When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks

Octavian Suciu, Radu Mãrginean, Yiğitcan Kaya,

Hal Daumė III, Tudor Dumitras.

USENIX Security Symposium 2018

Web Security & Privacy Lab

Some figures refer to the author slide

KAIST

Summary

- Problem
  - Existing attacks make diverse, potentially unrealistic assumptions about the adversary attack in machine learning (ML) system
  - Need high accuracy ML attack model by a weaker adversaries
- Contributions
  - Define FAIL model that is a general framework for ML attack in various adversarial knowledge and control over the victim
  - Propose StingRay that is targeted poisoning attack algorithm of high attack accuracy in FAIL model
  - Show that a prior poisoning attack is less effective under FAIL model while StingRay shows high accuracy of attack

Contributions & Motivations

- Contributions:
  - Define FAIL model that is a general framework for ML attack in various adversarial knowledge and control over the victim
  - Propose StingRay that is targeted poisoning attack algorithm of high attack accuracy in FAIL model
  - Show that a prior poisoning attack is less effective under FAIL model while StingRay shows high accuracy of attack
- The attacker has too much information and controls about a victim
  - Existing attacks make diverse, potentially unrealistic assumptions about the adversary attack in machine learning (ML) system
  - Need high accuracy ML attack model by a weaker adversaries

Motivation

The previous attacker (paper) has too much information and controls about a victim

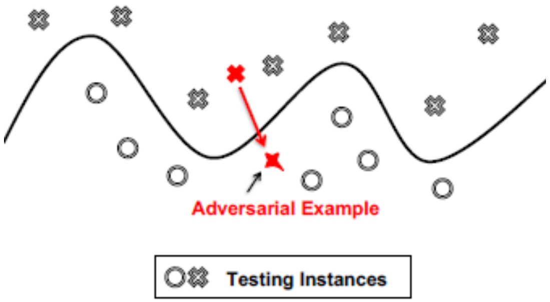
- Know victim’s data set
- Know victim’s ML algorithm
- Know victim’s feature information
- Know victim’s data set

Web Security & Privacy Lab

KAIST

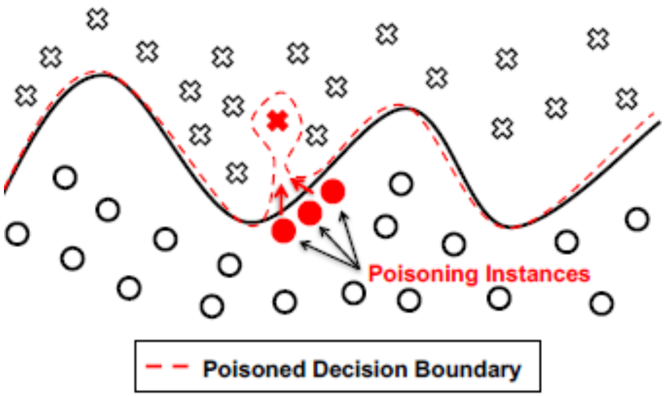
Targeted attacks against machine learning model

- Evasion attacks



(b)

- **Targeted poisoning attacks**



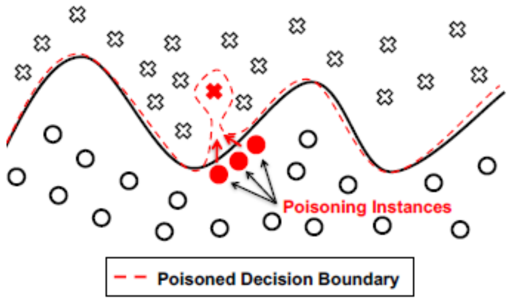
(c)

Problem Statement

- How can we systematically model adversaries based on realistic assumptions about their capabilities?
- How realistic is the targeted poisoning threat?
- Define and evaluate a more general transferability across a wide range of adversary models
- **Transferability**
  - Attack samples crafted locally, on a surrogate model that reflects the adversary's limited knowledge, allowing them to remain successful against the target model
  - Black-box attacks often investigate transferability in the case where the local and target models use different training algorithms

