REVERSING AND OFFENSIVE-ORIENTED TRENDS SYMPOSIUM 2019 (ROOTS)
28TH TO 29TH NOVEMBER 2019, VIENNA, AUSTRIA

Shallow Security: on the Creation of Adversarial Variants to Evade Machine Learning-Based Malware Detectors

Fabrício Ceschin
Federal University of Paraná, BR
@fabriciojoc

Marcus Botacin
Federal University of Paraná, BR
@MarcusBotacin

Heitor Murilo Gomes
University of Waikato, NZ
www.heitorgomes.com

Luiz S. Oliveira
Federal University of Paraná, BR
www.inf.ufpr.br/lesoliveira

André Grégio
Federal University of Paraná, BR
@abedgregio

Who am I?

Background

- Computer Science Bachelor (Federal University of Paraná, Brazil, 2015).
- Machine Learning Researcher (Since 2015).
- Computer Science Master (Federal University of Paraná, Brazil, 2017).
- Computer Science PhD Candidate (Federal University of Paraná, Brazil).

Research Interests

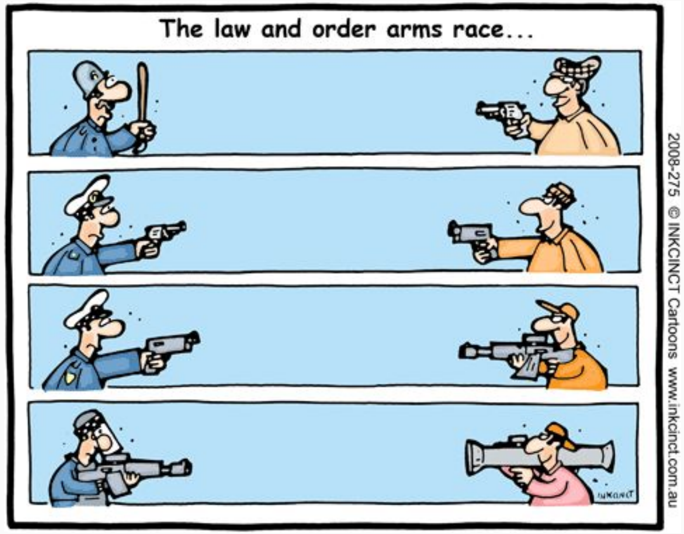
- Machine Learning applied to Security.
- Machine Learning applications:
    - Data Streams.
    - Concept Drift.
    - Adversarial Machine Learning.

Introduction

Motivation, the problem, initial concepts and our work.

The Problem

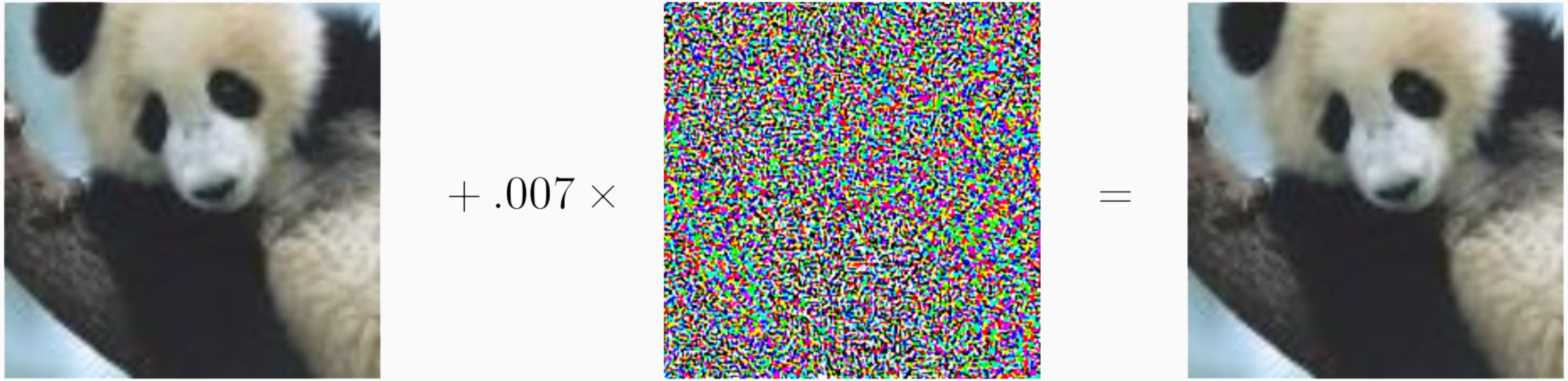
- **Malware Detection:** growing research field.
  - Evolving threats.
- **State-of-the-art:** machine learning-based approaches.
  - Malware classification in families;
  - Malware detection;
  - Dense volume of data (data stream).
- **Arms Race:** attackers VS defenders.
  - Both of them have access to ML.



The Problem

- **Defenders:** developing new classification models to overcome new attacks.
- **Attackers:** generating malware variants to exploit the drawbacks of ML-based approaches.
- **Adversarial Machine Learning:** techniques that attempt to fool models by generating malicious inputs.
    - Making a sample from a certain class being classified as another one.
    - Serious problems for some scenarios, like **malware detection**.

Adversarial Examples



$$\boldsymbol{x} \quad + .007 \times \quad \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)) \quad = \quad \boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$$

$\boldsymbol{x}$	$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$	$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"panda"	"nematode"	"gibbon"
57.7% confidence	8.2% confidence	99.3 % confidence

Adversarial Examples

- **Image Classification:** adversarial image should be **similar** to the original one and yet be classified as being from another class.
- **Malware Detection:** adversarial malware should **behave** the same and yet be classified as **goodware**.
- **Challenge:** automatically generating a fully functional adversarial malware may be difficult.
  - Any modification can make it behave different or not work.

Our Work: How did everything start?

- **Machine Learning Static Evasion Competition:** modify fifty malicious binaries to evade up to **three** open source malware models.
- Modified malware samples must retain their **original functionality**.
- **The prize:** NVIDIA Titan-RTX.

ENDGAME. VMRAY MRG Effitas EFFICACY ASSESSMENT & ASSURANCE

Our Work: What did we do?

- We bypassed **all** the three models creating modified versions of the 50 samples originally provided by the organizers.
- Implemented an **automatic exploitation method** to create these samples.
- Adversarial samples also bypassed **real anti-viruses** as well.
- **Objective:** investigate models robustness against adversarial samples.
- **Results:** models have severe weaknesses so that they can be easily bypassed by attackers motivated to exploit real systems.
    - Insights that we consider important to be shared with the community.

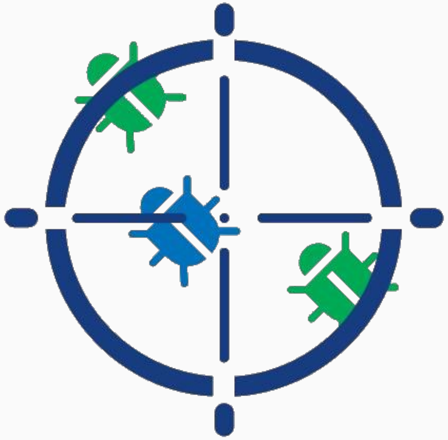
The Challenge

Rules, dataset and models.

The Challenge: How did it work?

