Building Better Detection with Privileged Information

Z. Berkay Celik[1], Patrick McDaniel[1], Rauf Izmailov[2], Nicolas Papernot[1], and Ananthram Swami[3]

[1]SIIS Laboratory, Department of CSE, The Pennsylvania State University
{zbc102,mcdaniel,ngp5056}@cse.psu.edu
[2]Applied Communication Sciences, Basking Ridge, NJ, USA
rizmailov@appcomsci.com
[3]Army Research Laboratory, Adelphi, MD, USA
ananthram.swami.civ@mail.mil

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 test-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 don't train (and thus don't learn from) features that are unavailable or too costly to collect at run-time. In this paper, we leverage recent advances in machine learning to integrate privileged information – features available at training time, but not at run-time– into detection algorithms. We apply three different approaches to model training with privileged information; knowledge transfer, model influence, and distillation, and empirically validate their performance in a range of detection domains. Our evaluation shows that privileged information can increase detector precision and recall: we observe an average of 4.8% decrease in detection error for malware traffic detection over a system with no privileged information, 3.53% for fast-flux domain bot detection, 3.33% for malware classification, 11.2% for facial user authentication. We conclude by exploring the limitations and applications of different privileged information techniques in detection systems.*

1 Introduction

Detection systems are a bedrock tool for system and enterprise defense [21]. Such systems provide predictions about the existence of an attack in a target domain using information collected in real-time. The detection system uses this information to compare the run-time environment against known normal or anomalous state. In this way, the detection system "recognizes" when the environmental state becomes –at least probabilistically– dangerous. What constitutes dangerous is learned; detection algorithms construct models of attacks (or non-attacks) from past observations using a training algorithm. Thereafter, the detection systems use that model for detection (i.e., at run-time). Yet, a limitation of this traditional model of detection is that it focuses on features that will be available at run-time. Many useful features are either too expensive to collect in real-time or only available after the fact. In traditional detection, past such information is ignored for the purposes of detection. We argue that future detection systems need to learn from *all* information relevant to an attack, whether available at run-time or not.

Consider a recent event that occurred at a peer institution. In the Fall of 2014, the Penn State University college of engineering fell victim to sophisticated cyber-attacks that resulted in substantial exfiltration of personal information and intellectual property [19]. Working with the University staff, security analysts from Mandiant [14] clandestinely conducted a six-month investigation. During that time, a vast amount of information was collected from networks and systems across the university, e.g., network flows, system logs files, user behaviors. An analysis of the collected data revealed the presence of two previously undetected advanced persistent threat (APT) actors on the college's network. Yet, the analysis is largely non-actionable by detection systems; because the vast array of derived features would not be available at run-time, they can't be used to train Penn State's detection systems.

This example highlights a challenge for future intrusion detection; *how can detection systems integrate hard-earned intelligence relevant to an attack that is not measurable at run-time?* Here, we turn to recent advances in training under privileged information [24,25]. The goal of this effort is to leverage *privileged information* (features available at training time, but not at run-time) to improve the accuracy of detection.

In this paper, we explore at alternate approach to training intrusion detection systems that exploits privileged information. In this, we make the following contributions:

- We introduce three classes of detection algorithms that integrate privileged information at training time to improve detection precision and recall.
- We provide an empirical evaluation of techniques on a large variety of real-world datasets. These experiments show that privileged information decreases the detection error an average of 4.8% for malware traffic detection, 3.53% for fast-flux domain bot detection, 3.33% for malware classification, 11.2% for face authentication, over benchmarked systems that do not use privileged information.
- We analyze dataset properties and algorithm parameters that maximize detection gain, and present guidelines and cautions for the use of privileged information in realistic deployment environments.

2 Privileged Information in Detection

Our focus is on systems that incorporate learning models in detection mechanisms. Below, we provide our problem, a definition of privileged information, and introduce our techniques. Unlike much-related work, our focus is not detection of a particular attack, we aim to improve detection strength of any detection system where a system collects information relevant to an attack at training time, yet that is not available at run-time.

2.1 Problem Definition

Detection systems adapt traditional algorithms such as support vector machines and multi-layer perceptron neural networks for generating detection models from security data. The models aim at learning patterns to estimate an unknown dependency or structure of a system with a limited number of observations from historical data (or training data). Training data consists a pair of features and a class such as anomalous or normal. For instance, in network level malware detection, the features are extracted from packets or flows of incoming/outgoing traffic. In mobile malicious application identification, features may contain textual descriptions and permissions. These feature sets are used to learn a detection model. Thus, systems are concerned with estimating good predictive models from features to use it for accurately predicting newly arrived 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. This requires building a model in which the same feature space used to derive the model is required to make predictions on unseen samples. Yet, this limits the detection on solely features that can be collected reliably at detection time. We argue that there may be a plethora of information relevant to an attack at training. For instance, security data repositories stores to a vast amount of information from packets, log files and other sources for future use [4, 30]. Such products are available today. Archsight captures raw data from packets, web proxies, DHCP servers, VPN servers, and more at rates of up to 100K events per second, compresses, and stores up to 42 TB of data for future analysis [9]. However, much useful information becomes meaningful after further analysis. Thus, the main problem is turning this data actionable by detection systems; even they have limitations at run-time. We articulate and give examples of such limitations in Section 2.3.

Our goal is to enable detection systems to leverage the ancillary information available at training time, regardless of whether that information is available at detection time after. Therefore, we relax the strong assumption of limiting training to features that can be collected on unseen samples at run-time. The motivation behind the information utilization is to leverage the ancillary information at training time to build better models for run-time that outperforms those built on the standalone samples. In contrast to the traditional model of detection, this reveals the inherent tension between information utilization, detection accuracy, and robustness of a system.

2.2 Approach

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* [24]. The intuition suggests an intelligent teacher providing a student examples along with additional explanations such as comments, comparisons, and so on. This idea is formalized under the Learning with Privileged Information (LUPI) paradigm [25]. However, *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* [24].

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.

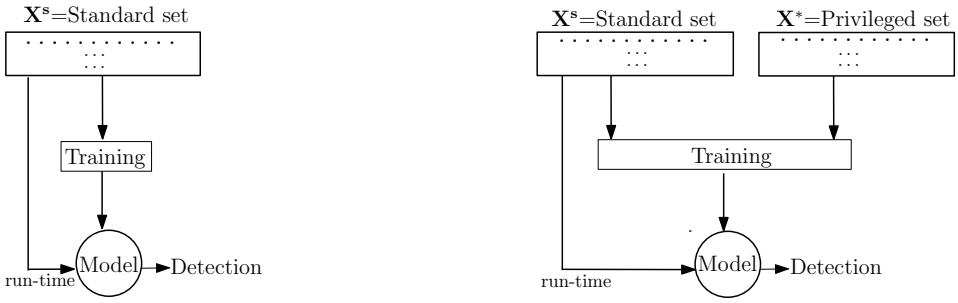

Figure 1: **Comparison of a traditional detection and proposed solution:** (Left) Existing detection systems generate model trained on standard set and evaluate unseen samples using the standard set. (Right) The proposed solution trains both standard set and privileged set, yet model requires only standard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

Definition 1 (*Privileged information in a security setting*). The privileged information represents the meaningful features relevant to a particular task that are bounded by constraints on measuring 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 provide no guarantees regarding the availability and quality of the resultant information; thus, they are not present at run-time.

In a traditional detection setting, a training set consists a pair of features and a class in (m+1)-dimensional feature vectors 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 $f(\mathbf{x})$. Thus, systems are concerned with estimating reliable predictive models from features for accurate prediction of unseen samples. 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. This represents the notion of information availability 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 leverage recent advances in learning theory to integrate privileged information into detection algorithms. The internal model generation process we use is composed of three different techniques:

- *Knowledge transfer*- This general algorithm considers a mapping function where each privileged feature is defined as a target and the standard set as 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 use 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 ones, yet does not require them after deployment.

