RiskTeller: Predicting the Risk of Cyber Incidents

Leyla Bilge, Yufei Han, Matteo Dell'Amico

Symantec Rsearch Labs

CCS 2017

Web Security & Privacy Lab

KAIST

Summarize the paper

- Problem: Few previous studies predicts the risk of infection. One study demonstrated a 20% FPR
- Goal: Let's predict which machines are at the risk of infection
- Contribution
  - Leverage both supervised and semi-supervised learning
  - Design 89 features that are extracted from file appearance logs
  - RiskTeller achieved a 96% TPRs with only 5% FPRs
- Meaning
  - It is feasible to quantify the risk of future infection with a high accuracy

Motivation

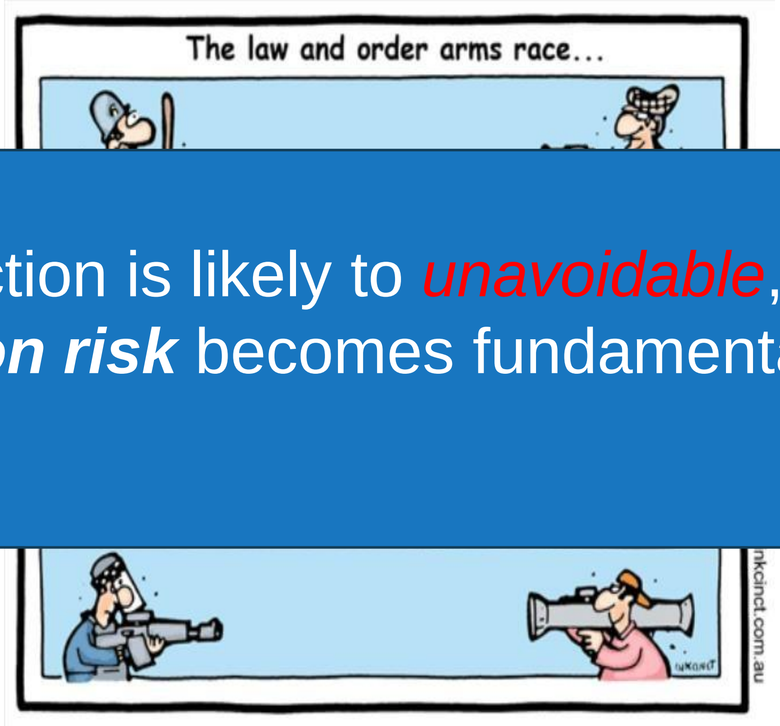
- The cyber-threat ecosystem faced dramatic changes
- Attackers use sophisticated tools and techniques to breach systems

Web Security & Privacy Lab

KAIST

Motivation

- The cyber-threat ecosystem faced dramatic changes
- Attackers use sophisticated tools and techniques to breach systems



Since a malware infection is likely to *unavoidable*, ***predicting the infection risk*** becomes fundamental

How to predict? (example)

- Low security awareness
- Lack of security training
- Host usage patterns

Why cyber risk prediction?

- Facing cyber attacks is now the norm rather than an exception
- Businesses need to be prepared to minimize damage when attacks eventually strike
  - Deploy multiple layers of security (security services, advisor, employee training programs, defense program, etc.)
  - Very expensive!
- Cyber insurance companies have been seeking for better risk prediction methodologies to persuade security companies