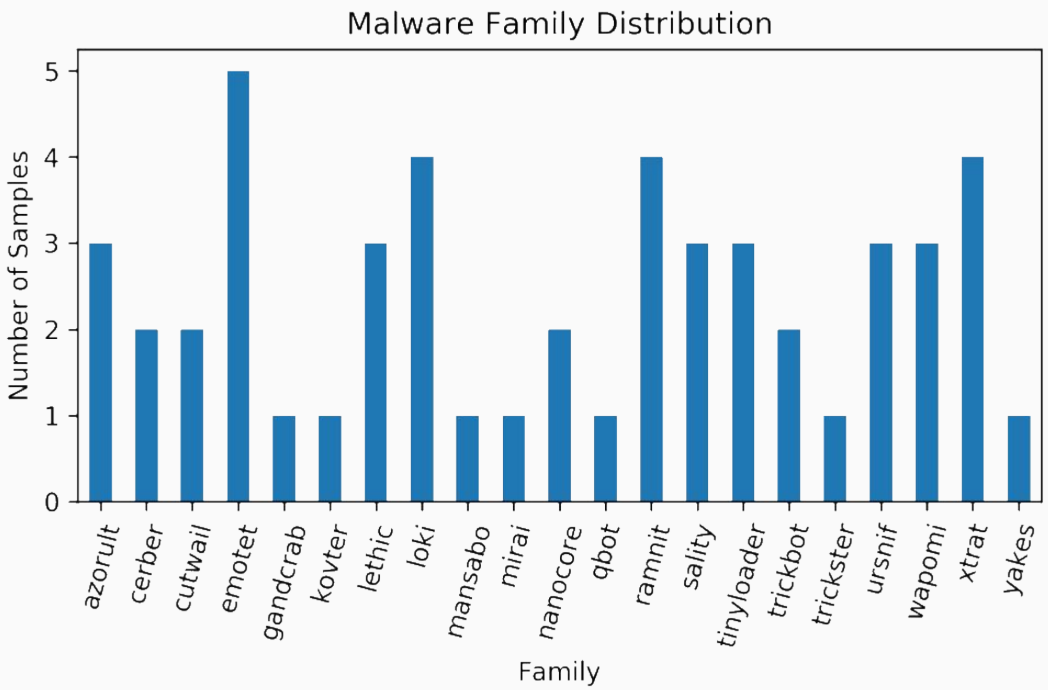
- **Fifty** binaries are classified by three distinct ML models.
- Each bypassed model for each binary accounts for **one** point (**150** points in total).
- All binaries are executed on a **sandboxed environment** and must produce the same **Indicators of Compromise** as the original ones.
- Our team figured among the **top-scorer** participants.
  - Second position!

Nickname	Total best score per user
Will	150
deep_secret	150
Jakub	133

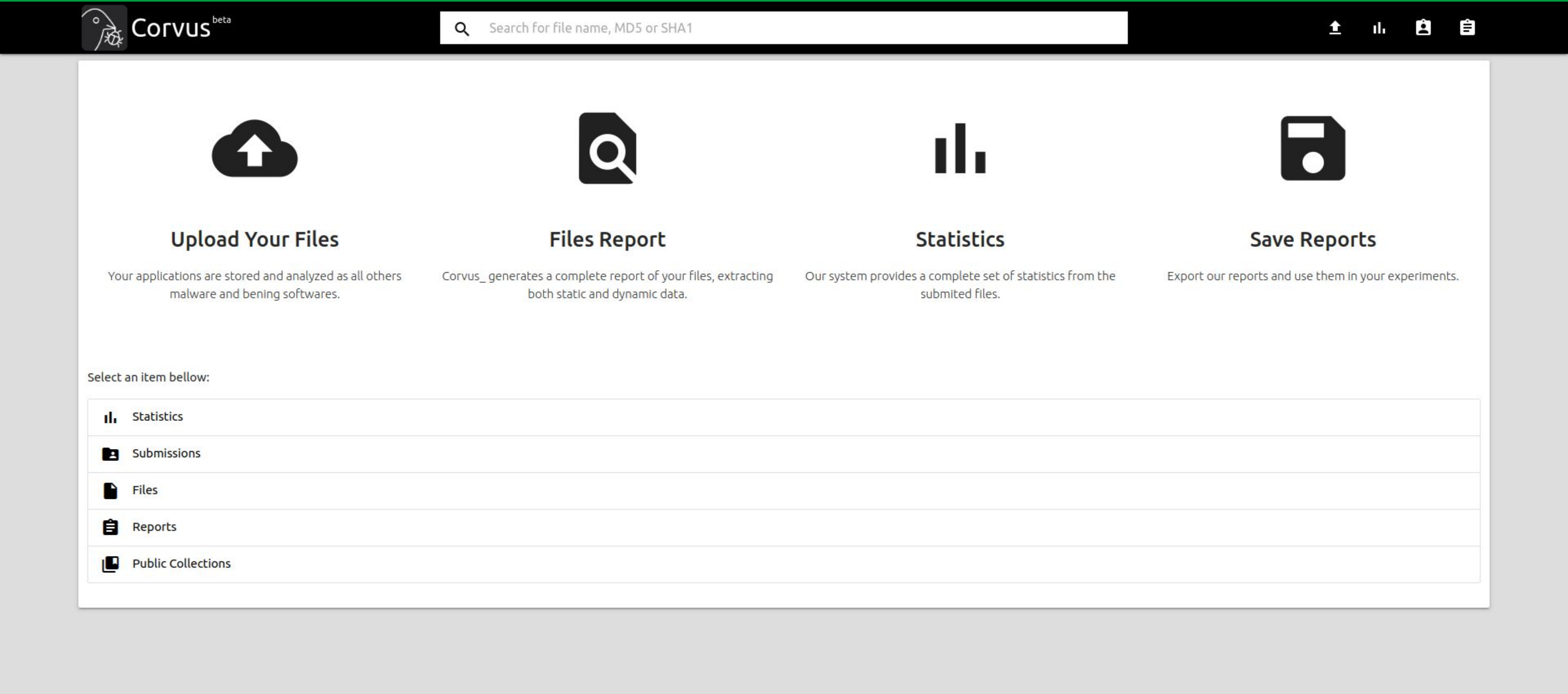


Dataset: Original Malware Samples

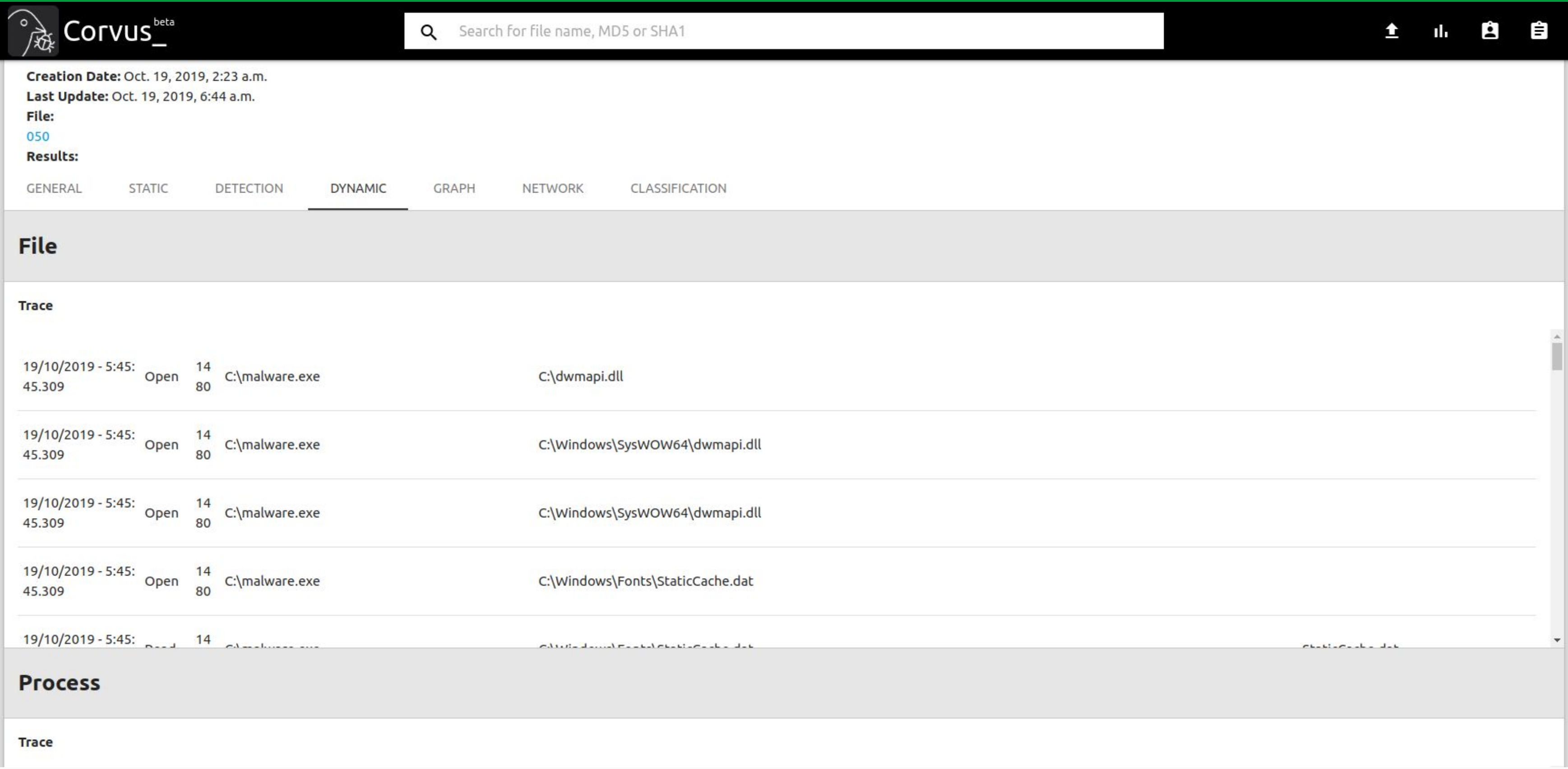
- **Fifty** PE (Portable Executable) samples of varied malware families for Microsoft Windows.
  - Diversified approaches to bypass sample's detection.
- **VirusTotal** & **AVClass**: 21 malware families.
- Real malware samples executed in sandboxed environments.



