RiskTeller: Predicting the Risk of Cyber Incidents

Leyla Bilge
Symantec Research Labs
leyla_bilge@symantec.com

Yufei Han
Symantec Research Labs
yufei_han@symantec.com

Matteo Dell'Amico
Symantec Research Labs
matteo_dellamico@symantec.com

ABSTRACT

The current evolution of the cyber-threat ecosystem shows that no system can be considered invulnerable. It is therefore important to quantify the risk level within a system and devise risk prediction methods such that proactive measures can be taken to reduce the damage of cyber attacks. We present RiskTeller, a system that analyzes binary file appearance logs of machines to predict which machines are at risk of infection months in advance. Risk prediction models are built by creating, for each machine, a comprehensive profile capturing its usage patterns, and then associating each profile to a risk level through both fully and semi-supervised learning methods. We evaluate RiskTeller on a year-long dataset containing information about all the binaries appearing on machines of 18 enterprises. We show that RiskTeller can use the machine profile computed for a given machine to predict subsequent infections with the highest prediction precision achieved to date.

1 INTRODUCTION

Over the last two decades, the cyber-threat ecosystem faced dramatic changes. On the one hand, attackers now use sophisticated tools and techniques to breach systems [32]; on the other hand, the stakes at risk become larger every year. Experian reports that almost half of business organizations suffer at least one security incident per year [8]; despite the efforts in developing and defending secure systems, hardly anybody in the industry – especially in large organizations – can feel that their infrastructure is invulnerable. Rather than wondering *if* they would be victims of malicious actions, IT administrators are shifting towards trying to understand *when* it will happen, and what the consequences will be.

Since malware infections may be unavoidable, the problem of *predicting the risk* becomes fundamental: understanding which are the most risky parts of a given system allows to act proactively, and focus on hardening them; estimating the infection risk is fundamental in the rapidly growing area of cyber-insurance [27]. Cyber-insurance ensures that organizations and private users alike can mitigate the cost of otherwise potentially devastating attacks, but an efficient cyber-insurance market is only feasible if effective ways of estimating and predicting risk exist.

To date, the research community extensively studied various aspects of the malware problem mainly focusing on three topics: analysis, detection and prevention. However, only a few works presented prediction models [4, 16, 19, 30, 37, 38] that could allow proactive measures to be adopted to avoid the damage. These works analyzed the users' demographics [16, 38], network connectivity behavior [38], web browsing behavior [4, 38], website features [30], network mismanagement details [19] and historical incident reports of organizations [19, 37] to predict cyber incidents. We instead employ a dataset that provides fine graned information about the security posture of each enteprise machine and allows us to achieve higher prediction accuracy than previous works.

We present *RiskTeller*, a system that analyzes per-machine file appearance logs to predict which machines are at risk of getting infected by malware. The data is collected from 600K machines belonging to 18 enterprises. During our analysis, these machines reported a total of 4.4 billion binary file appearance events, resulting in the largest dataset used for risk prediction to date. We use this data to generate, for each machine, a profile of 89 features. Features are based on the volume of events, their temporal patterns, application categories, rarity of files, patching behavior, and past threat history. These features give us a comprehensive synthesis of each machine's usage patterns, and a good sense of the security awareness of its users. All together, these features play a fundamental role for the quality of the results produced by RiskTeller.

Our ground truth is based on observing malicious files and infection records according to the end point protection software installed in the period that follows feature extraction: we built it carefully, avoiding to create labels due to mislabeled files. In an ideal scenario where the labeled data is sufficiently comprehensive to capture all clean and risky machine profiles in the wild, supervised machine learning algorithms can achieve the best accuracy assuming that the features are selected appropriately. However, such an ideal ground truth is nearly impossible to obtain as no malware detection solution can attain perfection due to the known arms-race with the cyber attackers. Our ground truth leverages a large set of labeled files (to our knowledge the largest ground truth used so far with over 1B labeled files) which gives us the confidence that our results are reliable; we nevertheless question the quality of our ground truth and investigate whether there is still room for improvement. To address this problem, in addition to a traditional supervised machine learning algorithm, we devise a *semi-supervised* algorithm that attributes fuzzy labels to unknown user profiles based on their similarity to the labeled ones; this choice allows us to leverage on full information available in the dataset and enrich the ground truth when it is imbalanced or limited in size.

In summary, the contributions of our paper are as follows:

(1) We propose RiskTeller, a system that leverages both supervised and semi-supervised learning methodologies to predict which machines are at risk with highest accuracy achived to date.

CCS 2017, October 30-November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-4946-8/17/10…$15.00
https://doi.org/10.1145/3133956.3134022

(2) We design 89 features that are extracted from per-machine file appearance logs to produce *machine profiles*, which capture the machine's patterns of usage and security awareness of enterprise users.
(3) We design a *semi-supervised machine learning algorithm*, which leverages on profile similarity to infer fuzzy labels for unlabeled machines based on similar labeled machines to enrich the ground truth.
(4) We perform a comprehensive *evaluation* that shows how RiskTeller can predict which machines will get infected with high precision, and that the former two steps are fundamental in getting high-quality results.

2 WHY CYBER RISK PREDICTION?

For around a decade, the security community has been warning about an imminent "cyber crime era"; the constant stream of news about cyber security incidents, often reaching the front pages of mainstream media, shows that the cyber crime era finally arrived. Unfortunately, facing cyber attacks is now the norm rather than an exception, and it is hence necessary to set up proactive defenses. Businesses need to be prepared to minimize damage when attacks eventually strike; to this end, they need to deploy multiple layers of security including managed security services, trusted security advisors, employee training programs etc. in addition to traditional cyber threat defense mechanisms. Since this can be very expensive, businesses may need to prioritize; predicting the entities that are more likely to get attacked and ultimately infected is an important element on the prioritization step. For example, a special security training could be provided to the employees that are at higher risk of cyber attacks through malware sent via email, such that when they receive mail from attackers they do not open the attachment.

Recently, several security companies started to incorporate cyber insurance into their multi-layer cyber security approach, to ensure that the recovery after cyber attacks is less painful. Due to the high demand for cyber insurance, the market has been steadily growing and putting the insurance companies in a great competition to assess and predict the risk the most accurately. Typically [2], risk assessment for cyber insurance uses underwriting tools whose main goal is to assess the risk of a company through questionnaires with the least possible questions, or via publicly available data (obtained, e.g., through active scanning or by looking for the IP addresses of a company in blacklists). In both cases, the information available to accurately assess a company's security posture is limited; hence, insurance companies have a significantly lower accuracy than what can be achieved using internal security telemetry. Similarly, in other insurance fields the accuracy of risk assessment that has been widely adopted is quite low [7, 10, 28] (around 70% prediction accuracy); this is why insurance companies have been seeking for better risk prediction methodologies that leverage private information obtained from the insured entities (e.g., fitness tracking data for health insurance, driving habits data collected through special hardware for car insurance, etc.). In our work, we analyze internal telemetry collected from companies to predict which computers are most at risk, and thanks to the profiles we build for each computer, we are able to achieve high prediction accuracy.

We need to emphasize that predicting future events is a different and more difficult problem than detecting current malicious events. For detection, false positives can be very expensive (e.g., a user may not be able to perform their job if an essential software is erroneously recognized as malware); the goal is hence maximising the true positive ratio while keeping the false positives very low. On the other hand, for prediction the main goal is quite the opposite: an enterprise would want to know all the machines that *could* be infected, to apply appropriate hardening measures or provide security training to users; compared to the detection domain, false positives are more difficult to avoid, but the cost of false positives is lower. Previous works in the prediction field produced more than 20% false positives to predict over 95% of the incidents correctly [19]; these numbers are perfectly acceptable for insurance companies. However, the competition in the market raises the bar and asks for lower false positives. Our work aims at meeting this expectation.

3 DATASET

Our work builds its foundations by mining large-scale data that helped us discover interesting behavioral differences between clean machines and those that are more likely to get infected by malware.

The first dataset we employ consists of reports for the appearance of new binary files (e.g., due to file downloads or in-house file creations after compilation), generated by enterprise customers of a large antivirus company who opted in to share their data, such that through large-scale data analysis new methodologies can be developed to increase the existing malware detection capabilities. To protect customer identities, all sensitive information such as customer id, IP addresses, and enterprise names are anonymized.

The binary file appearance logs are collected from more than 100K enterprises. Every day, the data centers receive reports about around 100M file appearances of 14M distinct binaries. We were able to obtain only a subset of this data, covering 4.4B binary appearance logs of 600K machines belonging to 18 enterprises: even if its size is considerably smaller than the full dataset processed by the AV company, our results show that it is sufficient to model the file appearance patterns of Internet users and to accurately predict cyber threats several months in advance.

