Detection under Privileged Information*

Z. Berkay Celik¹, Patrick McDaniel¹, Rauf Izmailov², Nicolas Papernot¹, Ryan Sheatsley¹,
Raquel Alvarez¹, and Ananthram Swami³

¹Pennsylvania State University, ²Vencore Labs, ³Army Research Laboratory

ABSTRACT

For well over a quarter century, detection systems have been driven by models learned from input features collected from real or simulated environments. An artifact (e.g., network event, potential malware sample, suspicious email) is deemed malicious or non-malicious based on its similarity to the learned model at runtime. However, the training of the models has been historically limited to only those features available at runtime. In this paper, we consider an alternate learning approach that trains models using “privileged” information—features available at training time but not at runtime—to improve the accuracy and resilience of detection systems. In particular, we adapt and extend recent advances in knowledge transfer, model influence, and distillation to enable the use of forensic or other data unavailable at runtime in a range of security domains. An empirical evaluation shows that privileged information increases precision and recall over a system with no privileged information: we observe up to 7.7% relative decrease in detection error for fast-flux bot detection, 8.6% for malware traffic detection, 7.3% for malware classification, and 16.9% for face recognition. We explore the limitations and applications of different privileged information techniques in detection systems. Such techniques provide a new means for detection systems to learn from data that would otherwise not be available at runtime.

KEYWORDS

Detection systems; privileged information; machine learning

ACM Reference Format: Z. Berkay Celik, Patrick McDaniel, Rauf Izmailov, Nicolas Papernot, Ryan Sheatsley, Raquel Alvarez, and Ananthram Swami. 2018. Detection under Privileged Information. In *ASIA CCS'18: 2018 ACM Asia Conference on Computer and Communications Security, June 4–8, 2018, Incheon, Republic of Korea*. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3196494.3196502>

1 INTRODUCTION

Detection systems based on machine learning are an essential tool for system and enterprise defense [31]. 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 runtime environment against known normal or anomalous states. 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 at runtime. A limitation of this traditional approach is that it relies solely on the features (also referred to as inputs) that are available at runtime. In practice, many features are too expensive to collect in real-time or only available after the fact—and are thus ignored for the purposes of detection.

Consider a recent event that occurred in the United States. In the Summer of 2017, the credit reporting agency Equifax fell victim to sophisticated cyber attacks that resulted in substantial exfiltration of personal information and intellectual property [4]. Working with the government staff and security analysts conducted a clandestine investigation. During that time, a vast amount of information was collected from networks and systems across the agency, e.g., network flows, system logs files and user activity. An analysis of the collected data revealed the presence of previously undetected advanced persistent threat (APT) actors on the agency’s network. Yet, the collected analysis is largely non-actionable by detection systems post investigation; because the vast array of derived features would not be available at runtime, they cannot be used to train Equifax’s detection systems.

In other contexts, features may be available at runtime but infeasible or undesirable to collect because of environmental or system constraints. For example, the collection of a large numbers of features in environments of mobile phones [3], Internet of Things (IoT) [29], sensor networks [16], embedded control systems [30], or ad-hoc networks [5] is often too slow or requires too many resources to be feasible in practice.

These examples highlight a challenge for future intrusion detection: *how can detection systems integrate intelligence relevant to an attack that is not available at runtime?* Here, we turn to recent advances in machine learning that support models that learn on a superset of features used at runtime [33, 34]. The goal of the work described in this paper is to leverage these additional features, called *privileged information* (features available at training time, but not at runtime), to improve the accuracy of detection. Using this approach designers and operators of detection systems can leverage additional effort during system calibration to improve detection models without inducing additional runtime costs.

Pioneered recently by Vapnik, Izmailov, and others, learning under privileged information eliminates the need for symmetric features in training and runtime—thereby expanding the space of learning models to include “ancillary” and “non-runtime” information. However, to date, the application of these techniques in practical domains has been limited, and within the context of security non-existent. In this work, we explore how this new mode of learning can be leveraged in detection systems. This requires an exploration of not only use of these new learning models but also their applicability to security domains and the requirements of those domains on feature engineering. Our experience in this effort over the last two years has demonstrated that blind application of

*The development of detection systems under privileged information was a highly complex endeavor. An extended version of this paper is available with substantially more description, detail and commentary [10].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASIA CCS '18, June 4–8, 2018, Incheon, Republic of Korea

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5576-6/18/06...\$15.00

<https://doi.org/10.1145/3196494.3196502>

privileged application can lead to poor detection—yet judicious and careful use can substantially improve detection quality.