Threat model

- **Targeted poisoning attacks**
- Alice as victim classifier
- $h(t) = y_t$
- Bob as an owner of the target instance:
  - possesses an instance $\mathbf{t} \in \mathbf{T}$ with label y_t, called the target, which will get classified by Alice
- Mallory as an attacker:
  - Partial knowledge of Alice's classifier
  - Read-only access to target's feature representation
  - Do not control either $\mathbf{t}$ or label y_t which is assigned by the oracle such as VirusTotal



(c)

Threat model: Attacker Goals

- Mallory's first goal is:
  - Introduce a targeted misclassification on the target by deriving a training set, $y_{\mathbf{d}}$ is Mallory's desired label for t
- Mallory's second goal is:
  - Minimize the effect of the attack on Alice's overall classification performance, small PDR
  - Performance Drop Ratio (PDR)

$$PDR = \frac{performance(h)}{performance(h^*)}$$

Realistic adversaries

- Imperfect knowledge about the ML model
- Limited capabilities in crafting adversarial samples
- For successful attack
  - Samples crafted under these conditions must transfer to the original (target) model
- Suggest formalize the adversary's strength in the **FAIL** attacker model in four dimensions

Constraints

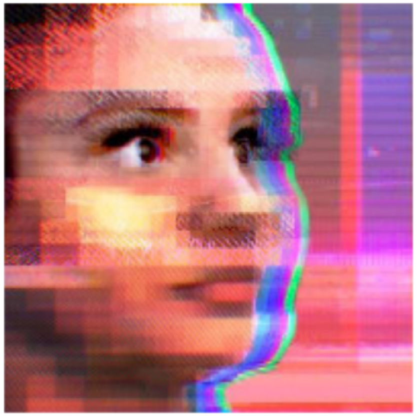
1. Poison samples must have clean labels
   - Adversary can inject training samples but cannot determine how they labeled
2. Poison samples must be individually inconspicuous
   - Similar to the existing training instances
3. Poison samples must be individually inconspicuous
4. Poison samples must exhibit a generalized form of transferability
   - Adversary tests the samples on a surrogate model, trained with partial knowledge along multiple dimensions, defined by the FAIL model

Discussion about poison attack

Assumption about poison attack: adversary can inject training samples but cannot determine how they labeled

Microsoft's Tay chatbot poisoned through tweets

(https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist)



Question: Is it a valid assumption to inject a training samples?

FAIL attacker model

- **F**eature knowledge $F_k = \{i \in 1 \ldots n : x_i \text{ is known}\}$: the subset of features known to the adversary.
- **A**lgorithm knowledge A': the learning algorithm that the adversary uses to craft poison samples.
- **I**nstance knowledge S': the labeled training instances available to the adversary.
- **L**everage $F_m = \{i \in 1 \ldots n : x_i \text{ is modifiable}\}$: the subset of features that the adversary can modify.

- **F** and **A** dimension constrain the hypothesis space
- **I** dimension affect the accuracy of instance space
- **L** dimension affects ability to craft attack instance

FAIL attacker model

F dimension: *What features could be kept as a secret? Could the attacker access the exact feature values?*

A dimension: *Is the algorithm class known? Is the training algorithm secret? Are the classifier parameters secret?*