- *Distillation*- This technique leverages the distillation designed for Deep Neural Networks (DNNs) to learn the weights of the network. We distill the information extracted from privileged information as probability outputs of classes with the standard set sample labels into the resulting model.

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 set 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. 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 detection time.

2. *Learning a model from standard set:* This system learns a model solely using the standard set. This corresponds to traditional detection which considers the models trained on features that are available both before and after deployment of a detector.

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

$$R_{opt}(\theta) = \int_X \int_Y L((\mathbf{x},\theta),y)p(\mathbf{x},y)\,d\mathbf{x}\,dy \qquad (1)$$

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.

We show the performance of complete set and standard set and compare with techniques based on three metrics; accuracy, recall, and precision. *Accuracy* is the sum of the true positive and true negatives over a total number of samples. *Recall* is the number of true positives over the sum of false negatives and true positives, and *precision* is the number of true positives over the sum of false positives and true positives. Higher values of accuracy, precision, and recall indicates higher quality of the detection output. The goal of our techniques is to improve metrics with the use of privileged information and obtain better results than the standard set, and similar to the complete set.

2.3 Examples

In the following, we provide several concrete examples from recent literature to illustrate the possibility of using privileged information in traditional detection systems. We note that most previous work on combating detection models may include various privilege information in their existing models to augment their detection accuracy.

Example 1 - 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 [28]. These make the algorithms impractical for time critical tasks because of the several minutes/hours of collection and processing. In such cases, features that involve complex and expensive data collection can be defined as privileged information to produce models with low cost and high accuracy.

Example 2 - 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 [22]. Thus, these features can be identified as privileged to outperform the accuracy of a traditional detection system.

Example 3 - Data collection incurs resource consumption. The growth of mobile malware requires the presence of robust malware detectors on mobile devices. However, collecting data for numerous type of attacks may cause high power consumption [3]. One might consider using complete data for comprehensive detection; however, this process may drain the battery quickly and make users disable the detection mechanism. Instead, some features can be defined as privileged to prevent users limiting the trade-off between usability and security.

Example 4 - 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 a long network congestion, data disruption or restricted access due to privacy issues in data acquisition may prevent continuous monitoring [30]. 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 use privileged information to overcome the constraints defined on a set of features by learning a detection model that exploits the confidence information revealed by the privileged set. To do so, we introduce the techniques, which we explore using datasets presented next for evaluation.

3 Detection Under Privileged Information

In this section, we introduce three approaches to model training with privileged information; knowledge transfer, model influence, and distillation. Several privileged information-augmented detection datasets are developed

Application	Standard set	Detection time constraint	Privileged set	Goal
Fast-flux bot detection	DNS packet information	Time consuming operations: IP coordinate database lookup, whitelist processing	Domain name, IP entropy features	Detect fast IP-changing domain names
Malware classification	Binary files	Software dependency, error-prone collection	Assembly source code from a disassembler	Classify malware binaries to respective families
Malware traffic detection	Network traffic flow information	Dynamic port numbers payload encryption, masquerading	Port numbers, packet flags information, payload analysis	Detect malware packet flows embedded amongst benign traffic
User face authentication	Colored, blurred or gray facial images	Human or computational efforts	Facial actions, bounding boxes, textual descriptions	Authenticate users from face images

Table 1: **Real-world datasets and availability of features at training and detection time:** The constraints associated with a set of features allow us to capture the privileged information and integrate it for model generation. These datasets are evaluated with techniques introduced in Section 2.2.

and described. The accuracy, precision and recall of the detection systems are empirically validated in a range of detection domains. Table 1 presents the concrete representative samples of each domain, their goals and introduce the privileged information they include.

3.1 Real-world Datasets

Fast-flux bot detection. Our first dataset comes from a fast-flux bot detector [5]. Fast-flux servers are adopted by attackers to hide the actual IP addresses of the malicious servers that are used for malicious activities. The authors jointly build a features space that is extracted from domain names, DNS packets, packet timing intervals, WHOIS and IP coordinate database processing [18, 27]. The dataset consists of many rather than a few features to increase the resilience that targets discrimination of Content Delivery Networks (CDNs) from fast-flux domain names, as they share many technical similarities. However, the extraction of some features entails computational delays. For example, finding the KL-Divergence of a given domain from a whitelist of domain names or IP coordinate database/WHOIS processing may take several minutes/hours to collect. Thus, we define these as privileged features to provide real-time detection.

Malware classification. As a second dataset, we use the dataset from Microsoft malware classification challenge [16]. The dataset classifies malware families into their respective families, which is used to detect new samples of the malware. The dataset 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 IDA disassembler tool [11] logs of binary assembly source code (asm). The metadata manifest of each malware presents additional information such as memory allocation and function calls. We rely on simple but meaningful feature extraction. We split the dataset into two groups: byte files and asm files. The byte files are used to construct a feature space by counting the frequencies of each hexadecimal duos (i.e., byte bigrams). However, we consider that 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 accurate feature extraction. Thus, we include the frequency count of distinct tokens in asm files as privileged information. For instance, we include functions such as `mov()`, `cmp()` in the text section as a bag of word counts. These tokens are privileged examples of holistic descriptions of malware binaries.

Malware traffic detection. The goal of the dataset is network traffic flow anomaly detection, wherein the malware traces are salted with the benign traffic of LBNL[1] and University of Twente[2] repository. The dataset consists 21 different recent malware families which were active from 2007 to 2014. The raw dataset used in experiments are publicly available for research purposes, and can be downloaded[3456], and recently analyzed in previous works of [2, 7, 20, 23]. The analyzed malware families are used for spam distribution, DDoS attacks, and click fraud. We use the same labeled 19-dimensional flow-based feature space with class labels that is recently analyzed in [6, 17]. The dataset relies only on tamper-resistant features making it difficult for sophisticated malware to avoid detection. However, unlike the proposed features, we enhance the feature

[1] http://www.icir.org/enterprise-tracing
[2] https://traces.simpleweb.org/traces/TCP-IP/location6/
[3] http://mcfp.felk.cvut.cz/
[4] http://contagiodump.blogspot.co.uk
[5] http://www.iscx.ca/datasets
[6] http://labs.snort.org/papers/zeus.html

space with the use of eliminated features such as TCP flag information (e.g., total packets with PSH (push) flag set to 1), port numbers and packet timing (e.g., total connection time) as privileged information. This eliminates the concern of crafting adversarial samples at run-time.

User face authentication. To show the efficiency in image oriented security applications, we implement a facial recognition authentication system. Our goal is to recognize an image containing a face with an identifier corresponding to the individual depicted in the image. We train models over the labeled faces in the wild dataset [10]. This set contains 13,000 real-person facial images labeled with each person's name. Various additional datasets use the original images from this collection to build annotations, funneled versions, aligned and cropped images [12, 26]. These additional information of images obtained by the aid of software or human analysis helps us defining auxiliary privileged information to improve the correct authentication of the users.

3.2 Knowledge Transfer

As a first technique, we consider a general algorithm to transfer knowledge from the space of privileged information to the space where detection function is constructed. The algorithm works by deriving a mapping function to estimate each feature in privileged set from a subset of standard set [24]. The estimation is straightforward: the relationship between them is exploited by defining each privileged feature as a target and standard set as an input to a function $f(\cdot)$, e.g., in the form of regression or any other function. This allows a system to apply the same detection model learned before deployment with the union of standard and estimated privileged set as it performs detection with the model inferred from complete set.

