Feature Cultivation in Privileged Information-augmented Detection

Z. Berkay Celik
Department of CSE
The Pennsylvania State University
zbc102@cse.psu.edu

Patrick McDaniel
Department of CSE
The Pennsylvania State University
mcdaniel@cse.psu.edu

Rauf Izmailov
Vencore Labs
Basking Ridge, NJ, US
rizmailov@appcomsci.com

ABSTRACT

Modern detection systems use sensor outputs available in the deployment environment to probabilistically identify attacks. These systems are trained on past or synthetic feature vectors to create a model of anomalous or normal behavior. Thereafter, run-time collected sensor outputs are compared to the model to identify attacks (or the lack of attack). While this approach to detection has been proven to be effective in many environments, it is limited to training on only features that can be reliably collected at detection time. Hence, they fail to leverage the often vast amount of ancillary information available from past forensic analysis and post-mortem data. In short, detection systems do not train (and thus do not learn from) features that are unavailable or too costly to collect at run-time. Recent work proposed an alternate model construction approach that integrates forensic "privilege" information—features reliably available at training time, but not at run-time—to improve accuracy and resilience of detection systems. In this paper, we further evaluate two of proposed techniques to model training with privileged information: knowledge transfer, and model influence. We explore the cultivation of privileged features, the efficiency of those processes and their influence on the detection accuracy. We observe that the improved integration of privileged features makes the resulting detection models more accurate. Our evaluation shows that use of privileged information leads to up to 8.2% relative decrease in detection error for fast-flux bot detection over a system with no privileged information, and 5.5% for malware classification.

CCS Concepts

•Computing methodologies → Learning paradigms; •Security and privacy → Intrusion detection systems;

Keywords

Privileged information, intrusion detection.

1. INTRODUCTION

Detection systems use traditional learning algorithms such as support vector machines and neural networks to learn detection models. The models aim at learning patterns to estimate an underlying dependency, structure or behavior of a system with a limited number of observations from historical data (also referred as training data). A training data is a collection of a pair of features and class labels such as anomalous or normal. For instance, in network level malware detection, the features may be extracted from packets of incoming/outgoing traffic. In mobile malicious application identification, features may contain textual descriptions and permissions. The detection systems seek to learn models from the features that can accurately make predictions on unseen samples.

The quality of the detection model is largely dependent on extracting all information relevant to a particular task so as to improve the accuracy. However, learning algorithms typically used for training of detection systems build a model assuming that the features used to make predictions at run-time will be identical to the one used for training. This model restricts the detection systems to the features that are reliably measurable once deployed to make predictions. For instance, security data repositories store a myriad of information from packets, log files, and other sources, but they cannot be all leveraged by detection systems in real-time [1, 2]. Much of the useful information only becomes available after further investigation and analysis. This latency thus prevents detection systems from including potentially key features at run time. Therefore, the fundamental problem is using such features at training; even if constraints at run-time prevent them being used by the detection systems.

Our starting point is recent work by Celik et al. [3], which used privileged information (features available at training time, but not at run-time), to improve the accuracy of detection. Their analysis of privileged information demonstrated that the additional discriminating knowledge extracted from privileged features systematically enhances the resulting model's generalization capabilities outside of their traditional feature set. The authors explored the use of privileged information and noted briefly that the cultivation of the privileged features was essential for good accuracy. In this paper, we explore that aspect of technique selection that is cultivated by the features of the dataset. In this, we make three primary contributions:

- We demonstrate the utility of privileged information with several examples from recent literature.
- We apply two different approaches to train detection systems that exploit privileged information, and present guidelines and cautions for the cultivation of privileged information to maximize detection gain.
- We show that correct cultivation of privileged information in detection systems decreases relative detection error on average 8.2% for fast-flux bot detection over systems with no privileged information and 5.5% for malware classification.

IWSPA'17, March 24 2017, Scottsdale, AZ, USA

ISBN 978-1-4503-4909-3/17/03…$15.00

DOI: http://dx.doi.org/10.1145/3041008.3041018

2. PRELIMINARIES

Our focus is on systems that incorporate learning models in detection algorithms. Below, we provide a definition of privileged information in a security setting, introduce the techniques to build privileged information augmented detection algorithms and demonstrate the utility of privileged information with examples.

2.1 Privileged Information

The formal definition of privileged information is proposed in learning theory as a form of student learning with an intelligent teacher. Vapnik and Izmailov motivate the insight by *better than a thousand days of diligent study is one day with a great teacher* [4]. The intuition suggests an intelligent teacher providing a student with examples along with additional explanations such as comments, comparisons, and so on. This idea is formalized under the Learning Using Privileged Information (LUPI) paradigm [5]. The crucial point in this paradigm is that *the privileged information is available when the teacher interacts with a student and is not available at test time when student operates without the supervision of teacher* [4].

Our intuition is that there are many examples of security sensitive applications that can benefit from privileged information. Privileged information is not a replacement for a secondary, in-depth analysis; but it does provide useful information for building a unified detection model along with features available at run-time with an increase in both in false positives and negatives rates. In the following, we give a formal definition and show how we improve generalization of detection models, thus increase the accuracy of detection.

Definition 1 (*Privileged information in a security setting*). The privileged information represents the features relevant to a particular task that are bounded by constraints on using them after system deployment. The main constraints include but are not limited to high resource consumption, computational overhead, human labor, tamper-resistible systems, and error-prone processes. These constraints affect the availability of corresponding input features; thus, it cannot be assumed that such features will be reliably available when making predictions for the purposes of detection at run-time.