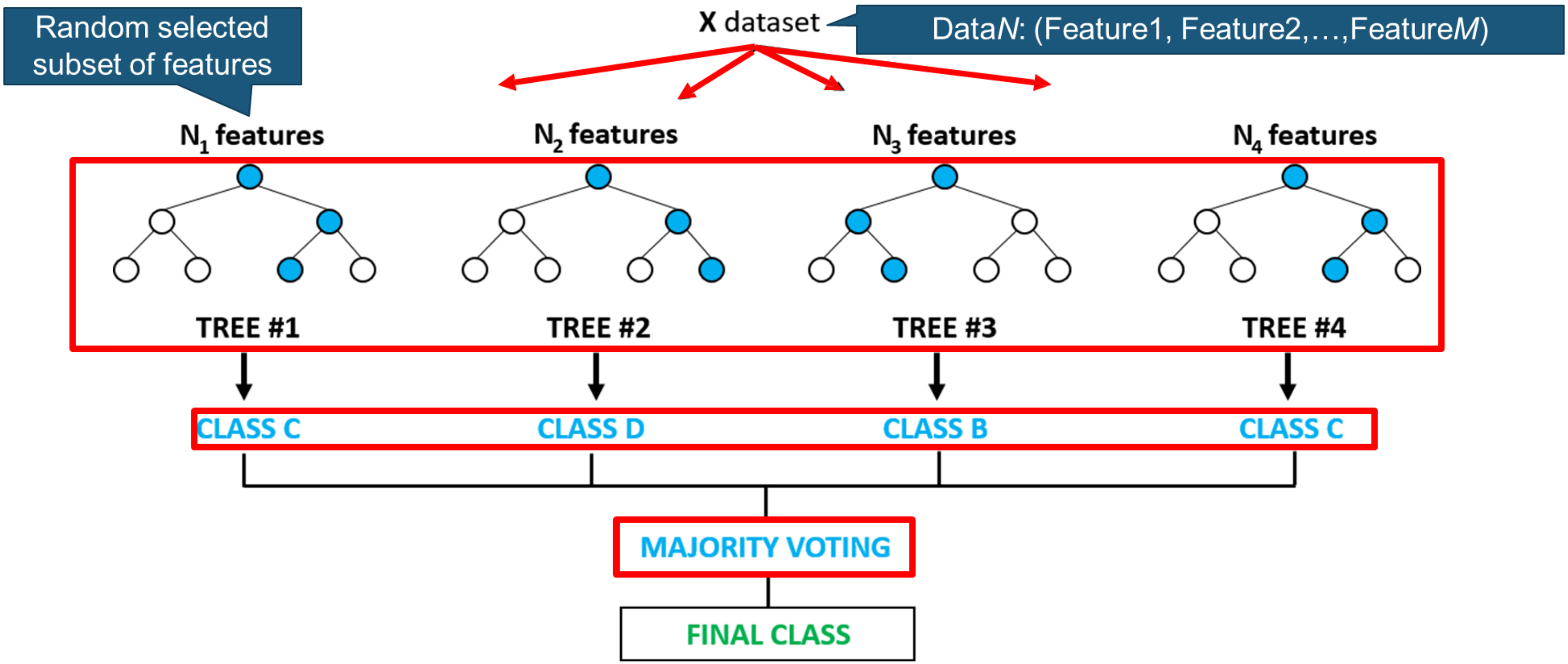
Web Security & Privacy Lab

KAIST

Background – Machine Learning

- Supervised Learning
  - Training **labeled data** includes desired outputs
  - Trying to predict a specific quantity
- Unsupervised Learning
  - Training **unlabeled data** does not include desired outputs
  - Trying to understand the data
- Semi-supervised Learning
  - Training **labeled + unlabeled data** includes a few desired outputs

Background – Random Forest Classifier



Background – Detection VS Prediction

- **Predicting future events** is a more difficult problem than **detecting on-going malicious events!**
- Detection
  - False positives can be very expensive
  - Goal: maximizing the true positive ratio while keeping the false positives very low
- Prediction
  - Compared to the detection domain, the cost of false positives is lower.
  - An enterprise would want to know all the machines that could be infected

RiskTeller

- Employ a dataset that **provides fine grained information about the security posture** of each enterprise machine
- Analyze internal telemetry collected from companies to predict which computers are most at risk
- Analyze **per-machine file appearance logs** to predict which machines are at risk of getting infected

Dataset

- Mining large-scale data that discover interesting behavior differences between clean and risky machines
- Binary appearance logs
  - E.g., Due to file downloads or compilation
  - Generated by enterprise employees
  - Collected by the AV company data centers
  - Collected from more than 100K enterprises
  - Every day, receive reports about 100M logs of 14M distinct binaries
  - Obtain only a subset of this data, covering 4.4B logs of 600K machines belonging to 18 enterprises

Why?

Web Security & Privacy Lab

KAIST

Data Preprocessing

- The fields extracted in from the binary file appearance logs are:
  a. 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 and for the time when it was reported to the data centers
  f. File signer subject in the certificate

Web Security & Privacy Lab

KAIST

Data Preprocessing

- Data normalization and cleaning
  - Remove version numbers in filenames (using regular expression)
  - Remove suffixes generally appended to duplicate files (e.g., “()”, “[]”)
  - Find which applications binary files belong to: resort to their directory name
    - Use the CSIDL (Constant Special Item Id List) to identify the name of special folders
    - To identify an application, use depth-3 paths starting from CSIDL_PROGRAM_FILES (e.g., Chrome directory is \CSIDL_PROGRAM_FILES\google\chrome)

Why use this heuristic?