Corvus: Our Malware Analysis Platform

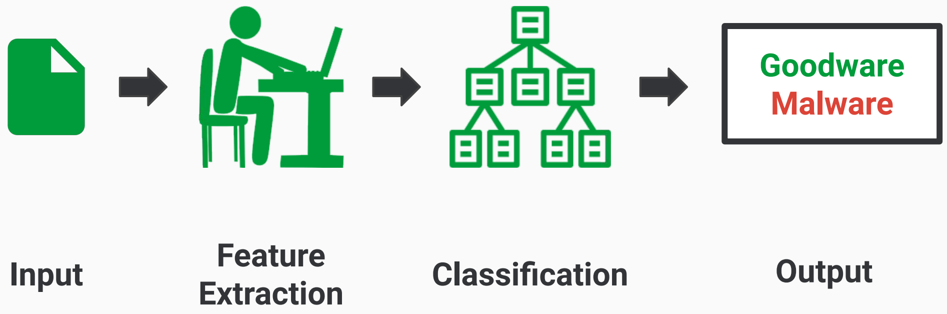


Corvus: Report Example



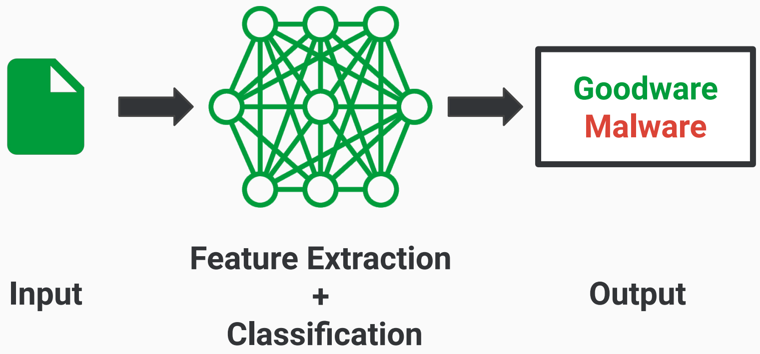
Machine Learning Models: LightGBM

- Gradient boosting decision tree using a feature matrix as input.
- Hashing trick and histograms based on binary files characteristics (PE header information, file size, timestamp, imported libraries, strings, etc).



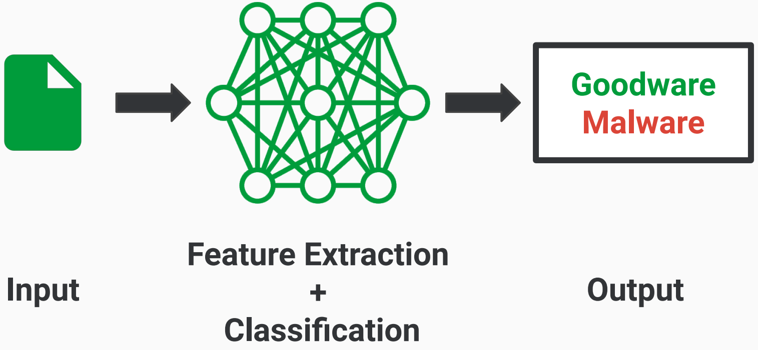
Machine Learning Models: MalConv

- End-to-end deep learning model using raw bytes as input.
- Representation of the input using an 8-dimensional embedding (autoencoder).
- Gated 1D convolution layer, followed by a fully connected layer of 128 units.
- Softmax output for each class.



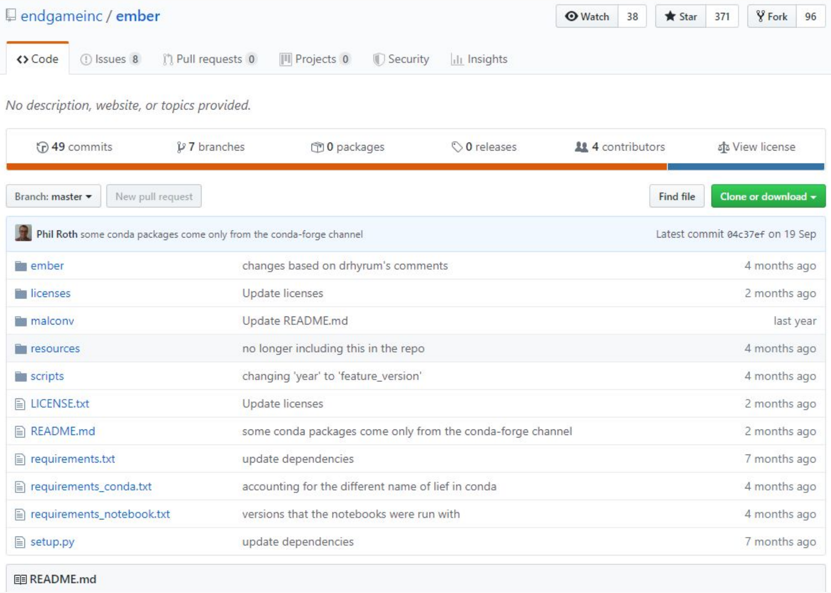
Machine Learning Models: Non-Negative MalConv

- Identical structure to MalConv.
- **Only non-negative weights:** force the model to look only for malicious evidences rather than looking for both malicious and benign ones.



Dataset used to Train the Models

- Ember 2018 dataset.
- Benchmark for researchers.
- 1.1M Portable Executable (PE) binary files:
  - 900K training samples;
  - 200K testing samples.
- Open Source dataset:
  - https://github.com/endgameinc/ember



Corvus: Classifying Samples Submitted Using Machine Learning Models

Corvus beta

Search for file name, MD5 or SHA1

Report #702

Creation Date: Oct. 19, 2019, 2:23 a.m.
Last Update: Oct. 19, 2019, 6:44 a.m.
File:
050
Results:

GENERAL | STATIC | DETECTION | DYNAMIC | GRAPH | NETWORK | CLASSIFICATION

Results

BINARY

KNN (K=3, NFS-BRMalware)	**confidence**: 100.00% **suspicious**: False	**Decision Tree (NFS-BRMalware)**	**confidence**: 100.00% **suspicious**: False
SVC (Kernel=Linear, NFS-BRMalware)	**confidence**: 49.08% **suspicious**: False	**MalConv (Ember: Raw Bytes, Threshold=0.5)**	**confidence**: 99.92% **suspicious**: True
Random Forest (100 estimators, NFS-BRMalware)	**confidence**: 61.00% **suspicious**: True	**Non-Negative MalConv (Ember: Raw Bytes, Threshold=0.35)**	**confidence**: 58.13% **suspicious**: True
LightGDM (Ember: File Characteristics, Threshold=0.8336)	**confidence**: 100.00% **suspicious**: True		

Biased Models?

- How does these models perform when classifying files of a pristine Windows installation?
- **Raw data:** high False Positive Rate (FPR) when handling benign data.