We begin the description with a simpler restricted case in which the knowledge transfer always produces the correct response. Assume for now that $f(\cdot)$ is perfect, i.e., it allows for true value estimation of a privileged feature, and we can assume that $f(\widehat{\mathbf{X}}_i, \beta) = x_i^*$ almost surely for any privileged feature x_i^*; this case is covered in Algorithm 1. Suppose that a model is learned with both standard and privileged sets. At run-time, $f(\cdot)$ can be used to substitute privileged features with the true values. Notice that this is exactly the optimal quantity that is minimized by the value returned by Algorithm 1. Under these assumptions, Algorithm 1 is a perfect algorithm for recovery. Thus, a system able to estimate the true values of the privileged set by using a subset of standard features available at detection time. Using this technique, detection systems are able to construct the complete set with the true values of privileged features. This contributes to the information utilization of a model generation and, therefore, enhances the generalization compared to the model trained solely on standard set.

Algorithm 1: Knowledge transfer from a standard set to privileged set

Input : Standard training set
$\mathbf{X}^s = \vec{x_1}, \ldots, \vec{x_L}, \vec{x_i} \in \mathbb{R}^N$
Privileged training set
$\mathbf{X}^* = \vec{x_1^*}, \ldots, \vec{x_L^*}, \vec{x_i^*} \in \mathbb{R}^M$
$\theta \leftarrow$ knowledge transfer parameters

1 Find a best reduced set $\widehat{\mathbf{X}} \subseteq \mathbf{X}^s$, for all $1 \leq i \leq m, x_i^* \in \mathbf{X}^*$

2 Select set $\widehat{\mathbf{X}}_i$ for x_i^*

3 Mapping function f evaluates $(\widehat{\mathbf{X}}_i, x_i^*)$ as given $f_i(\widehat{\mathbf{X}}_i, \beta|\theta) = x_i^*$, $f_i \in \mathcal{F}_m$

4 Output β for x_i^* that minimizes f_i

In practice, $f(\cdot)$ typically learns the patterns of training set which are used to estimate privileged features. We now generalize $f(\cdot)$ is not assumed to be perfect. There is a possibility that $f(\cdot)$ itself does not contain enough useful information about the correlation between certain standard features and the privileged features. The restrictions may be the multicollinearity or overfitting based on the mapping options. For illustrative purposes, consider a model taking one privileged feature, and maps an exact function $f : \mathbf{X^s} \rightarrow x_i^*$. This results in an abundant near similar feature. If we assume that model is learned from a probabilistic classifier such as Naive Bayes, adding a similar feature may assign more weight to that particular sample's target class, as independence assumptions between the features are considered in model. Thus, determining how well a mapping function allows one to predict privileged set may be a cumbersome endeavor similar to the model selection of a detection task, and generally requires further analysis, which is the purpose of the evaluation that we discuss next.

Evaluation. Let our goals be to find rules $y = f(\mathbf{x})$ to 1) classify domain names into two categories in fast-flux (FF) bot detection: benign or malicious; and 2) classify malware families into their respective families in malware classification (MC). Here, complete and standard feature space of datasets are used for comparison. The detection rule without privileged information has to be in the same space of standard features. However, at training time, privileged information of spatial and network-based features in FF and asm files in MC dataset are also available.

To do so, we implement two options as a knowledge transfer: *regression-based* and *similarity-based* to derive $f(\cdot)$ from the training set. A polynomial regression model assumes that each privileged feature is a function of a sub-

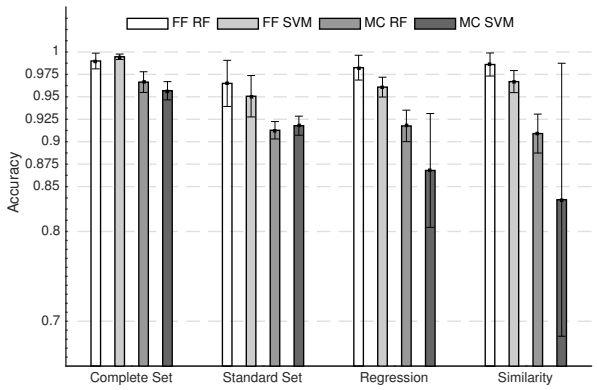

Figure 2: **Knowledge transfer accuracy over baseline standard set.** Regression and similarity mapping options yields. Fast-flux bot detection (FF) and Malware Classification (MC) datasets trained on Random Forrest Classifier (RF) and Support Vector Machines (SVM), as these models are typically preferred for security purposes. Standard set and available privileged information are discussed in Section 3.1.

		Fast-flux Bot Detection			Malware Classification		
		Acc	P	R	Acc	P	R
Complete	RF	98.99± 0.88	99.4	99.4	*96.64± 1.2*	99.3	95.2
Set	SVM	*99.46± 0.3*	99.34	100	95.68± 1	98.63	94.62
Standard	RF	96.5± 2.57	98.68	96.75	91.2± 0.97	91.34	94.35
Set	SVM	95.07± 2.3	94.38	95.5	91.78± 1.1	93.17	93.55
Similarity	RF	**98.6± 1.29**	99.34	98.68	90.09± 2.2	97.27	88.02
(KT)	SVM	96.7± 1.2	98.68	97.07	83.54± 15.2	88.3	85.61
Regression	RF	98.25± 1.38	99.01	98.68	**91.76± 1.8**	94.36	92.5
(KT)	SVM	96.08± 1.1	98.68	96.13	86.8± 6.3	95.56	85.02

Table 2: Accuracy (Acc), precision (P) and recall (R) for FF and MC. All numbers shown are percentages. The best result of the complete set and knowledge transfer (KT) option for each task is highlighted in italics and boldface, respectively.

set of standard features, i.e., there exists a coefficient vector $\beta \in \mathbb{R}^M$ such that $x_i^* = f_i(\mathbf{x}^s, \beta) + b + \varepsilon$ for some bias term b and random residual error ε. We use polynomial regression that fits a nonlinear a relationship to each privileged feature and picks the one that minimizes the error. The resulting mapping function $f_i(\dot{)}$ learned at training time is used to derive an estimation of each privilege feature at detection time.

As a second mapping option, we implement a weighted similarity-based method. This option is used to estimate the privileged features from the most similar samples in the training set. We find the k most similar subset of standard set samples that are selected by using the Euclidean distance between an unseen sample and training instances. Then, the privileged set features are replaced by assigning weights that are inversely proportional to the similarities of their neighbors. In practice, these options typically learn the patterns of data and should largely suffice for deriving the nearly precise mapping functions.

The results are given in Figure 2, which shows the performance of Random Forest Classifier (RF) and Support Vector Machines (SVM) along with transferring knowledge on privileged features on the FF and MC datasets in a form of multiple regression and weighted-similarity. The numbers are given ten independent runs of stratified cross-validation, and measured the difference between training and validation performance with parameter optimization (e.g., k parameter in similarity option). The accuracy of complete dataset in both classifiers are close to the ideal case where the baseline performance is obtained by always guessing the most probable class with 68% accuracy on FF dataset, 55% accuracy on MC dataset.

We found that both knowledge transfer options come close to ideal accuracy on FF dataset (1% less accurate with complete set), and exceeds the standard set accuracy. This means that regression and similarity do a better job predicting privileged information than solely using the standard set available at detection time. For FF dataset, regression and similarity options outperform standard set and gives within 4% more accurate than FF and SVM model trained on solely standard set. However, the results for MC are quite different. Neither regression or similarity option was able to predict the privileged features near precisely. Both the regression and similarity slightly degrades the accuracy; they do slightly worse (3-5%) on both RF and SVM models compared to standard accuracy. This is due to the fact that regression suffers from overfitting to uncommon data points and similarity lacks fitting data points that distinctly lie an abnormal distance from the range of standard features. This is confirmed by checking the sum of square error difference between the estimated and true values. These results indicate that privileged features of MC are difficult to predict using these options. Derivation of more advanced mapping options may require avoiding this limitation.