The fields we are particularly interested in from the binary file appearance logs are: *(a)* (anonymized) enterprise and machine identifiers, *(b)* SHA2 file hash, *(c)* file name and directory, *(d)* file version, *(e)* timestamps for the first appearance of the file on the machine (local timezone) and for the time when it was reported to the data centers (PST), and *(f)* file signer subject in the certificate.

3.1 Data Preprocessing

As discussed earlier, the data fields we include in our analysis are enterprise and machine identifiers, file hash, name, directory, and version, timestamps for file appearance and report, reputation and name of the file signer. To extract better value from this information, we perform some data normalization and cleaning.

We normalize file and directory names to identify those likely to perform the same functions. We remove version numbers through simple regular expressions (version numbers in general appear in filenames as groups of numerical characters separated by dots with occasional alphabetical characters, e.g., "`v2.45.7r5`"), and also remove suffixes generally appended to duplicate files, such as numbers wrapped inside brackets ("`()`", "`[]`", and "`{}`").

Unfortunately, our datasets do not include specific information about which applications binary files belong to: we therefore resort to directory name to infer them. In Windows systems, applications use the CSIDL (Constant Special Item Id List)[1] to identify the name of special folders in each Windows installation, and our datasets in a majority of the cases include normalized directory names (we normalize the remaining following the guidelines of CSIDL) rather than the full path including variable information such as drive identifiers, usernames, and OS-specific strings. Applications are installed in the `CSIDL_PROGRAM_FILES` directory, and our heuristic to identify an application is to use depth-3 paths starting from that directory to identify applications; for example, the directory corresponding to Google Chrome is `\CSIDL_PROGRAM_FILES\google\chrome`. This is the string that we use to recognize applications.

3.2 Ground Truth

We split our datasets in two consecutive periods: a first *feature extraction* period from which we compute features that will be fed to our classifier, and a *labeling* period from which we identify a ground truth of "clean" and "infected" machines. Riskteller performs prediction in a temporal sense, meaning that the classifier only uses features from the feature extraction period to predict labels that are based on events happening in the successive labeling period.

In addition to the binary appearance logs dataset, we leverage three more datasets that we use to build our ground truth. As we discuss in Section 5, our predictive model leverages supervised and semi-supervised machine learning techniques to differentiate the clean machines from those that are likely to encounter malware infections. As the quality of machine learning techniques strongly depends on the quality of the ground truth, we endeavor to define clean and risky machine profiles in a way that minimizes the probability of errors.

We build our ground truth using three different datasets:

(1) A labeled dataset we obtained from the AV company at the end of 2015, consisting of 16M known benign and 214M known malware file hashes; note that this dataset is updated continuously whenever new labels are acquired.
(2) A dataset of file hashes that were identified as malware according to the AV product, and that has never been manually exonerated (i.e., marked as false positive) by users. By the end of 2015 the AV telemetry dataset, which consists of detailed reports about the type of malware the files are associated with, contained 800M hashes. The fact that these files were never marked as false positives gives us confidence that very few of these files are actually benign.
(3) A dataset of malware infections detected through the network activity of machines. For example, if the machine initiates a command-and-control activity with known Zeus C&C servers, the machine is flagged as infected. We obtain the network-based infection data from the telemetry dataset generated by the IPS product of the company.

In Section 6.1.2 we describe in more detail how we define thresholds to identify clean and risky profiles; here we limit ourselves to a brief summary. We define as "clean" a machine with a very limited number of unknown files, zero files known to be malware, and no infection record. On the other hand, we consider "risky" a machine that, during the labeling period, has multiple known malware files in its file appearance logs and/or records of infections according to the IPS telemetry dataset.

4 BUILDING THE MACHINE PROFILES

As the main goal of our work is to be able to predict which machines are most likely to get infected, we perform an in-depth investigation on the binaries appearing on machines, with an effort to identify specific behaviors that predict future infections. We do not seek to pinpoint the exact *causes* of infections, but rather characteristics that are *correlated* with them and that are useful to represent how the machine is used; hence, many features we analyze are chosen to represent the security awareness of the machine's users, and the usage patterns of those machines. For example, we extract a number of features about the patching behavior with respect to a limited number of applications. While the direct cause of a particular infection could be a vulnerability of an application which is not included in our analysis, a higher-level explanation would be that the user(s) of the machine do not patch existing vulnerabilities promptly enough.

As mentioned before, the binary appearance logs consist of metadata collected from the end-hosts regarding all new binaries that appeared during the period of analysis. While in general a majority of these binaries are downloaded from various sources, some of them could be generated on the host itself: some benign examples are binaries compiled on the system by users who are application developers, or those created by specific applications such as web servers. Unfortunately, our datasets do not allow us to distinguish which binaries are downloaded or copied from other sources, and which ones are created on the computer; however, our category-based features (Section 4.1.4) allow us to distinguish machines based on the type of applications installed, capturing implicitly the main reason of binary appearance. For the sake of brevity, hereinafter, we will refer to all binary downloads/installations/compilations/creations as binary appearance events, or simply events.

In the following, we explain how we preprocess data to prepare for the feature extraction step and provide details about the features we use for prediction.

4.1 Feature Discovery

After the data preprocessing step, we obtain a list of events, each representing the appearance of a new binary on a machine during our analysis window. For each machine, we create a profile consisting of 89 different features synthesized from these events, which we use to predict each machine's risk of future infection. Table 1 provides a comprehensive list of these features, grouped by categories. In the following, we describe each category.

4.1.1 Volume-Based Features. Our first category of features covers general statistics calculated from new binaries appeared during our analysis window. We include the total number of events (i.e.,

[1] https://msdn.microsoft.com/en-us/library/windows/desktop/bb762494(v=vs.85).aspx

Table 1: RiskTeller features for predictive modeling.

Feature Category	Feature #	Features
Volume-based (§ 4.1.1)	1–3	# of events, # of distinct file hashes/filenames
	4–6	fraction of events from top signers/top file hashes, average # of events per active day
	7–12	# of distinct applications, quartiles of per-application fraction
Temporal (§ 4.1.2)	13–17	fraction of events during daytime/evening/night/weekdays/weekends
	18–19	diurnal # of events: median/standard deviation
	20–21	monthly # of events: median/standard deviation
Vulnerabilities/patching (§ 4.1.3)	22–24	# of patched vulnerabilities/applications, the most patched application
	25–29	quartiles of CVSS scores for patched vulnerabilities
	30–34	quartiles of the vulnerability window length for patched applications
	35–37	# of vulnerabilities, unpatched applications, app with highest vulnerability count
	38–42	quartiles of CVSS scores for unpatched vulnerabilities
	43–47	quartiles of the vulnerability window length for unpatched applications
Application categories (§ 4.1.4)	48–52	top-5 application categories with most events
	53–57	fraction of events per top-5 category
	58	fraction of system diagnostics tools
	59	fraction of system administration tools
	60	fraction of attack tools
Infection history (§ 4.1.5)	61–63	fraction of events for malicious/benign/unknown files
Prevalence-based (§ 4.1.6)	64	fraction of events with singleton signers
	65–69	fraction of events with prevalence [1, 10]/[11, 100]/[101, 1000]/[1 001, 10 000]/[10 001, ∞) signers
	70	fraction of events with signers seen in only one enterprise
	71–74	fraction of events with signers seen in [1, 10]/[11, 100]/[101, 1 000]/[1 001, ∞) enterprises
	75	fraction of prevalence-1 files
	76–79	fraction of prevalence [1, 10]/[11, 100]/[101, 1 000]/[1 001, ∞) files
	80	fraction of files seen only in one enterprise
	81–84	fraction of files seen on [1, 10]/[11, 100]/[101, 1 000]/[1 001, ∞) enterprises
	85	fraction of files seen only on one machine
	86–89	fraction of files seen on [1, 10]/[11, 100]/[101, 1 000]/[1 001, ∞) machines

appearance of a new binary on a machine), of distinct binaries (recognized through SHA2 file hashes), filenames, and of applications on each host.

We identify the most popular software vendors and the most frequently appearing binaries by identifying the 50 most frequently appearing file signers and 150 most frequent file hashes respectively querying the whole data; we then compute the fraction of events pertaining to these top signers and hashes. To capture the level of activity while the machine is used, we also compute the average number of binary appearance events during each day in which events were generated.

Previous work has shown that people with abundant and varied browsing behavior suffer higher risks [4]: to identify similar patterns with respect to binaries appearing on machines, we compute the percentage of events generated by each application (identified as described in Section 3.1), and synthesize these values through 6 values: the number of distinct applications, plus the 5 quartiles of the per-application percentage of events: minimum, maximum, median, plus 25th and 75th percentiles.