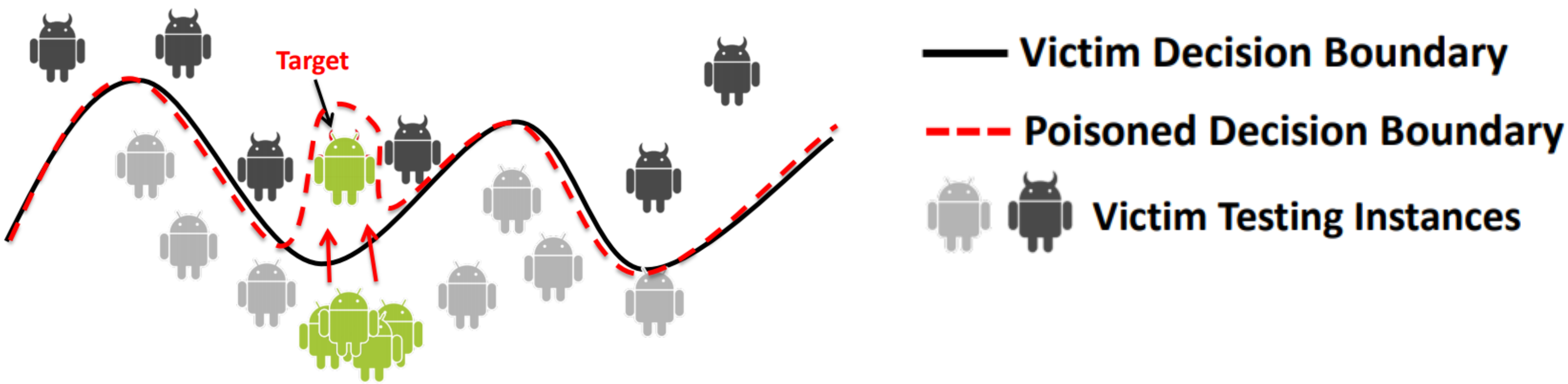
I dimension: *Is the entire training set known? Is the training set partially known? Are the instances known to the attacker sufficient to train a robust classifier?*

L dimension: *Which features are modifiable by the attacker? and What side effects do the modifications have?*

Constraints

1. Poison samples must have clean labels
   - Adversary can inject training samples but cannot determine how they labeled
2. Poison samples must be individually inconspicuous
   - Similar to the existing training instances
3. Poison samples must be individually inconspicuous
4. Poison samples must exhibit a generalized form of transferability
   - Adversary tests the samples on a surrogate model, trained with partial knowledge along multiple dimensions, defined by the FAIL model

StingRay Attack



StingRay is a general framework for crafting poison samples

StingRay is model agnostic (CNN, SVM, random forest)

StingRay Attack Algorithm

Algorithm 1 The StingRay attack.

procedure $\text{StingRay}(S', Y_{S'}, \mathbf{t}, y_t, y_d)$
  $I = \varnothing$
  $h = A'(S')$
  repeat
    $\mathbf{x_b} = \text{GetBaseInstance}(S', Y_{S'}, \mathbf{t}, y_t, y_d)$
    $\mathbf{x_c} = \text{CraftInstance}(\mathbf{x_b}, \mathbf{t})$
    if $\text{GetNegativeImpact}(S', \mathbf{x_c}) < \tau_{NI}$ **then**
      $I = I \cup \{\mathbf{x_c}\}$
      $h = A'(S' \cup I)$
  until $(|I| > N_{min}$ **and** $h(\mathbf{t}) = y_d)$ **or** $|I| > N_{max}$
  $PDR = \text{GetPDR}(S', Y_{S'}, I, y_d)$
  if $h(\mathbf{t}) \neq y_d$ **or** $PDR < \tau_{PDR}$ **then**
    return $\varnothing$
  return I
procedure $\text{GetBaseInstance}(S', Y_{S'}, \mathbf{t}, y_t, y_d)$
  for $\mathbf{x_b}, y_b$ **in** $\text{Shuffle}(S', Y_{S'})$ **do**
    if $D(\mathbf{t}, \mathbf{x_b}) < \tau_D$ **and** $y_b = y_d$ **then**
      return $\mathbf{x_b}$

S.I $h(\mathbf{t}) = y_d$: the desired class label for target

S.II $D(\mathbf{t}, \mathbf{x_b}) < \tau_D$: the inter-instance distance metric

D.III $\bar{s} = \frac{1}{|I|} \sum_{\mathbf{x_c} \in I} s(\mathbf{x_c}, \mathbf{t})$, where $s(\cdot, \cdot)$ is a *similarity* metric: crafting target resemblance

D.IV $NI < \tau_{NI}$: negative impact of poisoning instances

S.V $PDR < \tau_{PDR}$: the perceived performance drop

D.VI $|I| \geq N_{min}$: the minimum number of poison instances

B.VII $|I| \leq N_{max}$: maximum number of poisoning instances

Example of Crafting Instances – Android Malware Detector Drebin

```
api_call::setWifiEnabled
permission::WRITE_CONTACTS
permission.CALL_PHONE
permission::ACCESS_WIFI_STATE
permission::READ_CONTACTS
intent.action.SEARCH
intent.action.MAIN
```

