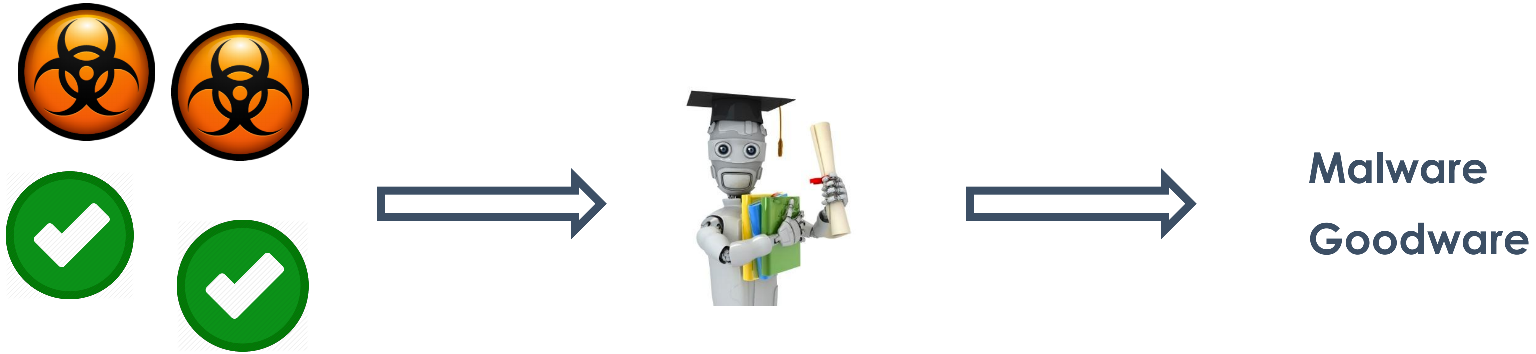
FireEye

Exploring Adversarial Examples in Malware Detection

Octavian Suciu*, Scott E. Coull and Jeffrey Johns

Machine Learning for Malware Classification



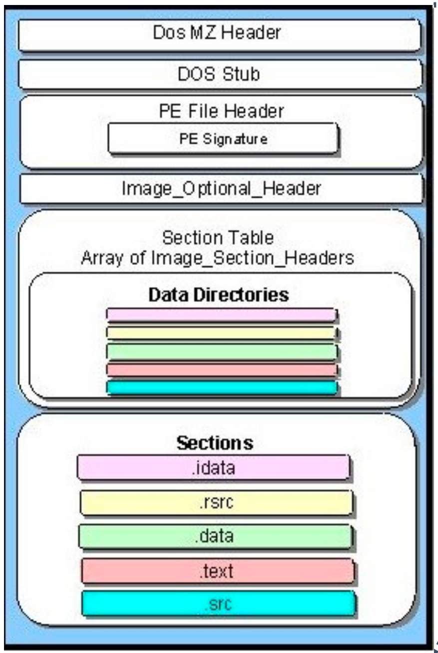
- Evasion attacks against malware detectors contributed to an arms race spanning decades
- Extensive work on understanding evasion attempts affecting traditional ML-based detectors
- Defenders are increasingly employing new approaches such as end-to-end learning

We study the robustness of deep learning-based malware detectors against evasion attempts

Outline

- **Malware detectors based on deep learning**
- Domain challenges for evasion
- Append Attack
- Slack Attacks

Feature Extraction in Static Malware Classification



\x90\x00\x03\x00\x00\x04\x1C

Binary Program

Code length = 1141 bytes
Touched file = “%WINDIR%\System32\en-US\wscript.exe”
String = “http://bad.site”

Features

Feature Engineering

String = “http://bad.site”

Malware

String = “http://lessbad.site”

Goodware

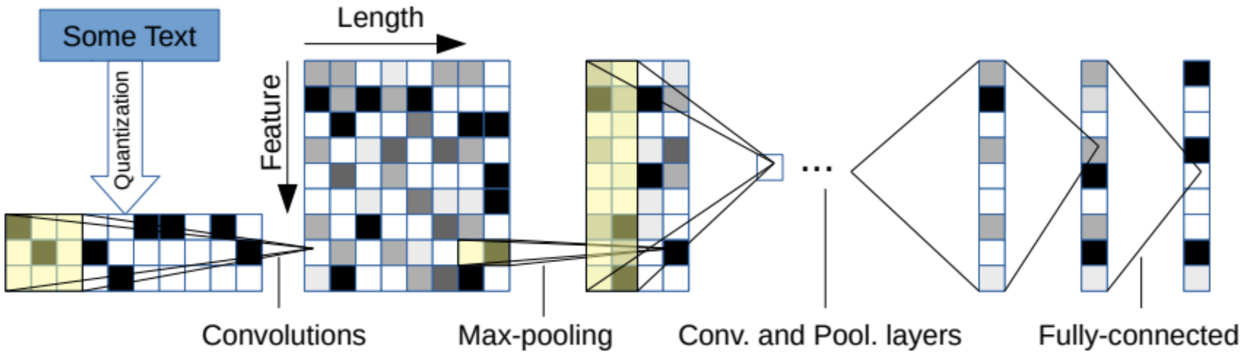
Feature Engineering is challenging and time consuming