More concretely, in this paper, we explore an alternate approach to training intrusion detection systems that exploit privileged information (See Figure 1). We design algorithms for three classes of privileged-augmented detection systems that use: (1) *Knowledge transfer*, a general technique of extracting knowledge from privileged information by estimation from available information, (2) *Model influence*, a model-dependent approach of influencing the model optimization with additional knowledge obtained from privileged information; and (3) *Distillation*, an approach of summarizing the additional knowledge about privileged samples as class probability vectors. We further explore feature engineering in a privileged setting. To this end, we measure the potential impacts of privileged features on runtime models. Here, we use the degree to which a feature improves a model (*accuracy gain*—a feature’s additive contribution to accuracy) as a quality metric for selecting privileged features for a target model. We develop an algorithm and system that selects features that maximize model accuracy in the presence of privileged information. Finally, we compare the performance of privileged-augmented systems with systems without privileged information. We evaluate four recently proposed detection systems: (1) face authentication, (2) fast-flux bot detection, (3) malware traffic detection, and (4) malware classification. Our contributions are:

- We augment several diverse detection systems using three classes of privileged information techniques and explore the strength and weaknesses of these techniques.
- We present the first methods for feature engineering in privileged-augmented detection for security domains and identify inherent tensions between information utilization, detection accuracy, and model robustness.
- We provide an evaluation of techniques on a variety of existing detection systems on real-world data. We show that privileged information decreases the relative detection error of traditional detection systems up to 16.9% for face authentication, 7.7% for fast-flux bot detection, 8.6% for malware traffic detection, and 7.3% for malware classification.

2 PRIVILEGED INFORMATION DETECTION

This section introduces three approaches to integrate privileged features into detection algorithms: knowledge transfer, model influence, and distillation. Following the definition of privileged information, we consider a detection setting where the features $\mathcal{X}$ used for detection are split into two sets to characterize their availability for detection at training time. The *standard features* includes the features that are reliably available both at training and runtime (as in conventional systems), while *privileged features* have constraints that prevent using them for detection at runtime. More formally, we assume that detection models will be trained on some data $\{(x_i, x_i^*, y_i)\}_{i=1}^n$, and they will make detection on some data $\{x_j\}_{j=n+1}^{n+m}$. Therefore, our goal is a creation of algorithms that efficiently integrate privileged features $\{x_i^*\}_{i=1}^n$ into detection models, without requiring them at runtime.

2.1 Knowledge Transfer

We consider a general algorithm to transfer knowledge from the space of privileged information to space where the detection model is constructed [33]. The algorithm works by deriving a *mapping function* to estimate each privileged feature from a subset of standard features. The algorithm used to identify a mapping function f_i .

Figure 1: Overview of a typical detection system (left) and proposed solution (right): Given training data including both malicious (+) and benign samples (−), modern detection systems use training data to construct a model. In contrast, we construct a model both using training and privileged data, yet the model only requires the training data features to detect unknown sample as benign or malicious.

The estimation is straightforward: the relationship between standard and privileged features is learned by defining each privileged feature x_i^* as a target and standard set as an input to a mapping function $f_i \in \mathcal{F}$. The mapping functions can be defined in the form of any function such as regression or similarity-based (we give examples in Section 4.1). The use of the mapping function allows a system to apply the same model learned from the complete set at training time with the union of standard and estimated privileged features on unknown samples. By using f_i , detection systems are able to construct the complete features with the correct values of privileged features at runtime—intuitively, each f_i is used to generate a synthetic feature that represents an estimate of a privileged feature. As a result, the accurate estimation of privileged features contributes to using complete and relevant features in a model training and, therefore, enhances the generalization of models compared to those trained solely on standard features. Note that the estimating power of f_i is bounded by the size and completeness of the training data (with respect to the privileged features), and thus the use of f_i in the model should be calibrated based on measurements of estimation quality (See Section 4.4).

2.2 Model Influence

Model influence incorporates the useful information obtained from privileged features to the correction space of the detection model by defining additional constraints on the training errors [33, 34]. Intuitively, the algorithm learns how privileged information influences outputs on training input feature vectors towards building a set of corrections for the space of inputs—in essence creating a correction function that takes as input runtime features and adjusts model outputs. Note that while we adapt model influence to support vector machines (SVM) herein, it applies to other ML techniques. Here the system is trained on standard and privileged features but only uses standard features at runtime.

We influence the detection boundary of a model trained on standard features $f_i = w^T x_i^s + b$ at x_i^s with the correction function defined by the privileged features $f_i^* = w^{*T} x_i^* + b^*$ at the same location (labeled as influence from privileged features). In this manner, we use privileged features as a correction function of the slack variables ξ_i defined in the objective of SVM. In turn, the useful information obtained from privileged features is incorporated as a measure of confidence for each labeled standard sample. The formulation is named as SVM+ and requires $O(\sqrt{n})$ samples to converge

compared to $O(n)$ samples for SVM which is useful for systems with a sparse data collection [7, 33]. We refer readers to our extended paper [10] for a complete formulation and implementation.

2.3 Distillation

Model compression or distillation are techniques to transfer knowledge from a complex Deep Neural Network (DNN) to a smaller one without loss of accuracy [17]. The goal of the distillation is to use the class knowledge from both class labels (i.e., hard 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 define a similarity metric over the classes apart from the samples' correct classes. Lopez-Paz et al. recently introduced an extension of model distillation used to compress models built on a set of features into models built on a different set of features [22]. We adapt this technique to detection algorithms.