VirusTotal highlights some features as more **suspicious** than others

malicious label

Example of Crafting Instances – Choosing a Base Instances

```
api_call::setWifiEnabled
permission::WRITE_CONTACTS
permission.CALL_PHONE
permission::ACCESS_WIFI_STATE
permission::READ_CONTACTS
intent.action.SEARCH
intent.action.MAIN
```

```
api_call::setWifiEnabled
permission::ACCESS_WIFI_STATE
activity::MainActivity
permission::READ_CONTACTS
```

Choose base instances with some similarity to target

benign label

Web Security & Privacy Lab

KAIST

Example of Crafting Instances – Individual Inconspicuousness

api_call::setWifiEnabled
permission::WRITE_CONTACTS
permission.CALL_PHONE
permission::ACCESS_WIFI_STATE
permission::READ_CONTACTS
intent.action.SEARCH
intent.action.MAIN

api_call::setWifiEnabled
permission::ACCESS_WIFI_STATE
activity::MainActivity
permission::READ_CONTACTS

Reusing existing instances mitigates lack of leverage on some features

Web Security & Privacy Lab

KAIST

Example of Crafting Instances – Collective Inconspicuousness

```
api_call::setWifiEnabled
permission::WRITE_CONTACTS
permission.CALL_PHONE
permission::ACCESS_WIFI_STATE
permission::READ_CONTACTS
intent.action.SEARCH
intent.action.MAIN
```

```
api_call::setWifiEnabled
permission::ACCESS_WIFI_STATE
activity::MainActivity
permission::READ_CONTACTS
```

Poison instances bypass three defenses: RONI, targeted RONI and Micromodels

```
api_call::setWifiEnabled
permission::ACCESS_WIFI_STATE
activity::MainActivity
permission::READ_CONTACTS
```

```
api_call::setWifiEnabled
permission::ACCESS_WIFI_STATE
activity::MainActivity
permission::READ_CONTACTS
```

Web Security & Privacy Lab

KAIST

Example of Crafting Instances – Uncontrolled Labels

```
api_call::setWifiEnabled
permission::WRITE_CONTACTS
permission.CALL_PHONE
permission::ACCESS_WIFI_STATE
permission::READ_CONTACTS
intent.action.SEARCH
intent.action.MAIN
```

```
api_call::setWifiEnabled
permission::ACCESS_WIFI_STATE
activity::MainActivity
permission::READ_CONTACTS
intent.action.MAIN
```

89% of the 19,000 crafted apps are labeled as benign by VirusTotal

```
api_call::setWifiEnabled
permission::ACCESS_WIFI_STATE
activity::MainActivity
permission::READ_CONTACTS
```

```
api_call::setWifiEnabled
permission::ACCESS_WIFI_STATE
activity::MainActivity
permission::READ_CONTACTS
intent.action.SEARCH
```

unknown label

Attack implementation: Image classification

- Dataset:
  - Neural network (NN) based image classification
  - CIFAR-10, 60,000 RGB images of 32x32 pixels
  - Split into 10 classes
  - 50,000 instances for training, 10,000 instances for testing
  - the attacker has an image t with true label $\boldsymbol{y}_{\mathbf{t}}$ (e.g. a dog) and wishes to trick the model into classifying it as a specific class $\boldsymbol{y}_{\mathbf{d}}$ (e.g. a cat)

Attack implementation: Android Malware Detection

- The Drebin android malware detector
- Linear SVM classifier
- The Drebin dataset
  - 123,453 Android apps
  - 5,560 malware samples
  - 10 AV engines on VirusTotal
  - The feature space has 545,333 dimensions
- Two feature extractor:
  - AndroidManifest XML file
  - Dex file

Attack implementation: Twitter-based exploit prediction

- Features extracted from:
  - Twitter
  - Public vulnerability databases
- The dataset contains:
  - 4,140 instances
  - 268 are labeled as positive
- The classifier uses 72 features from 4 categories:
  - CVSS Score;
  - Vulnerability Database;
  - Twitter traffic;
  - Twitter word features

Attack implementation: Data breach prediction

- Random Forest classifier
- The classifier uses:
  - 2,292 instances
  - 382 positive-labeled examples
- The 74 existing features are extracted from externally observable network misconfiguration symptoms as well as blacklisting information about hosts in an organization's network.