4.1.2 Temporal Behavior. We aim to understand whether longer working hours or working outside the official working hours is correlated with facing higher risk to encounter malware infections. Our hypothesis is that people who generally use their machines during weekends or in the evenings are more likely to use their machines for personal, more varied purposes in addition to work-related ones, possibly engaging in riskier activities that might result in malware infections. As we will show in the following, it is indeed the case that machines with more binary appearance events during night time are more risky.

The features we extract are the fraction of events that happen during the day, evening, night, weekdays and weekends (notice that the timestamps we use correspond to local time, therefore timezone differences do not affect the accuracy of these features). We consider daytime as the 06:00–18:59 interval, evening as 19:00–00:59, and night as 01:00–05:59. According to the customs of the majority of the world, we define weekdays as Monday to Friday, and weekends as Saturdays and Sundays. Additionally, to capture the regularity or irregularity of machine usage in time, we compute the median

Table 2: Applications with vulnerable versions identified.

Vendor	Product	# CVE IDs
Adobe	Air	128
	Flash Player	3 708
	Reader	261
Google	Chrome	806
Microsoft	Internet Explorer	1 018
	Silverlight	36
	Skype	28
Mozilla	Firefox	9 536
Oracle	MySQL	108

and standard deviation for the number of events observed each day (diurnal) and each month (monthly).

4.1.3 Vulnerabilities and Patching Behavior. Based on the history of binaries that appear on each machine, we can infer when they have installed vulnerable application versions, and extract features about the severity of vulnerabilities and the *vulnerability windows*: time intervals during which machines are running software with known vulnerabilities. As common sense suggests, the vulnerability patching behavior and the severity of existing vulnerabilities on machines can be highly correlated with the probability of future malware infections. Indeed, our results support this, as we find that not patching vulnerabilities, besides indicating low security awareness, is a very good predictor of future infection risk.

Our analysis includes known vulnerabilities for 9 different applications, described in Table 2. As discussed in the following, the process of matching vulnerability information with data from national vulnerability database (NVD) and our file appearance logs data requires a non-negligible manual effort; therefore, we focus on applications which are widely installed, and often exploited through vectors such as drive-by-downloads, e-mail attachments, etc. [23]. A comprehensive overview of *all* vulnerable applications installed on a machine would be essentially unfeasible with the datasets we have; we consider, however, that identifying the patching behavior with respect to the widely used applications listed in Table 2 is sufficient to understand and capture user behavior, and that their patching behavior for other applications is likely to be similar to the one for the applications we consider.

Following the spirit of the work by Nappa et al. [22], we first identify software through a manually defined *leading filename* combined with information about the file signer (e.g., we consider that Google Chrome is identified by binaries named `chrome.exe` and signed by Google); we then obtain file version information from the file logs and/or VirusTotal.[2] We obtain information about vulnerabilities through the NIST's National Vulnerability Database (NVD).[3] By parsing NVD data, we obtain information about vulnerable file versions and the severity of each vulnerability through the *CVSS* (Common Vulnerability Scoring System) score,[4] which ranks vulnerabilities through scores between 0 and 10, where high-severity vulnerabilities have scores of at least 7. We manually resolved inconsistencies between version numbers in the file appearance logs or VirusTotal and those in the NVD database.

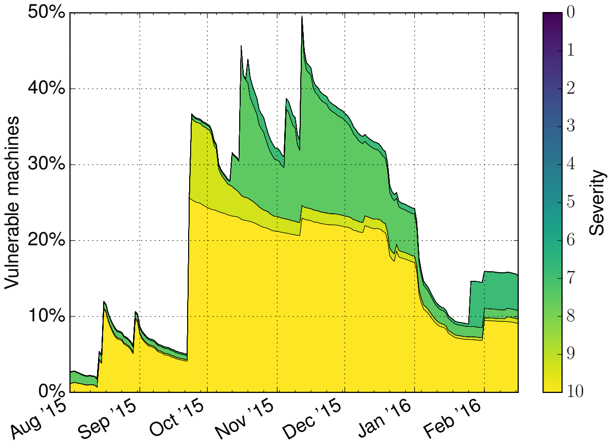


Figure 1: Fraction of vulnerable machines and their severity for a single company. In this period, a large fraction of severe vulnerabilities were due to Adobe Flash.

Thanks to this process, we are able to identify the software versions installed on each machine and known vulnerabilities of those pieces of software. In many cases, severe vulnerabilities on widely installed software (e.g., Adobe Flash) result in large numbers of vulnerable machines, and vulnerable software versions often require up to months to become updated (see Figure 1).

Once we are able to identify vulnerable versions of applications together with the time they were installed and updated, we extract 26 features about the severity of patched and unpatched vulnerable applications we observe in the log, the time it takes for the user to patch known vulnerable applications and the window of exposure for the remaining unpatched vulnerabilities. Details about each feature can be found in Table 1: we remind that with quartiles, we refer to minimum, maximum, median, plus 25th and 75th percentiles.

4.1.4 Application Category-Based Features. We extract a set of features related to the categories of applications seen on machines, based on the intuition that these categories can be used for machine profiling, and different machine profiles suffer different risk levels from cyber-threats. Some previous works have already touched upon machine profiling in other contexts: [36] assessed the risk of different job profiles with respect to targeted attacks; [22] focused on the vulnerability patching behavior of three user profiles (software developers, professionals, security experts); [24] described a correlation between cyber-attacks and three predefined user profiles (professionals, software developers and gamers). Motivated by these works, we perform a detailed analysis to obtain a set of machine profiles corresponding to the type of applications installed: our goal is to understand which specific machine profiles are more

[2] https://www.virustotal.com/

[3] https://nvd.nist.gov/

[4] https://nvd.nist.gov/cvss.cfm

prone to encounter cyber-attacks, and whether features including the most downloaded application categories on machines can be useful for our predictive model.

Table 3: Application categories.

Category	# of Apps	Category	# of Apps
Architecture	59	Government	142
Asset Management	574	Health	1 243
Automobile	172	HR	796
Bank	166	Insurance	246
Business	1 266	IT	353
Chat	87	Legal	547
Chemical	29	Logistics	146
Construction	371	Oil	145
Sales	1 050	Point of Sale	251
Data / DB	254	SDK	490
Education	101	Secretary	100
Engineering	73	Security	294
Finance	1 206	Statistics	71

We first match application categories to the files downloaded on each machine: to this end, we create a ground truth of over 10K applications that fall into 26 different categories. Table 3 lists the details about the application categories we incorporated to our analysis; we obtained this comprehensive ground truth that covers different types of job roles by manually querying Capterra[5] to retrieve the list of most popular applications in each category.

Our dataset, unfortunately, does not provide information about application names; therefore, we follow a simple heuristic, which we validated through manual checking. For each application, Capterra provides a vendor name together with an application name: to match these applications within our dataset, we matched through regular expressions the application name with the installation directory, and the vendor name with the file signer subject. Through this matching, we are able to identify the 5 application categories representing most events on each machine, and their frequency. We use this data as features.

In addition to the aforementioned job-related application categories, we also include three application categories that are often abused by attackers for malevolent purposes such as gathering information about compromised hosts, cracking passwords, exfiltrating sensitive data, etc. We focus on these (legitimate) applications due to a growing number of advanced threats that incorporate system administration or diagnosis tools in various stages of their attacks: for example, in an attack worth millions of dollars of stolen credit card data from the Target supermarket chain, the attackers used popular network sniffers to steal credit card data, and other legitimate tools such as `ftp` and `scp` to exfiltrate it [15]. Motivated by such cases, we give a closer look at the machines that have tools that can be useful for attackers, and understand whether having these tools could be correlated with the risk of future infections.