In a traditional detection setting, a training set consists of a pair of features and a class in (m+1)-dimensional feature vector as $\mathbf{X} = (\mathbf{x}, y) \in \mathbb{R}^m \times \mathbf{Y}$ where y is a target class such as malicious or benign. Training set is used to learn a detection model $f(\mathbf{x})$. As opposed to traditional detection systems, we consider a detection setting where the feature space, $\mathbf{X}$, is split into two categories at training time to characterize the information utilization of a system.

Definition 2 (*Standard and privileged set*). A Standard set $\mathbf{X^s} = \{\mathbf{x}_i, i = 1, \ldots, T, \mathbf{x}_i \in \mathbb{R}^M\}$ is composed of features that are available both before and after deployment of a detection system, while features of privileged set $\mathbf{X}^* = \{\mathbf{x}_i, i = 1, \ldots, T, \mathbf{x}_i \in \mathbb{R}^N\}$ have constraints that makes them unavailable at run-time.

We use both standard and privileged set for a model generation. Figure 1 illustrates the detection at training and run-time time with and without privileged information. The primary purpose of this feature space is to leverage the amount of data that is relevant to an attack and, therefore, improve the generalization of models. However, building a system with privileged information is challenging since it cannot be combined with the standard set at run-time. Therefore, traditional detection eliminates the privileged set to infer knowledge in the models. We use two approaches to integrate privileged information into detection algorithms. The internal model generation process we use is composed of two different techniques [3]:

- *Knowledge transfer*- This general algorithm considers a mapping function where each privileged feature is defined as a target and the standard set as the input vector. The goal is "estimating" the privileged set with plausible values based on a nearly precise understanding of the relationship between each privileged feature and a subset of standard features.

- *Model influence*- This technique takes a radically different formulation in which we influence the model generation with privileged information. We implement Support Vector Machines Plus (SVM+) algorithm to influence standard feature set errors with privileged information. In turn, resulting detection model includes the knowledge extracted from privileged set, yet does not require them for prediction at run-time.

Each technique is based on different assumptions about the underlying training set and detection model generation. However, *central to all of the techniques is extracting vital information from the privileged features and leveraging them to learn models without needing them at run-time*. To establish a baseline for comparison, we define two detection strategies based on the available features trained to learn a model. This corresponds to a system that has either access to both standard and privileged sets $(\mathbf{X^s}, \mathbf{X}^*)$ or only to the standard set $(\mathbf{X^s})$ at run-time. To establish a baseline for comparison, we define two detection strategies based on the features trained to learn a model. These strategies have either access to both standard and privileged sets $(\mathbf{X^s}, \mathbf{X}^*)$ or only to the standard set $(\mathbf{X^s})$ at run-time. We define the strategies as follows:

1. *Learning a model from complete set:* This ideal system learns a detection model from both standard and privileged sets. This corresponds to a model inferred from the relevant features to perform detection. However, this model is impractical to run at deployment due to the constraints introduced in Definition 1 are present on a subset of features at run-time.

2. *Learning a model from standard set:* This system learns a detection model solely using the standard set. This corresponds to existing detection systems which make predictions using the features that are reliably available at run-time.

In strategy 1, expected loss of a system with samples drawn from an unknown probability distribution $p(x,y)$ is minimized as follows:

$$R(\theta) = \int_X \int_Y L((\mathbf{x}, \theta), y) p(\mathbf{x}, y)\, d\mathbf{x}\, dy \tag{1}$$

where $L(f(\mathbf{x}, \theta), y)$ is the loss function and θ is the detection model hyperparameters. Equation 1 leverages the all information inferred from complete training set. In the presence of constraints defined on a subset of features, we aim to obtain expected loss of techniques similar to Equation 1 which aims at better detection than using solely standard set as introduced in strategy 2.

2.2 Examples

We demonstrate the utility of privileged information with several examples from recent literature. We remark that previous works on combating detection systems may include privileged information in their existing algorithms to augment their accuracy of detection.

Data collection is not practical. Many systems audit data generated from various sources such as operating systems, application software or network devices for future analysis. The correlation of data from these sources may result in finding better patterns. However, algorithms may be overwhelmed by the bulk volume and the computational costs of processing the raw data [6]. These make

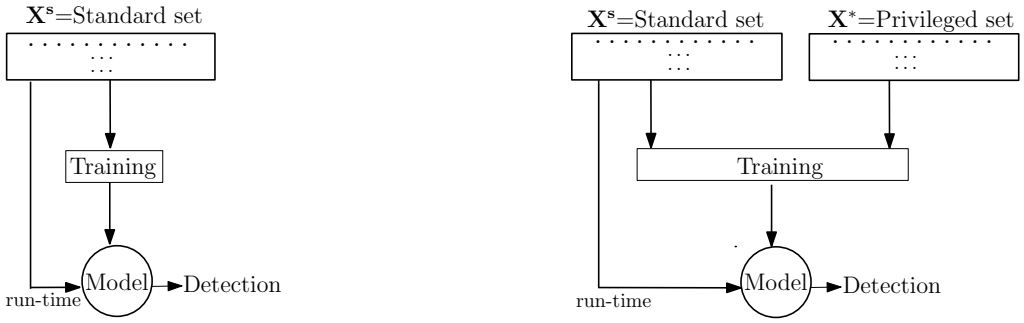

Figure 1: **Comparison of a traditional detection and proposed solution in [3]:** (Left) Existing detection systems generate model trained on standard set and evaluate unseen samples using the standard set. (Right) The proposed solution trains both standard set and privileged set, yet model only requires the standard set for performing detection on unseen samples.

the algorithms impractical for time critical tasks because of the several minutes/hours of data collection and processing. In such cases, features involving complex and expensive data collection can be defined as privileged to provide real-time detection.