Table 2 presents all results on complete set, standard set, and quantifies the impact of application of knowledge transfer options on accuracy, precision and recall. We have argued that $f(\cdot)$ might be optimal in one particular sense, i.e., we are able to find a nearly precise relation between standard and privileged features. This decreases the expected misclassification rate on average both in false positives and negatives (as in FF experiments) with the plausible values of privileged set over benchmarked detection with no privileged information. However, we found that it might be a complicated task to specify this relation for which the standard features convey little useful infor-

mation, so the expected misclassification rate minimized here may diverge substantially from the true rate (as in MC experiments). In these cases, knowledge transfer may perform poorly and exploiting the relation becomes a daunting task to estimate contributing privileged features. These constraints may add a challenge to find a valid knowledge transfer option for particular datasets and to estimate satisfying approximated privileged features. For these reasons, we introduce the second technique to relax these assumptions as detailed next.

3.3 Model Influence

In this section, we explore the algorithms in which privileged information is effectively used in the structure or parameters of the detection model at training time. As opposed to the previous technique, we take a radically different approach and ask the essential question *Can we perform detection in a single step without estimation of privileged set ?* To address this question, we implement algorithmic realization of Learning using Privileged Information (LUPI) paradigm [24, 25].

In this technique, the goal is to inject the discriminating power of privileged information to the correction space of the detection model by defining additional constraints on the training errors. In this way, a detection model is learned from both standard and privileged sets, yet it requires only standard set to perform detection on unseen samples. In contrast to the traditional detection, we now consider a training set that is formed as $(\mathbf{X}^s, \mathbf{X}^*, y)$ and generated from an unknown probability distribution $p(\mathbf{X}^s, \mathbf{X}^*, y)$. We attempt to find the best model $f \rightarrow \mathbf{x}^s : y$ such that the expected loss is:

$$R(\theta) = \int_{X^s} \int_{Y} L((\mathbf{x}^s, \theta), y) p(\mathbf{x}^s, \mathbf{x}^*, y) \, d\mathbf{x}^s \, dy \qquad (2)$$

In order to achieve this goal, we implement the Support Vector Machine Plus (SVM+) algorithm, which extends the optimization objective of the SVM algorithm. To understand how we influence the detection model with the privilege information, we show the Lagrangian of SVM+ as follows (see formulation of SVM+ algorithm in Appendix A and implementation details in Appendix B):

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

Equation 3 allows us to influence the detection boundary of standard set $f_i = w^\intercal x_i^s + b$ at x_i^s with the correction function defined by the privileged set $f_i^* = w^{*\intercal} x_i^* + b^*$ at the same location. The correction functions of privileged set considered as the slack variables ε of SVM whereas determined by the privileged set that is only present before deployment. Thus, we are able to include the privileged set to provide a measure of confidence for each labeled sample by introducing them in correction space.

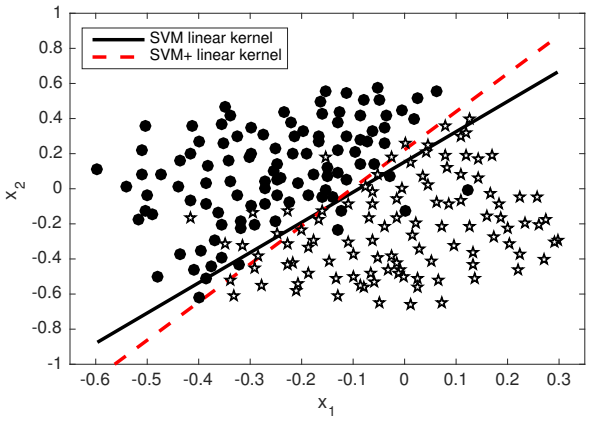

Figure 3: **Visualizing the privileged set impact:** This 2D representation illustrates the detection boundary of a two-class synthetic dataset. The use of privileged information separates the classes more accurately with the same hyperparameters in standard set. This example can be generalized to higher dimensional features with more advanced kernel implementations.

To summarize, the objective function of SVM+ allows us to encode directly the discriminating power of privileged set in detection boundary by simultaneously optimizing both standard and privileged feature spaces. As opposed to general technique knowledge transfer (KT), we eliminate the cumbersome task of finding the nearly precise mapping functions between standard and privileged sets, thus, we reduce the problem to a single-task as a unified framework. Furthermore, the optimization problem of SVM+ requires a significantly smaller number of samples, $O(\sqrt{n})$ samples to converge compared to $O(n)$ samples for traditional learning methods (e.g., SVM). This is helpful for applications with a sparse data collection.

To illustrate the impact of the privileged set, we consider a synthetically generated dataset. The dataset includes two classes, and linear separation of classes is not possible. A sample (x_i, x_i^*, y) is generated from $x_i \sim \mathcal{N}(u, \sigma) + \varepsilon$ with a random mean and standard deviation. Privileged set is defined as $x_i^* \leftarrow x_1, \ldots, x_n$ where n is selected $|2|$. Additional noise samples are also added intentionally to misclassify some samples. The example aims at illustrating the impact of including relevant privileged features on detection accuracy, which have many examples of real-world applications. Figure 3 shows a representative example of decision boundaries found by SVM (solid), and in SVM+ (dashed). On average, we ob-

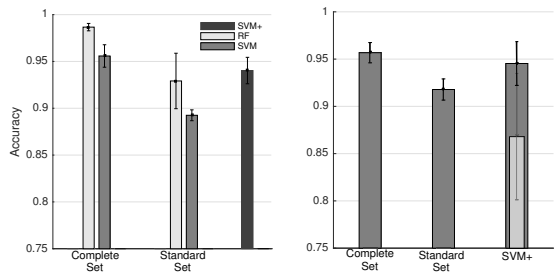

Figure 4: **The accuracy gain of SVM+ after including privileged set:** (Left) Comparison of results for detecting Zeus malware flows amongst benign web flows with SVM and RF models. (Right) Malware classification with SVM model. In all graphs, five independent experiment results are reported with the mean and standard deviation. Overlay bar graph on right compares the accuracy with the best-resulting similarity based knowledge transfer option.

		Malware Traffic Analysis		
		Acc	P	R
Complete Set	RF	*98.67± 0.3*	99.72	98.89
	SVM	95.58± 1.2	98.79	94.64
Standard Set	RF	92.92± 2.96	97.4	95.27
	SVM	89.25± 0.58	94.05	94.58
Model Influence	SVM+	**94.03± 1.4**	94.8	98.84
Similarity (KT)	SVM	93.28± 2.1	94.42	98.35
Regression (KT)	RF	92.56± 0.9	95.07	96.97

Table 3: Quantifying the impact of model influence on accuracy (Acc), precision (P), and recall (R). SVM+ gives the better results than standard set and the best resulting KT options.

tained 90% for SVM and 93% accuracy for SVM+. This shows that privileged set accurately transfers information to standard space, and the model becomes more robust to outliers. Similar examples of datasets composed of high dimensional feature spaces or application of kernel functions may result in better separation of classes.

Evaluation. To formally characterize the notion of detection significance used by model influence, we consider the traffic analysis of benign and botnet that are formed from the members of real-world legitimate web (HTTP(S)) and Zeus botnet traffic[7]. From these classes, we wish to discriminate the Zeus traffic that involves malicious activity such as attempts to connect C&C center or filter private data. We can deduce from network-level traffic whether infected hosts perform malicious activities with a feature vector constructed from flow-based statistics as in [5, 29]. However, this botnet reveals the fact of HTTP mimicry wherein detection is evaded by carefully crafting legitimate sequence of packet flows while performing malicious activities. This makes detection difficult and results in high false positive and negative rates where Zeus traffic is classified as legitimate web flows [17]. To this end, we turn to information gain as a way of including privileged information such as packet flags, port numbers, and packet timing information from packet headers. This information is not preserved in [2, 5, 17], as they can be subverted easily in subsequent versions by the attacker. However, using them as privileged set allows us to build robust detection without any concern at run-time.

