process open*) type represents an operation of opening a running local process object by another local process. (We note that any process launch results in a process access event; our security monitoring sensor recognizes such cases and ignores process access events automatically generated by process launches.) Analysis of events of this type is useful for detecting adversarial process manipulation techniques, such as DLL injection, thread execution hijacking and process hollowing, including important entries in the MITRE ATT&CK knowledge base: credential dumping [70], impairing defences [71], and so on. Similarly, to the PLD case, two key attributes of process access events are *actor* (opening) process and *actee* (being opened) process. Another attribute relevant for security is *desired access rights*; we denote it by *access_mode*. In Microsoft Windows, for instance, desired access value is a combination of several predefined permission flags [72]. As a simple example, the presence of all the defined flags in the attribute's value is a sign of a potentially misconfigured actor which does not follow *the least privilege principle* and deserves attention of security analysts.

Attribute name	**Details**
Actor	An identifier of the local process opening another local process (the current choice is to use the process file name as its ID).
Actee	An identifier of the local process to be opened (file name of the process).
access_mode	A bitmask specifying desired access rights.

Table 1. Attributes for process access events.

The PLD approach to identifying anomalous actor - actee pairs does not take into account the behavioural 'uncertainty' of actor processes (the same applies, of course, to actee processes). A hypothesis that came up in the discussions with security experts and seems to be confirmed by experiments is that many false positives of PLD models can be explained by highly 'random' process launching behaviour of actors (which informally means that it is difficult to guess what process a given actor will launch or - in the APA case - open). Examples of 'random' actors are executables belonging to Microsoft Windows Installer, Console Windows Host, Client / Server Runtime Subsystem.

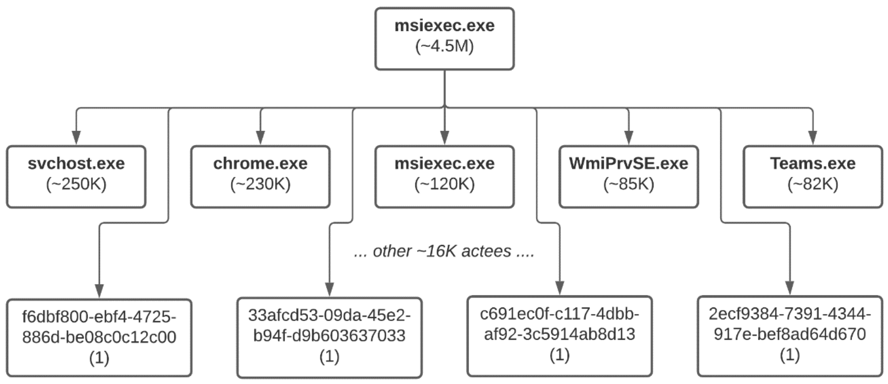

msiexec.exe (~4.5M)
svchost.exe (~250K)
chrome.exe (~230K)
msiexec.exe (~120K)
WmiPrvSE.exe (~85K)
Teams.exe (~82K)
... other ~16K actees ....
f6dbf800-ebf4-4725-886d-be08c0c12c00 (1)
33afcd53-09da-45e2-b94f-d9b603637033 (1)
c691ec0f-c117-4dbb-af92-3c5914ab8d13 (1)
2ecf9384-7391-4344-917e-bef8ad64d670 (1)

Figure 6. Processes manipulated by Microsoft Windows Installer.

Figure 6 depicts the actees and the corresponding process open statistics for one of the most unpredictable actors: msiexec.exe, which is a part of the Microsoft Windows Installer technology. The collected data show that msiexec.exe opens approximately 16,000 actee processes, 99% of which have random file names and are actually installation-time-specific temporary executables.