Automatically Learning Feature Representations

- ML-based solutions require extensive feature engineering
  – List of features must constantly evolve to capture adaptive adversaries
- One solution: end-to-end learning
  – Automatically learn important features from raw data

Learning from Raw Data

Character-level Convolutional Neural Networks for text classification [Zhang+, 2015]



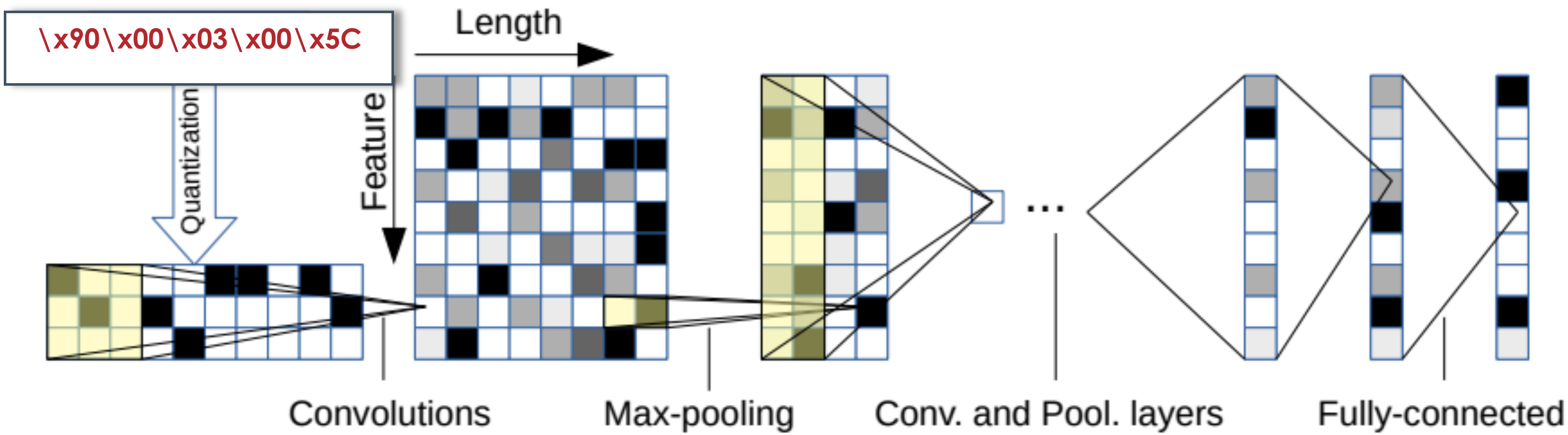
- Embeddings: characters mapped to fixed-size vectors
- Convolutions: receptors for character compositions (e.g. words)
- Max-pooling: filters for non-informative features (e.g. common words)
- Fully connected: non-linear classifier

Analogy between Text and Programs

Natural Language:	Executable programs:
the quick brown fox	\x90\x00\x03\x00\x00\x04\x1C
text characters	bytes
words	instructions
sentences	functions