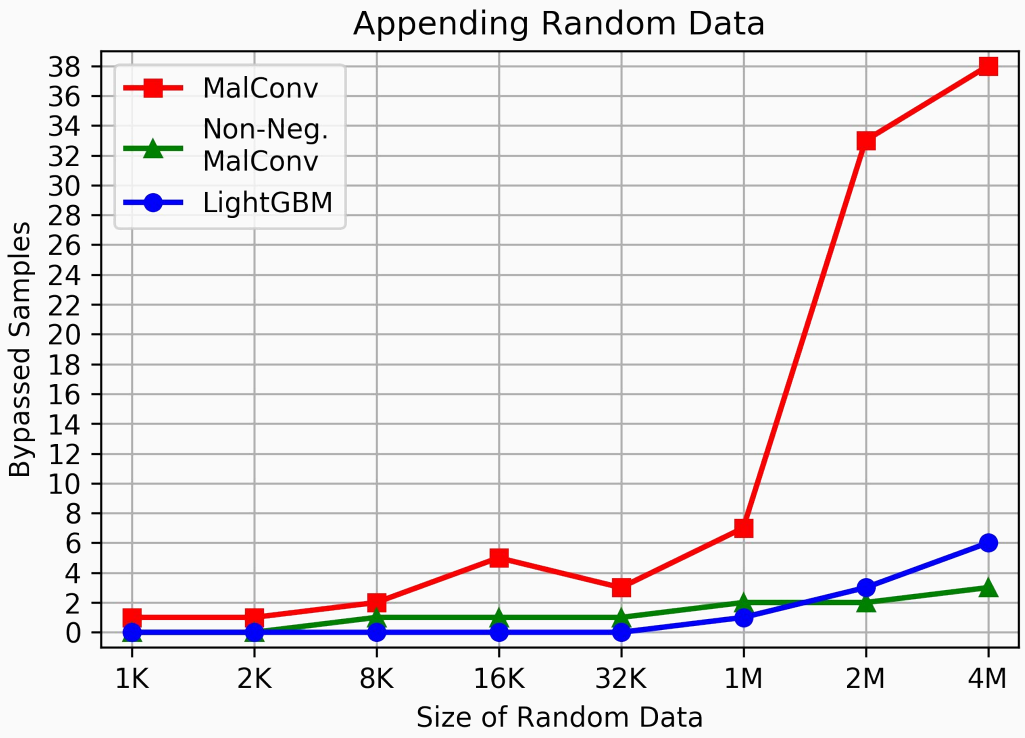
FileType	False Positive Rate (FPR)		
	MalConv	**Non-Neg. MalConv**	**LightGBM**
EXEs	71.21%	87.72%	0.00%
DLLs	56.40%	80.55%	0.00%

Model's Weaknesses

Series of experiments to identify model's weaknesses.

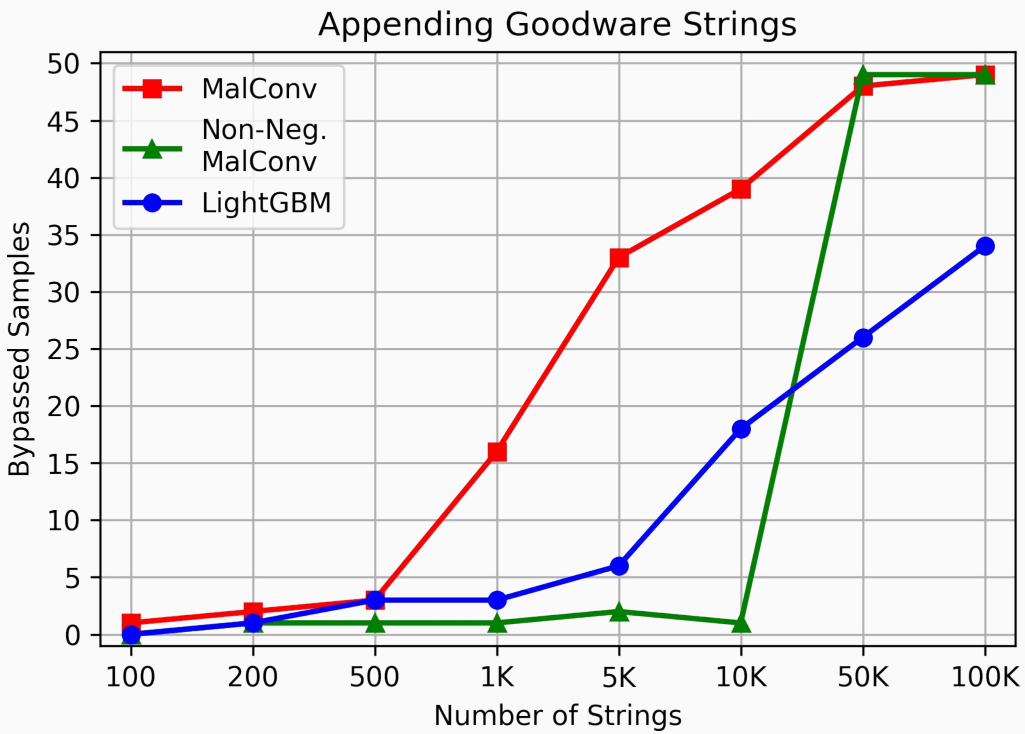
Appending Random Data

- Generating growing chunks of random data, up to the limit of 5MB defined by the challenge.
  - MalConv, based on raw data, is more susceptible to this strategy.
  - Severe for chunks greater than 1MB.
  - Some features and models might be more robust than others.
  - Non-Neg. MalConv and LightGBM were not so affected.



Appending Goodware Strings

- Retrieving strings presented by goodware files and appending them to malware binaries.
- All models are significantly affected when 10K+ strings are appended.
- Result holds true even for the model that also considers PE data (LightGBM), which was more robust in the previous experiment.



Changing Binary Headers

- Replacing header fields of malware binaries with values from a goodware.
    - Version numbers and checksums.
- Decision took by Microsoft when implementing loader: ignores fields.
- Bypassed **only six** samples.
- Model based on PE features learned other characteristics than header values.

```
# open base gw
base_pe = pefile.PE(GWR_BASE)
# iterate over output samples , changing their PE HEADER
for m in adv_list :
    # open adversarial sample
    m_pe = pefile.PE(m)
    # update adversarial header
    m_pe.OPTIONAL_HEADER.MajorLinkerVersion =
    base_pe.OPTIONAL_HEADER.MajorLinkerVersion
    m_pe.OPTIONAL_HEADER.MinorLinkerVersion =
    base_pe.OPTIONAL_HEADER.MinorLinkerVersion
    m_pe.OPTIONAL_HEADER.CheckSum = base_pe.OPTIONAL_HEADER.CheckSum
    m_pe.OPTIONAL_HEADER.MajorOperatingSystemVersion =
    base_pe.OPTIONAL_HEADER.MajorOperatingSystemVersion
    m_pe.OPTIONAL_HEADER.MinorOperatingSystemVersion =
    base_pe.OPTIONAL_HEADER.MinorOperatingSystemVersion
    m_pe.OPTIONAL_HEADER.MajorImageVersion =
    base_pe.OPTIONAL_HEADER.MajorImageVersion
    m_pe.OPTIONAL_HEADER.MinorImageVersion =
    base_pe.OPTIONAL_HEADER.MinorImageVersion
    # write updated sample
    m_pe.write ( m )
```

Packing and Unpacking samples with UPX

- UPX compresses entire PE into other PE sections, changing the external PE binary's aspect.
- Evaluated by packing and unpacking the provided binary samples.
- Classifiers easily bypassed when appending strings to UPX-**extracted** payloads, but not when directly appended to the UPX-**packed** payloads.
- **Bias against UPX packer:** any UPX-packed file is considered malicious.
- **Evaluation:** randomly picking 150 UPX-packed and 150 non-packed samples from malshare database and classified them.

Packing and Unpacking samples with UPX

- UPX-packed versions are more detected by all classifiers.
- Classifiers biased towards the detection of UPX binaries, despite their content.