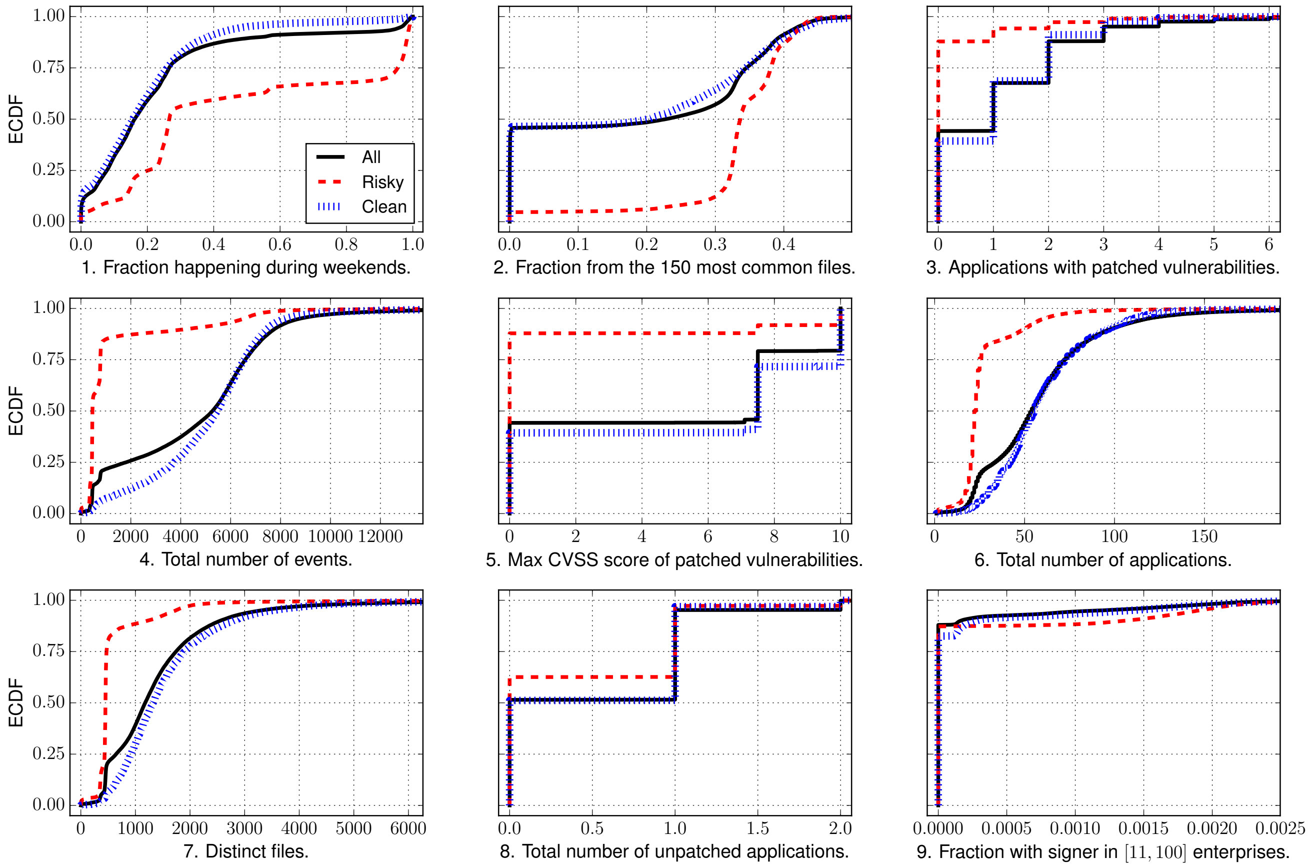
Based on online documentation about cyber attacks similar to the aforementioned Target case, we collected a list of 115 applications used by attackers or similar to them. For example, knowing that `ftp` is used for data exfiltration, we also add `sftp`, `scp`, `pscp` and `winscp` to our list since they can be used in the same way. The first category of applications include 18 *system diagnosis* tools such as `ping`, `netstat`, `diskinfo`, `tcpview`, `whois`, `dig`, etc. The second category consists of 64 *system administration* tools such as data transmission tools, device scanners, IP port scanners, penetration testing tools, remote access applications and sniffers. The last category, *attack tools*, includes applications that are either directly attack-related or can help the attacker achieve more than the previous two category of applications. For example, pass-the-hash attack tools, man-in-the-middle attack tools, password crackers, exploitation tools, hijacking tools, tools to perform dictionary attacks belong to this category. We identify these tools employing the same heuristic discussed before for the other type of application categories. Once we identify the files that belong to these critical category of applications, we keep the fraction of files in each category as features.

4.1.5 History of malware and goodware events. As discussed earlier, we split our year of data in a period for feature extraction and a subsequent one for labeling; in the latter labeling period we derive our ground truth to distinguish "clean" and "infected" machines. It is reasonable to imagine that past infection history is correlated with future events; to evaluate that, we include a set of features about hashes of known benign/malicious files in the feature extraction period. Perhaps surprisingly, our results show that these features are within the least informative ones.

4.1.6 Prevalence-Based Features. If one looks at the binaries installed on a single machine, a majority of them are in common with many other machines, and they are signed by a rather small number of different vendors. Indeed, low-prevalence files and signers are an indicator of file suspiciousness: while there are legitimate and benign reasons they can be generated even in large numbers (e.g., files compiled on a given machine), malware tends to have lower prevalence than benign software. Moreover, one of the best ways to label an unknown file is to consider it similar to the files it co-occurs with [5, 34]: for low-prevalence files this is intrinsically difficult, since the low number of machines they are installed on causes low confidence on results. In particular, it is very difficult to clearly label prevalence-1 (or "singleton") files, which occur exactly once on a single machine. In our context, the fact that a machine has a large number of low-prevalence files gives us reasons to be suspicious about that machine.

For each event, we compute *(i)* the number of events and enterprises in which the file signer is seen, *(ii)* the number of events in which the file hash is seen, *(iii)* the number of enterprises and machines in which the file hash is seen. Based on this information, we bucket these values and create 26 features as reported in Table 1. As we shall see in the following, a larger number of files that are rare or created by rare signers entails a larger risk.

[5] http://www.capterra.com/



1. Fraction happening during weekends.

2. Fraction from the 150 most common files.

3. Applications with patched vulnerabilities.

4. Total number of events.

5. Max CVSS score of patched vulnerabilities.

6. Total number of applications.

7. Distinct files.

8. Total number of unpatched applications.

9. Fraction with signer in $[11, 100]$ enterprises.

Figure 2: Empirical cumulative distribution functions (ECDFs) for the numerical features considered as most significant in the feature selection step, ranked by decreasing importance. We differentiate user profiles marked as risky, clean, and the overall feature distribution. "Fraction" refers to the fraction of events per machine having the given property.

4.2 A Look at the Dataset

We now observe the feature distribution of our dataset; for obvious space reasons, it is impossible to report them for all the 89 features listed in Table 1. In Figure 2, we show the overall cumulative distribution functions (CDFs) of the 9 most relevant features reported in Table 5, along with the distribution observed for machine profiles marked as either risky or clean; the feature labeled as 1 is the most important. We remark that these are not necessarily the 9 features that exhibit the most differences between infected and clean users but rather the ones that, when considered together, best allow separating infected machines from clean ones.

In general, the overall distribution and that of clean profiles are similar; on the other hand, infected users behave differently: hence, we can see the "signature" of risk in the behaviors that are more consistent with risky users.

From plot 4 in Figure 2, we infer that risky users install substantially less binaries on their machines than other users. Plots 2 and 6 corroborate this by showing that, on machines with risky profiles, many files are very common and the total number of applications is lower: it appears that machines that remain unused are more at risk. Machines at risk are also updated very rarely, if ever: even if the number of vulnerabilities on machines with risky profiles is comparable to others (plot 8), those vulnerabilities are almost never patched on machines with risky profiles (plots 3 and 5). Two other characteristics associated with risk are higher usage during weekends(plot 1), and a small but significant fraction of cases where files are signed by entities appearing in few enterprises (plot 9).

The overall picture that emerges from this analysis is that risk is strongly correlated with machines that are likely to be installed and then forgotten, sitting unused with vulnerable and unpatched operating systems and/or applications. Other risk patterns we observe are higher usage during weekends, and in some cases "weird" files signed by rarely seen vendors.

This picture corroborates the idea that our feature can distinguish features that can help us isolate the characteristics of risky behavior. In the following, we describe the algorithms we designed to assign risk scores to machines.

5 PREDICTIVE ANALYTICS

The features defined in the previous section result in machine profiles that are the inputs of our machine learning algorithms. The first algorithm is a commonly used supervised machine learning classifier: Random Forest Classifier. The second algorithm performs semi-supervised learning such that even with ground truths that are unbalanced and/or limited in size we can still predict future malware infections. Before going into the details of these two algorithms, we note that we transform the category-based features that are not numerical through *one-hot* (or *one-out-of-k*) encoding. A categorical feature having the i^{th} of k possible values becomes a set of k binary features where the i^{th} value is set to 1, and all others are set to 0.