We address the problem of privileged information using distillation by minimizing Equation 1, which learns a detection model by simultaneously imitating the privileged predictions of the privileged model and learning the targets of the standard set. The algorithm for learning such a model is outlined as follows:

$$f_t(\theta) = \underset{f \in F_t}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n \left(\overbrace{(1-\lambda) \mathcal{L}(y_i, \sigma(f(x_i^s)))}^{\text{detection}} + \overbrace{\lambda \mathcal{L}(s_i, \sigma(f(x_i^s)))}^{\text{imitate privileged set}} \right) \quad (1)$$

We first learn a privileged model f_s by using the privileged samples available at training time. We then compute the soft labels by applying the softmax function (i.e., normalized exponential) $s_i = \sigma(f_s(x_i^*)/T)$. The output is a vector which assigns a probability to each class of the privileged samples. We note that class probabilities obtained from privileged model provide additional information for each class. Here, temperature parameter T controls the degree of class prediction smoothness. Higher T enables softer probabilities over classes and vice versa. As a final step, Equation 1 is sequentially minimized to distill the knowledge transferred from privileged features as a form of probability vectors (soft labels) into the standard sample classes (hard labels). In Equation 1, the λ parameter controls the trade-off between privileged and standard features. For $\lambda \approx 0$, the objective function approaches the standard set objective, which amounts to detection solely on standard features. However, as $\lambda \rightarrow 1$, the objective transfers the knowledge acquired by the privileged model into the resulting detection model. Therefore, learning from the privileged model does, in many cases, significantly improve the learning process of a detection model.

3 PRIVILEGED INFORMATION SYSTEMS

In this section, we explore algorithms for feature engineering (selecting privileged features for a detection task) and demonstrate their use in diverse experimental systems.

3.1 Selecting Privileged and Standard Features

The first challenge facing our model of detection is deciding which features should be used as privileged information. Asked another way, *given some potentially large universe of offline features, which are the most likely to improve detection?* To address this, we develop an iterative algorithm that selects features that maximize model accuracy. Selection is made on the calculated *accuracy gain* of each feature—a measure of the additive value of the features concerning an existing feature set for detection accuracy.

Our feature selection algorithm measures the potential impacts of privileged features that help detect the hard-to-classify examples. Generally speaking, easy examples fall in a distribution that can be

explained by some set of model parameters, and hard examples do not precisely fit the model—and are either misclassified or near a decision boundary [32]. As a consequence, accurate classification of hard examples is one of the main challenges of practical systems, as they are the main source of detection errors due to the incorrect or insufficient information about normal or anomalous system states.

The first step of feature engineering—as is true of any detection task—is identifying all of the available features that potentially may be used for detection. Specifically, we collect the set of domain-specific features based on using domain knowledge and surveying the recent efforts in that domain. It is from that set that we will identify the privileged features to be used for training. Note that defining privileged features sometimes requires a level of domain expertise. However, trained security experts will find most privileged features straightforward after defining the runtime constraints on features. Additionally, the initial privileged set may include irrelevant features that carry little or no useful information for the target detection task; thus we identify the privileged set using an algorithm. The algorithm starts with standard features of a detection system and sequentially adds one privileged feature from the set which maximizes correct classification of hard examples, i.e., the feature whose addition to the existing set has the greatest positive impact on accuracy (measured accuracy gain). The accuracy gain of hard examples is found using SVM. This process is repeated until the potential feature set is empty, a maximum number of features is reached, or the accuracy gain is below a threshold for usefulness.

3.2 Experimental Systems

In this section, we introduce four security-relevant systems for face authentication, fast-flux bot detection, malware traffic detection, and malware classification. This diverse set of detection systems serves as a representative benchmark suite for our approaches. Additional details about these systems and their features are presented in Appendix C. We also consider several illustrative uses of privileged information in Appendix A.

Experimental System: Face Authentication - To explore the efficiency of approaches in image domains, we modeled a user authentication system based on recognition of facial images. Our goal is to recognize an image containing a face with an identifier corresponding to the individual depicted in the image. We use images from a public dataset that includes face images labeled with each person's name [19]. We build the features from 1348 facial images with at least 50 images per user.

Privileged information. It is recently found that face recognition systems used for access control in particular energy and computation constrained camera sensors can be easily bypassed by an attacker [14]. In this, the lack of useful features or number of images used to train the systems is the main reason for duping the systems into falsely authenticate/recognize users. We use two types of privileged features for each image in addition to the original images in model training: cropped and funneled versions of the images [21, 35]. These images provide additional information for a users' face by image aligning and localizing. While it is technically possible these features can be obtained by an aid of software or human expert at runtime, they are much more likely to not be available in low energy and slow processing sensors (and thus we define them as privileged).¹

Experimental System: Fast-flux Bot Detection - The Fast-flux bot detector is used to identify hosts that use fast-changing DNS entries to hide the existence of server hosts used for malicious

¹We interpret the accuracy gain as a defense for hardening misclassification of users.

System	Approach	Relative detection gain		
		Accuracy	Precision	Recall
Fast-flux Bot Detection	Knowledge Transfer	7.7%	5.3%	3.4%
Malware Classification	Model Influence	7.3%	2.2%	3.7%
Malware Traffic Analysis	Distillation	8.6%	2.2%	5%
Face Authentication	Distillation	16.9%	9.3%	9.2%

Table 1: Summary of accuracy, precision and recall gain over traditional detection via privileged information-augmented systems. The best resulting approach is illustrated.

activities. The raw data consists of 4GB DNS requests of benign and active fast-flux servers collected in early 2013 [11]. We build a detection system by using the 19 features used in recently proposed botnet detectors [20, 28, 36]. This system relies on features obtained from domain names, DNS packets, packet timing intervals, WHOIS domain lookup and IP coordinate database. The resulting dataset includes many features to increase separation of Content Delivery Networks (CDNs) from fast-flux servers, as similarities between them are the main source of detection errors.