Data collection incurs resource consumption. The growth of mobile malware requires the presence of robust malware detectors on mobile devices. One might consider collecting data for numerous type of attacks; however, comprehensive data collection may drain the battery quickly and make users disable the detection mechanism [7]. Instead, some features can be defined as privileged to prevent users limiting the trade-off between usability and security.

Data processing involves human intervention. The acquisition of data at detection time may require a skilled domain expert or a physical experiment. For instance, manual analysis of semantic information measured from attack alerts. The cost associated with the data acquisition process thus renders a fully labeled training set feasible, whereas acquisition of some data at detection time is relatively impossible [8]. Thus, these features can be identified as privileged to outperform the accuracy of a traditional system.

Data available after the fact. Data acquisition from geographically diverse modules can be a strict requirement depending on the aggressiveness of auditing and the granularity of network monitoring. However, imperfections such as long network congestion, data disruption or restricted access due to privacy issues in data acquisition may prevent continuous monitoring [2]. In these cases, some features can be identified as privileged to eliminate the strict requirement of obtaining them in a timely fashion.

It turns out that we can define a set of features as privileged to overcome the constraints that prevent using them for detection at run-time. To do so, we present the techniques that efficiently exploits the confidence information revealed by the privileged set.

3. METHODOLOGY

We seek here to evaluate the capacity of privileged features to improve detection, thus building towards a generally applicable technique to refine the input space of a dataset. We use two techniques to model training with privileged information: knowledge transfer, and model influence. We empirically evaluate the accuracy of techniques on two detection systems. Our evaluation shows that correct cultivation of privileged information improves the detection gain of privileged-augmented detection.

3.1 Knowledge Transfer

We start with a general approach based on using multiple mechanisms of knowledge transfer (KT) [3, 4]. Knowledge transfer formulates a mapping function from standard inputs by defining privileged inputs as target variables. For instance, consider an input x_i^* is identified as privileged, and the remaining features are standard inputs. We learn a mapping function $f(\cdot)$ for x_i^* where the output of the function is the predicted value of the privileged input, and the input of the function is the subset of the standard inputs. These functions are learned and stored to estimate the value of a privileged input. This process can be generalized as part of a broader definition of a set of privileged inputs.

The key point in knowledge transfer is finding a precise mapping function. To do so, as a first knowledge transfer option, we implement variants of multiple regression. The mapping function of regression is defined as $x_i^* = f(\mathbf{X}_j^s, \beta) + \varepsilon$ where $f(\cdot)$ is a regression function, β is a vector of k coefficients and ε is an error term. For each privileged input, we select a set of standard inputs $\mathbf{X}_j^s \subset \mathbf{X}^s$ to learn the underlying relationship between x_i^* and $\mathbf{X}_j^s$. In this setting, we model the relationship of a privileged input and standard inputs with two regression models [9]. First, we use maximum likelihood estimation (Mle) to learn the coefficients by fitting the polynomial to the standard inputs. We use the sum of the squares error to minimize the error between the predicted values of privileged inputs with the corresponding actual values. The errors are considered as Gaussian distributed. Second, to control the over-fitting phenomenon in the previous method, we use maximum-a-posteriori estimation (Map). In this, we add a penalty term to the error function to avoid the coefficients to be so large. We model the priors as Gaussian distribution for insufficient or poorly distributed training sets.

As a second mapping function, we implement a weighted similarity-based (wSim) method. This option is used to estimate the privileged inputs from the most similar samples found in training set. We find the closest subset of the samples that are selected by using a similarity measure between privileged sample and training samples. We first arrange k nearest samples in increasing order of their distances. Then, the privileged inputs are replaced by assigning weights that are inversely proportional to the similarities of their neighbors. This allows us to infer privileged inputs with only the most similar observed inputs by weighting their similarity rather than by using the closest k samples. The privileged input x_i^* of a sample is inferred from k nearest neighbors samples in training set as follows:

$$x_i^* = \sum_{j \in K_{ij}} s(\mathbf{x_i^*}, \mathbf{x_j}) x_j \; / \; \sum_{j \in K_{ij}} s(\mathbf{x_i^*}, \mathbf{x_j}) \qquad (2)$$

where s is the similarity measure, calculated between the observed values of the samples $\mathbf{x_i^*}$, and $\mathbf{x_j}$. After examining a number of similarity measures, we found that Euclidean distance gives the most accurate predictions in our experiments (See Section 4).

In practice, regression-based and similarity-based options are expected to learn the patterns of training set which should mostly suffice for determining the nearly precise mapping functions to estimate the privileged inputs. However, we might quickly lose track of such patterns for any number of reasons; therefore unstable detection outputs may occur. We next bring up a discussion to show the reasons and provide solutions.

Discussion- Our implementation of nonlinear multiple regression variants may cause problems if multicollinearity among the inputs is not identified. Multicollinearity refers to high correlation of independent input with a combination of at least one independent input [10]. Thus, multiple regression options are well suited only when there is no perfect correlation between standard and privileged inputs. Otherwise, regression options may result in abundant near similar inputs, and may artificially alter the output of different models. Consider a probabilistic classifier that explicitly assumes independence between the inputs for detection. Adding a similar input may assign more weight to that particular sample's target class. This may subvert output of the system unexpectedly. We note that this assumption is not limiting because inputs crafted appropriately with input selection or projection techniques may eliminate them.