Ground Truth

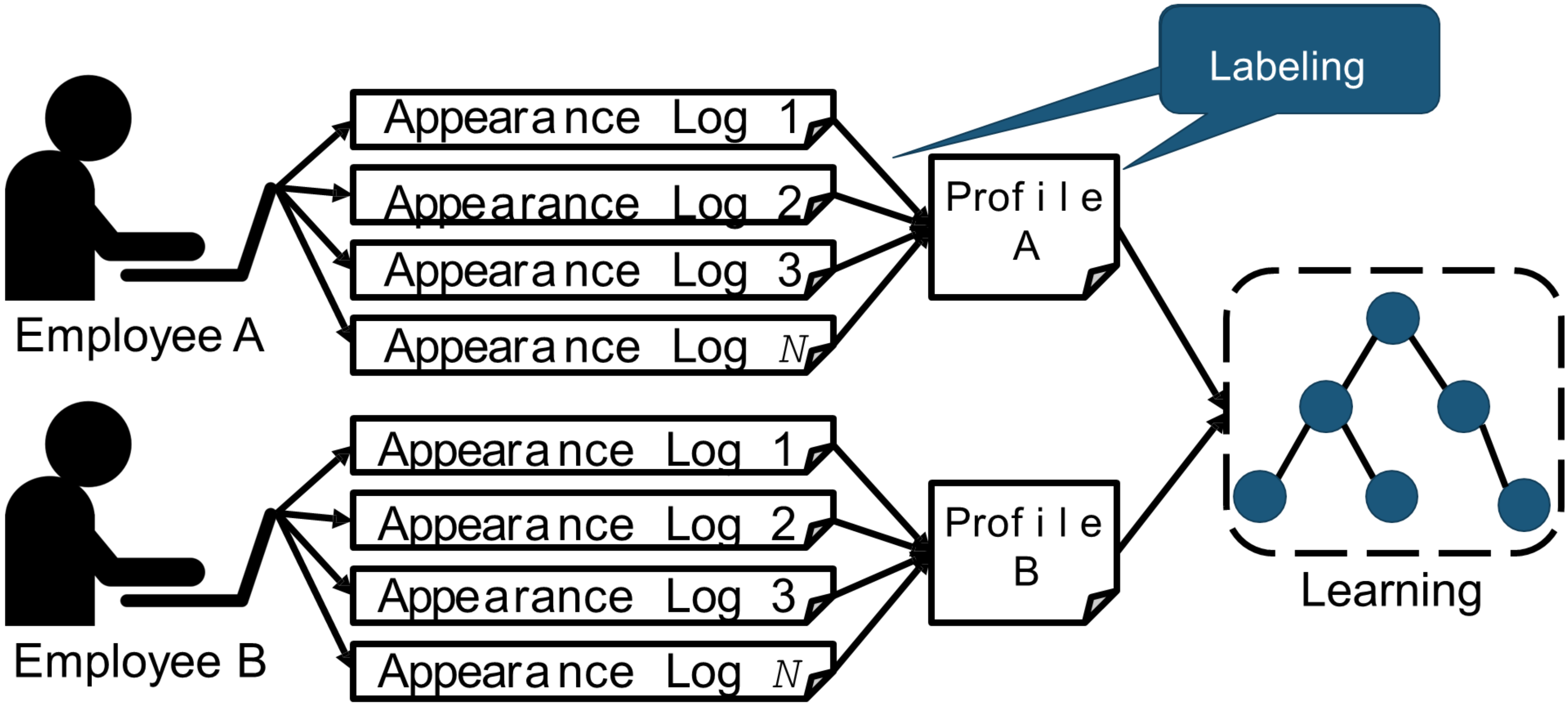
- Split datasets in two consecutive periods
  - Feature extraction: compute features that will be fed to classifier
  - Labeling: identify a ground truth of "clean" and "risky" machines
- Additional references to determine the ground truth
  - 16M known benign and 214M known malware **file hashes**
  - **File hashes** that were identified as malware according to the AV product: 800M file hashes
  - **Infection reports** for the machines from the IDS product
- How?
  - If there is a host with no records with the above file hash nor report => The host is "clean"
  - Over a threshold => "risky"

Building the machine profiles

- Do not seek to pinpoint the exact causes of infections, but rather characteristics that are correlated with them

- For each machine, create a profile consisting of 89 different features synthesized from logs

Building the machine profiles



Volume-Based Features

- General statistics calculated from new binaries appeared

- From the 50 most frequently appearing file signers
- From the 150 most frequent file hashes

Too coarse-grained feature?