Average Results

#	Δ	SR %	$\bar{\tau}_D$
FAIL:Unknown features			
1	32%	67/3	0.070
2	62%	86/7	0.054
F**A**IL:Unknown algorithm			
3	shallow	99/10	0.035
4	narrow	82/20	0.027
FA**I**L:Unavailable training set			
5	35000	93/18	0.032
6	50000	80/80	0.026
FA**I**L:Unknown training set			
7	45000	90/18	0.029
8	50000	96/19	0.034
FAI**L**:Read-only features			
9	18%	80/4	0.011
10	41%	80/34	0.022
11	62%	80/80	0.026

Table 3: JSMA on the image classifier

Δ	SR %	PDR	Instances
FAIL:Unknown features			
39%	87/63/67	0.93/0.96/0.96	8/4/10
66%	84/71/74	0.94/0.95/0.95	8/4/9
F**A**IL:Unknown algorithm			
shallow	83/65/68	0.97/0.97/0.96	17/14/15
narrow	75/67/72	0.96/0.97/0.96	20/16/17
FA**I**L:Unavailable training set			
35000	73/68/76	0.97/0.96/0.96	17/16/14
50000	78/70/74	0.97/0.97/0.97	18/16/15
FA**I**L:Unknown training set			
45000	82/69/74	0.98/0.96/0.96	16/10/15
50000	70/62/68	0.95/0.96/0.96	17/8/17
FAI**L**:Read-only features			
25%	80/70/72	0.97/0.97/0.97	19/16/15
50%	80/71/76	0.97/0.97/0.97	18/16/13
75%	83/78/79	0.97/0.97/0.96	16/16/12

Table 4: StingRay on the image classifier

Δ	SR %	PDR	Instances
FAIL:Unknown features			
109066	79/3/5	0.99/0.99/1.00	73/50/53
327199	77/12/13	0.99/0.99/1.00	51/50/15
F**A**IL:Unknown algorithm			
SGD	42/33/42	0.99/0.99/0.99	65/50/31
dSVM	38/35/48	0.99/0.99/0.99	78/50/61
FA**I**L:Unavailable training set			
8514	69/27/27	0.90/0.99/0.99	57/50/42
85148	50/50/50	0.99/0.99/0.99	77/50/61
FA**I**L:Unknown training set			
8514	53/21/24	0.93/0.99/1.00	62/50/49
43865	36/29/39	1.04/0.99/0.99	100/50/87
FAI**L**:Read-only features			
851	73/12/13	0.67/0.99/1.00	50/50/10
8514	49/16/17	0.90/0.99/1.00	61/50/47
85148	32/32/32	0.99/0.99/0.99	79/50/57

Table 5: StingRay on the malware classifier

Tables 3, 4, 5: FAIL analysis of the two applications. For each JSMA experiment, we report the attack SR (perceived/potential), as well as the mean perturbation $\bar{\tau}_D$ introduced to the evasion instances. For each StingRay experiment, we report the SR and PDR (perceived/actual/potential), as well as statistics for the crafted instances on successful attacks (mean/median/standard deviation). Δ represents the variation of the **FAIL** dimension investigated.