Dataset	MalConv	Non-Neg MalConv	LightGBM
Originally Packed			
UPX	63.64%	55.37%	89.26%
Extracted UPX	59.50%	53.72%	66.12%
Originally Non-Packed			
Original	65.35%	54.77%	67.23%
UPX Packed	67.43%	56.43%	88.12%

Packing Samples with a Distinct Packer

- Bias against the popular UPX? Use another packer!
- **Evaluation:** packing provided samples with TeLock.
  - Compresses and encrypts the original binary sections into a new one;
  - The original content cannot be identified by the classifiers.
- Proven to be effective, bypassing all models when appending data.
- However, some samples such as the ones from the Extreme RAT family do not execute properly when packed with this solution.

Embedding Samples in a Dropper

- Embedding the binary in a new section, not encrypted nor compressed, avoiding unpacking issues.
- **Evaluation:** embedding samples in the Dr0p1t dropper.
- Along with data appendix, it bypassed all detectors without breaking sample's execution.
- However, it generated binaries greater than 5MB, incompatible with the challenge rules.

Automatic Exploitation

Creating an automatic exploitation method.

Automating Models Exploitation

- Our findings about the models:
    1. Some samples (RATs) do not work well when data is appended.
    2. LightGBM detects when unusual headers and sections are present.
    3. LightGBM model can be bypassed by packing and/or embedding the original binary within a dropper with standard header and sections.
    4. Appending data to packed and embedded samples allows bypassing the Malconv models without affecting the dropped code execution.
- **Objective:** Generate variants able to bypass detection automatically.

Automating Models Exploitation

- Automated the process of packing/embedding all payloads within a new file.
  - Standard header and sections.
- Then, we append goodware data to this file.
- Maximum file size: 5MB.
  - TeLock and Dr0p1t were not an option.
- We implemented our own dropper.
  - Embedding the original malware sample as a PE binary resource.

Dropper

1. Retrieves a pointer to the binary resource (line 3 to 5);
2. Creates a new file to drop the resource content (line 7);
3. Drop the entire content (line 8 to 10);
4. Launches a process based on the dropped file (line 13).

- Bypass all models (data appending).

```
1  int main(){
2      HMODULE h = GetModuleHandle(NULL);
3      HRSRC r = FindResource(h, ...);
4      HGLOBAL rc = LoadResource(h,r);
5      void* data = LockResource(rc);
6      DWORD size = SizeofResource(h,r);
7      FILE *f = fopen("dropped.exe","wb");
8      for(int i=0;i<size;i++){
9          unsigned char c1 = ((char*)data)[i];
10         fprintf(f,"%c",c1);
11     }
12     fclose(f);
13     CreateProcess("dropped.exe", ...);
```

Adversarial Malware Generation: Definition

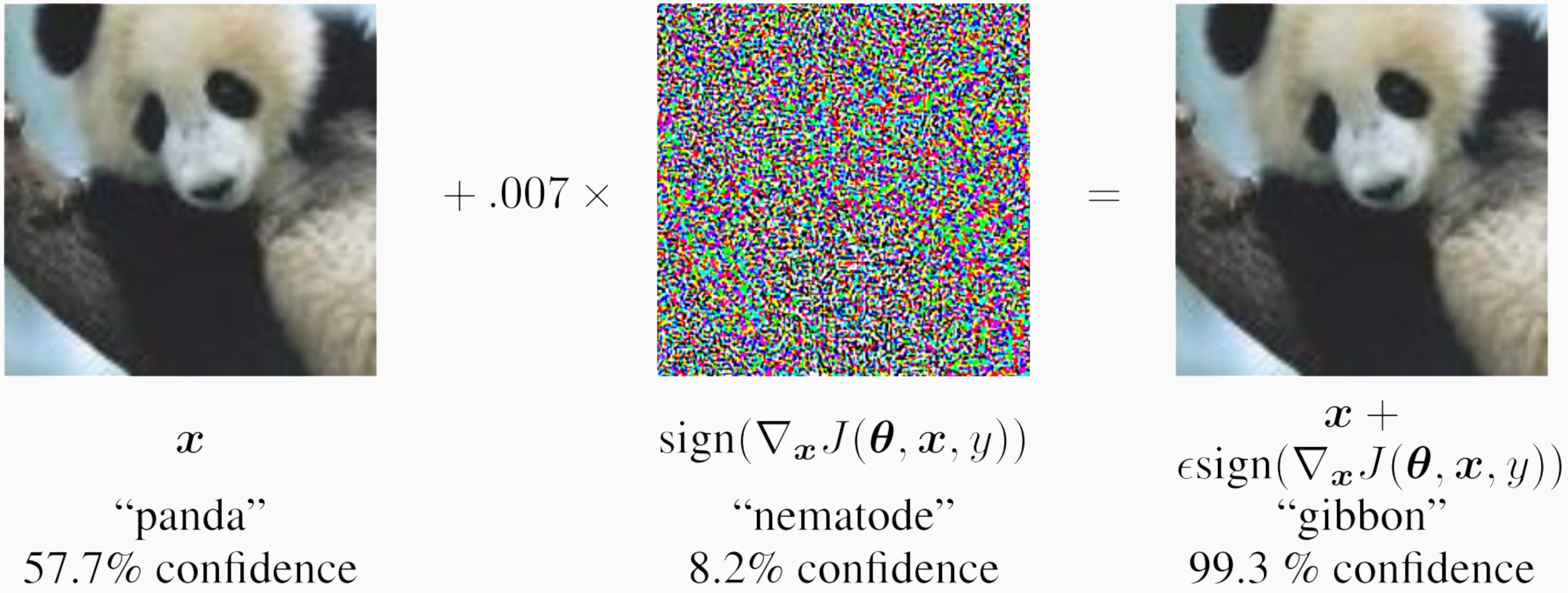
- To generate an adversarial malware ($\mathbf{mw+}$):
  - Original Malware (mw);
    - Input malware file.
  - Embedding Function (f);
    - Generates an entirely new file with standard PE headers and section to host the original malware payload as a resource.
  - Goodware Samples (gw);
    - Set containing n samples: all system files from a pristine Windows installation.
  - Extraction Function ($data$);
    - Retrieve strings and/or bytes information of a file.

Adversarial Malware Generation: Equation

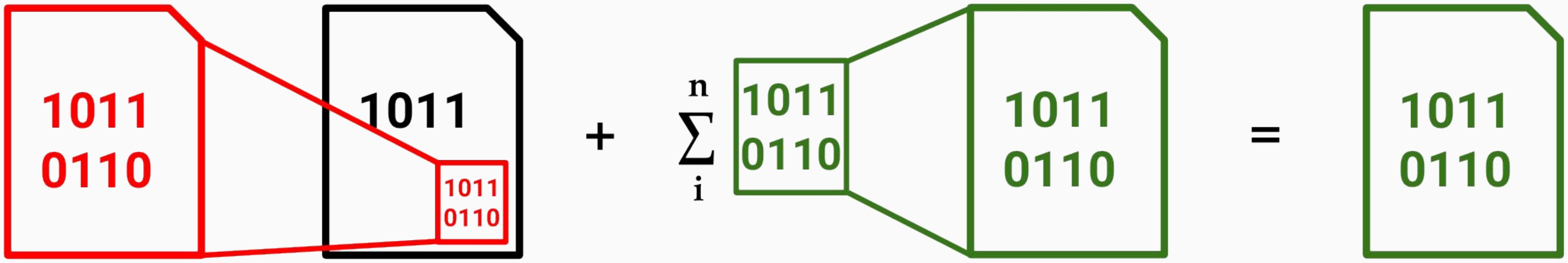
- Extracted chunks $data(gw_i)$ are appended to the new file created using the function $f(mw)$ to ensure a **bias towards the goodware class**.
- Function outcome is an adversarial malware sample ($mw+$).
- Possible to iterate this procedure so as to consider multiple goodware samples, thus repeatedly appending data to the end of $f(mw)$.

