Robust URL Classification With Generative Adversarial Networks

Martino Trevisan
Idilio Drago
URL classification – What is this?

*URL classification* means **accounting** a URL to a particular class.

- Video Streaming (e.g., Netflix, CNN, …)
- Software Update (Windows, Antivirus)
- Malware

Useful for:

- Content filtering
- Cyber Security
- Accounting / traffic measurements
URL classification – Example

Video Streaming

Fixed pattern
+ Variable parts (CDN nodes, timestamp, ...)

Malware: Tidserv Fast Flux Phishing

Malware: Tidserv Fast Flux Phishing

Fixed pattern
+ Variable parts (CDN nodes, timestamp, ...)

Malware: Tidserv Fast Flux Phishing

Path is pseudo-random
+ Hostname not fixed

Malware: Tidserv Fast Flux Phishing
URL classification – Example

Video Streaming

Classical Solution:
Manually create a Regular Expression

Classical Solution:
Train a classifier provided
(i) samples of the malware
(ii) Normal traffic (non-malware)

Malware: Tidserv Fast Flux Phish

Malware:

Tidserv Fast Flux Phishing

Malware:

wuptywcj.cn/XZc2GhZD7z4ymgo3dmVyPTUuMCZzPTAmYmlkPWE0OTY1YjNlMTFlZmMmYZz13d3cuZ29vZ2xlLml0JnE9dWlzcCZ4ODY9MzI=27x
wuptywcj.cn/xKP3jVbd7qj6s5dmVpTUuMCMYmMzJjYWQ2OGQmYWlkPMTwMDI4JnNpZD0LJnE92Z1pGJhbNnJg4Nj0zMs==27x
rlyg0-6nbcv.com/Kvb13nWd6P4XrFs3dmVyPTQuZDA5N2RiYmRlYmVkZiZhaWQ9NTAwMTgmc2lkPTAmc2dsZS5pdCZxPWZhY2Vib29rZ27c
wuptywcj.cn/KkV3WhhD7m5ZD0WJmVuZz13d3cuZ29vZ2xlLml0JnE9bmF0YWx popupGMltyNjI2xpYSUyMC0lMjB0b3JuJITwHJhZHV6
Generative Adversarial Networks [1]

Unsupervised Learning
• Given training samples, the model learns to generate other samples:
  • Realistic - that look like the original ones
  • But are not exactly the same

Generative Adversarial Networks [1]

Composite model:
1. Generator: take random noise => Generate realistic samples
2. Discriminator: distinguish generated from original samples

They are adversaries, in constant battle during the training process

They are often neural networks

Generative Adversarial Networks
Use GANs for URL classification (and generation)

Key Idea:
• Collect example data of URL of interesting class
• Train a GAN for each class
• Use discriminators to identify new URLs in a live stream of URLs

Pros:
• No manual intervention
• No samples of other URLs / normal data

Cons:
• It is still an empiric model
• If something goes wrong, you do not know why
Use GANs for URL classification (and generation)

For each class of interest:
- Collect example data
  1. Train a GAN – both models are simple feedforward NN
  2. Use the discriminator to classify new URLs
Use GANs for URL classification (and generation)

For each class of interest:
• Collect example data
1. Train a GAN
2. Use the discriminator to classify new URLs

Discriminator Thresholds
New URLs

Classification

abc.com/img.jpg
def.net/a.css
abc.com/a.jpg
def.net/b.css
xyz.com/img.jpg
abc.net/b.jpg
xyz.com/a.jpg
def.net/img.jpg
Our experiments

4 URL classes – real URLs from an operational network

<table>
<thead>
<tr>
<th>Name</th>
<th>#URLs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>8 620</td>
<td>Video Streaming chunks</td>
</tr>
<tr>
<td>Checkpoint</td>
<td>17 451</td>
<td>CheckPoint firewall updates</td>
</tr>
<tr>
<td>Windows</td>
<td>5 277</td>
<td>Windows update archives</td>
</tr>
<tr>
<td>Tidserv</td>
<td>227</td>
<td>TidServ malware</td>
</tr>
<tr>
<td>Others</td>
<td>24 667</td>
<td>Other URLs</td>
</tr>
</tbody>
</table>

Split in:
- Training set: train 4 GANs
- Test set: used at classification time

Used to quantify the ability of classification

We imagine a scenario where you have a deluge of Normal URLs, and we want to pinpoint interesting ones (e.g., content filtering, advanced accounting, …)
Train a GAN for each class. Use trained discriminators on new URLs.

By design, discriminator output is in $[0,1]$. Ideally, we want to have:

- 1: for URLs belonging to the class
- 0: for the others

It is trivial to fix a threshold!!!

We use the boxplot rule [2]

Classification performance

- Take the test set URLs
- Add the Normal URLs
- Compute classical performance metrics

Precision and recall are generally high (>90%)

Not used at training time 😊

Tidserv (fast-flux malware) has high recall, but low precision 😬
That’s because we have only 227 URL sample, and URLs are very variable
Note the high recall 😊
And the generators?

For each URL class, we trained a discriminator and a generator. How generated URLs look like?

High-level patterns are correctly learned 😊

URL semantic is not respected 😞
In conclusion

GANs are a promising technique for unsupervised learning, used mostly for image processing.

We show that can be used also with network data (URLs in this case)
  • Good for classification
  • Promising for generation

Still preliminary work:
  • More Classes, more URLs
  • Comparison with other approaches
  • Scalability?

Possible fields of application in networking
  • Privacy: automatically identify ads/tracker URLs
  • Cyber security: generate / identify attacks
  • Network emulation: realistic 3G/4G emulation