Privileged information. In this system, even though the complete features are relevant for fast-flux detection, obtaining some features at runtime entails computational delays. For example, processing of WHOIS records, maintaining up-to-date IP coordinate database and whitelist of domain names takes several minutes/hours. Thus, we define eleven features obtained from these sources as privileged to assure real-time detection.

Experimental System: Malware Traffic Detection - Next, we modeled a malware traffic anomaly detection system based on network flow statistics used in recent detection systems [12, 15, 25]. The system aggregates 20 flow features for detecting botnet command and control (C&C) activity among benign applications. For instance, the ratio between maximum and minimum packet size from server to client and client to server find out to be a distinctive observation between benign and malicious samples. We add 173 botnet traffic of Zeus variants that is used for spam distribution, DDoS attacks, and click fraud to the 1553 benign applications (web browsing, chat, email, etc.) of Lawrence Berkeley National Laboratory (LBNL) [27] and University of Twente [2].

Privileged information. In this system, the authors eliminate the features that can be readily altered by an attacker, as the model trained with tampered features allows an attacker to manipulate the detection results easily [23, 37]. The impact of altering the features at run time on detection accuracy is recently studied [12]. For instance, consider that destination port numbers or packet interval time are used as a feature. An adversary may “easily” change them in subsequent malware versions to evade detection systems. Also, the authors do not use payload content to obtain features because the attacker can use encrypted traffic to prevent deep packet inspection. Thus, we deem such eight features as a privileged, as inference does not consider their tampered values at runtime.

Experimental System: Malware Classification - The Microsoft malware dataset [24] is an up-to-date publicly available corpus. The dataset includes nine malware classes including hexadecimal representation of the malware’s binary content, and a class representing one of nine family names. The dataset used in our experiments includes 1746 malware samples extracted from 200GB malware files. We build a real-time malware classification system by using the binary content file. Following a recent malware classification system [1], we construct features by counting frequencies of each hexadecimal duos (i.e., byte bigrams). These features found out to provide distinctive between different families because of exploiting the code dissimilarities among families.

Privileged information. This dataset also includes a metadata manifest log file. The log file contains information such as memory allocation, function calls, strings, etc.. The logs along with the malware files can be used for classifying malware into their respective families. Thus, similar to the byte files, we obtain the frequency count of distinct tokens from asm files such as `mov()`, `cmp()` in the text section. These tokens allow us to capture execution differences between different families. However, in practice, obtaining features from log files introduces significant overheads in the disassembly process. Further, various types or versions of a disassembler may output byte sequences differently. Thus, this process may result in inaccurate and slow feature processing in real-time systems [26]. To address these limitations, we include features from disassembler output as privileged for accurate and fast classification.

4 EVALUATION

We demonstrate the detection gain of including privileged features. The accuracy, precision and recall gain of the best resulting approach for each detection system is summarized in Table 1. As detailed throughout, we find that the use of privileged information can improve—often substantially—detection performance.

Overview of Experimental Setup. We compare the performance of privileged-augmented systems against two baseline (nonprivileged) models: the standard set model and the complete set model. The *standard set model* is a conventional detection system that does not include the privileged features for training or runtime, but uses all of the standard features. The *complete set model* is a conventional system that includes all the privileged and standards features for training or runtime. Note that the ideal privileged information approach would have similar performance as the complete set.

To learn standard and complete set models, we use classifiers of Random Forest (RF) and Support Vector Machines (SVM) with a radial basis function kernel. These classifiers give better performance in the previously introduced systems and are also preferred by the system authors. The parameters of the models are optimized with exhaustive or randomized parameter search based on the dataset size. We show detection performance of complete and standard set models and compare their results with our approaches based on three metrics: accuracy, recall, and precision.

4.1 Knowledge Transfer

We compare the performance of privileged-augmented detection system using knowledge transfer (KT) over standard and complete set models. In this experiment, we classify domain names into benign or malicious in fast-flux bot detection system (FF).

To realize KT, we implement two mapping functions to estimate the privileged features from a subset of standard features: *regression-based* and *similarity-based*. We find that both mapping functions learn the patterns of a training data and mostly suffice for the derivation of the nearly precise estimated privileged features. First, a polynomial regression function is built to find a coefficient vector $\beta \in \mathbb{R}^d$ such that there exists a $x_i^* = f_i(x^s, \beta) + b + \epsilon$ for some bias term b and random residual error ϵ . The resulting function is then used to estimate each privileged feature at detection time given the standard features as input. We use polynomial regression that fits a nonlinear relationship to each privileged feature and picks the one that minimizes the sum of squares error. To evaluate the effectiveness of the regression, we implement a second mapping function named weighted-similarity. This function is used to estimate the privileged features from the most similar samples in the training set. We first find the k most similar subset of standard samples that