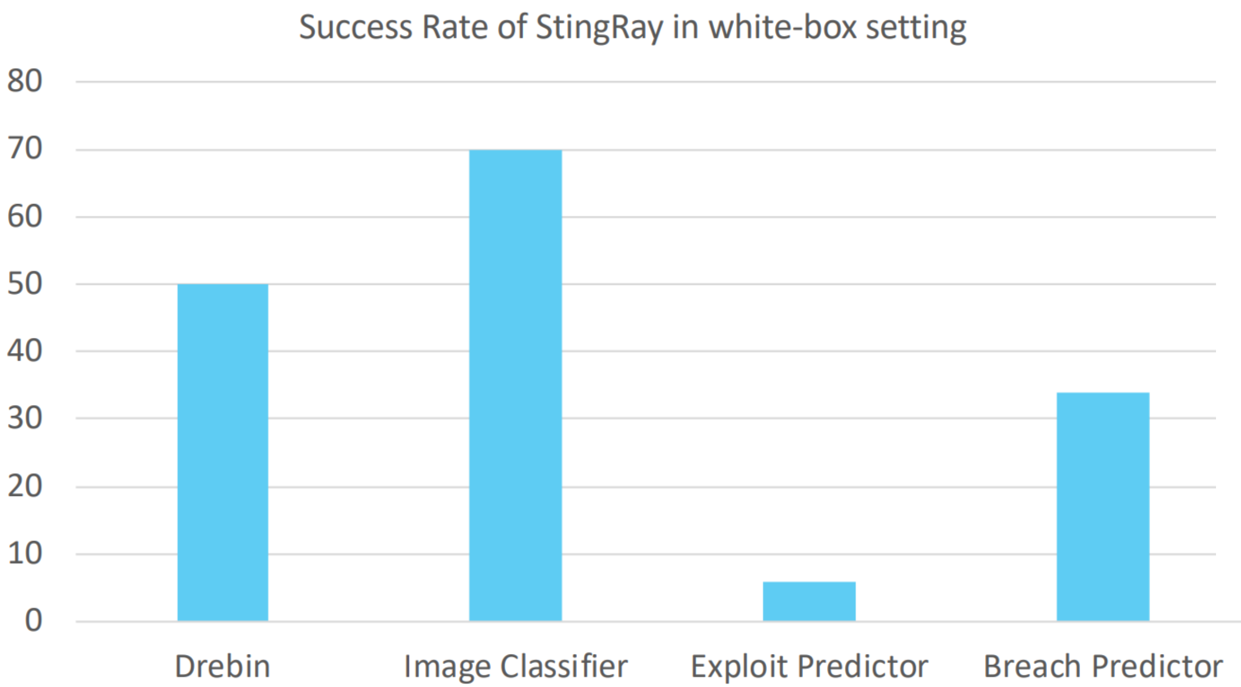
#6 is white box adversary

Web Security & Privacy Lab

KAIST

StingRay – White-Box Performance

Success Rate (SR): Percentage of attacks that are successful on the victim

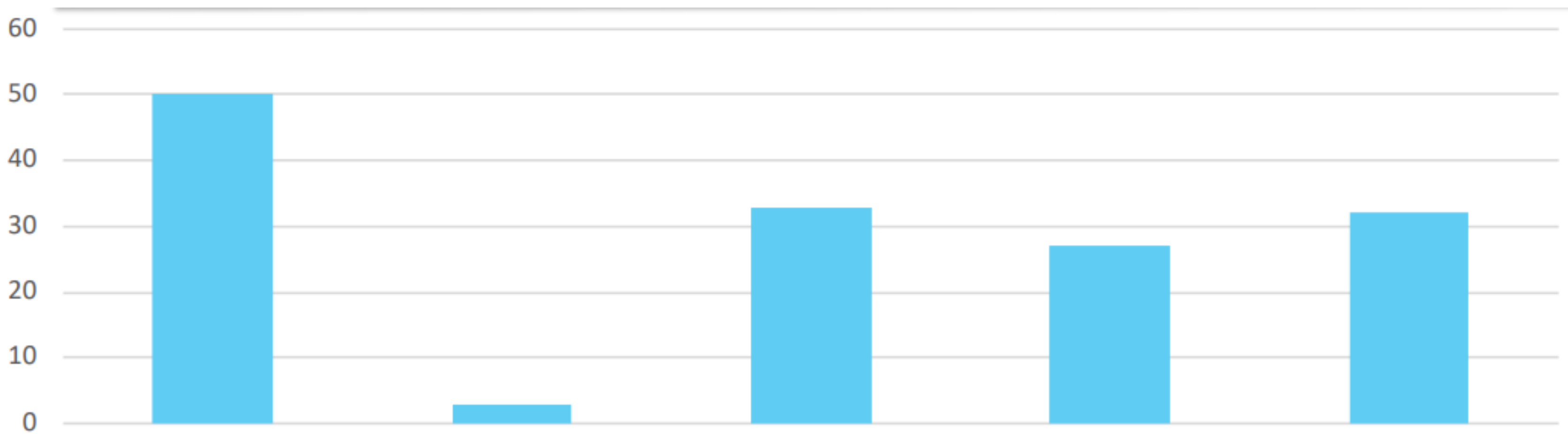


	StingRay
	$\lvert I \rvert$/SR%/PDR
Images	16/70/0.97
Malware	77/50/0.99
Exploits	7/6/1.00
Breach	18/34/0.98