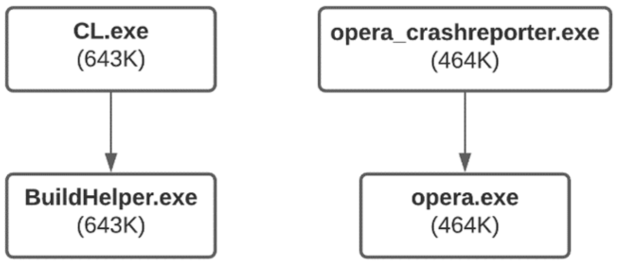


Figure 7. Examples of deterministic actor-actee pairs.

Figure 7 presents the other extreme: examples of actor-actee pairs which always appear together. In particular, the actors of these pairs can be considered fully predictable. So, any process access event with one of these two actors and a different actee would deserve a security investigation.

It was discussed in [73] that the PLD method could potentially be improved by taking into account the 'uncertainty' of the process launching behaviour of actors. That idea was implemented in the APA model by introducing an (empirical) entropy factor that penalizes for highly uncertain actor's behaviour (cf. the PLD_score formula in D3.4):

$$score_{pair}(a,\ b) = \frac{\frac{\#(a,\ b) +\ \alpha_1}{t +\ \beta_1}}{\frac{\#a +\ \alpha_2}{t +\ \beta_2} \times \frac{\#b +\ \alpha_3}{t +\ \beta_3}} \times e^{H(a)}$$

where

- $\#(a)$ is the number of the occurrences of a as the 1st element of a pair in the training set
- $\#(b)$ is the number of the occurrences of b as the 2nd element of a pair in the training set
- $\#(a,\ b)$ is the number of the occurrences of pair $(a,\ b)$ in the training set
- t is the total number of elements in the training set
- α_1, α_2, α_3 and β_1, β_2, β_3 are smoothing and other constants, often chosen to optimize the model performance

and the added multiplier’s power is defined as an empirical entropy:

$$H(a) = -\sum_{x \in B} \frac{\#(a,x)}{\#a} \log\left(\frac{\#(a,x)}{\#a}\right)$$

where *B* is the set of all the 2nd elements of the pairs which have a as the 1st element.

Another new element in the APA model is the use of the access_mode attribute. In addition to computing the $score_{pair}(a,\ b)$ for (actor, actee) pairs, we do the same for (ordered) pairs of the two other types: (actor, access_mode) and (actee, access_mode). Our experiments showed that aggregating these three score values for a process access event by selecting the minimal of them as the event score brings significant improvements in the APA model performance.

The entropy factor brings an obvious asymmetry to the $score_{pair}(a,\ b)$ formula. The order of the elements in the three types of pairs was selected experimentally and can be re-considered later.
Figure 8 shows the three selected APA attributes and the relations among them.

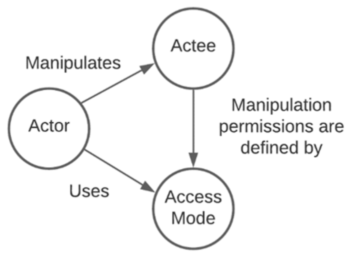


Figure 8. Selected representation of process access events in the entity relationship form.

APA Model in Action

Here is a somewhat artificial but illustrative example of the effects of the APA model enhancements. The model alerts on the presence of the mimikatz tool (a post-exploitation tool popular among attackers) in the endpoint data collected during a red teaming exercise. The process access event detected by the model is (msmpeng.exe, mimikatz.exe, 0x101001), which translates into: "A scanning service process of Windows Defender opens a local process mimikatz.exe with the desired access rights to terminate the actee process and retrieve certain information about it".

Field index	Field name	Rating in the top anomalous process access events list		
		I	II	III
1	Actor name	msmpeng.exe	svchost.exe	svchost.exe
2	Actee name	mimikatz.exe	ngen.exe	mscorsvw.exe
3	Desired access_mode value	0x101001	0x1028	0x1028
4	$\log_{10}(score_{pair}(\text{actor}, \text{actee}))$	~ 4	~ -1	~ 1
5	$\log_{10}(score_{pair}(\text{actor}, \text{access\_mode}))$	$\boldsymbol{\sim -7}$	~ 1	~ 1
6	$\log_{10}(score_{pair}(\text{actee}, \text{access\_mode}))$	~ 2	$\boldsymbol{\sim -4}$	$\boldsymbol{\sim -4}$

Table 2. An excerpt from the "top anomalous process access events" list for a red teaming test data set.