	Fast-flux Bot Detection				Malware Traffic Detection			Malware Classification		
	Model	Acc.	Pre.	Rec.	Acc.	Pre.	Rec.	Acc.	Pre.	Rec.
Complete Set	RF	99±0.9	99.4	99.4	98.7±0.3	99.7	98.9	96.6±1.2	99.3	95.2
	SVM	99.4±0.3	99.3	100	95.6±1.2	98.8	94.6	95.7±1	98.6	94.6
Standard Set	RF	96.5±2.6	98.7	96.8	92.9±3	97.4	95.3	91.2±1	91.3	94.4
	SVM	95±2.3	94.4	95.5	89.2±0.6	94	94.6	91.8±1.1	93.2	93.6
KT (Similarity)	RF	98.6±1.3	99.3	98.7	93.3±2.1	94.4	98.4	90.1±2.2	97.3	88
	SVM	96.7±1.2	98.7	97.1	92.6±0.9	95.1	96	83.5±11.2	88.3	85.6
Model Influence	SVM+	97.3±1.3	97	99.3	94±1.4	94.8	98.8	94.6±2.3	93.3	97.8
Distillation	DNN	97.5±0.3	97.4	99.3	95.7±0.6	96.1	99.3	92.6±0.7	92.6	95.3

Table 2: Summary of results: Accuracy (Acc.), Precision (Pre.) and Recall (Rec.) The best result is highlighted in bold.

are selected by using the Euclidean distance between an unknown sample and training instances. Then, the privileged features are replaced by assigning weights that are inversely proportional to the similarities of their neighbors. We note that other distance metrics gives worse accuracy than Euclidean distance.

Table 2 shows the accuracy of Random Forest Classifier (RF) and Support Vector Machines (SVM) on standard, complete models, and KT in the form of weighted-similarity (We report similarity results as it yields better results than regression). The average accuracy of ten independent runs of stratified cross-validation is given by measuring the difference between training and validation performance with parameter optimization (e.g., k parameter in similarity). The complete model accuracy of both classifiers is close to 99%. We found that mapping functions are effective in finding a nearly precise relation between standard and privileged features. This decreases the expected misclassification rate on average both in false positives and negatives over benchmark detection with no privileged features. Both KT mapping options come close to the complete model accuracy on FF dataset (1% less accurate), and significantly exceeds the standard set accuracy (2% more accurate).

4.2 Model Influence

Next, we evaluate the performance of model influence-based privileged information in detecting Zeus botnet in real-world web (HTTP(S)) traffic. Here, the system attempts to detect the malicious activity of a Zeus botnet that connects to C&C centers and filters private data. Note that the Zeus botnet uses HTTP mimicry to avoid detection. As a consequence, the sole use of standard features makes detection of Zeus difficult, resulting in high detection error (where Zeus traffic is mostly classified as legitimate web traffic). To this end, we include privileged features of packet flags, port numbers, and packet timing information from packet headers (See Appendix C for the complete list of features). We observe that while these features can be spoofed by adversaries under normal conditions, using them as privileged information may counteract spoofing (because inference does not consider their runtime value).

We evaluate accuracy gain of model influence over the standard model. We use a polynomial kernel in the objective function to perform a non-linear classification by implicitly mapping the features into a higher dimensional feature space. We note that we avoid overfitting by tuning of the regularization parameter. Table 2 presents the impact of model influence on accuracy, precision, and recall and compares with standard and complete models. We found that using the privileged features inherent to malicious and benign samples in model training systematically better separates the classes. This positive effect substantially improves both false negative and false positive rates. The accuracy of model influence is close to the optimal accuracy and reduces the detection error on

average 2% over RF trained on the standard set. This positive effect is more observable in SVM, and the accuracy gain yields 4.8%.

4.3 Distillation

We evaluate distillation on the experimental face authentication system. The standard features of a face authentication system consist of original images of users (i.e., background included) with 3 RGB channels. We obtain the privileged features of each image from funneled and downscaled bounding boxes of images. In this way, we better characterize each user's face by specifying the face localization and eliminating the background noise. In the following experiments, we construct both standard and privileged models using a deep neural network (DNN) 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 provides superior results for image-specific applications. We train the network with ten runs of 400 random training samples.

The resulting model achieves an average of 89.2% correct classification rate on the privileged set, which is better than the standard set with 66.5%. We observe that distilling the privileged set features into our detection algorithm gives better accuracy than standard set accuracy with optimal T and λ parameters. The accuracy is maximized when $T = 1$, the gain is on average 6.56%. The best improvement is obtained when $T = 1$ and $\lambda = 0.2$ with 11.2% increase over the standard set model accuracy. However, increases in T negatively affect detection accuracy. This is because as T increases, the objective function puts more weight on learning from the standard features which upsets the trade-off between standard and privileged features.

4.4 Comparison of Approaches

We compare the relative performance of the approaches on three data sets.² Distillation is implemented using Deep Neural Networks (DNN), and regression and weighted similarity mapping functions are used for knowledge transfer. Table 2 presents the results of knowledge transfer, model influence, and distillation and compares against complete and standard models. We present calibration of approaches to maximize the detection gain in Appendix B.

The accuracy of model influence, distillation and knowledge transfer on the fast-flux detector and malware traffic detection is stable. All approaches yield accuracy similar to the ideal accuracy of the complete set model, and often the increased accuracy is the result of the correct classification of true positives (intrusions). This results in on average up to 5% relative gain in recall with the model influence and distillation over conventional models. In contrast, knowledge transfer often increases the number of samples detected

²We do not compare performance on face recognition because processing the number of the input features (e.g., pixels) was intractable for several solutions.

by systems as being actually malicious (e.g., 99.3% precision in the fast-flux detector), meaning that the number of false alarms is reduced over conventional detection. The results confirm that the approaches are able to balance the conventional detection and its accuracy by using privileged features inherent to both benign and malicious samples: either reducing the false positives or negatives. We note that these results are obtained after carefully tuning the model parameters. We further discuss their parameter tuning and impact of results on systems in Appendix B.