5.1 Random Forest Classifier

Random Forest Classifiers (RFCs) are ensemble learning machines consisting of several decision trees which output a weighted vote of the decision output of each individual tree [3, 11]. The classifier aims at reducing the variance of the learning model through bias-variance trade-off. We use RFCs due to the following merits:

1. RFCs can handle categorical and numerical features and do not require feature normalization.
2. They behave well with new and previously unseen testing data, providing unbiased estimates of the generalization error; hence, they give a good approximation to the true classification boundary. As reported on many datasets, they produce accurate regression and classification outputs.
3. Trees are helpful to provide an intuitive understanding on how the prediction is made.
4. The output of RFCs is an intrinsically well-calibrated probability that the model assigns to belonging to a given class.
5. Most importantly, RFCs are intrinsically scalable and run very efficiently on large-scale datasets similar to ours.

We run the RFC with 800 trees, chosen, as Liaw and Wiener [18] advise, as the threshold where improving the number of trees does not improve classifier accuracy.

5.2 Semi-Supervised Learning

While supervised machine learning algorithms perform better with balanced ground truths having enough labeled elements, semi-supervised learning (SSL) algorithms excel when the ground truth datasets are unbalanced and/or small. Preparing a good ground truth in many cases requires manual labeling or relies on some automated tools that might result in faulty labels. The results of previous works [21, 25, 29, 33] on several security problems demonstrated SSL's promising performance in reducing manual labeling overheads and preserving classification accuracy.

We propose a novel SSL-based inductive learning engine for RiskTeller, which conducts classifier training to estimate ground truth and improve the classifier's prediction accuracy at the same time. Our SSL module is fed with n-dimensional *feature vectors* $\{X_i \in \mathbb{R}^n : i \in 1, \ldots, m\}$ describing user-behavioral profiles on m machines. Without loss of generality, we assume that the first l of them are labeled: $R_i = 0$ if profile X_i is risky, $R_i = 1$ otherwise. Here, we show how we build a risk prediction model F such that the *risk score* $F(X_i)$ quantifies the infection risk for machine i.

We establish our design of the risk score on two principles. First, *risk scores are bounded in* $[0, 1]$ *and continuous*: a value of 1 indicates that an unquestionable evidence of infection is detected on the machine; a value of 0, conversely, indicates that the machine is free from any potentially malicious files, because only known benign files have been observed on the machine. Between these extreme situations, risk score measures the likelihood of infection on a machine for which we do not have conclusive evidence of being either clean or infected: those with larger scores are more likely to be, or eventually become, infected. The second principle is that *similar user profiles yield close risk scores*: if two feature vectors X_a and X_b are close to each other in the feature space, we infer that those machines are used in a similar way, and hence they will have similar risk scores.

We define the risk score as $P_i = P(\text{infection}|X_i)$, the posterior probability of becoming infected given the feature vector X_i. Being a probability, this value is bounded in $[0, 1]$ and continuous, with extreme values corresponding to unquestionable indicators of infection or cleanliness; higher values correspond to higher infection risks, consistently with the first principle outlined above.

We implement the second principle through an optimization framework. We seed the system through the "clean" and "infected" labels of Section 6.1.2, which we use as *a priori* knowledge by assigning to them risk scores of respectively 0 and 1; we then propagate risk scores to all user profiles based on similarity between feature vectors, optimizing an objective function

$$C_P = \sum_{i,j} w_{i,j} \left(P_i - P_j\right)^2 + \alpha \sum_i \left(P_i - 0.5\right)^2 \tag{1}$$

constrained by $P_i = R_i$ for the labeled profiles $i \in 1, \ldots, l$, and where $w_{i,j}$ is the similarity between feature vectors X_i and X_j. The first term of Equation 1 enforces that the risk scores of similar profiles should be as close as possible, enforcing our second principle. The last term regularizes the risk score distribution, according to maximum entropy theory [31]: it encourages risk scores to be as distributed as possible on the $[0, 1]$ axis while respecting the first term and the constraints on labeled items, avoiding degenerate solutions where all or most P_i items are close to 0 or 1. Our chosen P_i^* values are therefore those that minimize C_{P^*}:

$$P^* = \operatorname*{argmin}_P C_P \text{ s.t. } P_i = R_i \forall i \in 1 \ldots l. \tag{2}$$

We solve this problem using the alternative minimizing procedure [31]: we introduce a pseudo-variable $\{Q_i\}$ $(i = 1, \ldots, m)$ resulting in a joint optimization problem with respect to P and Q:

$$\begin{gathered} P^*, Q^* = \operatorname*{argmin}_{P,Q} \sum_{i,j} w_{i,j} \left(P_i - Q_j\right)^2 + \alpha \sum_i \left(P_i - 0.5\right)^2 \\ \text{s.t. } P_i = Q_i = R_i \forall i \in 1, \ldots, l. \end{gathered} \tag{3}$$

Our solution is to update P_i and Q_i where $i > l$ alternatively until convergence, giving an iterative procedure composed by two successive steps shown in Equations 4 and 5.

$$Q_i^n = \frac{\sum_{j \neq i} w_{i,j} P_j^{n-1}}{\sum_j w_{i,j}} \text{ if } i > l, R_i \text{ otherwise.} \tag{4}$$

$$P_i^n = \frac{\sum_{j \neq i} w_{i,j} Q_j^n}{\sum_j w_{i,j}} \text{ if } i > l, R_i \text{ otherwise.} \tag{5}$$

The starting scores are

$$P_i^1 = Q_i^1 = R_i \text{ if } i \leqslant l, 0.5 \text{ otherwise}. \tag{6}$$

We finally learn function F through a random forest classifier consisting of 800 trees. We train the classifier to fit F such that P_i^* is the sample weight of the corresponding unlabeled user profile X_i. From Equation 5 we see that, if profile i is not labeled, we set it as a weighted average of the other values X_j, where closer values have a higher impact. To guarantee that the output of F is in the $[0, 1]$ range, we adopt the soft-voting based probabilistic output of the random forest, which is the mean predicted class probability of all the trees in the forest. As such, given the usage profiling features X_i of each user, $F(X_i)$ is the final risk score that we derive. It is noteworthy that the probabilistic output of the random forest doesn't require additional calibration, unlike support vector machines or boosting. It is defined as the mean predicted class probability for each tree in the forest. For each tree, the class probability is the fraction of samples of that class that drop into the same leaf. When the training set is large enough, the output probability asymptotically approaches the true probability of class membership.

6 EXPERIMENTS AND RESULTS

We evaluate RiskTeller through an extensive set of experiments. First, we analyze RiskTeller's parameters (Section 6.1) to choose the best settings; then, we evaluate its ability to predict machine infection (Section 6.2). We then study the significance of features and feature categories (Section 6.3), and the significance of the semi-supervised risk prediction algorithm we propose, in particular when few items are labeled for the ground truth (Section 6.4).

6.1 RiskTeller Parameters

RiskTeller has two key sets of parameters that we evaluate here: the first regards M and N, the length of the periods we use for feature extraction and labeling; the second regards the thresholds used for setting the ground truth labels. Here, we show the effects of these parameters, and how we set them for the rest of this analysis.

6.1.1 Feature Extraction and Labeling Period Length. We use our data for two different goals: feature extraction and assigning labels. Since our goal is predicting infections that occur *after* we extract features, we separate data in two consecutive periods: the *feature extraction* period, based on which we create the machine profiles described in Section 4, and the *labeling* period, based on which we extract the ground truth labels that we attempt to predict with RiskTeller through the machine profile information.

We fragment our year of data in two consecutive periods, where the feature extraction period lasts M months and the labeling period lasts N months. As our dataset has one year of data, we are constrained by $M+N \leqslant 12$; when $M+N < 12$, we prepare different datasets starting at the beginning of each month. For example, for $M = 6$ and $N = 4$, we create three different datasets with feature extraction (labeling) periods of respectively January–June (July–October), February–July (August–November), and March–August (September–December).

We aim to investigate the impact of M and N on risk prediction performance, and to choose their optimal setting; we organize the experiment as follows. For each (M, N) pair such that $M + N \leqslant 12$, we recover the datasets created above and identify a ground truth of clean and risky profiles based on their behavior in the labeling period. We split the labeled users into a labeled training set L and a validation set V. We then build the risk prediction model using both L and all the unlabeled user profiles U, as described in Section 5. After that, we apply the prediction model on the validation set V to test accuracy. We repeat the split of L and V as a 10-fold cross-validation process.

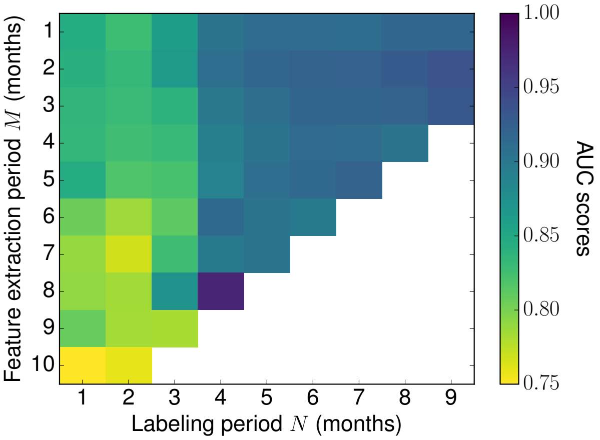


Figure 3: AUC scores obtained with different M and N values.