To address multicollinearity problem, we construct *input selection maps* which are used to reduce the size of standard inputs to derive the privileged ones. Input selection maps include variance inflation factors (`VIF`) which point the dependencies among multiple standard inputs, as opposed to only correlations among pairs of standard inputs [10]. The identification of the dependencies is crucial, as we map privileged inputs from the multiple standard inputs. We obtain `VIF` to indicate the variance proportion (`VP`) along with the condition indexes (`CI`) which indicate set of standard inputs that are associated highly collinear relations with a privileged input [11]. By examining the input selection maps, we transfer the knowledge from standard inputs when `CI` along with `VP` is quite satisfactory, i.e., not fairly large ($\texttt{VP} \geq 0.5$, $\texttt{CI} \geq 30$) [10]. This notion of input selection defines a metric for the robustness of inferring privileged inputs from standard inputs which in turn improves the detection accuracy (We give examples in Section 4).

On the other hand, the weighted similarity-based option has a drawback of not building a model until the time that privileged inputs are present. In regression variants, we generate each mapping function at training time. Then, these functions are stored to estimate each privileged input at run-time. However, similarity option repeats similar searches over dataset at detection time which might be computationally expensive for large training sets. In these cases, we apply stratified sampling to reduce the size of the training set.

These constraints add an extra challenge to estimate privileged inputs. Thus, the options should be carefully applied to counter the risk of oversimplifying the multilevel nature of dataset under study. We next present model influence to relax these constraints.

3.2 Model Influence

Recall that knowledge transfer options estimate values for privileged inputs. However, the constraints previously introduced needs to be tackled. Reflecting on this point, we now ask the essential question: *Can we perform detection in a single step without estimation of privileged inputs?* To address this question, we use the recently introduced paradigm called LUPI [3, 5, 4].

The LUPI paradigm illustrates a concept of human experience inspired by learning with a teacher. Paradigm defines an access to the samples, and an additional explanation from a teacher at training time. This concept is leveraged for improving detection in the presence of privileged inputs [3]. We introduce standard inputs $\mathbf{X}^s$ as the samples that we access, and privileged inputs $\mathbf{X}^*$ as the explanation from the teacher. We implement the algorithmic realization of paradigm called SVM+ which extends the formula of state-of-the-art SVM algorithm. To understand how we influence the detection model with the privileged inputs, first we show the formulation of model influence (Proof and implementation details is given in technical report [12]):

$$\mathcal{L}(w,w^*,b,b^*,\alpha,\delta) = \underbrace{\frac{1}{2}||w||^2 + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i y_i f_i}_{\text{objective function}} + \underbrace{\frac{\gamma}{2}||w^*||^2 + \sum_{i=1}^{n}(\alpha_i + \delta_i - C)f_i^*}_{\text{influence from privileged inputs}} \quad (3)$$

Equation 3 influences the detection boundary of standard inputs $f_i = w^\intercal x_i^s + b$ at x_i^s with the correction function defined by the privileged inputs $f_i^* = w^{*\intercal} x_i^* + b^*$ at the same location. The correction functions of privileged inputs are considered as the slack variables ε of SVM algorithm whereas determined by the inputs that are only available before deployment. This enables a detection model to utilize privileged inputs by providing a measure of confidence for each labeled sample in correction space. We also use radial basis function (RBF) to perform a non-linear classification by implicitly mapping the inputs into higher dimensional input space.

Model influence differs from knowledge transfer in two ways. First, detection model is learned using both standard and privileged inputs before deployment. However, it applies only standard inputs and does not infer values. Second, while model influence defines its optimization problem that restricts the application of various models, knowledge transfer options are flexible to the application of various models. Overall, given a set of privileged inputs before deployment, model influence allows learning models for detection without the need of privileged inputs.

Discussion- Model influence follows a different route and eliminates assumption of generating a relevant mapping function defined between standard and privileged inputs. Thus, there are no strict relations required to exploit between standard and privileged inputs. This eliminates a cumbersome process of finding a proper mapping function as in knowledge transfer. That is, if similarity measures of standard and privileged inputs are not appropriate in correction space, these privileged inputs are rejected. In this way, detection model is influenced when privileged inputs find out to be contributing to the detection. This makes model influence more robust to the samples with the inputs that distinctly lies abnormal distance from the range of the standard inputs. However, knowledge transfer options may conform privileged inputs within the range of the standard inputs. For instance, inferring a privileged outlier input that is not observed in the training set as in similarity-based option of knowledge transfer.

Another advantage of model influence is converging from a significantly smaller number of samples. The model requires $O(\sqrt{n})$ samples compared to $O(n)$ samples for traditional learning methods. However, the convex optimization problem of Equation 3 works with the dual representation and is limited to small to a medium number of samples. Recent attempts have proposed to accelerate the computation by using specialized kernels [4].

4. EVALUATION

We evaluate techniques on two real-world security datasets: fast-flux bot detection (Section 4.1) and malware classification (Section 4.2). We first describe the experimental setup and evaluation steps, and then evaluate the systems.

Experimental Setup- We first build the input space for each of the given raw datasets. Then, we use three state-of-the-art classifiers to learn the detection models: Random Forest Classifier (RF), Support Vector Machines (SVM) and k-Nearest Neighbors (kNN) algorithm. Then, we identify the set of inputs that have constraints at runtime. These privileged inputs are integrated into detection systems systematically with approaches. The evaluation process includes the application of techniques and compares them with two strategies:

- A system learns a model from *complete features* that comprised of standard and privileged features.
- A system learns a model solely from *standard features* in which training set is built by eliminating the privileged inputs from the complete features.

Evaluation Metrics- To formalize our discussion of privileged inputs, we evaluate the effectiveness of strategies both in terms of detection error and performance loss. Detection error shows the average rate of a model that incorrectly labels a sample after n runs over a test set. Performance loss is used to establish a baseline for the relative detection loss of models on using complete features. It summarizes the empirical results to determine the technique that drives the accuracy closer to the result of complete features. It is defined as follows:

$$PL = 100\frac{(1-R_c(\theta))-(1-R_i(\theta))}{(1-R_c(\theta))} \quad (4)$$

where R_c is the detection error obtained from complete features and R_i represents the detection error of the applied technique.

