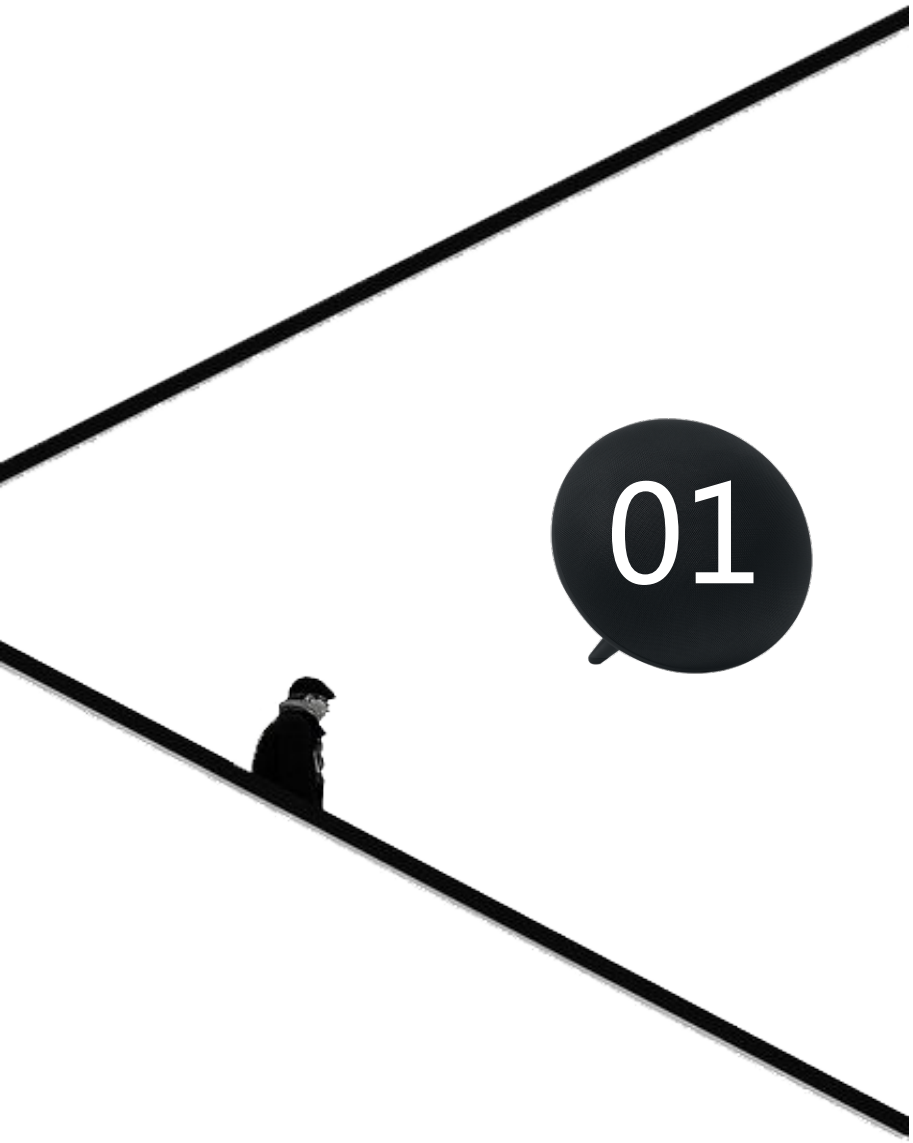
Improving Detection Accuracy For Malicious JavaScript Using GAN

by Junxia Guo

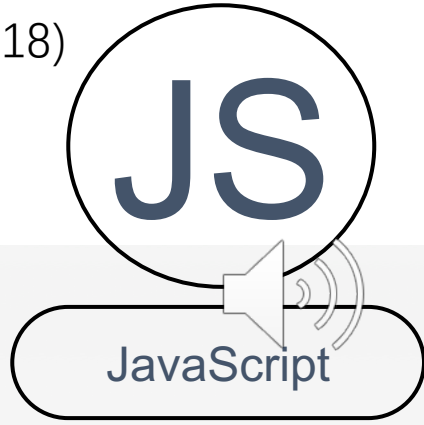
Contents



01 Background

Background

- Web applications are progressively more utilized for security-critical services, they have turned out to be a well-liked and precious target for the web-related vulnerabilities.
- As one of the key technologies to resist network attacks, JavaScript has become an open playground for the attackers to spread malware by injecting malicious JavaScripts. (about 4800 every month in 2018)