Distillation is easy to apply as its objective is independent of the machine learning algorithm (model-free) and often yields better results than other approaches. Its quality as a detection mechanism becomes more apparent when its objective function is implemented with deep neural networks (DNN) with a nonlinear objective [22]. This makes distillation give better results on average than other approaches. On the other hand, the design and calibration of model influence detection require additional effort and care in tuning parameters—in the experiments, this additional effort yields strong detection (as 94.6% in malware classification). Note that when the dataset includes a large number of privileged features or samples, training of model influence takes significantly more time compared to other approaches (See Appendix B).

Finally, it is important to note that the while knowledge transfer accuracy gain for fast-flux detection and malware traffic analysis is similar to other approaches, its malware classification results are inconsistent (i.e., 83.5% average accuracy with an 11.2% standard deviation). Neither regression nor similarity mapping functions were able to predict the privileged features near precisely, in turn, they slightly degrade the accuracy (7–8%) on both RF and SVM standard set models. This observation confirms the need to find and evaluate an appropriate mapping function for the transfer of knowledge discussed in Section 2.1. In this particular dataset, the mapping functions fail to find a good relation between standard and privileged features. 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 (confirmed by an increase in the sum of square errors of estimated and true values for the privileged features). We remark that derivation of more advanced mapping functions may solve this problem [9].

5 CONCLUSIONS

We have presented a range of techniques to train detection systems with privileged information. All approaches use features available only at training time to enhance the accuracy of detection models. We consider three approaches: (a) knowledge transfer to construct mapping functions to estimate the privileged features, (b) model influence to smooth the detection model with the useful information obtained from the privileged features, and (c) distillation using probability vector outputs obtained from the privileged features in the detection objective function. Our evaluation of several detection systems shows that we can we improve the accuracy, recall, and precision regardless of their high detection performance using privileged features.

ACKNOWLEDGMENTS

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] Mansour A. et al. 2015. Novel feature extraction, selection and fusion for effective malware family classification. *arXiv preprint arXiv:1511.04317*.
- [2] R. Barbosa, R. Sadre, A. Pras, and R. Meent. 2010. University of Twente traffic traces data repository. In *University of Twente Tech Report*.
- [3] J. Bickford et al. 2011. Security versus energy tradeoffs in host-based mobile malware detection. In *Mobile systems, applications, and services*.
- [4] Equifax Data Breach. 2018. <https://en.wikipedia.org/wiki/Equifax>. (2018). [Online; accessed 15-January-2018].
- [5] I. Butun et al. 2014. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*.
- [6] A. A. Cardenas et al. 2013. Big data analytics for security. *IEEE System Security*.
- [7] Z. B. Celik, R. Izmailov, and P. McDaniel. 2015. *Proof and implementation of algorithmic realization of learning using privileged information (LUPI) paradigm: SVM+*. Technical Report NAS-TR-0187-2015. NSCR, CSE, PSU.
- [8] Z. Berkay Celik, David Lopez-Paz, and Patrick McDaniel. 2016. Patient-driven privacy control through generalized distillation. *IEEE Symposium on Privacy-Aware Computing (PAC)*.
- [9] Z. B. Celik, P. McDaniel, and R. Izmailov. 2017. Feature cultivation in privileged information-augmented detection. In *ACM CODASPY IWSPA*.
- [10] Z. Berkay Celik, Patrick McDaniel, Rauf Izmailov, Nicolas Papernot, Ryan Sheatsley, Raquel Alvarez, and Ananthram Swami. 2018. Detection under privileged information (Extended Version). *arXiv:arXiv:1603.09638*
- [11] Z. B. Celik and S. Oktug. 2013. Detection of fast-flux networks using various DNS feature sets. In *ISCC*.
- [12] Z. B. Celik, J. Raghuram, G. Kesidis, and D. J. Miller. 2011. Salting public traces with attack traffic to test flow classifiers. In *Usenix CSET*.
- [13] Z. B. Celik, R. J. Walls, P. McDaniel, and A. Swami. 2015. Malware traffic detection using tamper resistant features. In *IEEE MILCOM*.
- [14] N. M. Duc and B. Q. Minh. 2009. Your face is not your password face authentication bypassing lenovo—asus—toshiba. *Black Hat Briefings*.
- [15] G. Gu et al. 2008. BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security*.
- [16] A. Hassanzadeh et al. 2013. PRIDE: Practical intrusion detection in resource constrained wireless mesh networks. In *Information and Communications Security*.
- [17] G. Hinton, O. Vinyals, and J. Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- [18] C. Hsu, C. Huang, et al. 2010. Fast-flux bot detection in real time. In *RAID*.
- [19] G. B. Huang et al. 2014. *Labeled faces in the wild: Updates and new reporting procedures*. Technical Report UM-CS-2014-003. UMass.
- [20] S. Huang et al. 2010. Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection. In *ASIACCS*.
- [21] Vidit J. and E. Learned-Miller. 2010. *FDDB: A Benchmark for Face Detection in Unconstrained Settings*. Technical Report UM-CS-2010-009. UMass.
- [22] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik. 2015. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*.
- [23] P. McDaniel, N. Papernot, and Z. B. Celik. 2016. Machine learning in adversarial settings. *Security & Privacy Magazine*.
- [24] Microsoft malware classification challenge. 2017. <https://www.kaggle.com>. (2017). [Online; accessed 10-May-2017].
- [25] D. J. Miller et al. 2012. Sequential anomaly detection in a batch with growing number of tests: Application to network intrusion detection. In *MLSP*.
- [26] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang. 2011. A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In *Security and Artificial Intelligence Workshop*.
- [27] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. 2005. A first look at modern enterprise traffic. In *Internet Measurement Conference*.
- [28] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi. 2008. Fluxor: Detecting and monitoring fast-flux service networks. In *DIMVA*.
- [29] S. Raza, L. Wallgren, and T. Voigt. 2013. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad hoc networks*.
- [30] J. Reeves, A. Ramaswamy, M. Locasto, S. Bratus, and S. Smith. 2011. Lightweight intrusion detection for resource-constrained embedded control systems. In *Critical Infrastructure Protection*.
- [31] K. Scarfone and P. Mell. 2007. Guide to intrusion detection and prevention systems (IDPS). *NIST special publication*.
- [32] V. Sharmanska, N. Quadrianto, and C. H. Lampert. 2013. Learning to rank using privileged information. In *International Conference on Computer Vision*.
- [33] V. Vapnik and R. Izmailov. 2015. Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research*.
- [34] V. Vapnik and A. Vashist. 2009. A new learning paradigm: Learning using privileged information. *Neural Networks*.
- [35] L. Wolf, T. Hassner, and Y. Taigman. 2011. Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *Pattern Analysis and Machine Intelligence*.
- [36] S. Yadav et al. 2010. Detecting algorithmically generated malicious domain names. In *Internet Measurement Conference*.
- [37] G. Zou, G. Kesidis, and D. J. Miller. 2011. A flow classifier with tamper-resistant features and an evaluation of its portability to new domains. *JSAC*.