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

- 5 quartiles of the per-application percentage of events
- Minimum, maximum, median, 26th and 57th percentiles

25th and 75th

The people with abundant and varied browsing behavior suffer higher risks [4]
=> Let's check whether people use a limited number of apps often or various apps

Temporal Behavior

- To understand whether longer working hours is correlated with facing higher risk to encounter malware infections

Hypothesis: generally use machines during weekends or in the evenings are more possibly engaging in riskier activities

06:00-18:59

19:00-00:59

01:00-05:59

Feature Category	Feature #	Features
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 and Patching Behavior

- The patching behavior and the severity of existing vulnerabilities can be highly correlated with the prediction

1. Manually identify software (by checking signer and file name)
2. Obtain file version information (by checking logs or VirusTotal)
3. By parsing NVD data, obtain vulnerable file version and CVSS score

Feature Category	Feature #	Features
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

Vulnerabilities and Patching Behavior

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

They selected only 9 application!
Q. Why it is hard to matching vulnerability information with NVD data and appearance logs?

Web Security & Privacy Lab

KAIST

Vulnerabilities and Patching Behavior

- The patching behavior and the severity of existing vulnerabilities can be highly correlated with the prediction

Feature Category	Feature #	Features
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 Category-Based Features

- To understand which specific machine profiles are more prone to encounter cyber-attacks

Feature Category	Feature #	Features
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

Application Category-Based Features

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

Create a ground truth of over 10K applications that fall into 26 different categ ories by manually queryi ng Capterra

Web Security & Privacy Lab

KAIST

Application Category-Based Features

- To understand which specific machine profiles are more prone to encounter cyber-attacks

Feature Category	Feature #	Features
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

Application Category-Based Features

- To understand which specific machine profiles are more prone to encounter cyber-attacks

Later, Transform numerical through *one-hot* encoding

18 tools such as ping, netstat, etc.

64 tools such as data transmission tools, device scanners, etc.

33 tools such as MITM attack tools, password crackers, etc.

Feature Category	Feature #	Features
Application categories (§ 4		top-5 application categories with most events
		fraction of events per top-5 category
		fraction of system diagnostics tools
		fraction of system administration tools
	60	fraction of attack tools

History of malware and goodware events

- It is reasonable to conjecture that past infection history is correlated with future events (Based on the ground truth)

Feature Category	Feature #	Features
Infection history (§ 4.1.5)	61–63	fraction of events for malicious/benign/unknown files

Prevalence-Based Features

- Malware tends to have lower prevalence than benign software
- The fact that a machine has a **large number of low-prevalence** files gives reasons to be suspicious about that

Feature Category	Feature #	Features
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

Prevalence-Based Features

For each event, compute

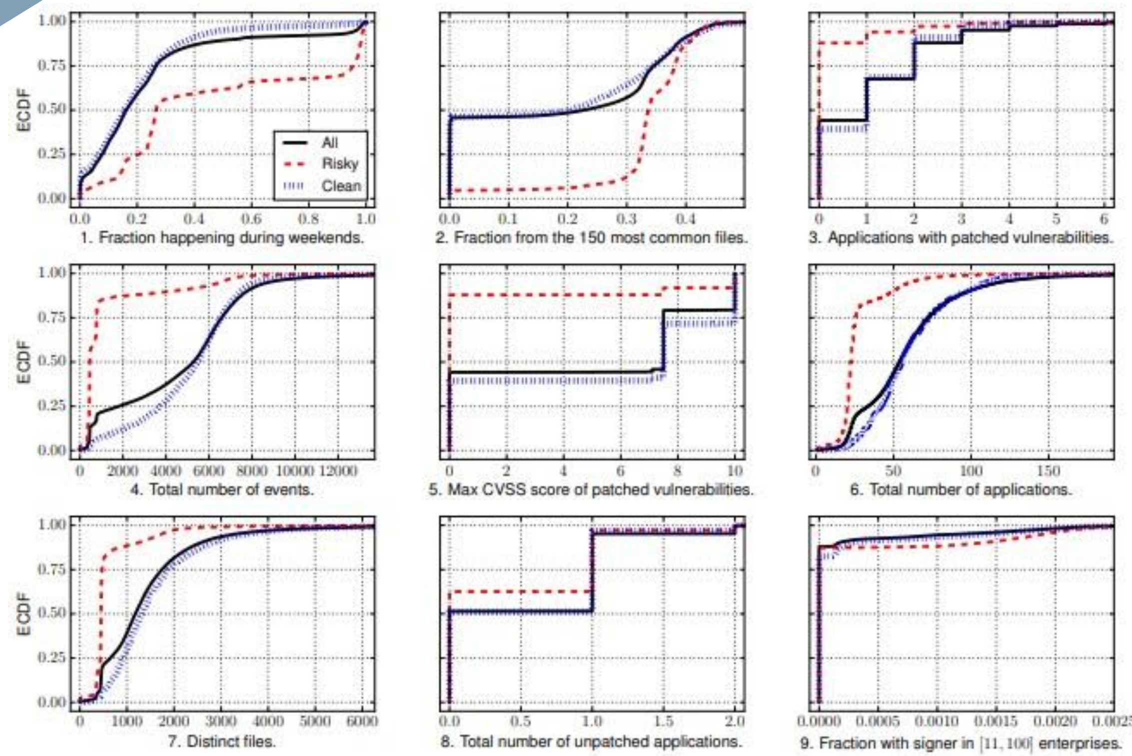
1. The number of events and enterprises in which the file signer is seen
2. The number of events in which the file hash is seen
3. The number of enterprises and machines in which the file hash is seen