We now evaluate accuracy gain of model influence over standard detection with respect to this criterion. Figure 4 (left) plots the SVM, RF and SVM+ results as models trained on complete, standard feature space and leveraging privileged information. The experiments are conducted with the application of polynomial kernel to perform a non-linear classification by implicitly mapping the features into higher dimensional feature space. We observe that including privileged set helps to decrease both false negative and false positive rates. This corresponds to decrease in average 1.1% detection error of RF trained on standard dataset. This positive effect is more observable in SVM model: the detection gain yields 4.8%. Table 3 presents all results of detection of bot flows amongst web traffic and compares the average precision and recall values with the best similarity based and regression based KT options. These results suggest the importance of exploiting statistical dependencies between standard and privileged set for discriminating Zeus from Web traffic.

Finally, we revisit the malware classification (MC) and Fast-flux bot detection (FF) experiments previously evaluated with the knowledge transfer technique. We run the same experiments and compare the results with SVM+. Recall that implementation of regression and similarity based options failed in MC to approximate contributing privileged set, thus, we observed that the models could not benefit from the privileged set in the model internals. SVM+ allows us to infer a model which is generated in the byte feature space with the cooperation of the asm files produced by the dissembler as a correcting space. However, the detector runs without the requirement of asm files after deployment. Figure 4 (right) presents the SVM+ results and compares with the best result of the similarity-based option of KT technique as an overlay bar plot. SVM+ yields 94.029%±1.4 accuracy with an average of 94.8% precision and 98.84% recall. This corresponds to 2.75% detection gain over model trained on SVM standard set, and 7.7% better than KT similarity-based option. This is due to the sensitivity of similarity based option to samples located at abnormal distanced from the range of the standard set samples. However, we found that higher generalization of SVM+ algorithm by considering the only contributing privileged set samples

[7]Botnet traffic includes Zeus V2, Zeus Pony Loader and Zeus Gameover that are appeared between 2011 and 2013 [2].

to the resulting model solves this problem. Similar results hold for the FF experiment. We observe 97.26±1.34 accuracy and an average of 97.02% precision and 99.32% recall which is slightly less than the KT options.

To conclude, we showed that the fact that the best performance achieved with the privileged set is based on useful information drawn by exploiting feature relations both in standard and privileged feature spaces. The resulting detection model better discriminates classes, and this positive effect substantially improves the accuracy, precision and recall over benchmarked detection with no privileged information. We next present generalized distillation for leveraging privileged information that is independent of the underlying model structure.

3.4 Distillation

Introduced by Hinton et. al., distillation is designed to transfer knowledge from a complex Deep Neural Network (DNN) to small one without loss of accuracy [8]. The motivation behind the idea suggested in [1] is closely related to knowledge transfer technique evaluated previously, yet it was formally introduced in [8] to reduce the computational complexity of the DNN architectures which is useful for resource constrained devices (e.g., smartphones). The goal of the distillation is extracting the class knowledge from both hard class labels and probability vectors of each (i.e., soft labels). The benefit of using class probabilities in addition to the hard labels is intuitive because probabilities of each class encode additional information apart from the samples' correct classes.

The idea was recently extended to help us leverage it in the context of the privileged information. Lopez-Paz et al. introduced a unified framework named as *generalized distillation* wherein it enables to learn multiple machines and data representations [13]. The main difference between general and the original distillation proposed by Hinton et al. is the knowledge extracted from privileged information instead of standard training samples. This framework enables us to use privileged information in the context of distillation which aims at improving the accuracy of detection over standard set.

We extract the knowledge from privileged set in the form of probability vectors. This tells us not only the most likely class but also gives us the information about the relative likelihood of each other class. Therefore, the knowledge provided by the privileged set is used in a cost function of a model that attempts to match each class probability in the output. The algorithm for learning such a model is shown Algorithm 2, and outlined as follows:

A model $f(\cdot)$ whose outputs are softmax values is first trained on the privileged set which is different than the traditional detection. Softmax values are obtained by application of softmax function (i.e., normalized exponential) to model f_i output as in Equation 5. It represents a vector s which assigns a probability to each class of the privileged set samples. In Equation 5, temperature parameter T controls the degree of class prediction smoothness. Higher T enables softer probabilities over classes and vice versa. The λ parameter in Equation 4 is used as a balance parameter to adjust the influence of hard and soft class labels. Thus, T and λ parameters play a central role in the underlying dataset, and one needs to optimize them before deployment of a system. As a final step, the Equation 4 is sequentially minimized to distill the knowledge transferred from privileged set as a form of probability vectors (soft labels) into the standard set (hard labels).

Algorithm 2: Detection model generation with privileged information distillation

Input : Standard training set
$\mathbf{X}^s = \vec{x_1}, \ldots, \vec{x_L}, \vec{x_i} \in \mathbb{R}^N$
Privileged training set
$\mathbf{X}^* = \vec{x_1^*}, \ldots, \vec{x_L^*}, \vec{x_i^*} \in \mathbb{R}^M$
Training labels $\mathbf{Y}$
$T \leftarrow$ Temperature parameter , $T > 0$
$\lambda \leftarrow$ Imitation parameter , $\lambda \in [0,1]$

1 Learn a model f_s using $(\mathbf{X}^*, y)$

2 Compute soft labels as $\mathbf{s_i} = \sigma(f_s(x_i^*)/T)$ for all $1 \leq i \leq l$

3 Learn detection model using Equation 4 $\{(\mathbf{X^s}, \mathbf{Y}), (\mathbf{X^s}, S)\}$

4 Output θ for $\mathbf{X^s}$ that minimizes model

$$f_t(\theta) = \operatorname{argmin} \frac{1}{n} \sum_{i=1}^{n} \Bigg(\overbrace{(1-\lambda)\mathcal{L}(y_i, \sigma(f(x_i^s)))}^{\text{hard class labels}} + \overbrace{\lambda \mathcal{L}(s_i, \sigma(f(x_i^s)))}^{\text{soft class labels}} \Bigg) \tag{4}$$

In Equation 4, s_i is the probability vector of classes and obtained from a model f_i with the σ is the softmax function defined as follow:

$$s_i = \sigma\Big(\frac{f_i(x_i^*)}{T}\Big) \tag{5}$$

To conclude, we note that distillation differs from model influence and knowledge transfer. First, it is not limited to the SVM's objective function. Second, while knowledge transfer attempts to estimate the privileged set with a representation of a mapping function, distillation is a trade-off between the privileged set probabilities and hard class labels using various learning algorithms [13]. We next show the positive effect of privileged set distillation on face authentication system.

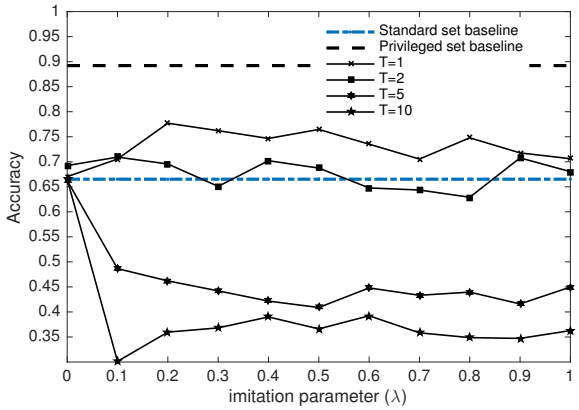

Figure 5: **Privileged set impact on user authentication accuracy.** Standard and privileged set accuracy plotted as a baseline. The effect of temperature (T) and imitation (λ) parameters are quantified on various values.