To report comparable results, we ran experiments by using stratified random sampling to split the dataset into from 10% to 60% training set and the remaining as a test set. All experiments are run over 3-10 independent realizations, and results are reported with the mean and standard deviation. Finally, we tune hyperparameters of the models via grid search based on 5-fold cross-validation over the training set, and then they are used to learn the models.

4.1 Case Study: Fast-Flux Bot Detection

Our first case is a fast-flux service bot detection (FFdetector) [13]. Fast-flux servers are adopted by attackers to hide the actual IP addresses of the servers for malicious activities. FFDetector is designed to detect the fast-flux servers using multiple sources of information. The combination of inputs from various sources is intended to improve the discrimination of content delivery networks (CDN) from fast-flux servers, as they share many technical similarities. The raw dataset consists of 4 GB DNS requests of benign and active fast-flux servers collected in early 2013. Each DNS packet is dissected to construct the19-dimensional numerical inputs. First two columns of Table 1 present the categorization of timing, domain name, spatial, network, and DNS answer inputs. The extraction of these inputs entails various dependencies. For instance, domain name based inputs require a reliable whitelist of benign domain names to measure the similarity of a domain. These dependencies entail computational delays in mission critical systems. For example, finding the KL-Divergence of a given domain from a whitelist of domain names or IP coordinate database and WHOIS processing may take several minutes/hours to collect. Thus, we define these as privileged features to assure real-time detection.

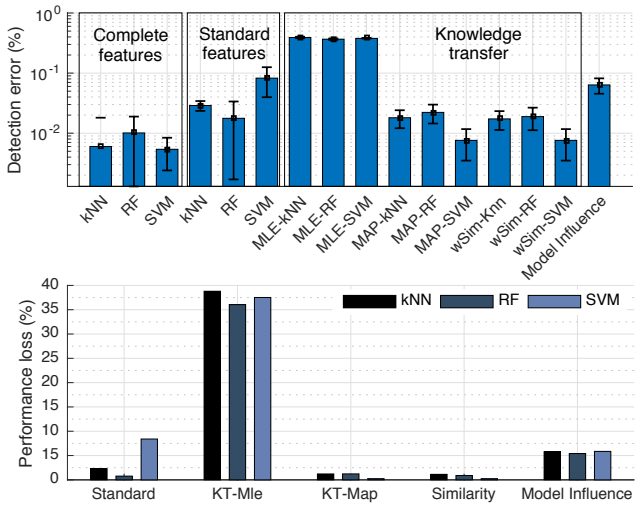

Figure 2: Fast-flux bot detection results. Log of detection error (upper) compares the approaches. Performance loss (bottom) presents the relative loss over complete dataset.

Results- We start by presenting the detection error of complete and standard sets. We then compare these results with the application of knowledge transfer and model influence. In this set of experiments, we consider spatial inputs as privileged inputs. Figure 2 presents the log-scale error bar plot of detection error where x axis lists the names of the applied technique. These results are summarized regarding their performance loss as shown in the lower part of Figure 2.

The upper leftmost of the figure shows the application of models to the complete dataset. We observe that RF, kNN, and SVM gives nearly perfect detection accuracy. Their average detection error is less than to 1%, and SVM yields the maximum average detection error of 0.54%. One way of sustaining detection with the absence of spatial inputs is retraining the models on standard features. First, we notice that the input space of FFdetector is well-defined, as the elimination of privileged inputs from training set gives higher detection error than that of complete dataset, as these inputs introduce loss of information. The average detection error of kNN, RF and SVM increases from 0.6%, 1.01%, 0.54% to 2.89%, 1.77%, 8.25% .

The most apparent change in the error is evoked by SVM, whereas changes found in RF and kNN are close to those observed in complete dataset. To understand the impact of privileged inputs, we verify their utilization in internal structures of the models. Through investigation of RF trees, we observe that the spatial inputs are substituted by standard inputs in classification trees because the discrimination power (i.e., in terms of information gain) of that input is similar to the privileged one. Thus, RF is able to replace each spatial input with a corresponding standard input which gives similar results for those of complete dataset. kNN detection loss is also recovered by a large combination of inputs. That is the DNS answer, and network input categories are often sufficient to determine the nearest neighbors of the actual class in the absence of spatial inputs. Finally, we find that the main source of the increase in detection error of SVM is false positives resulted from incorrect classification of fast-flux servers to CDNs, as spatial inputs act as primary inputs to distinguish fast-flux servers from CDNs.

Now, our goal is eliminating the retraining and detection loss introduced by standard dataset, which is the central purpose of the techniques. This allows us to apply the derived detection models inferred from the complete dataset before deployment. We start

Input Category	Description	Complexity	Dependency
DNS answer	Number of unique A records Number of NS records	O(N)	Packet analysis
Domain name	Edit distance Kullback-Leibler divergence (unigrams and bigrams) Jaccard index (unigrams and bigrams)	O(ND) O(ND^2W) O(N^2D^2)	Whitelist of benign domain names
Spatial	Time zone entropy of A records Time zone entropy of NS records Minimal service distances (μ and σ)	O(NM)	IP coordinate database lookup (external source)
Network	Number of distinct autonomous systems Number of distinct networks	O(N)	WHOIS processing (external source)
Timing	Network delay (μ and σ) Processing delay (μ and σ) Document fetch delay (μ and σ)	O(N)	HTTP requests

N = Number of test domain names W = Number of domain names in whitelist
D = Max domain name size M = IP coordinate size

Table 1: Fast-flux bot detection system feature descriptions (μ is for mean, and σ is for standard deviation).