Since the mimikatz binary is not common, the entropy-enhanced PLD approach (row #4 in Table 2) does not rate events with mimikatz as actee as highly anomalous (note that the actor in this event - msmpeng.exe - is very common). For the same reason, the (actee, access_mode) pair's score is moderately high (therefore the pair is not considered highly anomalous). However, the (actor, access_mode) pair is a very rare combination of a common executable and a common requested access mode and considered anomalous. Thus, the overall score of the event is low, which will draw attention of the security personnel.

5 Conclusion

In this deliverable, we presented the first results of Task 5.2: "Federated learning of a global model based on shared locally trained models". We started with an exposition of the state-of-the-art in combining machine learning models for cybersecurity applications, covering ensembling and distributed and federated learning and including several publicly known use cases in deployed solutions. It appears that both academic research and industrial applications in this domain are not very mature at the moment, so the SAPPAN efforts seem relevant and timely.

We then outlined the context of this task in the overall scope of the SAPPAN project and explained the general concept of the task. Several showcases illustrating different approaches to building global detection models and the first results of our experiments were presented after that. The showcases include DGA detection, application profiling, and anomalous behaviour detection in endpoints. In the anomalous behaviour detection line of work, we presented PLD models trained in a federated learning-like fashion, the use case for combining local and global PLD models, and the evidence of a good potential of the PLD models combining approach. We then introduced the APA model - as an extension of the PLD approach - for detecting anomalous process access events in endpoints. Despite some promising results and evidence, clearly, more work is still required for evaluating the models' performance and for shaping and implementing improvement ideas.

6 Acknowledgements

We would like to thank the CESNET and Masaryk University colleagues for their contributions. We would also like to thank all the SAPPAN consortium members and the Artificial Intelligence Center of Excellence team at F-Secure for their support, comments and helpful discussions.

7 Bibliography

[1] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Accessed on January 24th, 2021.

[2] J.-H. Li. Cyber security meets artificial intelligence: a survey. [BBCC'19] D.S. Berman, A.L. Buczak, J.S. Chavis, C.L. Corbett. A Survey of Deep Learning Methods for Cyber Security. In *Information* 10, no. 4 (2019).

[3] A. Handa, A. Sharma, S.K. Shukla. Machine learning in cybersecurity: a review. In *WIREs Data Mining and Knowledge Discovery* 9, no. 4 (2019).

[4] D.S. Berman, A.L. Buczak, J.S. Chavis, C.L. Corbett. A Survey of Deep Learning Methods for Cyber Security. In Information 10, no. 4 (2019).

[5] U. Fiore, F. Palmieri, A. Castiglione, A. De Santis. Network anomaly detection with the restricted Boltzmann machine. In *Neurocomputing* 122, pp. 13-23 (2013).

[6] J. Cannady. Artificial neural networks for misuse detection. In *Proceedings of the National Information Systems Security Conference* (1998).

[7] Zhihua Cui, Fei Xue, Xingjuan Cai, Yang Cao, Gai-ge Wang, and Jinjun Chen. Detection of malicious code variants based on deep learning. IEEE Transactions on Industrial Informatics, 14(7):3187–3196, 2018

[8] Quan Le, Oisín Boydell, Brian Mac Namee, and Mark Scanlon. Deep learning at the shallow end: Malware classification for non-domain experts. Digital Investigation, 26:S118–S126, 2018

[9] T.T. Nguyen and V.J. Reddi. Deep Reinforcement Learning for Cyber Security (2020). Available at arxiv.org/pdf/1906.05799. Accessed on January 29th, 2021.

[10] https://www.microsoft.com/security/blog/2019/06/24/inside-out-get-to-know-the-advanced-technologies-at-the-core-of-microsoft-defender-atp-next-generation-protection/. Accessed on January 27th, 2021.