Evaluation. We implement a user facial authentication system to evaluate this technique. The standard set consists the full images of users (i.e., background included) with 3 RGB channels. The privileged information associated to each user image is the down scaled to bounding boxes that specify the facial localization of the faces. This gives additional information about each user's face by eliminating statistical correlation from the background noise. Note that in some cases background parts of images may unrealistically increase the accuracy because of the distinctive positive effect of faces and backgrounds. However, we verify that the subset of the images in our training set does not suffer from this effect.

We construct both standard and privileged model using a deep neural network with two hidden layers of 10 rectifiers linear unit with a softmax layer for each class. This type of architecture is commonly applied in computer vision and gives better results for image-specific applications. The experiments trained on a subset of data that consists of at least 50 images per person. This corresponds to a subset of 1348 images in our dataset. We train the network with for three epochs, in batches of 25 samples, and report the mean values of ten runs of 400 random sampling of training images.

Figure 5 plots the average model accuracy of standard and privileged sets as a baseline. The resulting model achieves an average of 89.2% correct classification rate on privileged set, which is better than the standard set with 66.5% accuracy. The next experiments measure the effect of temperature T and imitation λ parameters in distillation gain which affects the improvement of using privileged set over standard set accuracy. Here, our objective is identifying the optimal parameters of detection gain for this dataset. We observe that distilling the privileged set gives better detection accuracy than using standard set for

Malware Traffic Analysis			Fast-flux Bot detection			Malware Classification		
Acc	P	R	Acc	P	R	Acc	P	R
95.69± 0.6	96.1	99.33	97.47± 0.3	97.35	99.32	92.56± 0.7	92.57	95.31

Table 4: Distillation prediction results on optimal temperature T and λ parameters.

specific T and λ parameters. The accuracy is maximized when T is between 1 and 3 for the most λ values. We observe the effect of this trade-off most clearly when $T = 1$. The gain in accuracy is on average 6.56%. This yields the best improvement at $T = 1$ and $\lambda = 0.2$ with 11.2% increase in accuracy compared to using only standard set. However, the more increase in T parameter degrades the detection accuracy. This is because increasing the temperature parameter after a saturating point, the distribution of the class probabilities becomes smoother and prevents the proper weighting between standard and privileged sets.

Finally, we revisit the malware traffic analysis, malware classification and fast-flux bot detection experiments previously evaluated with the knowledge transfer and model influence. We run the experiment with the similar network as in face authentication experiment. Table 4 presents the results of distillation. We observe that with distilling at the proper temperature, we increased the detection performance by adding information from privileged set as a form of class probabilities. This yields increase in both false positives and negatives over benchmarked systems that do not use privileged information. We found that detection gains are stable and similar to results of the knowledge transfer and model influence techniques.

To summarize, we benefit from generalized distillation in security primitives to include the explicit relevant privileged information representation in the form of class probabilities to the resulting detection model. We found that this contributes to a better generalization of the model, in turn, improves the detection performance. We followed a different route than knowledge transfer and SVM+ algorithm by using the softmax output layer to achieve this goal. We next summarize the techniques and present the differences and implementation details for realistic deployment environments.

4 Summary and Discussion

We introduced three techniques to model training with privileged information; 1) Knowledge transfer: a general technique of extracting knowledge from privileged information by estimation from available information, 2) Model influence: a model dependent technique of influencing the model optimization with additional knowledge obtained from privileged information; and 3) Distillation: a more flexible technique of distilling the knowledge of

privileged samples as class probability vectors into the standard model. We empirically validated the performance of the privileged information-augmented detection algorithms in a range of detection domains.

The preceding analysis of privileged information demonstrated that the additional discriminating knowledge extracted from privileged set systematically enhances the resulting model's generalization capabilities outside of their standard set. Ideally, one would apply any technique introduced previously. However, one might need to consider the structure of the datasets for technique selection to maximize the detection gain. Thus, we do a preliminary investigation, discussing the dataset properties and algorithm parameters that present guidelines and cautions for the use of privileged information in realistic deployment environments. We summarize our experiences in Table 5.

Technique	Model Dependency	Detection Time Overhead	Model Optimization	Training Set Size
Knowledge Transfer	✗	✓	◓	◓
Model Influence	✓	✗	●	●
Distillation	✗	✗	◓	◓

Legend: ✓ yes ✗ no ◓ model dependent ● relatively higher

Table 5: Guideline for technique selection. We articulate the experimental experiences based on dataset properties and algorithm parameters.

Detection model dependency. Model selection is a task of picking an appropriate model (e.g., classifier) to generate detection function from a set of potential models. Knowledge transfer can be applied to a model of choice, as privileged set features are inferred with the selected option. General distillation only requires a model with a softmax output layer for obtaining the probability vectors. However, SVM+ algorithm is a unified framework that incorporates privileged set in its optimization problem, thus, it is restricted to its objective function.

Detection time overhead. Knowledge transfer options may introduce detection delays while applying the mapping function for estimation of privileged features. As an example, weighted similarity option implemented in Section 3.2 delays estimation until detection time without generating a mapping function at training time. This may introduce a detection bottleneck if dataset includes a large number of instances. We solve this problem by application of stratified sampling to reduce the size of the dataset. Furthermore, using pre-built function based options such regression-based technique in knowledge transfer minimizes the computational delays. On the other hand, if real-time detection is the primary concern, we recommend using model influence and generalized distillation to learn the detection functions a priori; hence, they introduce no overhead at detection time.

Model optimization. The application of techniques involves careful tuning of model parameters and hyperparameters which would yield the best possible model. For example, fine-tuning of the parameters of temperature in general distillation and kernel functions in model influence may increase the detection performance. The amount of parameters required to be optimized both the knowledge transfer and generalized distillation let one choose a priori the amount depending on the selected model. However, SVM+ algorithm has twice as many parameters as SVM algorithm, as two kernel functions are used simultaneously to learn detection boundary in standard and privileged feature spaces. For evaluation, we apply a grid search for small training sets with only a few parameters, and evolutionary search for large-scale datasets for parameter optimization.

Training set size. Training set size effects the time period of learning the model. The amount of time needed to run both knowledge transfer and generalized distillation is negligible, as they require similar models as standard systems apply. However, objective functions of model influence implementation may become infeasible or take a long time when the dimension of the feature space is very small, or dataset size is quite large. To solve this problem, instead of using general solvers such as convex optimization package CVX, we recommend using packages that are designed specifically for solving the quadratic programming (QP) problems (e.g., Matlab `quadprog()` function). Recent attempts have also proposed to accelerate the computation by using specialized spline kernels [24]. We give the implementation details in such packages in Appendix B.

5 Conclusion

We explored an alternate approach to training intrusion detection systems that exploits privileged information. At the core of our system, we use a privileged set that is available at training time, but not at run-time. We used three techniques to encode the useful information extracted from privileged set to the resulting detection model: knowledge transfer, model influence, and distillation. First, we used mapping functions to estimate the privileged set features. Second, we smooth the resulting model with the information extracted from the privileged set. Third, we are able to improve upon model dependency with distillation using probability vector outputs obtained from the privileged set. The common point of all techniques constitutes information utilization that is used to improve the accuracy of detection.

By evaluating the techniques over a variety of detection tasks, we are able to improve the accuracy, recall and precision over benchmarked systems with no privileged information: we showed an average of 4.8% decrease in detection error for malware traffic detection, 3.53% for fast-flux domain bot detection, 3.33% for malware classification, 11.2% for facial user authentication. Moreover, we highlighted limitations and application of techniques for realistic deployment environments.

To the best of our knowledge, we presented the first study of the challenges of sanitizing, correlating and analyzing privileged information in security settings. We highlighted the inherent tension between information utilization, detection accuracy, and robustness of a system. Our future efforts will attempt designing new systems that are able to benefit from advances in learning theory to learn better detection models.

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.