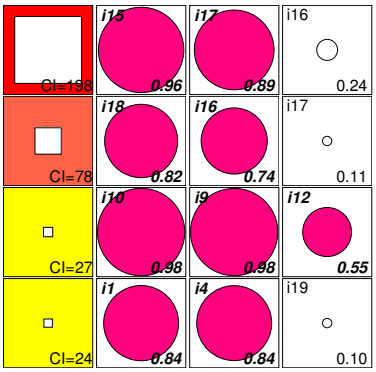

Figure 3: Visualizing the input selection map for time zone entropy of A records in a fast-flux detection. Eligible standard inputs for knowledge transfer are those which have condition index (CI) ≤ 30 and variance proportion (VP) ≤ 0.5.

with the application of knowledge transfer. Recall that to employ knowledge transfer options; we first need to derive a subset of standard inputs to generate mapping function for each privileged inputs. To do so, we construct input selection maps to find the most relevant standard inputs to estimate the privileged ones.

Figure 3 shows an example of input selection map for privileged input time zone entropy of A records. The leftmost column shows the four largest condition index (CI) values with corresponding variance proportions (VP) demonstrated by the circles. The background colors, the size of the rectangles and circles are used to label severity of the CI and VP values [11]. We eliminate the standard inputs from i_{15} to i_{18}, the inputs in the first two rows wherein CI ≥ 30 and VP ≥ 0.5. These inputs tell us that they are highly correlated with the privileged input of time zone entropy of A records. This process can be generalized for a set of privileged inputs.

We now present the results of the regression based options of Mle and Map in estimating the privileged inputs. We observe that all options, except Mle, benefit statistical dependencies between privileged and standard inputs, and reduces the detection error of kNN, and SVM from 2.89% and 8.25% to 1.81% and 0.76%. However, standard dataset trained with RF classifier yields good performance in some cases. For instance, we observe that Wsim and Map give 1.89% and 2.2% after recovery, whereas RF trained on standard dataset achieves 1.77%. While it is dataset dependent, the decline in detection error of Map and Wsim is promising. However the results of Mle is surprising, it gives an average of 39.2%, 36.7% 37.9% detection error for kNN, RF and SVM classifiers.

We observe that Mle regression suffers primarily from well-known confounding effects of substantial biases. This causes overfitting of the mapping functions for estimating privileged inputs. Thus, the test set shows a sharp increase in detection error. This is confirmed by checking the predicted values of each privileged input and responses of the classifiers. The inputs of FFDetector dataset is mostly composed of real-valued inputs, and a small change in privileged inputs influences the detection output. As an example, the entropy of the leaf of a RF tree may totally change if spatial input minimal service distance prediction is biased to one class. However, Map provides unbiased weight estimates for the privileged inputs by assigning residual terms. This reduces the variability of the estimated weights for all models.

It is important to note that the results of model influence are quite promising. It utilizes the spatial inputs accurately. Because it uses the objective function derived from SVM, it is fair to compare their results. We observe that the average detection error of model influence shows improvement over SVM, reducing from 8.25% to 6.36%. We discuss the effectiveness of model influence in detail later in the corresponding section.

4.2 Case Study: Malware Classification

As a second case study, we use the publicly available Microsoft malware classification dataset [14]. The goal of the dataset is to classify the nine malware families into their respective families, which is used to detect new samples of the malware. The complete dataset is split into a labeled training and an unlabeled test set. There are approximately 11000 observations in both files. The files are roughly 50MB in size, results in 200GB of training and test data. Each file consists of two set of files for each polymorphic versions of the nine malware classes: byte files and metadata manifest. The byte files include the raw hexadecimal representation of the malware binary contents. Metadata manifest contains the interactive disassembler tool (IDA) [15] logs of binary assembly source code (asm). The metadata manifest of each malware presents additional information such as memory allocation and function calls.

We use the labeled training set to construct the input space. The input extraction of malware dataset is more complicated than FFDetector, as two large separate raw data sets are given. We split the

```
00409C00 18 53 52 00 D1 E8 F7 D0 83 E0 01 C3 A1 18 53 52
00409C10 00 C1 E8 11 F7 D0 83 E0 01 C3 A1 18 53 52 00 C1
00409C20 E8 02 F7 D0 83 E0 01 C3 A1 18 53 52 00 C1 E8 03
00409C30 F7 D0 83 E0 01 C3 A1 18 53 52 00 C1 E8 04 F7 D0
00409C40 83 E0 01 C3 A1 18 53 52 00 83 E0 60 33 C9 3C 60
00409C50 0F 95 C1 8B C1 C3 A1 18 53 52 00 C1 E8 07 F7 D0
00409C60 83 E0 01 C3 A1 18 53 52 00 C1 E8 08 F7 D0 83 E0
00409C70 01 C3 A1 18 53 52 00 C1 E8 09 F7 D0 83 E0 01 C3
```

```
var_C = dword ptr -0Ch
Dst = dword ptr 8
MaxCount = dword ptr 0Ch
push ebp
mov eax, large fs:0
jmp short loc_401883
sub esp, 0Ch
cmp esi, 0FFFFFFFEh
mov eax, dword_43D260
xor eax, ebp
mov [ebp+var_10], esp
```

```
FF BD BE 55 B8
FC CC 9F : 16
D6 78 AA B5 F1
Push call mov Eax
xor short Retn add proc
```

Figure 4: An example of a malware family raw files of hexadecimal representations, and disassembler output. We use frequency of byte bigrams and tokens for standard and privileged set, respectively (Right).

dataset into two groups: byte files and asm files. The byte files are used to construct an input space by counting the frequencies of each hexadecimal duo (i.e., byte bigrams), and the metadata manifest file is used as a textual depiction of the underlying byte files. We also extract the frequency count of distinct tokens in asm files. As an example, functions such as `mov()`, `cmp()` in the text section is included as a bag of word counts. These tokens capture the execution differences of malware binaries. Figure 4 presents a representative example of raw data and extracted inputs: raw hexadecimal representation of the malware's binary content, a log containing metadata manifest extracted from the binary by the disassembler, and standard inputs from byte files and privileged inputs from asm files.