$$mw+ = f(mw) + \sum_i^n data(gw_i)$$

Adversarial Malware Generation: Scheme



Adversarial Malware Generation: Scheme



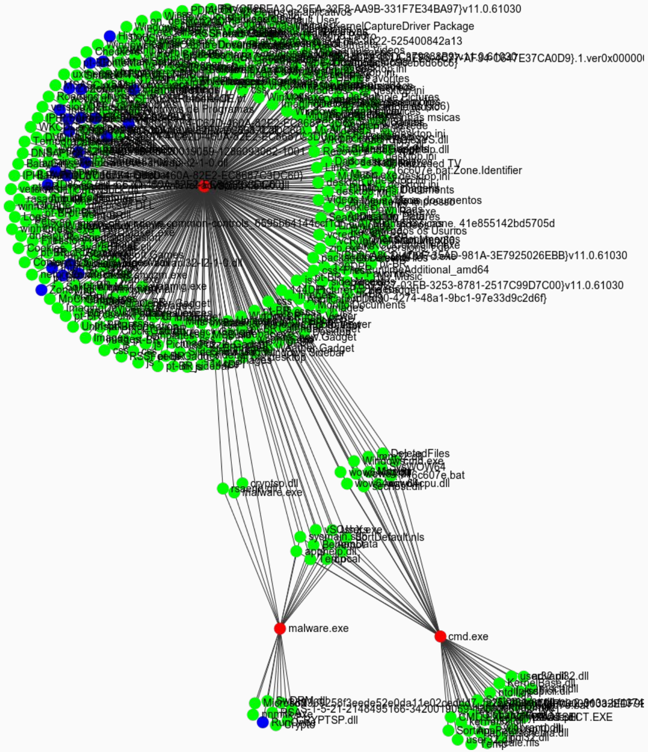
mw	f(mw)	data(gw_i)	gw_i	mw+
Malware	**Embedding Function**	**Goodware Data**	**Goodware**	**Goodware**
91.69% avg confidence			80.95% avg confidence	93.28% avg confidence

Adversarial Malware Generation: Results

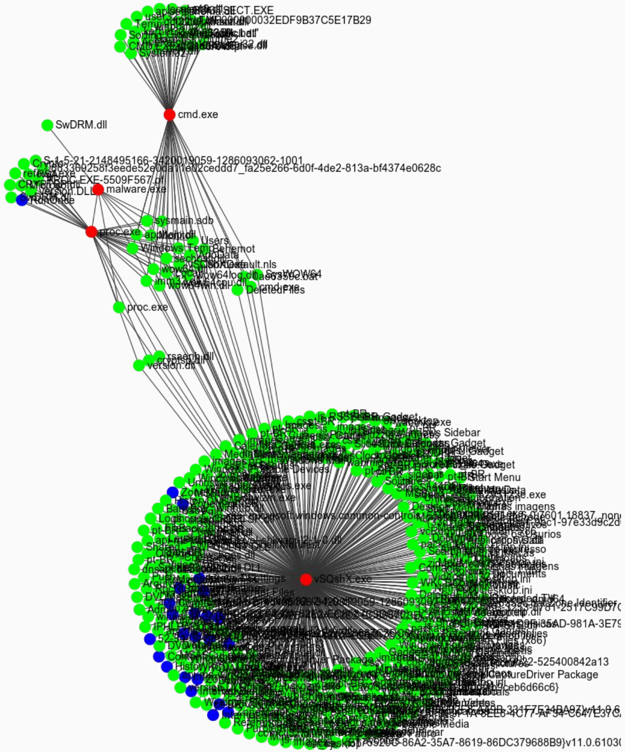
	Malware (mw)		Goodware (gw_i)		Adversarial Malware ($mw+$)	
Model	**Class**	**Confidence**	**Class**	**Confidence**	**Class**	**Confidence**
MalConv	Malware	99.99%	Goodware	69.54%	Goodware	81.22%
Non-Neg. MalConv	Malware	75.09%	Goodware	73.32%	Goodware	98.65%
LightGBM	Malware	100.00%	Goodware	99.99%	Goodware	99.97%
Average	Malware	91.69%	Goodware	80.95%	Goodware	93.28%

Corvus: Malware Execution Graph (Using Execution Trace)

Original Malware



Adversarial Malware



Corvus: Original Samples Collection with ssdeep Similarity

Corvus beta

Search for file name, MD5 or SHA1

B

Collection: Evade Malware ML (Original Samples)

Creation Date: Oct. 14, 2019, 2:30 p.m.
Update Date: Oct. 14, 2019, 8:38 p.m.
Created by: anonymous
Similarity Status: Finished
Reports:

REPORTS | SIMILARITY | GRAPHS

File 1	File 2	Similarity
c444e7e503cd7fd0631c96228eef365a (037)	d3a3e3de34adaa20a27588201e7a4f16 (039)	85%
d3a3e3de34adaa20a27588201e7a4f16 (039)	c444e7e503cd7fd0631c96228eef365a (037)	85%
9428cbf71d4f06c66882773acd55dda1 (038)	f6619f932b36a940f8f6e89988434e3d (035)	65%
f6619f932b36a940f8f6e89988434e3d (035)	9428cbf71d4f06c66882773acd55dda1 (038)	65%
12ee889f3a4da0ad4431f67b30b8279e (013)	534d13022fb65a1c0243d01324b62749 (049)	50%
534d13022fb65a1c0243d01324b62749 (049)	12ee889f3a4da0ad4431f67b30b8279e (013)	50%
0b72e57d9011d2b15e6abc1e8ef0451a (027)	fcee486c8496785dc3e467d53b14f8fa (017)	44%
fcee486c8496785dc3e467d53b14f8fa (017)	0b72e57d9011d2b15e6abc1e8ef0451a (027)	44%

Corvus: Adversarial Samples Collection with ssdeep Similarity

Corvus beta

Search for file name, MD5 or SHA1

B

Collection: Evade Malware ML (Adversarial Samples)

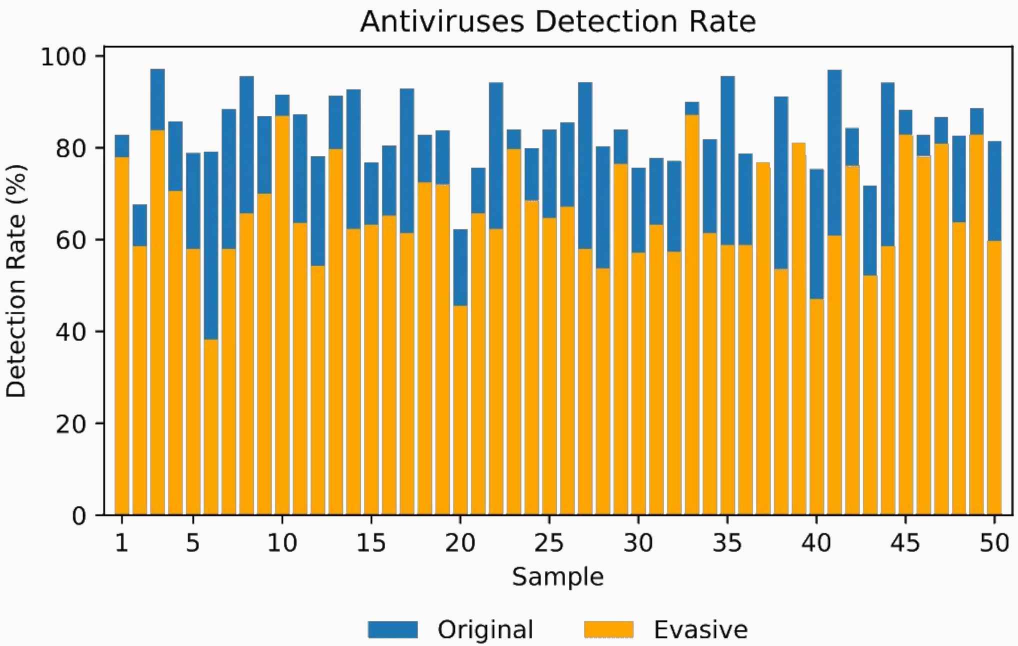
Creation Date: Oct. 11, 2019, 4:56 p.m.
Update Date: Oct. 13, 2019, 5:43 p.m.
Created by: anonymous
Similarity Status: Finished
Reports:

REPORTS | SIMILARITY | GRAPHS

File 1	File 2	Similarity
30f318aafc7cd69e767e9a48c659ecad (007)	e5e21320ed5fb45cbb0a2cbd662e28e8 (035)	97%
11da9f941fed91a25c054045ad4604f1 (003)	44718c856b17db945423ffacebcab12b (016)	97%
11da9f941fed91a25c054045ad4604f1 (003)	90e028745ad56da05614731b799d6a95 (032)	97%
11da9f941fed91a25c054045ad4604f1 (003)	1b8f9247397c5c26037971d00e6dc468 (034)	97%
11da9f941fed91a25c054045ad4604f1 (003)	32aefc46387678069306760f66991779 (047)	97%
44718c856b17db945423ffacebcab12b (016)	11da9f941fed91a25c054045ad4604f1 (003)	97%
44718c856b17db945423ffacebcab12b (016)	90e028745ad56da05614731b799d6a95 (032)	97%
44718c856b17db945423ffacebcab12b (016)	1b8f9247397c5c26037971d00e6dc468 (034)	97%

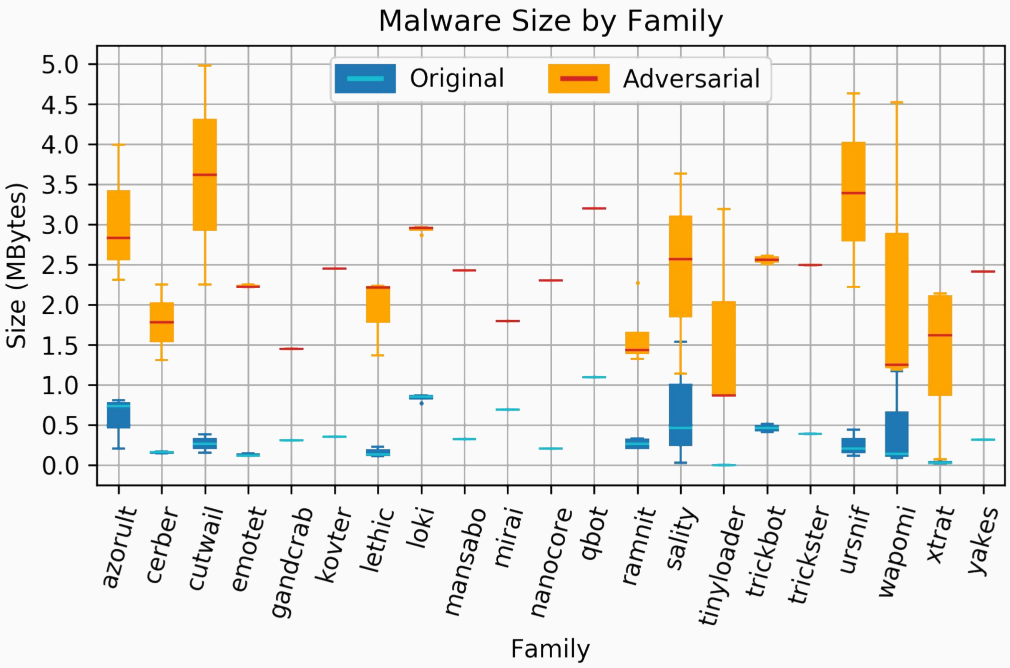
Adversarial Malware in Real World

- Could our strategy be leveraged in real world by actual attackers?
- **VirusTotal service:** detection rates for adversarial samples.
- **Results:** our approach also affected real AV engines.
  - Sample 6 dropping almost in half.
- **Explanation:** AV engines also powered by ML models.
  - Subject to same weaknesses and biases.



Adversarial Malware in Real World

- **Drawback:** binaries become larger than the original ones.
  - Additional data appended.
- Appended data is not even used by the malware.
  - Must be there to create a bias towards goodware class.
- Adversarial malware are, in general, at least around twice the size of original ones.
  - **Original:** around 1.5MB;
  - **Adversarial:** around 5MB.



Discussion

Weaknesses identified and pinpoint possible mitigation.

Susceptibility to Appended Data

- Major weakness of raw models.
- This simple strategy was enough to defeat the two raw data-based models.
- Concept learned by these models is not robust enough against adversarial attacks.

Appending Data Affects Detection but not PE Loading

- Windows loader ignores some PE fields and resolve them in runtime.
- Allows attackers to append content to the binaries without affecting their functionalities.
- More strict loading policies so as to mitigate the impact of this type of bypass technique.
- Loader should check if a binary has more sections than declared and/or if the section content exceeds the boundaries defined in its header.

Adversarial Malware are Much Bigger than Original Ones

- Additional data are needed to bypass classifiers, such as strings and bytes.
- Bias towards goodware class but also make their size greater.
- Can make it difficult for attackers to distribute them for new victims.
- **Challenge to be considered by any attacker:** sample with the minimum size as possible.

Develop Models Based on the Presence of Features Instead on Frequency

- Mitigate the impact of appended data on classification models.
- Classifiers changed decision from malware to goodware when goodware strings were added to the binary, masking the impact of malware strings.
- Malicious strings need to be still present in the binaries to keep its functional.

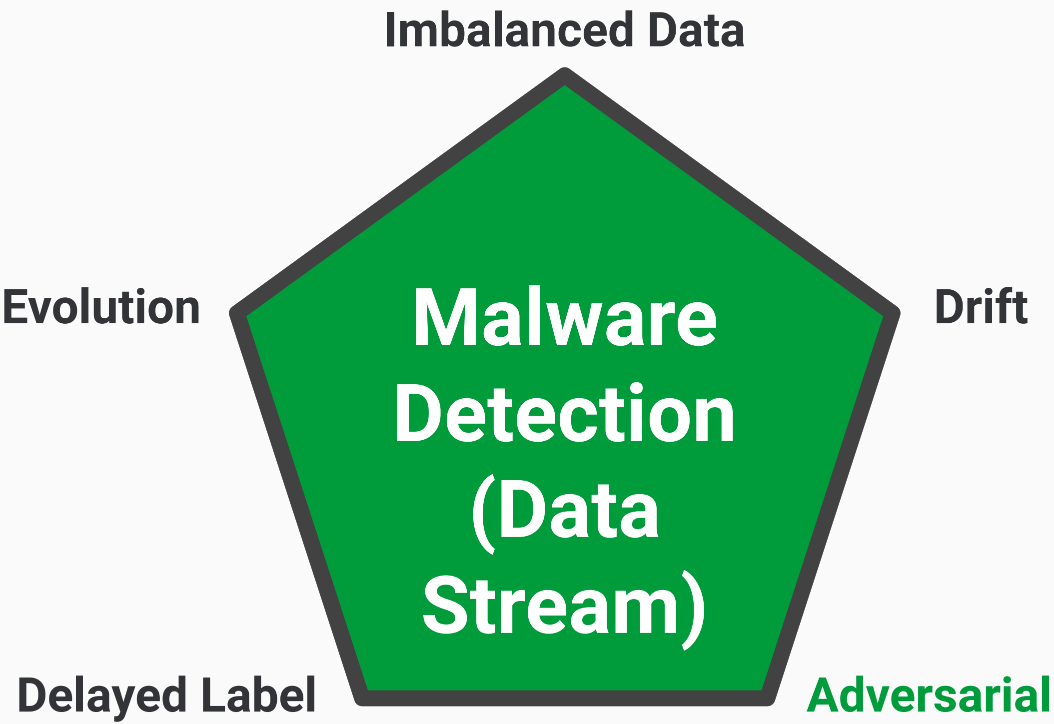
Domain-specific Models Might Present Biases and not Learn a Concept

- Model based on PE binaries features presented a bias against UPX packer.
- Packing benign software with UPX revealed that the detector learned to mistakenly always flag UPX binaries as malicious.

Adopting Malware Variants Robustness as a Criteria to Machine Learning Detectors

- Accuracy, F1 Score and Precision for what??
- Essential step to moving forward the malware detection field.
- Even deep learning models might be easily bypassed: less effective.
- Adoption of variants robustness testing as a criteria for future malware detectors.
- Process of correct evaluating a malware detector, which already includes handling concept drift and evolution, class imbalance, degradation, etc.

Malware Detection & Data Stream Challenges: How to Correctly Evaluate them?



Creating a Robust Representation

- Essential step for malware detection.
- Attackers might include goodware characteristics into their malware to evade any model.
- Representation that is invariant to these characteristics is fundamental to avoid adversarial malware.