[11] https://www.sophos.com/en-us/medialibrary/PDFs/factsheets/sophos-intercept-x-deep-learning-dsna.pdf. Accessed on January 27th, 2021.

[12] https://media.kaspersky.com/downloads/business/Broshure_B2B_MachLearnHumExpert_Eng.pdf. Accessed on January 27th, 2021.

[13] https://www.fireeye.com/blog/products-and-services/2018/07/malwareguard-fireeye-machine-learning-model-to-detect-and-prevent-malware.html. Accessed on January 27th, 2021.

[14] https://www.crowdstrike.com/blog/defending-against-malware-with-machine-learning/. Accessed on January 27th, 2021.

[15] N.N.A. Sjarif, S. Chuprat, M.N. Mahrin, N.A. Ahma, A. Ariffin, F.M. Senan, N.A. Zamani, A. Saupi. Endpoint Detection and Response: Why Use Machine Learning? In *International Conference on Information and Communication Technology Convergence* (ICTC, 2019).

[16] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, J. Rellermeyer. A Survey on Distributed Machine Learning. In *ACM Computing Surveys* 53**,** no. 2 (2020).

[17] H. Attak, M. Combalia, G. Gardikis, B. Gaston, L. Jacquin, D. Katsianis, A. Litke, N. Papadakis, A. Pastor, M. Roig, O. Segou, D. Papadopoulos. Application of distributed computing and machine learning technologies to cybersecurity. In *C&ESAR 2018: Artificial Intelligence and Cybersecurity* (2018).

[18] R. Kozik, M. Choras, M. Ficco, F. Palmieri. A scalable distributed machine learning approach for attack detection in edge computing environments. In *Journal of Parallel and Distributed Computing* 119, pp. 18-26 (2018).

[19] X. Chen, J. Ji, C. Luo, W. Liao and P. Li, "When Machine Learning Meets Blockchain: A Decentralized, Privacy-preserving and Secure Design," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 1178-1187, doi: 10.1109/BigData.2018.8622598.

[20] N. Lower and F. Zhan. A Study of Ensemble Methods for Cyber Security. In *10th Annual Computing and Communication Workshop and Conference* (CCWC, 2020).

[21] P. Venosa, S. Garcia, F.J. Diaz. A Better Infected Hosts Detection Combining Ensemble Learning and Threat Intelligence. In: Pesado P., Arroyo M. (eds) *Computing Science* – CACIC 2019. Communications in Computer and Information Science 1184, Springer.

[22] G. Folino and F.S. Pisani. Evolving meta-ensemble of classifiers for handling incomplete and unbalanced datasets in the cyber security domain. In *Applied Soft Computing* 47, pp. 179-190 (2016).

[23] G. Folin and F.S. Pisani. Combining Ensemble of Classifiers by Using Genetic Programming for Cyber Security Applications. In: Mora A., Squillero G. (eds) *Applications of Evolutionary Computation*. EvoApplications 2015. Notes in Computer Science 9028, Springer.

[24] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, B. Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (AISTATS, 2017).

[25] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H.B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017).

[26] N. Benkovitch. Federated learning in the fight against e-mail threats. Kaspersky daily blog, December 8th, 2020. Available at https://www.kaspersky.com/blog/federated-learning-against-mail-threats/37936/. Accessed on January 22nd, 2021.

[27] C. Thapa, J.W. Tang, S. Abuadbba, Y. Gao, Y. Zheng, S.A. Camtepe, S. Nepal, M. Al-mashor. FedEmail: Federated learning in phishing email detection to enabling multiple organizational collaborations without sharing email data. Available at https://arxiv.org/abs/2007.13300. Accessed on January 24th, 2021.

[28] Preuveneers, D., Rimmer, V., Tsingenopoulos, I., Spooren, J., Joosen, W., & Ilie-Zudor, E. (2018). Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. Applied Sciences, 8, 2663.