Background

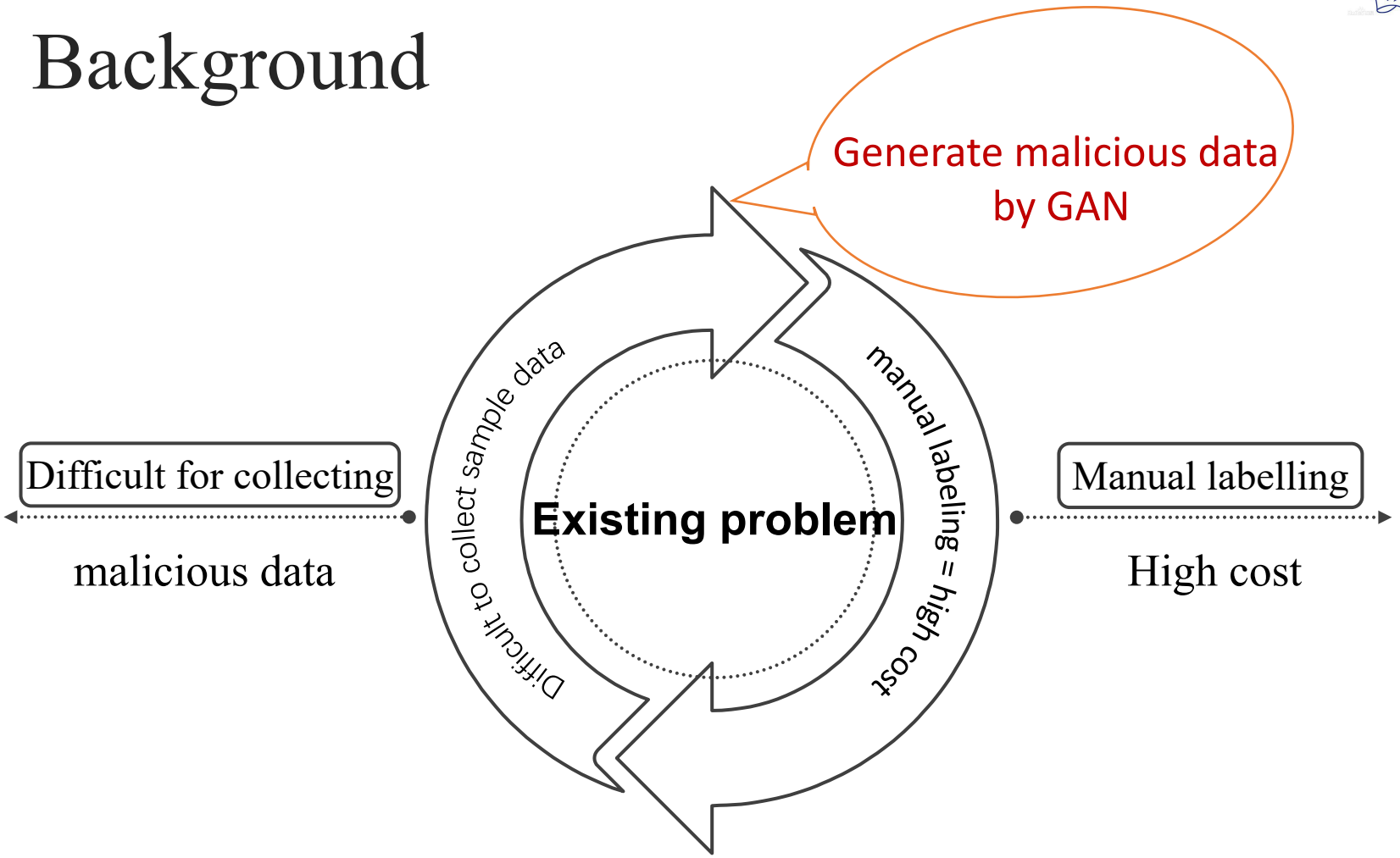
Static methods

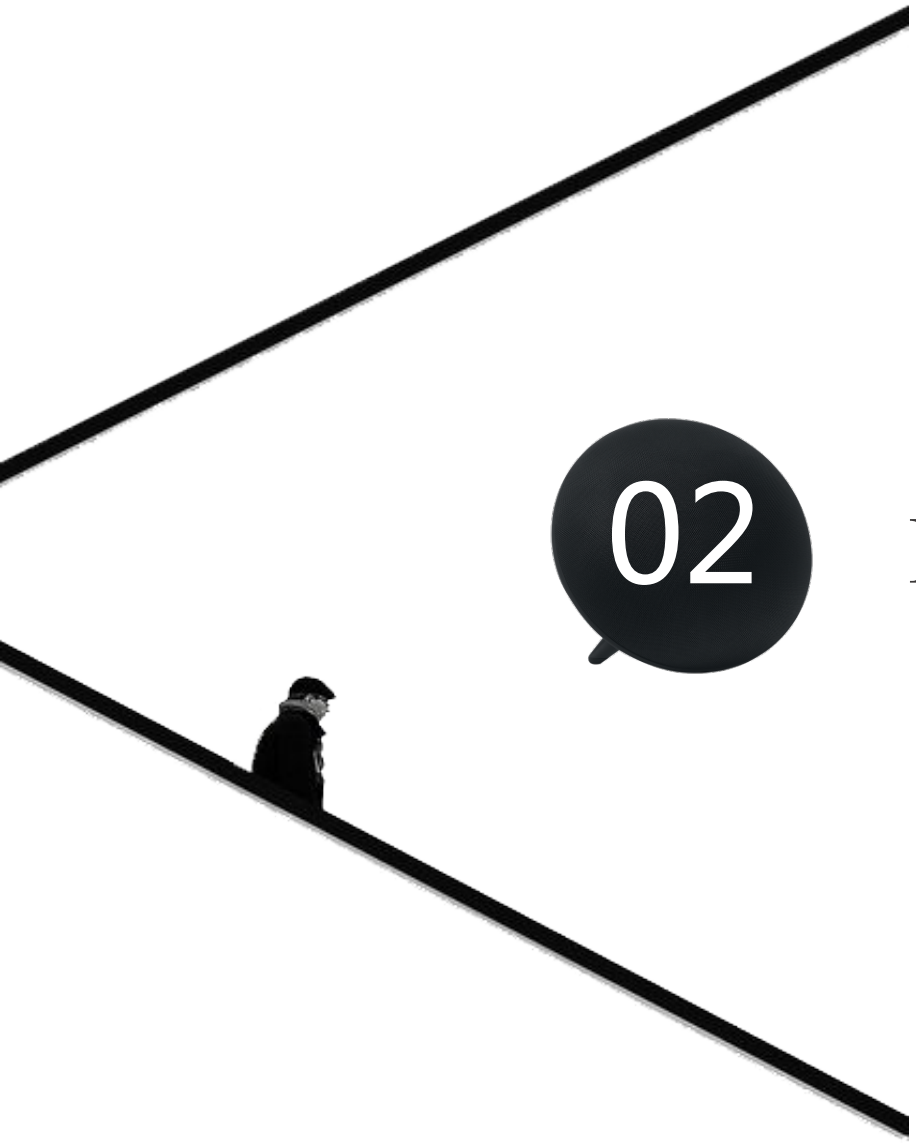
Dynamic methods

Static + Dynamic

- Researchers recently have enlisted machine learning approaches in the detection of malware, which is good at detecting previously unknown malware.
- However, in order to use machine learning in classifying regular and malicious codes, usually we need to collect a large number of regular and malicious JavaScript samples, label them manually, and then perform model training.
- In fact it is difficult to collect sample data.