A USES OF PRIVILEGED INFORMATION

Privileged information is not restricted to the domains discussed above, but are readily adaptable problems and machine learning settings. With respect to detection, we consider several illustrative uses of privileged information below:

Mobile Device Security - The growth of mobile malware requires robust malware detectors on mobile devices. Current systems collect data for numerous type of attacks; however, exhaustive data collection at runtime can have high energy costs and induce noticeable interface lag. As a consequence, users may disable the detection mechanism [3]. We note that high-cost features can be defined as privileged information to combat this problem.

Enterprise Security - Enterprise systems use audit data generated from a diverse set of devices and information sources for analysis [6]. For instance, SIEM products collect data from hosts, applications, and network devices in incredible volumes (e.g., 100K events per second yielding to 42 TB of compressed data). These massive datasets are mined for patterns identifying sophisticated threats and vulnerabilities. However, systems may be overwhelmed by feature collection and processing at runtime which makes their collection impractical for many settings. In such cases, features involving complex and expensive data collection can be defined as privileged to balance the real-time costs and accuracy.

Privacy Enhanced Detection - Many detection processes require the collection of privacy-relevant features, e.g., pattern and substance of user network traffic, use of the software [8]. Hence, it is important to reduce the collection and exposure of such data—legal and ethical issues may prevent continuously monitoring them in their original form. In these cases, a set of features can be defined as privileged to eliminate the requirement of obtaining and potentially retaining privacy-sensitive features at runtime from users and environments.

B CALIBRATION OF APPROACHES

We discuss the required dataset properties, algorithm parameters, training, and runtime overhead of using privileged information for detection. We also present guidelines and cautionary warnings for use of privileged information in realistic deployment environments. A summary of an approach selection criteria is presented in Table 1.

Model Dependency. Model selection is a task of picking an appropriate model (e.g., classifier) to construct a detection function from a set of potential models. Knowledge transfer can be applied to a model of choice, as privileged features are inferred with any accurately selected mapping function. Distillation requires a model with a softmax output layer for obtaining probability vectors. However, we adapt model influence to SVM’s objective function.

Detection Overhead. The mapping functions used in knowledge transfer may introduce detection delays while estimating the privileged features. For instance, weighted similarity introduced in Section 2.1 defers estimation until detection without learning a function at training time (i.e., lazy learner). This may introduce a detection bottleneck if dataset includes a large number of samples. To solve this problem, we apply stratified sampling to reduce the size of the dataset. Furthermore, constructing mapping functions at training time such as regression-based minimize the delay of estimating privileged features. For instance, in our experiments, weighted-similarity is used to estimate ten privileged features of 5K training samples in less than a second delay on 2.6GHz 2-core Intel i5 processor with 8GB RAM. Regression reduces this value to milliseconds. Therefore, if delay at runtime is the primary concern,

Approach	Model dependency	Detection time overhead	Model optimization	Training time overhead
Knowledge Transfer	✗	✓	●	●
Model Influence	✓	✗	●	●
Distillation	✗	✗	●	●

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

Table 1: Guideline for approach selection.

we suggest using model influence and distillation for learning the detection model, as they introduce no overhead at runtime.