Feature Category	Feature #	Features
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

Feature distribution of dataset

- Show the overall cumulative distribution functions (CDFs) of the 9 most significant features

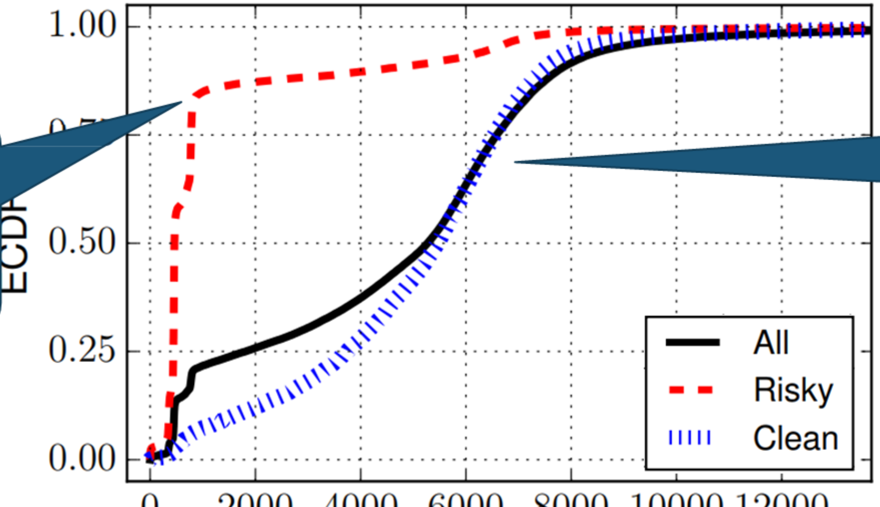
- X: allowable domain for the given features
- Y: cumulative distribution



Feature distribution of dataset

- Show the overall cumulative distribution functions (CDFs) of the 9 most significant features

Infer that risky users install less binaries on their machine than other users.