Background

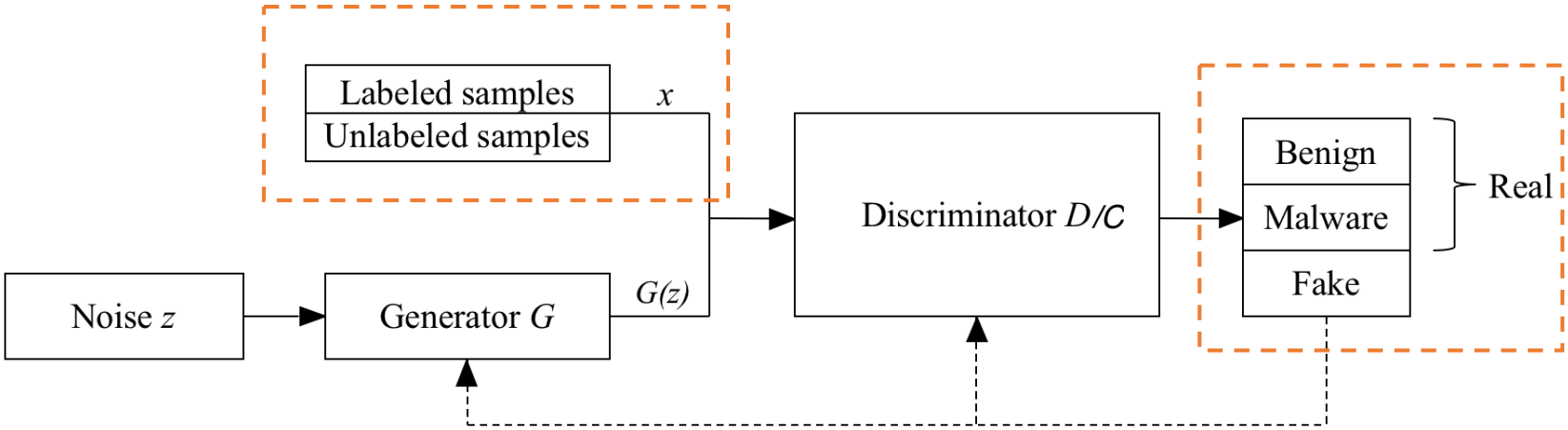

Generate malicious data by GAN
Difficult for collecting
malicious data
Difficult to collect sample data
Existing problem
manual labeling = high cost
Manual labelling
High cost



02

Main research

Main research



Framework for our approach

Main research

Change the Loss Function for our approach

$$L_{supervised} + L_{unsupervised}$$

Main research

$$Loss_D = L_{supervised} + L_{unsupervised}$$

where

$$L_{supervised} = -E_{z\sim p_{data}} log p_{model}(y|x, y\in\{0,1\})$$

$$L_{unsupervised} = -E_{x\sim p_{data}} \log\left(1 - p_{model}(y|x, y=2)\right)$$

$$+E_{x\sim G(z), z\sim noise} \log p_{model}(y|x, y=2)$$

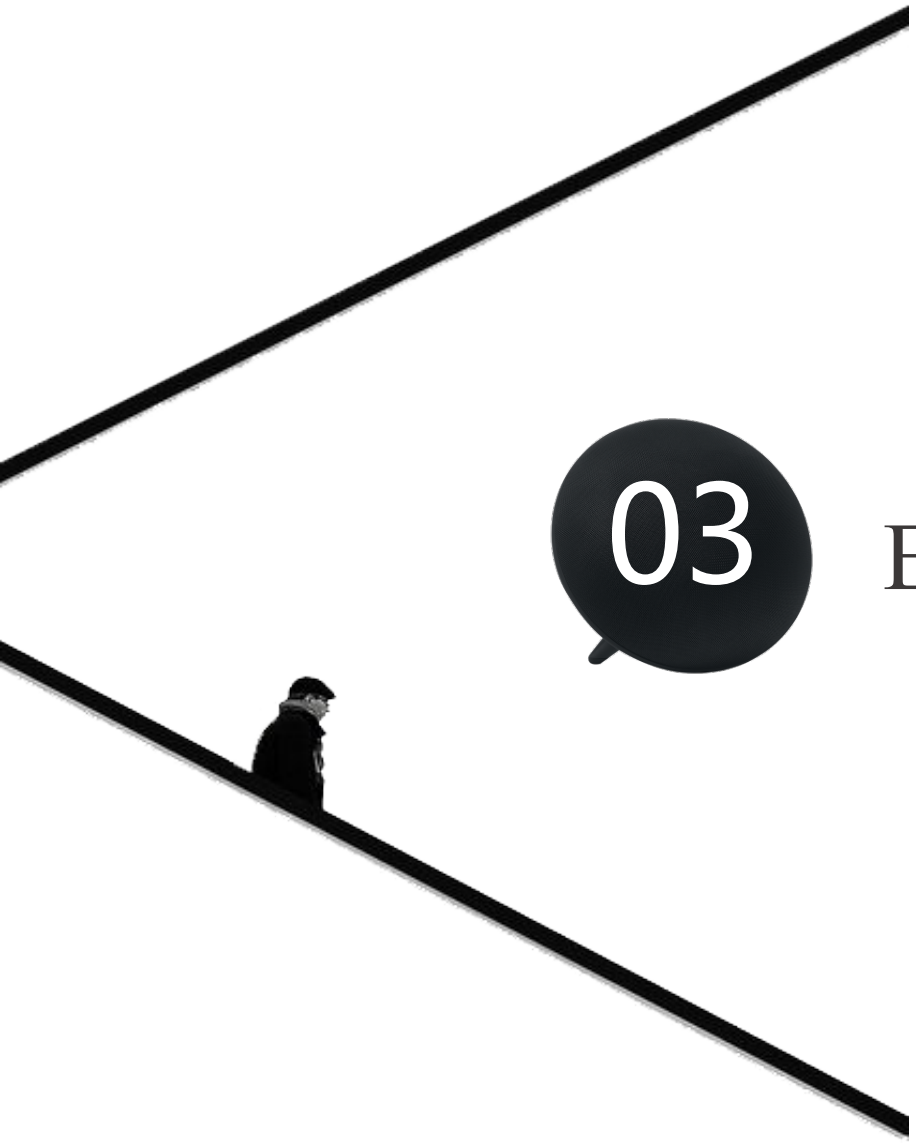
$$Loss_G = E_{z\sim noise} \log D(G(z))$$