Model Optimization. To obtain the best performance results, the parameters and hyperparameters of approaches need to be carefully tuned. For instance, fine-tuning of temperature and imitation parameters in distillation and kernel function hyperparameters in model influence approaches may increase the detection performance. Similar to conventional detection, the number of parameters required to be optimized both for knowledge transfer and generalized distillation can be determined a priori based on the selected model. However, model influence has twice as many parameters as SVM—two kernel functions are used simultaneously to learn detection boundary in standard and privileged feature spaces. We apply grid search for small training sets and evolutionary search for large-scale datasets for parameter optimization.

Training Overhead. Training set size affects the time required by model learning. The amount of additional time needed to run both knowledge transfer and generalized distillation is negligible, as they require similar models as existing systems apply. However, the objective function of model influence may become infeasible or take a long time when the dimension of the feature space is very small, or dataset size is quite large. For instance, in our experiments, distillation and knowledge transfer train 1K samples with 50 standard and privileged features on the same machine used for one minute including optimal parameter search. Model influence takes on average 30 minutes on the same machine used for measuring detection overhead. Packages that are designed specifically for solving the quadratic programming (QP) problems (e.g., MATLAB quadprog() function) can be used instead of general solvers such as convex optimization package CVX to reduce the training time. Further, specialized spline kernels can be used to accelerate the computation [33].

C DETAILS OF DETECTION SYSTEMS

We detail the standard and privileged features of the detection systems introduced in Section 3.2. Table 2 presents feature categories and definitions of fast-flux bot detector obtained from recent works [18, 20, 28, 36], and Table 3 presents the features of malware traffic detector obtained from recent works [12, 13, 25, 37]. Figure 1 and Figure 2 present the standard and privileged features of the face authentication and malware classification systems. The reader can refer to the references for the motivation of the feature selection.

Category	Definition	Feature dependency	Feature type
DNS Answer	Number of unique A records Number of NS records	DNS packet analysis	standard set
Timing	Network delay (μ and σ) Processing delay (μ and σ) Document fetch delay (μ and σ) Edit distance	HTTP requests	standard set
Domain name	Kullback-Leibler divergence (unigrams and bigrams) Jaccard similarity (unigrams and bigrams)	Whitelist of benign domain names	privileged set
Spatial	Time zone entropy of A records Time zone entropy of NS records Minimal service distances (μ and σ)	IP coordinate database lookup (external source)	privileged set
Network	Number of distinct autonomous systems Number of distinct networks	WHOIS processing (external source)	privileged set

Table 2: Fast-flux bot detection system standard and privileged feature descriptions (μ is mean and σ is std. dev.).

Abbreviation	Definition	Properties	Feature type
cnt-data-pkt	The count of all the packets with at least a byte of TCP data payload	-TCP length is observed	standard set
min-data-size	The minimum payload size observed	-Client to server -TCP length observed	standard set
avg-data-size	Data bytes divided by the total number of packets	-Client to server -0 if there are no packets -TCP length observed	standard set
init-win-bytes	The total number of bytes sent in initial window	-Packets with payload observed -Server to client -0 if there are no packets -Retransmitted packets not counted	standard set
RTT-samples	The total number of RTT samples found	-Client to server & server to client -0 if no ACK observed	standard set
IP-bytes-median	Median of total IP packets	-Frame length calculated	standard set
frame-bytes-var	Variance of bytes in Ethernet packets	-Client to server -IP length calculated	standard set
IP-ratio	Ratio between the maximum packet size and minimum packet size	-Client to server -Frame length calculated	standard set
pushed-data-pkts	The count of all the packets seen with the PUSH set in TCP header	-Client to server -IP length calculated	standard set
goodput	Total number of frame bytes divided by the differences between last packet time and first packet time	-Client to server & server to client -1 If a packet observed, and if no packets are observed 0 is reported -Client to server & server to client	standard set
duration	Total connection time	-Frame length calculated	standard set
min-IAT	Minimum packet inter-arrival time for all packets of the flow	-Client to server -Retransmitted bytes not counted	privileged set
urgent-data-pkts	The total number of packets with the URG bit turned on in the TCP header	Time difference between the last packet and first packet (SYN flag is seen from destination) -Client to server & server to client	privileged set
src-port	Source port number	-Client to server & server to client	privileged set
dst-port	Destination port number	-Undecoded	privileged set
payload-info	Byte frequency distributions	-Undecoded -If not HTTPS at training time -If payloads are available -Client to server & server to client	privileged set

Table 3: Malware traffic detection standard and privileged feature descriptions.

Figure 1: Example of face authentication features. Original image for standard features (Left), cropped (Middle), and funneled (Right) images used for privileged features.

00401000 56 8D 44 24 08 50 8B F1 E8 1C 1B 00 00 C7 06 08	mov edi, [esp+0Ch]
00401010 BB 42 00 8B C6 5E C2 04 00 CC CC CC CC CC CC CC	jz short loc_401506
00401020 C7 01 08 BB 42 00 E9 26 1C 00 00 CC CC CC CC CC CC	mov esi, [ecx+18h]
00401030 56 8B F1 C7 06 08 BB 42 00 E8 13 1C 00 00 F6 44	lea eax, [ecx+4]
00401040 24 08 01 74 09 56 E8 6C 1E 00 00 83 C4 04 8B C6	cmp esi, 10h
...	mov edx, [eax]
...	...

Figure 2: Excerpt from hexadecimal representations (right), and assembly view (left) of an example malware. Selected byte bigrams and tokens for this malware is shown in boxes.