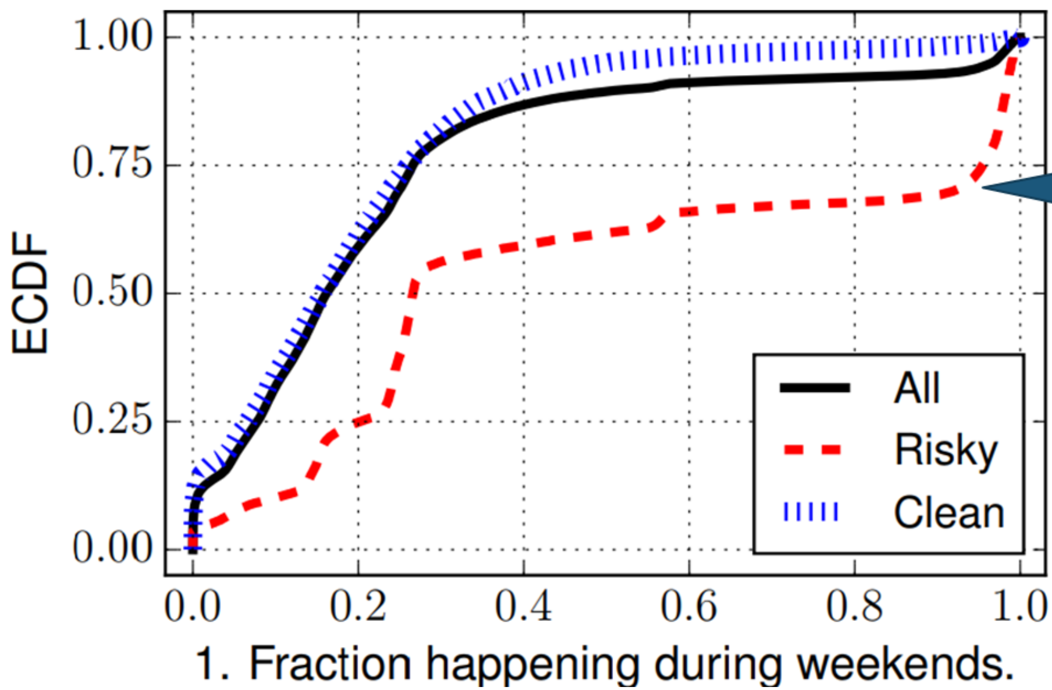


4. Total number of events.

In general, the overall distribution and that of clean profiles are similar

Feature distribution of dataset

- Show the overall cumulative distribution functions (CDFs) of the 9 most significant features



Risk are higher usage during weekends

Random Forest Classifier

- Aims at reducing the variance of the learning model through bias-variance trade-off
- Run the RFC with 800 trees as the threshold

Semi-Supervised Learning

- Excel when the ground truth datasets are unbalanced and/or small
- Reducing manual labeling overheads and preserving classification accuracy
- Design Principles
  1. Risk scores are bounded in [0,1]
     - A value of 1 indicates unquestionable of infection is detected on the machine
     - A value of 0 indicates that the machine is free from malicious files
  2. Similar user profiles yield close risk scores
     - If two profiles are close in feature space, infer that they will have similar risk scores

Web Security & Privacy Lab

KAIST

Gradient Descent

- If the multi-variable function *F(x)* is defined and diffe rentiable in a neighborhood of a point a
  - F(x) decreases fastest if one goes from a in the direction of t he negative gradient of F at a, $-\nabla F(\mathbf{a})$

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

  - If a particular choice of r and F(x) is a convex function, conv ergence to a local minimum can be guaranteed

- E.g. F(X) = X^2 + Y^2 Find a minimum value from a gi ven (x,y) = (1, 1)
  - Gradient $\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = (2x, 2y)$ (X', Y') = (1,1) – r*(2,2)

Semi-Supervised Learning

When Pi and Pj are similar, this term goes to zero

When Pi goes closer to 0.5, this term goes to zero

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

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

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

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

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

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

Web Security & Privacy Lab

KAIST

Experiments

They didn't specify experi ment setups (e.g., OS, co mputing power, etc.)

1. Parameters to choose the best settings
2. Ability to predict machine infection
3. Significance of features and feature categories
4. Significance of the semi-supervised risk prediction algorithm

Web Security & Privacy Lab

KAIST

RiskTeller Parameters

- Feature Extraction and Labeling Period Length
  - Fragment data in two consecutive periods, where the feature extraction period lasts M months and the labeling period lasts N months
  - When M + N < 12, prepare different datasets starting at the beginning of each month
  - 10-fold cross-validation
    - Split the labeled users into a labeled training set L and a validation set V
    - Build the risk prediction model using both L and unlabeled user profiles U (Semi-Supervised Learning)
    - Apply model on the set V

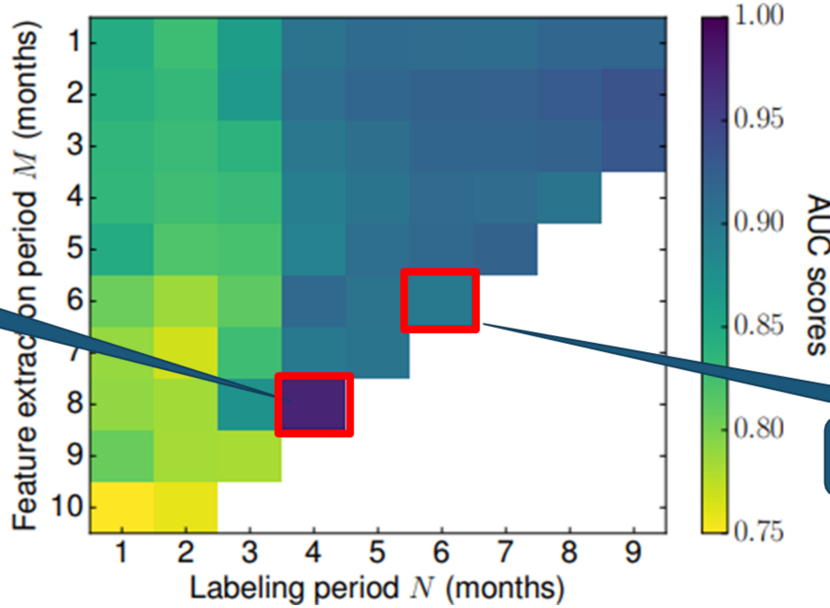
Did not specify the ratio of L and U!