set $D(x) = 1 - p_{model}(y|x, y=2)$,

$$L_{unsupervised} = -E_{x\sim p_{data}} \log D(x) + E_{x\sim G(z), z\sim noise} \log\left(1 - D(G(z))\right)$$

Discriminator Loss

Generator Loss



03 Experimental

Experimental

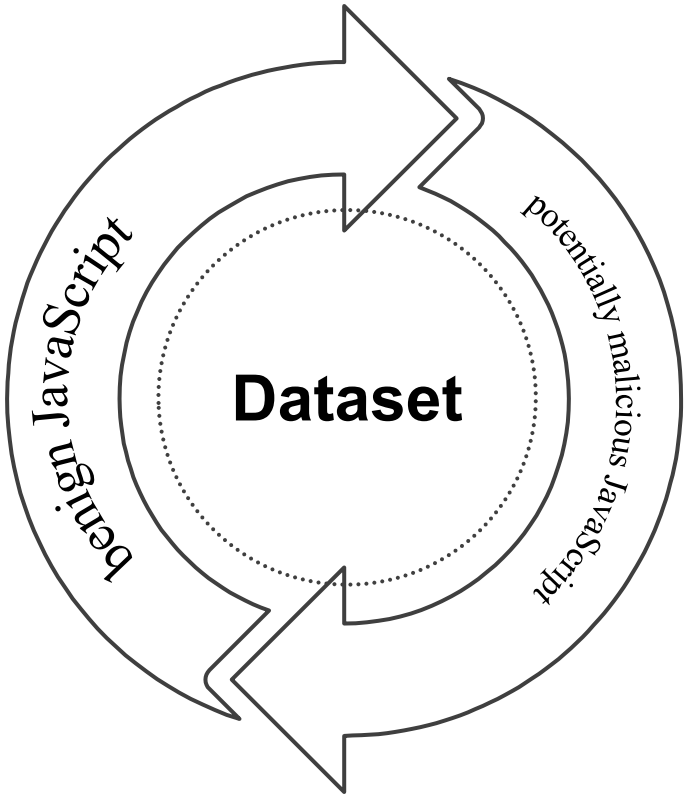
RQ1: Can the method we proposed in this paper make the classifiers work better?

RQ2: Can the method we proposed in this paper be really used in the detection of malicious JavaScript?