Web Security & Privacy Lab

KAIST

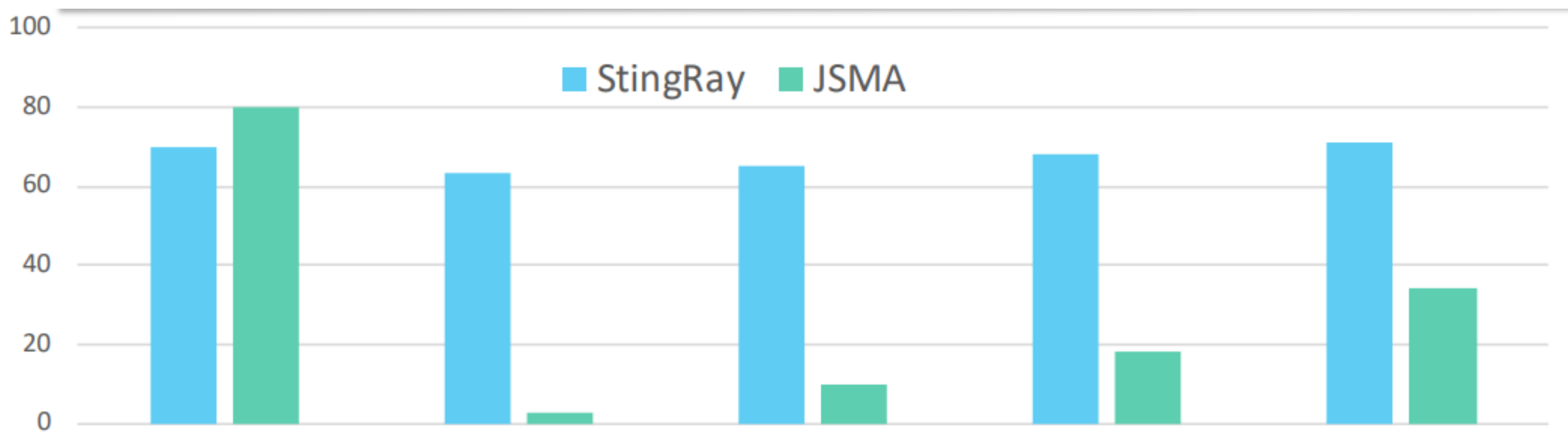
StingRay Attack on Drebin



Full	Limited F	Limited A	Limited I	Limited L
Unconstrained	Random 20% features known	SGD algorithm (vs SVM)	10% training set known	60% read-only features

Feature secrecy appears to be the most powerful limiting factor

StrngRay and JSMA – Image Classifier



Full	Limited F	Limited A	Limited I	Limited L
Unconstrained	30-40% center pixels known	CNN missing one layer	70% training set known	40-50% border pixels read-only

StingRay has high accuracy than JSAM on all dimensions

JSAM is more effective in white-box settings

StrngRay and JSMA – Image Classifier

Discussion!
How about unifying dimension?
For example, limited F+A or I+L

Full	Limited F	Limited A	Limited I	Limited L
Unconstrained	30-40% center pixels known	CNN missing one layer	70% training set known	40-50% border pixels read-only

StingRay has high accuracy than JSAM on all dimensions

JSAM is more effective in white-box settings

Conclusion

- **FAIL** adversary model provides a framework for exposing and systematizing assumptions
- **StingRay** - a targeted poisoning attack designed to bypass existing defenses. It shows that attack is practical for four classification tasks with three different classifiers.
- Feature secrecy as the most prominent factor in reducing the attack success rate