RiskTeller Parameters

- Feature Extraction and Labeling Period Length

- Area Under the Curve of ROC
- The closer AUC for a model comes to 1, the better it is.

Too short N!



Why this ratio?

Set $M = N = 6$

RiskTeller Parameter

To avoid misclassification(?), a machine that has at most *T_gray* unlabeled files is consider clean

- Thresholds for the Ground Truth
  - Define clean machine
    - No any infection records in the IPS dataset
    - Zero files known to be malware
  - Define risky machine
    - Only if it is associated with at least T_inf malicious events

Table 4: AUC va… he ground truth thresholds.

T_{inf}	T_{grey}	AUC	Machines Risky	Machines 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**

Prediction Results

96% TPRs with only 5% FPRs

- ROC curve obtained after a 10-fold cross validation

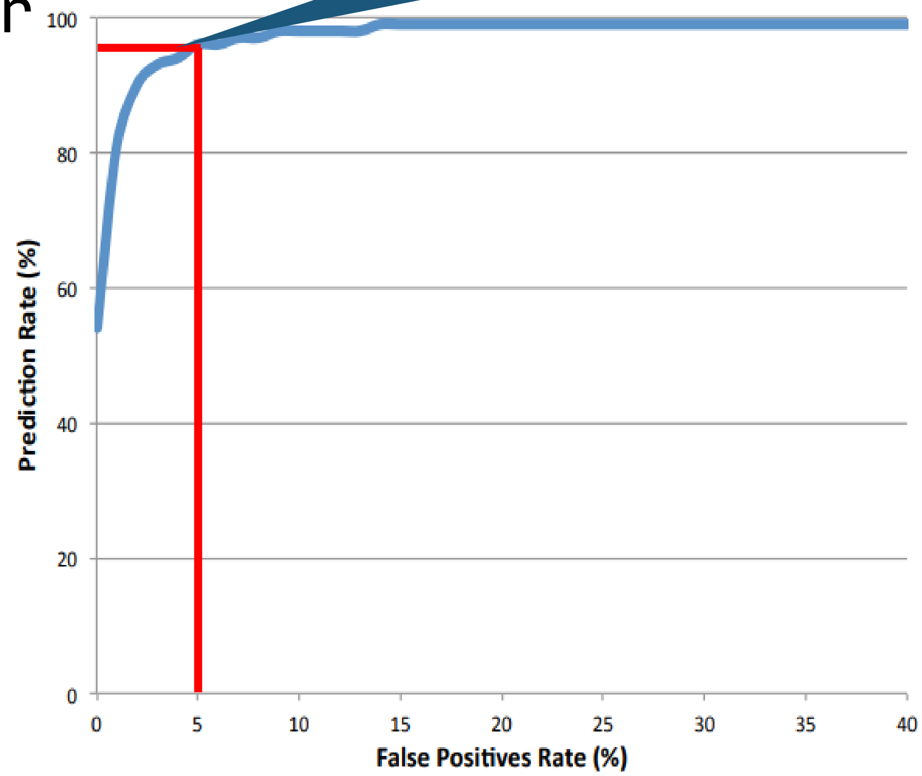


Figure 4: ROCs derived on the datasets

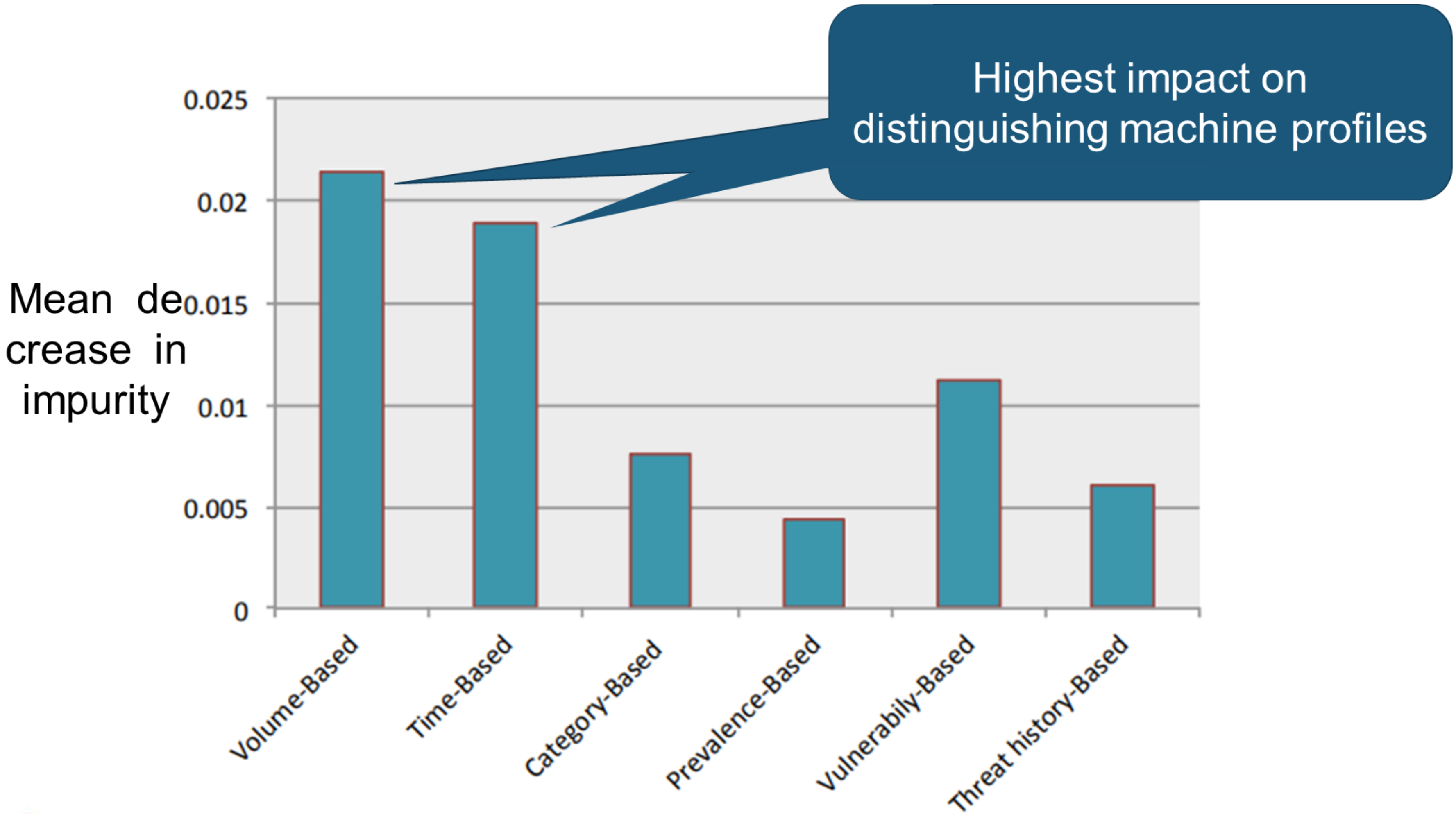
Feature Significance

- To list the most discriminative features, employ the mean decrease impurity methodology
- When training the trees, compute how much each feature decreases the weighted **impurity** in the trees
- After built the forest, they average the impurity decrease from each feature and rank them

Web Security & Privacy Lab

KAIST

Feature Significance



Feature Significance

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

Semi-Supervised Label Propagation