Experimental

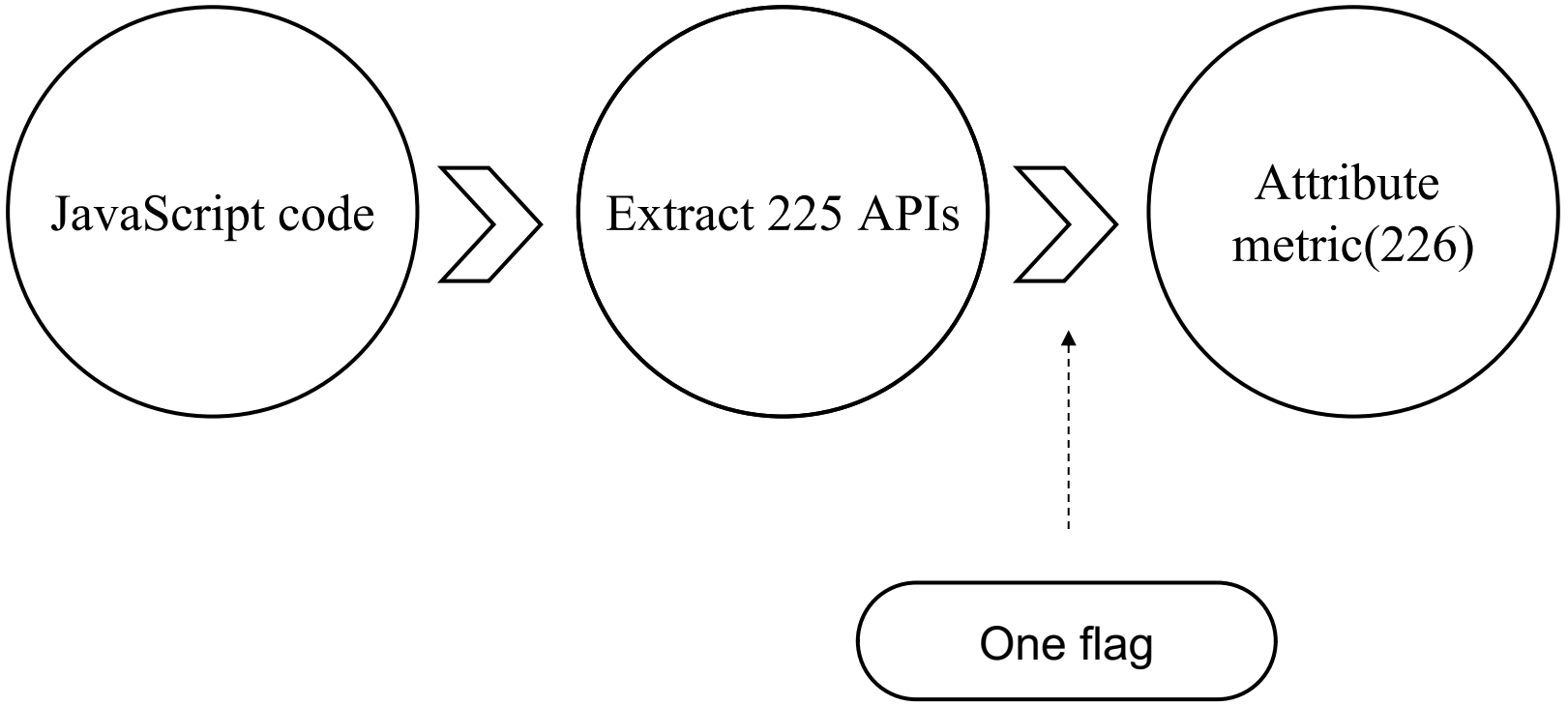
Setup

top 100 web-sites listed in Alexa



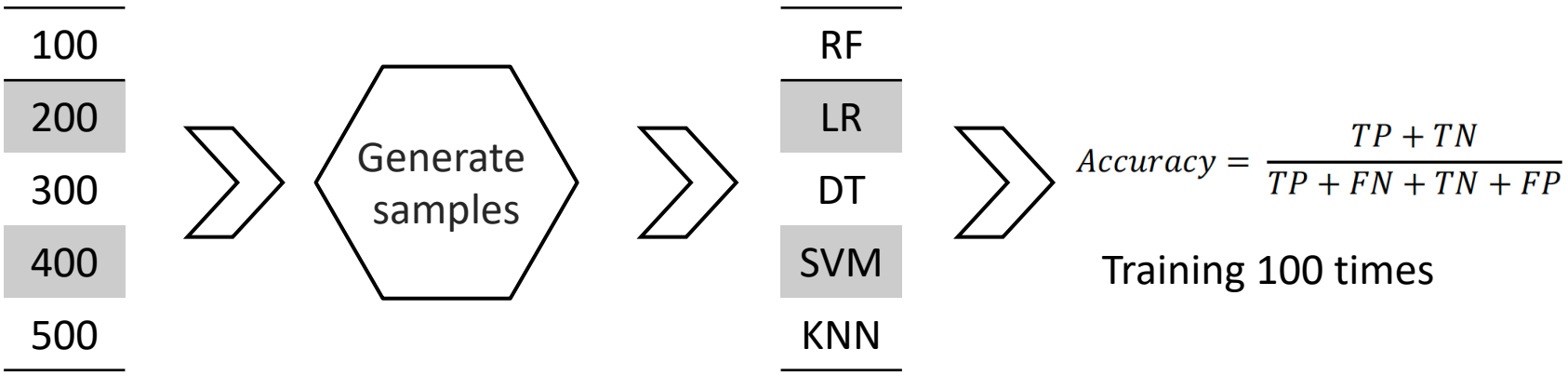
Common blacklists and services to track malicious URLs, for example Harmu