[29] K.-Y. Lin and W.-R. Huang. Using Federated Learning on Malware Classification. In *22nd International Conference on Advanced Communication Technology* (ICACT, 2020).

[30] R.-H. Hsu, Y-C. Wang, C.-I. Fan, B. Sun, T. Ban, T. Takahashi, T.-W. Wu, S.-W. Kao. A Privacy-Preserving Federated Learning System for Android Malware Detection Based on Edge Computing. In *15th Asia Joint Conference on Information Security* (AsiaJCIS, 2020).

[31] Y. Zhao, J. Chen, D. Wu, J. Teng, S. Yu. Multi-Task Network Anomaly Detection using Federated Learning. In *Proceedings of the Tenth International Symposium on Information and Communication Technology* (SoICT, 2019).

[32] S.A. Rahman, H. Tout, C. Talhi, A. Mourad. Internet of Things intrusion Detection: Centralized, On-Device, or Federated Learning? In *IEEE Network* 34, no. 6, pp. 310-317 (2020).

[33] T.D. Nguyen, P. Rieger, M. Miettinen, A.-R. Sadeghi. Poisoning Attacks on Federated Learning-based IoT Intrusion Detection System. In *Workshop on Decentralized IoT Systems and Security* (DISS, 2020).

[34] R. Zhao, Y. Yin, Y. Shi, Z. Xue. Intelligent intrusion detection based on federated learning aided long short-term memory. In *Physical Communication* 42 (2020).

[35] E. Khramtsova, C. Hammerschmidt, S. Lagraa, R. State. Federated Learning for Cyber Security: SOC Collaboration for Malicious URL Detection. In *IEEE International Conference on Distributed Computing Systems* (ICDCS, 2020).

[36] https://comparecamp.com/sophos-vs-symantec-endpoint-protection-comparison/. Accessed on January 27th, 2021.

[37] https://www.darktrace.com/en/resources/wp-machine-learning.pdf. Accessed on January 27th, 2021.

[38] https://assets.prod.cms.avira.com/cache-buster-1598423379/assets/oem.avira.com/resources/to%20delete/whitepaper_NightVision_EN_20170704.pdf. Accessed on January 27th, 2021.

[39] https://www.f-secure.com/content/dam/f-secure/en/business/common/collaterals/f-secure-whitepaper-blackfin.pdf. Accessed on January 27th, 2021.

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

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

[42] Matt Fredrikson, Somesh Jha and Thomas Ristenpart. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015.

[43] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter and Thomas Ristenpart. Stealing machine learning models via prediction apis. 25th USENIX Security Symposium (USENIX Security 16).

[44] Ian J Goodfellow and Jonathon Shlens and Christian Szegedy. Explaining and Harnessing Adversarial Examples. 2014.

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

[46] Congzheng Song, Thomas Ristenpart and Vitaly Shmatikov. Machine learning models that remember too much. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.

[47] 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.

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

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.

[54] 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.

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

[56] 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.

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

[58] 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.

[59] 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.

[60] 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.

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

[62] 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.

[63] 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.

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

[65] 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.

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

[67] 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.

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

[69] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar and Ú. Erlingsson. Scalable Private Learning with PATE. 2018.

[70] MITRE ATT&CK techniques: OS Credential Dumping, https://attack.mitre.org/techniques/T1003/. Accessed on January 29th, 2021.

[71] MITRE ATT&CK techniques: Impair Defenses, https://attack.mitre.org/techniques/T1562/. Accessed on January 29th, 2021.

[72] Microsoft documentation: Process Security and Access Rights, https://docs.microsoft.com/en-us/windows/win32/procthread/process-security-and-access-rights. Accessed on January 29th, 2021.

[73] Komashinskiy D., Karpuk D., Marshal S., Kirichenko A.: An anomaly detection approach to analysis of security monitoring data from endpoints. International Workshop on Next Generation Security Operations Centers (NG-SOC 2020) in conjunction with the 15th International Conference on Availability, Reliability and Security (ARES 2020).