References

[1] BA, J., AND CARUANA, R. Do deep nets really need to be deep? In *Advances in neural information processing systems* (2014), pp. 2654–2662.

[2] BERKAY CELIK, Z., WALLS, R. J., MCDANIEL, P., AND SWAMI, A. Malware traffic detection using tamper resistant features. In *Proc. IEEE Military Communications Conference (MILCOM)* (2015), pp. 330–335.

[3] BICKFORD, J., LAGAR-CAVILLA, H. A., VARSHAVSKY, A., GANAPATHY, V., AND IFTODE, L. Security versus energy tradeoffs in host-based mobile malware detection. In *Proceedings of the 9th international conference on Mobile systems, applications, and services* (2011), ACM, pp. 225–238.

[4] CARDENAS, A. A., MANADHATA, P. K., AND RAJAN, S. P. Big data analytics for security. *IEEE Security & Privacy* (2013).

[5] CELIK, Z. B., AND OKTUG, S. Detection of Fast-Flux Networks using various DNS feature sets. In *Proc. IEEE Symposium on Computers and Communications (ISCC)* (2013).

[6] CELIK, Z. B., RAGHURAM, J., KESIDIS, G., AND MILLER, D. J. Salting public traces with attack traffic to test flow classifiers. In *Proc. Usenix Cyber Security Experimentation and Test (CSET)* (2011).

[7] GARCÍA, S., GRILL, M., STIBOREK, J., AND ZUNINO, A. An empirical comparison of botnet detection methods. *In Computers & Security* (2014).

[8] HINTON, G., VINYALS, O., AND DEAN, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[9] Archsight logger, a comprehensive security information & event management (SIEM) solution. `http://www8.hp.com/us/en/software-solutions/arcsight-logger-log-management/`. [Online; accessed 9-February-2016].

[10] HUANG, G. B., AND LEARNED-MILLER, E. Labeled faces in the wild: Updates and new reporting procedures. Tech. Rep. UM-CS-2014-003, Dept. Comput. Sci.,University of Massachusetts, Amherst, May 2014.

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

[12] JAIN, V., AND LEARNED-MILLER, E. Fddb: A benchmark for face detection in unconstrained settings. Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[13] LOPEZ-PAZ, D., BOTTOU, L., SCHÖLKOPF, B., AND VAPNIK, V. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643* (2015).

[14] Mandiant consulting. `https://www.fireeye.com/services.html`. [Online; accessed 13-February-2016].

[15] Matlab documentation of quadprog() function. `http://www.mathworks.com/help/optim/ug/quadprog.html`. [Online; accessed 15-February-2015].

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

[17] MILLER, D. J., KOCAK, F., AND KESIDIS, G. Sequential anomaly detection in a batch with growing number of tests: Application to network intrusion detection. In *Proc. Machine Learning for Signal Processing (MLSP)* (2012).

[18] PASSERINI, E., PALEARI, R., MARTIGNONI, L., AND BRUSCHI, D. Fluxor: detecting and monitoring fast-flux service networks. In *Proc. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)* (2008).

[19] Penn state university college of engineering sophisticated cyberattacks. `http://securepennstate.psu.edu/engineering-051515/faqs`. [Online; accessed 15-February-2016].

[20] ROSSOW, C., ANDRIESSE, D., WERNER, T., STONE-GROSS, B., PLOHMANN, D., DIETRICH, C. J., AND BOS, H. Sok: P2pwned-modeling and evaluating the resilience of peer-to-peer botnets. In *IEEE Security and Privacy (SP)* (2013), pp. 97–111.

[21] SCARFONE, K., AND MELL, P. Guide to intrusion detection and prevention systems (IDPS). *NIST special publication* (2007).

[22] SHARMANSKA, V., QUADRIANTO, N., AND LAMPERT, C. H. Learning to rank using privileged information. In *Proc. International Conference on Computer Vision (ICCV)* (2013), IEEE.

[23] SHIRAVI, A., SHIRAVI, H., TAVALLAEE, M., AND GHORBANI, A. A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *In Computers & Security* (2012).

[24] VAPNIK, V., AND IZMAILOV, R. Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research* (2015), 2023–2049.

[25] VAPNIK, V., AND VASHIST, A. A new learning paradigm: Learning using privileged information. *Neural Networks* (2009), 544–557.

[26] WOLF, L., HASSNER, T., AND TAIGMAN, Y. Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *IEEE Pattern Analysis and Machine Intelligence* (2011), 1978–1990.

[27] YADAV, S., REDDY, A. K. K., REDDY, A., AND RANJAN, S. Detecting algorithmically generated malicious domain names. In *Proc. ACM Internet measurement* (2010), pp. 48–61.

[28] YEN, T.-F., OPREA, A., ONARLIOGLU, K., LEETHAM, T., ROBERTSON, W., JUELS, A., AND KIRDA, E. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proc. ACM Computer Security Applications* (2013), pp. 199–208.

[29] ZOU, G., KESIDIS, G., AND MILLER, D. J. A flow classifier with tamper-resistant features and an evaluation of its portability to new domains. *IEEE Selected Areas in Communications (JSAC)*, 7 (2011), 1449–1460.

[30] ZUECH, R., KHOSHGOFTAAR, T. M., AND WALD, R. Intrusion detection and big heterogeneous data: a survey. *Journal of Big Data* (2015), 1–41.

A Formulation of Model Influence

We provide the formulation and implementation of model influence algorithm SVM+ introduced in Section 3.3.

We can formally divide the feature space into two spaces at training time. Given L standard vectors $\vec{x_1},\ldots,\vec{x_L}$ and L privileged vectors $\vec{x_1^*},\ldots,\vec{x_L^*}$ with a target class $y=\{+1,-1\}$, where $\vec{x_i}\in\mathbb{R}^N$ and $\vec{x_i^*}\in\mathbb{R}^M$ for all $i=1,\ldots,L$. The kernels $K(\vec{x_i},\vec{x_j})$ and $K^*(\vec{x_i^*},\vec{x_j^*})$ are selected along with positive parameters κ, γ.

Our goal is finding the detection model $f\to\mathbf{x}^s:y$. The SVM+ optimization problem is formulated as [25]:

$$\begin{cases} \sum_{i=1}^{L}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{L}y_iy_j\alpha_i\alpha_jK(\vec{x_i},\vec{x_j}) \\ -\frac{\gamma}{2}\sum_{i,j=1}^{L}y_iy_j(\alpha_i-\delta_i)(\alpha_j-\delta_j)K^*(\vec{x_i^*},\vec{x_j^*})\to\max \\ \sum_{i=1}^{L}\alpha_iy_i=0,\ \sum_{i=1}^{L}\delta_iy_i=0 \\ 0\le\alpha_i\le\kappa C_i,\ 0\le\delta_i\le C_i,\ i=1,\ldots,L \end{cases} \tag{1}$$