We treat the dataset as a binary classification problem (e.g., malware classes 1 and 8) to save on computation time. We also apply input selection to reduce the 272 byte-level input space by using a correlation-based filtering mechanism. The correlation-based filter allows us to obtain the subset of inputs by measuring the relevance of each input as having a minimal correlation between classes but being extremely correlated to a particular class [16].

This particular dataset reveals an important consideration: inputs from asm files are not necessarily be available or accurate at detection time. Consider a case where different versions or types of disassembler are used to obtain the asm files. This may result in a great deal of human intervention—error-prone and software dependent process that provides no guarantees regarding the availability of the input space. Further, these constraints may introduce latency in obtaining them. We will walk through the application of experiments by considering byte inputs as standard inputs and inputs from asm files as privileged inputs.

Results- Figure 5 presents the experimental results. We find that complete set yields detection error with an average of 5.1%. 3.36% and 4.32%, whereas standard dataset yields 10.37%, 8.72%, and 8.22% for kNN, RF and SVM classifiers, respectively. Note that under traditional detection setting, the standard way of dealing with this problem is learning a detection model using inputs from only byte files (i.e., standard features). However, our goal is utilizing the privileged inputs to improve detection accuracy.

We first evaluate knowledge transfer options. We see that knowledge transfer options show unexpected variations in detection errors. Interestingly, most of the approaches fail to estimate privileged inputs accurately. For instance, kNN classifier produced the maximum detection error, yielding 48.1% and %31.4 average detection error for Mle and Map, which are only slightly better than random guessing between the two classes. We observe only detection gain in RF classifier with the application of Map. The average detection error of both reduces from 8.72% to 8.24% compared to results of RF trained on standard dataset.

To understand the performance differences, we inspect input selections maps. It turns out that standard dataset does not contain enough useful information about the correlation between certain

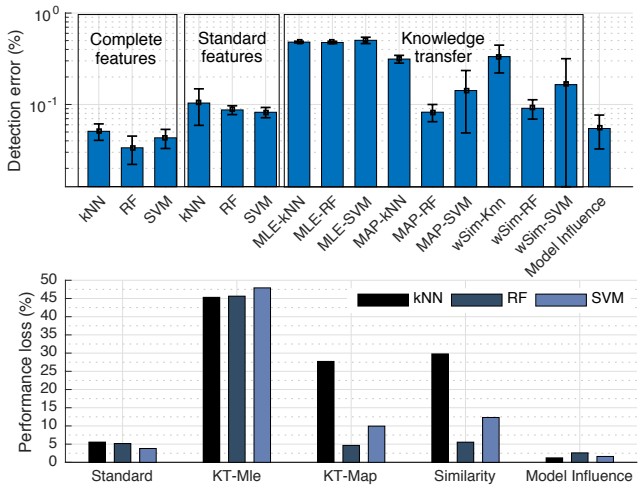

Figure 5: Malware classification results. Detection error (upper) compares the approaches. Performance loss (bottom) presents the relative loss over complete dataset.

standard inputs and the privileged inputs. Recall that we extract the asm files as a textual description of malware binaries. This shows that inputs derived from two different spaces may not have a precise mapping between them through knowledge transfer options. As a result, the disparity between these spaces fails to estimate privileged inputs and yields unpredictable detection errors. We note that this might be improved by application of more comprehensive mapping options derived from knowledge transfer.

We turn now to the striking results of model influence. As introduced in Section 3.2, model influence does not estimate privileged inputs by generating a mapping function. Instead, it encodes the discrimination power of privileged inputs into detection model. More specifically, we influence the detection model by introducing asm inputs in the correcting space of the optimization problem while rejecting the inappropriate similarity measures between two input spaces in the correction space. This results in a detection model which is generated in the byte input space with the cooperation of the asm files produced by the dissembler. However, the system runs without the requirement of asm files after deployment. In this manner, we relax the cumbersome process of finding accurate mappings of KT options. As illustrated in Figure 5, we observe this positive effect. Model influence gives superior results to other techniques, an average of 5.46% detection error, which is 2.1% less than the best result of the complete dataset obtained by RF. This yields 1.23%, 2.6%, and 1.61% performance loss. That being said, the privileged information reduces the detection error when selection of the techniques is determined by the input space of the dataset.

5. RELATED WORK

We validate the performance of knowledge transfer and model influence over detection systems of malware classification and fast-flux bot detection. There has been a wealth of research focused on these domains. For example, researchers have used specific patterns to group malware samples into families [17, 18, 19], and have used DNS information to understand and predict botnet domains [20, 21, 22, 23, 24]. The previous detection systems primarily focus on the features only available at run-time and tied to the specifics of these features to augment the detection performance. Unlike previous work, we demonstrate a systematic cultivation of privileged information into detection algorithms and increase their accuracy.