Figure 3 shows how AUC varies as a function of M and N. We can see that, in general, longer values of both M and N lead to higher AUC scores: larger time windows improve the stability of the statistical summary of user profiles and accumulate more solid evidence for identifying the ground truth labels, which in turn leads to higher AUC scores. Nevertheless, if more months are used for feature extraction, less months are used to identify the ground truth labels; as a result, the identified ground truth labels are more likely to miss some challenging and ambiguous user profiles, which are located near the classification boundary. Although the AUC score grows if these ambiguous user profiles are excluded from the training / testing process, this carries the risk of over-fitting in our risk prediction model. We remark that AUC scores are high and rather stable for $N \geqslant 5$ and $6 \leqslant M \leqslant 8$: this corroborates a conclusion stating that M and N should be similar in size, because a larger M increases the stability of feature extraction, and larger N increases the stability and quality of the labeling process. Therefore, in the remainder of our experiments we will set $M = N = 6$, which consistently provides good results.

6.1.2 Thresholds for the Ground Truth. Our ground truth relies on the detection capabilities of existing anti virus and intrusion prevention products of the AV company, hence we have to take into account the error rate of the identified infections, and the fact that sometimes the same file is detected multiple times (e.g., cases where users decide not to delete a file marked as malicious). Adding a machine profile to the set of risky ones because the machine was detected as infected a few times during the labeling period (e.g., 6 months) might result in wrong assessments. Here, we show the

Table 4: AUC varying the ground truth thresholds.

T_{inf}	T_{grey}	AUC	Machines	
			Risky	Clean
10	0	0.965	21 690	10 332
	3	0.968		14 638
50	0	0.978	16 393	10 332
	3	0.981		14 638
100	0	0.981	**14 272**	10 332
	3	**0.983**		**14 638**

impact on results of the thresholds we use to identify clean and risky profiles for our ground truth.

As discussed in Section 3.2, our ground truth comes from three large datasets including known malware files and list of malware infection records from IPS telemetry. We behave conservatively, avoiding to add machines to the ground truth if there are any reasons to doubt about the fact that they're either infected or clean.

We define a machine as clean if it has neither any infection records in the IPS telemetry dataset nor any files known to be malware. To avoid misclassifications, we also consider as clean a machine that has at most T_{grey} unlabeled files. As seen in Table 4, we used very strict thresholds to decrease the likelihood of labeling an infected machine as clean due to false negatives in our data. We behave in an analogously conservative way to label risky users: we mark a profile as risky only if it is associated with at least T_{inf} malicious events. To limit the risk of false positives, we again experiment with conservative values in Table 4. We achieve the best classification accuracy with $T_{\text{grey}} = 3$ and $T_{\text{inf}} = 100$, the thresholds we use for the remaining experiments.

6.2 Prediction Results

Once the main parameters are set, we evaluate the quality of RiskTeller's predictions through our random forest classifier; we have set a number of 800 trees as larger sizes do not improve neither the stability nor the quality of results. Our classifier was run on the dataset that includes file appearance logs from the machines of 18 enterprises. In Figure 4 we show the ROC curve obtained after a 10-fold cross validation. As the figure depicts, RiskTeller can predict 95% of the to-be-infected machines with only 5% false positives. This is a significant improvement over the related work [19] which had over 20% false positives for the same point in the ROC curve.

In addition to the cross-validation test, we conduct an experiment to demonstrate the forecasting capability of RiskTeller. We randomly select 50% of the to-be-infected machines to build the prediction models processing the first six months of training period. Then, we apply the models on the following 6 months to test whether the remaining 50% of the machines that got infected could be identified. We repeated this step 10 times to reduce the impact of the random machine selection. The resulting AUC score is 0.95; this validates our claim about RiskTeller's prediction capabilities.

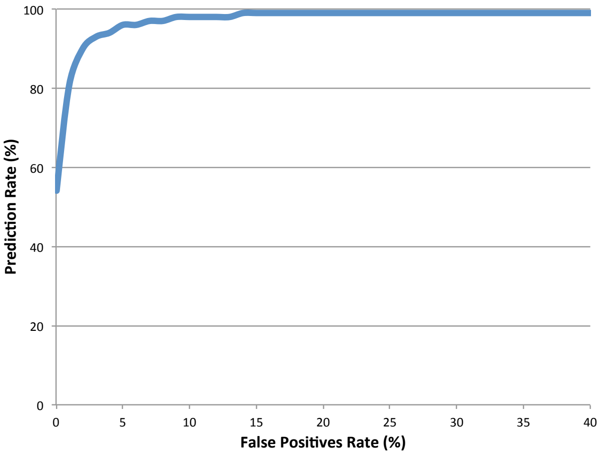


Figure 4: ROCs derived on the datasets

6.3 Feature Significance

To list the most discriminative features, we employ the mean decrease impurity methodology provided by the random forest classifiers. When training the trees in the random forest classifier, we compute how much each feature decreases the weighted impurity in the trees. Once the forest is built, we average the impurity decrease from each feature and rank them to identify the features that contribute the most to the classification accuracy. In Table 5 we list the strongest features among the 89 that RiskTeller processes, and Figure 5 presents the mean decrease in impurity per feature category. While all of the features do contribute to the ultimate classification results, the temporal file download/creation behavior of the users and the volume/diversity of file creation activities observed on the machines have the highest impact on distinguishing clean machine profiles from the risky ones. Surprisingly, the infection history features are not as correlated. Previous work on forecasting cyber security incidents [19] also found out that the historical threat data is not as useful for prediction and therefore, employed features about network misconfiguration details of the enterprises. Unlike that work, RiskTeller works at a finer granularity providing predictions at the machine level rather than at the enterprise level, and the prediction accuracy is larger.

6.4 Semi-Supervised Label Propagation

In this section, we perform additional experiments with semi-supervised learning (SSL) to highlight its merits. To this end, we manipulate our ground truth to simulate two issues commonly witnessed in real-world: the lack of balance between the sizes of classes in the labeled data and inadequate number of labeled data compared to the scale of the whole dataset. To obtain an imbalanced ground truth that is also small in size, we randomly choose p% of risky and q% of clean machine profiles in the ground truth to form new ground truths. The remaining labels are hidden from the classifier by signing them as unlabeled. Per different p and q values, we repeat the random sampling of the ground truth 10 times and take the average AUC and FPR values as the overall metric of the classification accuracy.

Table 5: Most discriminative features, grouping very similar ones.

Feature	Category	# in Table 1	Contribution
Fraction of events in weekdays/weekend	Temporal	16–17	0.075
Fraction of events from top-150 file hashes	Volume-based	5	0.060
# of patched apps	Vulnerabilities	22	0.041
Total number of events	Volume-based	1	0.026
Quartiles for CVSS scores of patched apps	Vulnerabilities	25–29	0.024
Distinct app count	Volume-based	7	0.023
Distinct file hashes	Volume-based	2	0.021
Unpatched app count	Vulnerabilities	36	0.020
Fraction of files signed by [101 − 1000] prevalence signers	Prevalence-based	67	0.018
Monthly median number of events	Temporal	20	0.018
Volume of downloads per app	Volume-based	53–57	0.017

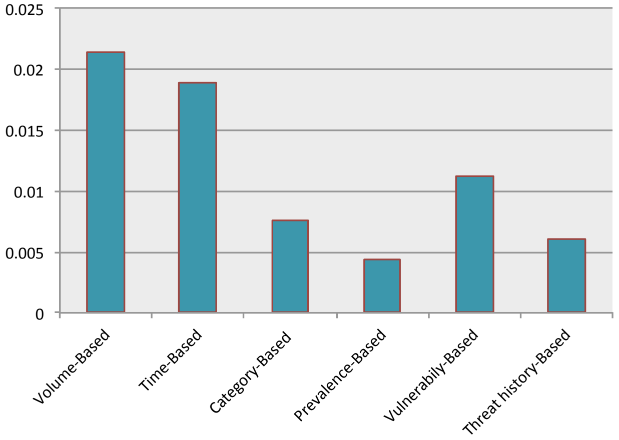


Figure 5: The contribution of different feature categories to the predictive modeling.