- Experiments with SSL to highlight its merits
- Manipulate ground truth to simulate two issues
  - The **lack of balance** between the sizes of classes in the labeled data
  - **Inadequate number** of labeled data

Semi-Supervised Label Propagation

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

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

Semi-Supervised Label Propagation

Ta... forest and semi-supervised me... ls.

Randomly choose *p*% of risky profiles

Per different *p* and *q* values, repeat the random sampling of the ground truth 10 times

p	*q*	FPR	Random F... 5%	10%	15%	5%	10%	15%
50%	0.1		77%	79%	80%	**90%**	**95%**	95%
50%	0.5		**90%**	93%	93%	84%	**94%**	**96%**
50%	1.0	...PR	**90%**	93%	94%	88%	**94%**	**96%**
20%	0.1		73%	83%	84%	**88%**	**93%**	**94%**
20%	[illegible]		[illegible]	[illegible]	[illegible]	84%	**93%**	**95%**
20%	[illegible]		[illegible]	[illegible]	[illegible]	**91%**	**95%**	**96%**

Randomly choose *q*% of clean profiles

Web Security & Privacy Lab

KAIST

Discussion

- Pros
  - They leverage **very large scale dataset** that help to discover behavioral patterns
  - **Comprehensive features** that can separate infected machines from clean ones
- Cons
  - Need some manual effort to extract features.
  - Because they leverage only binary appearance logs which have limited information, they rely on their heuristics (e.g. infer application using directory name)

Web Security & Privacy Lab

KAIST

Questions