The use of privileged information has recently attracted attention in a few research areas such as computer vision, and image processing. Wang et al. and Sharmanska et al. attempt to derive models from images with an auxiliary information provided as a privileged set [25, 8]. Hernandez-Lobato et al. treated learning with privileged information paradigm under the Gaussian process classification (GPC) [26]. Lopez-Paz et. al presented applications of privileged information in semisupervised, unsupervised and multitask learning [27], and Celik et al. used privileged information for addressing patients' privacy concerns in healthcare data analytics [28]. Unlike our techniques, their methodology is not designed to parse the security data but rather to determine if there is a possibility of application to a domain specific information. Thus, each has a significant limitation or assumption that does not apply well to security sensitive applications.

6. CONCLUSION

In this paper, we explored generally applicable techniques about training detection system models using privileged information. Through an explicit connection between a privileged set that is reliably available at training time, but not at run-time and current features of detection systems, we gave a straightforward analysis showing that privileged features can be used to improve generalization of detection models for better detection regardless of their high detection performance. We used two techniques to integrate the useful information extracted from privileged set into the resulting detection models: knowledge transfer, and model influence. First, we used knowledge transfer, a general technique for extracting knowledge from privileged information by estimation from available information. Second, we used model influence, a model dependent technique of influencing the model optimization with additional knowledge obtained from privileged information. By correctly cultivating the features in techniques, we improved the accuracy of detection systems regardless of their high detection performance. We showed up to 8.2% decrease in relative detection error for fast-flux bot detection over benchmark systems with no privileged information, and 5.5% for malware classification.

Acknowledgment

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

7. REFERENCES

[1] A. A. Cardenas, P. K. Manadhata, and S. P. Rajan. Big data analytics for security. *Proc. IEEE Security & Privacy*, 2013.

[2] Richard Zuech, Taghi M Khoshgoftaar, and Randall Wald. Intrusion detection and big heterogeneous data: a survey. *Journal of Big Data*, 2015.

[3] Z. Berkay Celik, Patrick McDaniel, Rauf Izmailov, Nicolas Papernot, and Ananthram Swami. Extending detection with forensic information. *arXiv:1603.09638*, 2016.

[4] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: Similarity control and knowledge transfer. *Journal of ML Research*, 2015.

[5] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 2009.

[6] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proc. Computer Security Applications*. ACM, 2013.

[7] Jeffrey Bickford, H Andrés Lagar-Cavilla, Alexander Varshavsky, Vinod Ganapathy, and Liviu Iftode. Security versus energy tradeoffs in host-based mobile malware detection. In *Proc. Mobile systems, applications, and services*. ACM, 2011.

[8] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Learning to rank using privileged information. In *Proc. International Conference on Computer Vision (ICCV)*, 2013.

[9] Christopher M. Bishop. *Pattern recognition and machine learning*. 2006.

[10] D. A. Belsley, E. Kuh, and R. E. Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons, 2005.

[11] Michael Friendly and Ernest Kwan. Where's Waldo? Visualizing collinearity diagnostics. *The American Statistician*, 2009.

[12] Z. Berkay Celik, Rauf Izmailov, and Patrick McDaniel. Proof and Implementation of Algorithmic Realization of Learning Using Privileged Information (LUPI) Paradigm: SVM+. Technical Report NAS-TR-0187-2015, CSE Department, PSU, December 2015.

[13] Z. Berkay Celik and Sema Oktug. Detection of Fast-Flux Networks using various DNS feature sets. In *Proc. IEEE Symposium on Computers and Communications (ISCC)*, 2013.

[14] Microsoft malware classification challenge. https://www.kaggle.com/c/malware-classification/. [Online; accessed 10-May-2015].

[15] Ida pro: Disassembler and debugger. http://www.hex-rays.com/idapro/.

[16] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proc. International Conference on Machine Learning (ICML)*, 2003.

[17] M Zubair Rafique and Juan Caballero. Firma: Malware clustering and network signature generation with mixed network behaviors. In *Proc. RAID*. 2013.

[18] Nir Nissim, Robert Moskovitch, Lior Rokach, and Yuval Elovici. Novel active learning methods for enhanced pc malware detection in windows os. *Expert Systems with Applications*, 2014.

[19] Mansour Ahmadi, Giorgio Giacinto, Dmitry Ulyanov, Stanislav Semenov, and Mikhail Trofimov. Novel feature extraction, selection and fusion for effective malware family classification. *arXiv preprint arXiv:1511.04317*, 2015.

[20] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From throw-away traffic to bots: detecting the rise of dga-based malware. In *Proc. USENIX Security*, 2012.

[21] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *Proc. NDSS*, 2011.

[22] Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Reddy, and Supranamaya Ranjan. Detecting algorithmically generated malicious domain names. In *Proc. ACM Internet measurement*, 2010.

[23] Z. Berkay Celik, Jayaram Raghuram, George Kesidis, and David J Miller. Salting public traces with attack traffic to test flow classifiers. In *Proc. Usenix Cyber Security Experimentation and Test*, 2011.

[24] Z. Berkay Celik, Robert J Walls, Patrick McDaniel, and Ananthram Swami. Malware traffic detection using tamper resistant features. In *Proc. IEEE Military Communications Conference (MILCOM)*, 2015.

[25] Ziheng Wang and Qiang Ji. Classifier learning with hidden information. In *Proc. IEEE Computer Vision and Pattern Recognition*, 2015.

[26] Daniel Hernández-Lobato, Viktoriia Sharmanska, Kristian Kersting, Christoph H Lampert, and Novi Quadrianto. Mind the nuisance: Gaussian process classification using privileged noise. In *Proc. Advances in Neural Information Processing Systems*, 2014.

[27] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

[28] Z. Berkay Celik, David Lopez-Paz, and Patrick McDaniel. Patient-driven privacy control through generalized distillation. *arXiv:1611.08648*, 2016.