We set p as 20, 50 and q as 0.1, 0.5 and 1 respectively in the experiments and run the RF and SSL modules of RiskTeller separately. Table 6 illustrates the average AUC and FPR values corresponding to each pair of p and q. As expected, the performances of the supervised random forest module deteriorate when the class imbalance grows and the number of limited users decrease. In contrast, the SSL method, which can estimate the labels of the unlabeled dataset to enrich the ground truth, remains consistently accurate.

In our final experiment, we investigate the prediction accuracy of RiskTeller on per-enterprise data. We stated earlier that the our dataset comprises of data collected from 18 different enterprises. Before initiating the classification experiments in each enterprise, we analyzed the ground truth details of each enterprise. As Figure 6 demonstrates, the ground truths can be very imbalanced (e.g., 13K risky and 1.6K clean profiles) or very small in size (e.g., 23 risky and 60 clean profiles). As expected, the basic random forest classifier does not perform well here (on average 45% TPR with 5% FPR). The SSL classifier obtains better results (on average 61% TPR with 5% FPR); the results are still definitely not as good as those that can be obtained when combining the data from different enterprises.

Table 6: TPR of the random forest and semi-supervised methods when sampling labels.

p	q	FPR	Random Forest 5%	10%	15%	Semi-Supervised 5%	10%	15%
50%	0.1%	TPR	77%	79%	80%	**90%**	**95%**	**95%**
50%	0.5%		**90%**	93%	93%	84%	**94%**	**96%**
50%	1.0%		**90%**	93%	94%	88%	**94%**	**96%**
20%	0.1%		73%	83%	84%	**88%**	**93%**	**94%**
20%	0.5%		**89%**	92%	94%	84%	**93%**	**95%**
20%	1.0%		**91%**	**95%**	**96%**	**91%**	**95%**	**96%**

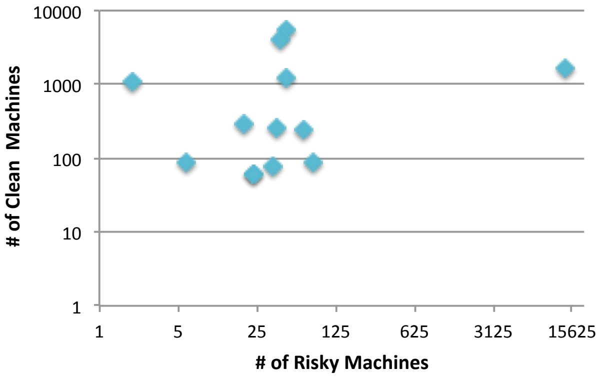


Figure 6: Ground Truth Size per enterprise.

7 DISCUSSION

A typical question that arises when machine learning (ML) is applied in the security domain is *evasion*, i.e., the possibility that malicious actors behave to avoid being discovered by the ML system. Luckily, in our case, this problem does not apply: RiskTeller uses features observed when machines are used by benign users in order to predict the risk of infections afterwards. Since features are collected on benign usage *before* the attacks we attempt to predict, evasion is not applicable to our case.

Another issue typically related to ML and security is *concept drift*: the risk that, as time passes, the statistical properties of the variable that ML models change, rendering the ML models less and less effective: for example, a malware detection system might lose effectiveness as malware families change over time. Instead of characterizing attacks, Riskteller models their likely *victims*. Our results —see Figure 3— show us that RiskTeller's results are stable over 6-month labeling periods, suggesting that characteristics of typical cyber-attack victims change slowly over time. On longer time frames, concept drift can be avoided by re-training our model.

Since we do not have data about the direct causes of attacks, we cannot use our system to learn *causality*; however, the correlations we observe can be interpreted to understand what are the cause that increase the risk of attack. Our analysis in Section 4.2 provides a clear picture of machines that are at risk of becoming infected: those with few or no security upgrades and unusual usage in terms of temporal patterns and binaries installed; it is quite intuitive to understand why such machines are at higher risk.

Our results show that RiskTeller can predict infections with variable precision depending on the dataset under consideration. Our prediction quality is higher than typical values in the insurance context, where prediction is a hard problem depending on human factors and outside events that are not observable in a model; for example, for bankruptcy prediction, see the ROC curves in the works by [7] and [28]: predictions that are less precise than those of RiskTeller are still useful to quantify risk and to price insurance.

We consider RiskTeller as also useful to identify machines at risk and drive choices for proactive hardening in enterprises: machine risk levels can be used independently and/or aggregated by organizations (e.g., departments) and job roles; these results can drive efforts to harden systems or educate users to risk, such that more effort is spent where it is most needed. One should not compare these results with those obtained in the related problem of malware detection: while detecting legitimate software as malware can sometimes make applications or machines unusable [6, 17, 20, 26, 35], a false positive in identifying a machine as at risk would just result in hardening that, *a posteriori*, could be considered not essential.

8 RELATED WORK

In computer security, a very large amount of work has been devoted to approaches that attempt to determine whether machines or systems are currently under attack, or distinguish malware from legitimate software. By comparison, the body of work targeting the prediction of future attacks and infections is comparably smaller.

In 1998, Korzyk [14] attempted to predict the growth in the number of vulnerability advisories in the early days of Internet security, using simple techniques such as time series analysis.

Some other works attempt to predict the behavior of attackers and defenders with the tools of game theory: Jones et al. [13] describe the interactions between efforts spent by attackers and defenders to predict their optimal strategies, while Axelrod and Iliev [1] investigate the issue of cyber-conflict timing, predicting the moment in which cyber-weapons such as zero-days exploits would be used. These approaches are most appropriate to describe high-stakes interactions between few rational players, and do not provide much insight with respect to the case of Internet security, with billions of machines that are susceptible to attack.

The IARPA CAUSE project [12] is a recent effort to use large amounts of Internet data together with machine learning to find the fingerprints (e.g., system probing) that predict a future attack. Unlike this project, our work focuses in the behavior of potential victims rather than the one of attackers.

More similarly to our approach, six works analyze data about users and systems to estimate the risk of cyber incidents: *(a)* Lalonde Levesque et al. [16] analyze a small set of 50 users' demographics and web browsing features to evaluate risk factors; *(b)* Canali et al. [4] perform a study that associated users to their risk through information about their web browsing logs; *(c)* Soska and Christin [30] collect features about websites to predict which ones will become malicious; *(d)* Yen et al. [38] analyze user demographics and behavioral features in a single large corporation to compute risk factors; *(e)* Liu et al. [19] predict cyber-security incidents within organizations by analyzing externally measurable features. In a recent study, *(f)* Veeramachanent et al. [37] analyze the historical incident records of enterprises to predict cyber incidents leveraging deep learning methods. We consider these approaches as valuable and complementary to the analysis that we carry out in this work. All these approaches provide informative predictions, but none of them is close to perfection: Soska and Christin obtain 66% true positives (TPs) with 17% false positives (FPs), Canali et al. obtain 74% TPs with 8% FPs, Liu et al. have 90% TPs with 10% FPs, and Veeramachanent et al. obtain 86.8% FPs with 4.4% TPs after being boosted with feedback from security analysts. Lalonde Levesque et al. and Yen et al. use logistic regression to compute risk factor and do not report FP/TP results, but their results suggest lower precisions. Each of these works predicts different types of incidents and hence these numbers are not directly comparable. However, they provide context showing that predicting incidents is a difficult problems, and that RiskTeller's TP/FP rates are generally better than those observed in similar studies. Moreover, the studies by Liu et al. and Veeramachanent et al. are inherently less granular and predict attack only at the level of an organization, while an approach such as ours gives more granular information, providing administrators an actionable way to highlight the most risky components of their infrastructure.

Semi-supervised learning has been used in the security context by Han and Shen [9], classifying automatically email-based spear-phishing campaigns. In this domain, the ground truth datasets are created by human analysts, and therefore require extensive human effort. Han and Shen propose to use a semi-supervised learning algorithm to reduce the labeling overheads. In our work, instead, we employ semi-supervised learning to verify the completeness of our ground truth and to enrich it in case it is limited and/or imbalanced in size.

9 CONCLUSION

In an era where the cyber incidents became unavoidable, cyber defenders started to shift their interest from reactive to proactive security, and enterprises and individuals purchase more and more cyber insurance packages so that when the incident happens, a part of their loss can be covered. One crucial task in both proactive security and cyber insurance is to estimate and predict the risk in advance. If methods for risk prediction exist, malware defense solutions can incorporate this piece of information to harden the systems of machines at risk such that it is harder for the attackers to compromise them. Also, cyber insurance schemes can benefit from risk estimates to price efficiently their offers. In this work, we focused on addressing this gap in the cyber ecosystem by proposing RiskTeller, a system that can predict the machines at risk in enterprises with high accuracy. To date, no previous work was able to achieve a 96% TPRs with only 5% FPRs for such predictions at a machine-level granularity.