Experimental



Experimental

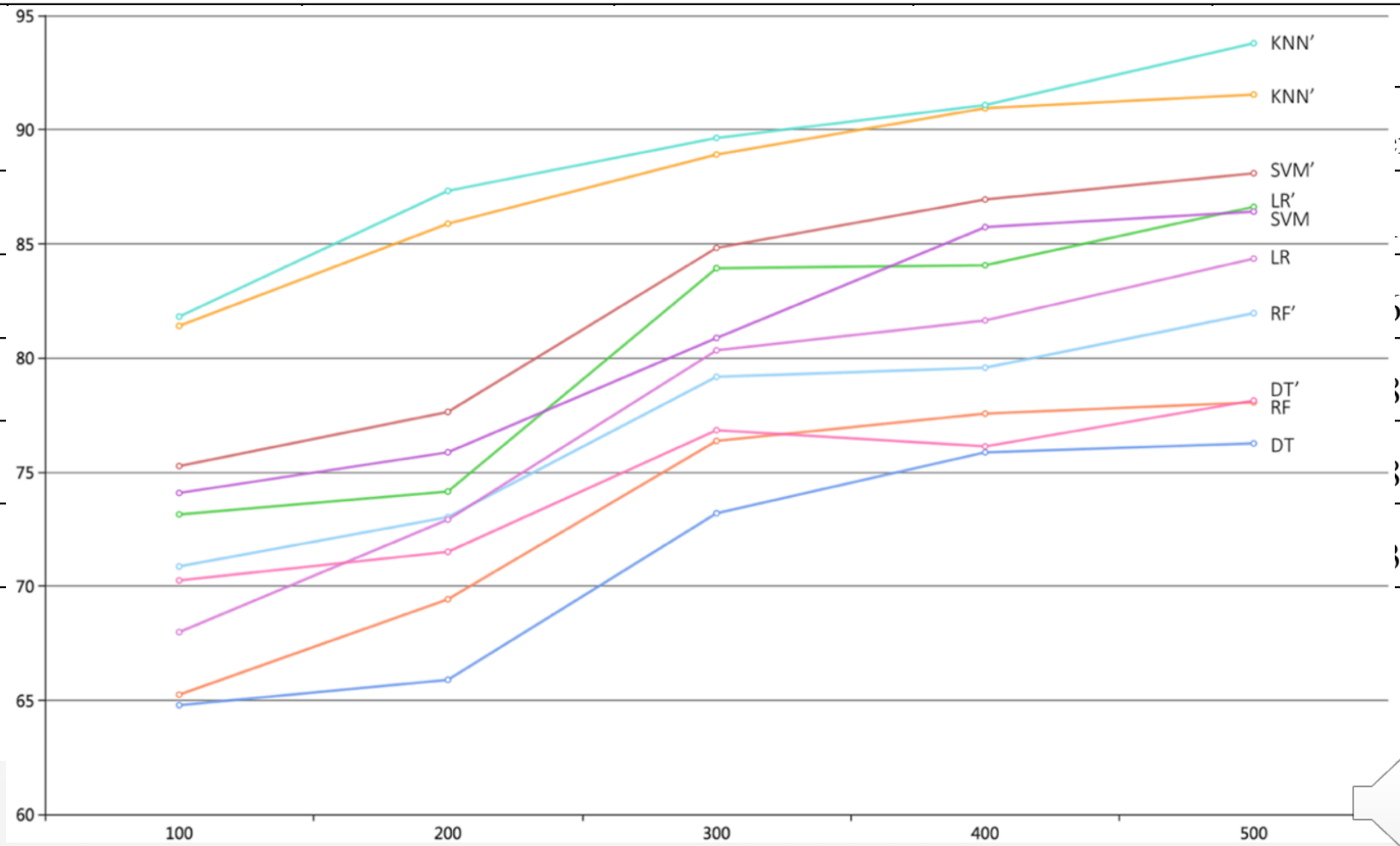
RQ1



Experimental

RQ1

						nerate
RF						.96
LR						5.62
DT						3.13
SVM						3.09
KNN						3.79



Experimental

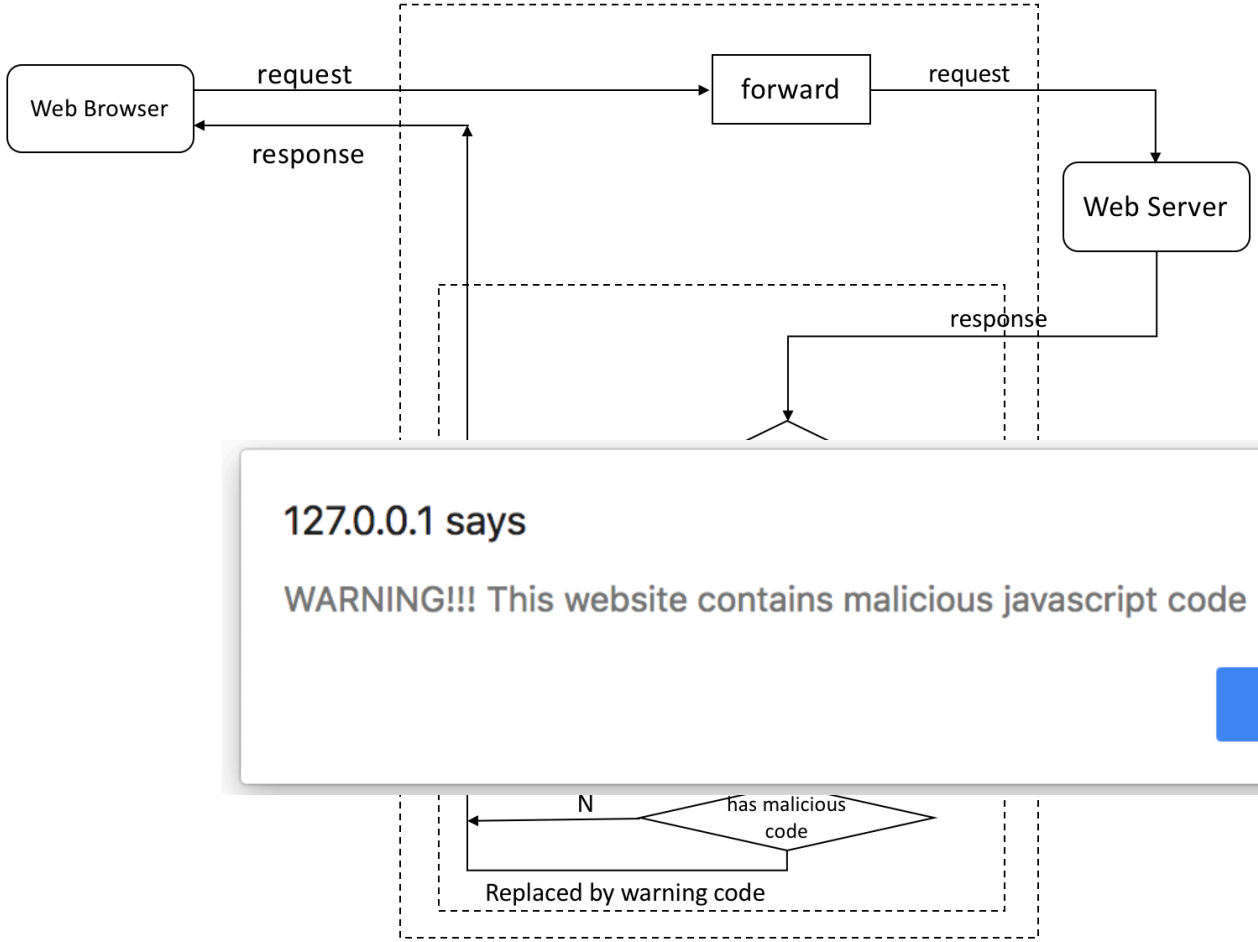
RQ1

- The proposed model has successfully learned to map the data distribution characteristics and can generate samples like the reality for these machine learning based malware detection algorithms.

- Each of the 100, 200, 300, 400, 500 sample sets can be considered as a small set. Thus, we can say that our method has good performance with small set of labeled samples.

Experimental

RQ2



roxy tool

04 Conclusion

Conclusion

- Proposed a method that uses GAN to generate samples and trained a model for detecting malicious JavaScript effectively.
- When using samples generated by GAN the classifiers all have a comparatively better performance with respect to accuracy.
- The MDProxy implemented in this paper can effectively intercept malicious JavaScript code in real-time.

Thanks for your attention!