The detection rule f for vector $\vec{z}$ is defined as:

$$f(\vec{z})=\text{sign}\left(\sum_{i=1}^{L}y_i\alpha_iK(\vec{x_i},\vec{z})+B\right) \tag{2}$$

where to compute B, we first derive the Lagrangian of (1):

$$\begin{aligned}\mathcal{L}(\vec{\alpha},\vec{\beta},\vec{\phi},\vec{\lambda},\vec{\mu},\vec{\nu},\vec{\rho})=&\sum_{i=1}^{L}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{L}y_iy_j\alpha_i\alpha_jK(\vec{x_i},\vec{x_j})\\&-\frac{\gamma}{2}\sum_{i,j=1}^{L}y_iy_j(\alpha_i-\delta_i)(\alpha_j-\delta_j)K^*(\vec{x_i^*},\vec{x_j^*})\\&+\phi_1\sum_{i=1}^{L}\alpha_iy_i+\phi_2\sum_{i=1}^{L}\delta_iy_i+\sum_{i=1}^{L}\lambda_i\alpha_i\\&+\sum_{i=1}^{L}\mu_i(\kappa C_i-\alpha_i)+\sum_{i=1}^{L}\nu_i\delta_i\\&+\sum_{i=1}^{L}\rho_i(C_i-\delta_i)\end{aligned} \tag{3}$$

with Karush-Kuhn-Tucker (KKT) conditions (for each $i=1,\ldots,L$), we rewrite

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\alpha_i}=&-K(\vec{x_i},\vec{x_i})\alpha_i-\gamma K^*(\vec{x_i^*},\vec{x_i}^*)\alpha_i+K^*(\vec{x_i^*},\vec{x_i^*})\gamma\delta_i\\&-\sum_{k\ne i}K(\vec{x_i},\vec{x_k})y_iy_k\alpha_k-\gamma\sum_{k\ne i}K^*(\vec{x_i^*},\vec{x_k^*})y_iy_k(\alpha_k-\delta_k)\\&+1+\phi_1y_i+\lambda_i-\mu_i=0\\\frac{\partial\mathcal{L}}{\partial\delta_i}=&-K^*(\vec{x_i^*},\vec{x_i^*})\gamma\delta_i+K^*(\vec{x_i^*},\vec{x_i^*})\gamma\alpha_i\\&+\sum_{k\ne i}K(\vec{x_i},\vec{x_k})y_iy_k\gamma(\alpha_k-\delta_k)\\&+\phi_2y_i+\nu_i-\rho_i=0\end{aligned} \tag{4}$$

where

$$\begin{aligned}&\lambda_i\ge0,\ \mu_i\ge0,\ \nu_i\ge0,\ \rho_i\ge0,\\&\lambda_i\alpha_i=0,\ \mu_i(C_i-\alpha_i)=0,\\&\nu_i\delta_i=0,\ \rho_i(C_i-\delta_i)=0,\\&\sum_{i=1}^{L}\alpha_iy_i=0,\ \sum_{i=1}^{L}\delta_iy_i=0\end{aligned} \tag{5}$$

We denote for $i = 1, \ldots, L$

$$F_i = \sum_{k=1}^{L} K(\vec{x}_i, \vec{x}_k) y_k \alpha_k, \qquad (6)$$
$$f_i = \sum_{k=1}^{L} K^*(\vec{x}_i^*, \vec{x}_k^*) y_k (\alpha_k - \delta_k)$$

and rewrite (4) in the form

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \alpha_i} = -y_i F_i - \gamma y_i f_i + 1 + \phi_1 y_i + \lambda_i - \mu_i = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \delta_i} = \gamma y_i f_i + \phi_2 y_y + \nu_i - \rho_i = 0 \\ \lambda_i \geq 0, \;\; \mu_i \geq 0, \;\; \nu_i \geq 0, \;\; \rho_i \geq 0, \\ \lambda_i \alpha_i = 0, \; \mu_i (C_i - \alpha_i) = 0, \;\; \nu_i \delta_i = 0, \; \rho_i (C_i - \delta_i) = 0 \\ \sum\limits_{i=1}^{L} \alpha_i y_i = 0, \;\; \sum\limits_{i=1}^{L} \delta_i y_i = 0 \end{cases} \qquad (7)$$

The first equation in (7) implies

$$\phi_1 = -y_j (1 - y_j F_j - \gamma y_j f_j + \lambda_j - \mu_j) \qquad (8)$$

for all j. If j is selected such that $0 < \alpha_j < \kappa C_j$ and $0 < \delta_j < C_j$, then (7) implies $\lambda_j = \mu_j = \nu_j = \rho_j = 0$ and (8) has the following form

$$\phi_1 = -y_j (1 - y_j F_j - \gamma y_j f_j)$$
$$\phi_1 = -y_j ((1 - \sum_{i=1}^{L} y_i y_j K(\vec{x}_i, \vec{x}_j)(\alpha_i)$$
$$- \gamma \sum_{i=1}^{L} y_i y_j K^*(\vec{x}_i^*, \vec{x}_j^*)(\alpha_i - \delta_i))$$

Therefore, B is computed as $B = -\phi_1$:

$$B = y_j (1 - \sum_{i=1}^{L} y_i y_j K(\vec{x}_i, \vec{x}_j)(\alpha_i)$$
$$- \gamma \sum_{i=1}^{L} y_i y_j K^*(\vec{x}_i^*, \vec{x}_j^*)(\alpha_i - \delta_i)) \qquad (9)$$

where j is such that $0 < \alpha_j < \kappa C_j$ and $0 < \delta_j < C_j$.

B Implementation of Model Influence

MATLAB function `quadprog`$(H, \vec{f}, A, \vec{b}, Aeq, \vec{beq}, \vec{lb}, \vec{ub})$ solves the quadratic programming (QP) problem in the form as follows:

$$\begin{cases} \dfrac{1}{2} \vec{z}^T H \vec{z} + f^T \vec{z} \to \min \\ A \cdot \vec{z} \leq \vec{b} \\ Aeq \cdot \vec{z} = \vec{beq} \\ \vec{lb} \leq \vec{z} \leq \vec{ub} \end{cases} \qquad (10)$$

Here, H, A, Aeq are matrices, and $\vec{f}, \vec{b}, \vec{beq}, \vec{lb}, \vec{ub}$ are vectors. $\vec{z}$ is defined as $\vec{z} = (\alpha_1, \ldots, \alpha_L, \delta_1, \ldots, \delta_L) \in R^{2L}$. We now rewrite (1) in the form of (10).

$$\frac{1}{2} \vec{z}^T H \vec{z} + f^T \vec{z} \to \min$$

where

$$f = (-1_1, -1_2, \ldots, -1_L, 0_{L+1}, \ldots, 0_{2L})$$

and

$$H_{ij} = \begin{pmatrix} H^{11} & H^{12} \\ H^{12} & H^{22} \end{pmatrix}$$

where, for each pair $i, j = 1, \ldots, L$,

$$H_{ij}^{11} = K(\vec{x}_i, \vec{x}_j) y_i y_j + \gamma K^*(\vec{x}_i^*, \vec{x}_j^*) y_i y_j,$$
$$H_{ij}^{12} = -\gamma K^*(\vec{x}_i^*, \vec{x}_j^*) y_i y_j,$$
$$H_{ij}^{22} = +\gamma K^*(\vec{x}_i^*, \vec{x}_j^*) y_i y_j$$

The second line of (10) is absent. The third line of (10) corresponds to the second line of (1) when written as

$$Aeq \cdot \vec{z} = \vec{beq}$$

where

$$Aeq = \begin{pmatrix} y_1 & y_2 & \cdots & y_L & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & y_1 & y_2 & \cdots & y_L \end{pmatrix},$$

$$\vec{beq} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The fourth line of (10) corresponds to the third line of (1) when written as

$$\vec{lb} \leq \vec{z} \leq \vec{ub}$$

where

$$\vec{lb} = (0_1, 0_2, \ldots, 0_L, 0_{L+1}, \ldots, 0_{2L}),$$
$$\vec{ub} = (\kappa C_1, \kappa C_2, \ldots, \kappa C_L, C_1, C_2, \ldots, C_L)$$

After all variables $(H, \vec{f}, A, \vec{b}, Aeq, \vec{beq}, \vec{lb}, \vec{ub})$ are defined as above, optimization toolbox guide [15] can be used to select `quadprog()` function options such as an optimization algorithm and maximum number of iterations. Then, output of the function can be used in detection rule f for a new sample $\vec{z}$ to make predictions as follows:

$$f(\vec{z}) = \text{sign} \left(\sum_{i=1}^{L} y_i \alpha_i K(\vec{x}_i, \vec{z}) + B \right) \qquad (11)$$