Checking File Resources and Embedded PE Files

- It should be part of ML feature extraction procedures.
- Allow classifiers to detect embedded malicious payload instead of being easily deceived by malware droppers.
- Example: https://corvus.inf.ufpr.br/reports/5378/#Static
    - Foremost & PEDetector

Converting Samples into Downloaders

- It might be a successful strategy.
- Malicious payload is retrieved from the Internet, undetected loader is submitted to ML.
- Reason about the whole threat model to cover all attack possibilities.
- **Downloader versions:** implemented but not submitted due to network-isolated sandboxes.

Adversarial Malware is a Particular Case of Adversarial Attacks

- Can be performed against multiple domains.
- **Same goal:** bypassing a classification.
- **Different techniques:** domain-specific.
- Adversarial images: look similar to the original ones (indistinguishable to human eye).
- Adversarial malware: same action as the original, even if they are different.
- Simply adding a noise to a malware might generate an invalid malware that does not work.

Conclusion

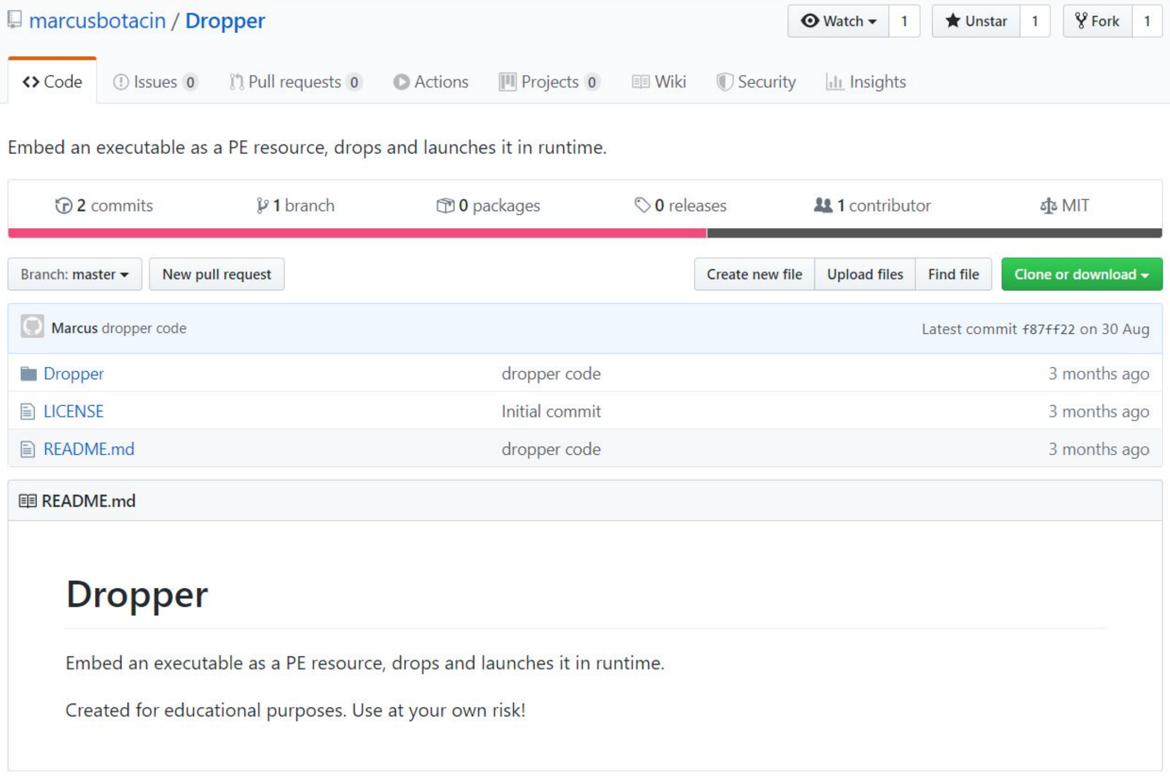
Final remarks, reproducibility and our online platform.

Conclusion

- Models leveraging raw binary data are easily evaded by appending additional data to the original binary files.
- Models based on the Windows PE file structure learn malicious section names as suspicious.
  - These detectors can be bypassed by replacing them.
- **Suggestion:** Adoption of malware variant-resilience testing as an additional criteria for the evaluation and assessment of future developments of ML-based malware detectors.
  - Applied to actual scenarios without the risk of being easily bypassed by attackers.

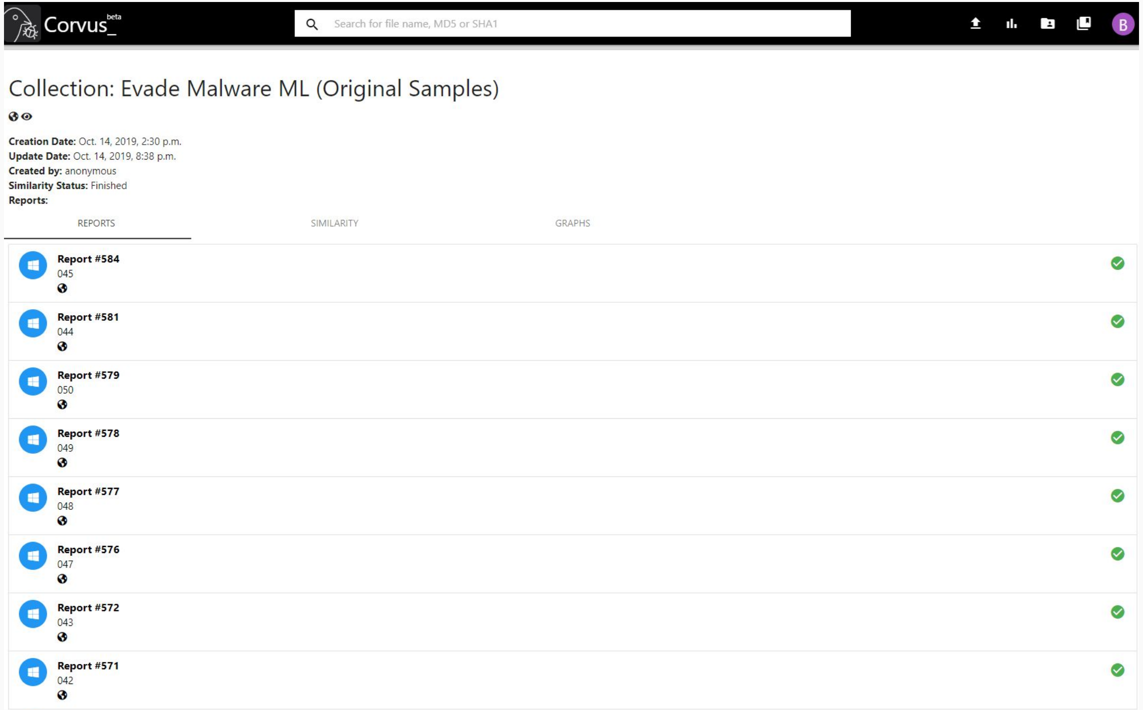
Reproducibility

- **Dropper:** prototype to embed malware samples into unsuspicious binaries released as open source on github.
    - https://github.com/marcusbotacin/Dropper



Reproducibility

- All analysis reports of evasive and non-evasive samples execution and their similarities are available on the **Corvus_** platform, developed by our research team.
    - https://corvus.inf.ufpr.br



SECRET | UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

REVERSING AND OFFENSIVE-ORIENTED TRENDS SYMPOSIUM 2019 (ROOTS)
28TH TO 29TH NOVEMBER 2019, VIENNA, AUSTRIA

Shallow Security: on the Creation of Adversarial Variants to Evade Machine Learning-Based Malware Detectors

Contact: fjoceschin@inf.ufpr.br or @fabriciojoc
Website: secret.inf.ufpr.br
Our Project: corvus.inf.ufpr.br

Fabrício Ceschin
Federal University of Paraná, BR
@fabriciojoc

Marcus Botacin
Federal University of Paraná, BR
@MarcusBotacin

Heitor Murilo Gomes
University of Waikato, NZ
www.heitorgomes.com

Luiz S. Oliveira
Federal University of Paraná, BR
www.inf.ufpr.br/lesoliveira

André Grégio
Federal University of Paraná, BR
@abedgregio