Despite the capabilities of RiskTeller to predict machines at risk we believe that, in the area of prediction for cyber security, much remains to be explored. RiskTeller predicts, in general, that a machine is at risk from malware, providing no hints about specific malware categories. In future work, we plan to extend our work to predict the risk of different machine profiles for threat types such as banking trojans, advanced threats, data breaches, ransomware, etc. Such a system requires effort from our community in terms of automated threat categorization on large scale data: while this could rely on AV labels, several pieces of work in the literature state that they are in general not very reliable. Another line of research that we wish to explore is risk prediction for individual users, whose usage behavior is less regular compared to the enterprise users; in future work, we plan to explore new techniques and evaluate other datasets to better capture those users' behavior and obtain better prediction accuracy.

REFERENCES

[1] Robert Axelrod and Rumen Iliev. 2014. Timing of cyber conflict. *Proceedings of the National Academy of Sciences* 111, 4 (2014), 1298–1303.

[2] Oleg Bogomolniy. 2017. Cyber Insurance Conundrum: Using CIS Critical Security Controls for Underwriting Cyber Risk. https://www.sans.org/reading-room/whitepapers/legal/cyber-insurance-conundrum-cis-critical-security-controls-underwriting-cyber-risk-37572. (2017).

[3] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[4] Davide Canali, Leyla Bilge, and Davide Balzarotti. 2014. On the effectiveness of risk prediction based on users browsing behavior. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*. ACM, 171–182.

[5] Duen Horng Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. 2011. Polonium: Tera-Scale Graph Mining and Inference for Malware Detection. In *SIAM International Conference on Data Mining (SDM) 2011*, Vol. 25. 131–142.

[6] Lucian Constantin. 2011. MSE false positive detection forces Google to update Chrome. The Inquirer, http://www.theinquirer.net/inquirer/news/2113892/mse-false-positive-detection-forces-google-update-chrome. (October 2011).

[7] J David Cummins, Martin F Grace, and Richard D Phillips. 1999. Regulatory solvency prediction in property-liability insurance: Risk-based capital, audit ratios, and cash flow simulation. *Journal of Risk and Insurance* (1999), 417–458.

[8] Experian. 2015. Data Breach Industry Forecast. https://www.experian.com/assets/data-breach/white-papers/2015-industry-forecast-experian.pdf. (2015).

[9] Yufei Han and Yun Shen. 2016. Accurate Spear Phishing Campaign Attribution and Early Detection. In *Proceedings of the 31st ACM Symposium on Applied Computing*.

[10] G. R. Hileman, S. M. Mehmud, and M. A. Rosenberg. 2016. Risk Scoring in Health Insurance. https://www.soa.org/Files/Research/research-2016-risk-scoring-health-insurance.pdf. (2016).

[11] Tin Kam Ho. 1995. Random Decision Forest. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition*. 278–282.

[12] IARPA. 2015. Cyber-attack Automated Unconventional Sensor Environment (CAUSE). https://www.iarpa.gov/index.php/research-programs/cause. (2015).

[13] Malachi Jones, Georgios Kotsalis, and Jeff S Shamma. 2013. Cyber-attack forecast modeling and complexity reduction using a game-theoretic framework. In *Control of Cyber-Physical Systems*. Springer, 65–84.

[14] Alexander D. Korzyk, Sr. 1998. A forecasting model for internet security attacks. In *National Information System Security Conference*.

[15] Aorato Labs. 2014. The Untold Story of the Target Attack Step by Step. https://aroundcyber.files.wordpress.com/2014/09/aorato-target-report.pdf. (September 2014).

[16] Fanny Lalonde Levesque, Jude Nsiempba, José M Fernandez, Sonia Chiasson, and Anil Somayaji. 2013. A clinical study of risk factors related to malware infections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 97–108.

[17] John Leyden. 2010. Horror AVG update ballsup bricks Windows 7. The Register, http://www.theregister.co.uk/2010/12/02/avg$_a$uto$_i$mmune$_u$pdate/. (December 2010).

[18] Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

[19] Yang Liu, Armin Sarabi, Jing Zhang, Parinaz Naghizadeh, Manish Karir, Michael Bailey, and Mingyan Liu. 2015. Cloudy with a chance of breach: Forecasting cyber security incidents. In *24th USENIX Security Symposium (USENIX Security 15)*. 1009–1024.

[20] Declan McCullogh. 2010. Buggy McAfee update whacks Windows XP PCs. CNET, http://www.cnet.com/news/buggy-mcafee-update-whacks-windows-xp-pcs/. (April 2010).

[21] Yuxin Meng, Wenjuan Li, and Lam-For Kwok. 2014. Enhancing email classification using data reduction and disagreement-based semi-supervised learning. In *Proceedings of IEEE International Conference on Communications 2014*. 622–627.

[22] Antonio Nappa, Richard Johnson, Leyla Bilge, Juan Caballero, and Tudor Dumitras. 2015. The attack of the clones: a study of the impact of shared code on vulnerability patching. In *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 692–708.

[23] Kartik Nayak, Daniel Marino, Petros Efstathopoulos, and Tudor Dumitraş. 2014. Some vulnerabilities are different than others. In *Research in Attacks, Intrusions and Defenses*. Springer International Publishing, 426–446.

[24] M. Ovelgonne, T. Dumitras, A. Prakash, V.S. Subrahmanian, and B. Wang. 2016. Understanding the Relationship between Human Behavior and Susceptibility to CyberAttacks: A DataDriven Approach. In *ACM Transactions on Intelligent Systems and Technology*.

[25] Clifton Phua, Vincent C. S. Lee, Kate Smith-Miles, and Ross W. Gayler. 2010. A Comprehensive Survey of Data Mining-based Fraud Detection Research. *Computing Research Repository* abs/1009.6119 (2010).

[26] Emil Protalinski. 2008. AVG incorrectly flags user32.dll in Windows XP SP2/SP3. Ars Technica, http://arstechnica.com/information-technology/2008/11/avg-incorrectly-flags-user32-dll-in-windows-xp-sp2sp3/. (November 2008).

[27] PWC. 2016. Insurance 2020 and beyond: Reaping the dividends of cyber resilience. http://www.pwc.com/gx/en/insurance/publications/assets/reaping-dividends-cyber-resilience.pdf. (2016).

[28] Alexander S Reisz and Claudia Perlich. 2007. A market-based framework for bankruptcy prediction. *Journal of Financial Stability* 3, 2 (2007), 85–131.

[29] Igor Santos, Javier Nieves, and Pablo G. Bringas. 2011. Semi-supervised Learning for Unknown Malware Detection. In *Proceedings of International Symposium on Distributed Computing and Aritifical Intelligence 2011*. 415–422.

[30] Kyle Soska and Nicolas Christin. 2014. Automatically detecting vulnerable websites before they turn malicious. In *23rd USENIX Security Symposium (USENIX Security 14)*. 625–640.

[31] Amarnag Subramanya and Jeff Bilmes. 2011. Semi-supervised learning with measure propagation. *Journal of Machine Learning. Research* 12 (2011), 3311–3370.

[32] Symantec. 2016. Internet Security Threat Report Vol. 21. https://www.symantec.com/security-center/threat-report. (April 2016).

[33] Christopher T. Symons and Justin M. Beaver. 2012. Nonparametric Semi-supervised Learning for Network Intrusion Detection: Combining Performance Improvements with Realistic In-situ Training. In *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence (AISec)*. ACM, New York, NY, USA, 49–58.

[34] Acar Tamersoy, Kevin Roundy, and Duen Horng Chau. 2014. Guilt by association: large scale malware detection by mining file-relation graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, 1524–1533.

[35] Aaron Tan. 2007. Flawed Symantec update cripples Chinese PCs. CNET, http://www.cnet.com/news/flawed-symantec-update-cripples-chinese-pcs/. (May 2007).

[36] Olivier Thonnard, Leyla Bilge, Anand Kashyap, and Martin Lee. 2015. Are you at risk? Profiling organizations and individuals subject to targeted attacks. In *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 13–31.

[37] Kalyan Veeramachanent, Ignaclo Arnaldo, Alfredo Cuesta-Infante, Korrapati Vamsl, Costa Basslas, and Li Ke. 2016. AI2: Training a big data machine to defend. In *Proceedings of the 2nd IEEE International Conference on Big Data Security*.

[38] Ting-Fang Yen, Victor Heorhiadi, Alina Oprea, Michael K Reiter, and Ari Juels. 2014. An epidemiological study of malware encounters in a large enterprise. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1117–1130.