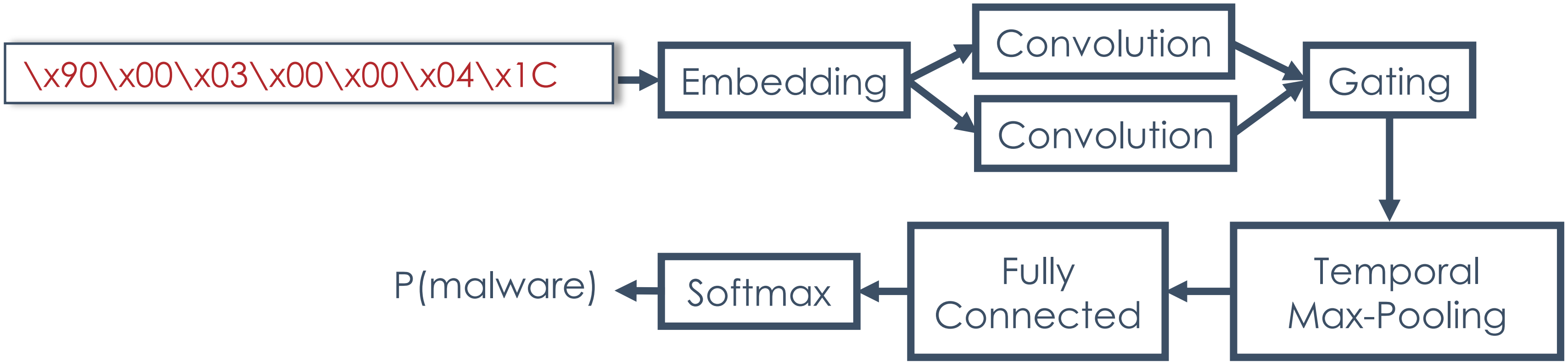
Byte-level Neural Networks for Malware Classification

Program Executable (PE) can be viewed as a sequence of bytes



MalConv: Malware Detector based on Raw Bytes

MalConv: Malware Detection by Eating a Whole EXE **[Raff+, 2017]**



- 2MB input padding, CNN 128 kernels with size=500 and stride=500
- Balanced Accuracy: **0.91** AUC = **0.98**

Is MalConv vulnerable to AML-based evasion attacks?

Training a Robust Classifier

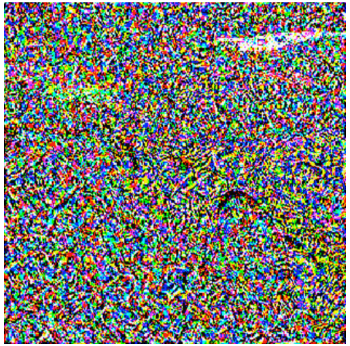
- Trainin MalConv on a production-scale dataset **(FULL)**
  - 12.5 M training samples with 2.2M malware
  - Training & testing sets have strict temporal separation
  - Frequent malware families are down-sampled to reduce bias
- Use published dataset [Anderson+, 2018] **(EMBER)**
  - 900 K training samples
  - Used pre-trained MalConv model shared with dataset
- Sample dataset comparable to prior work **(MINI)**
  - 4,000 goodware and 4,598 malware
  - Sampled from FULL

Outline

- Malware detectors based on deep learning
- **Domain challenges for evasion**
- Append Attack
- Slack Attacks

Evasion Attacks in Image Classification

+



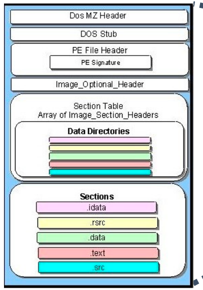
=

Toaster

$\boldsymbol{x}$ $\quad$ $\mathbf{Sign}(\nabla_x J(\theta, x, y))$ $\quad$ $x + \epsilon\mathbf{Sign}(\nabla_x J(\theta, x, y))$

- Gradient directs instance across decision boundary
  - **[Szegedy+, 2014]**, **[Papernot+, 2015]**, **[Carlini and Wagner, 2017]**
- **[Goodfellow+, 2015]**: Fast Gradient Sign Method
- Can we apply these attacks directly to the malware detection domain?

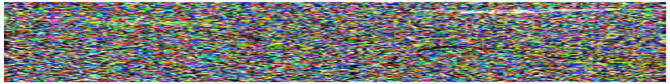
Applying AML Attacks to Binaries



\x90\x00\x03\x00\x00\x04\x1C — Original PE Sample

+

$\mathbf{Sign}(\nabla_x J(\theta, x, y))$



Adversarial Noise

=

??? ⟸ \xA3\x45\x03\xB3\x05\x04\x1C — Evasive PE Sample

Existing evasion attacks break the functionality of the executable

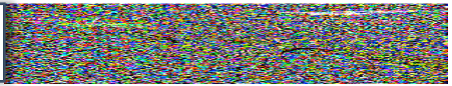
Outline

- Malware detectors based on deep learning
- Domain challenges for evasion
- **Append Attack**
- Slack Attacks

Append-based Attacks

\x90\x00\x03\x00\x00\x04\x1C

+



Adversarial Noise

=

\x90\x00\x03\x00\x00\x04\x1C\xA3\x21\x45\xB3\x05

Evasive PE Sample

- Appended noise preserves functionality by not modifying content of original bytes **[Kolosnjaji+, 2018]**

Naive Benign Append Attack

\x90\x00\x03\x00\x00\x04\x1C

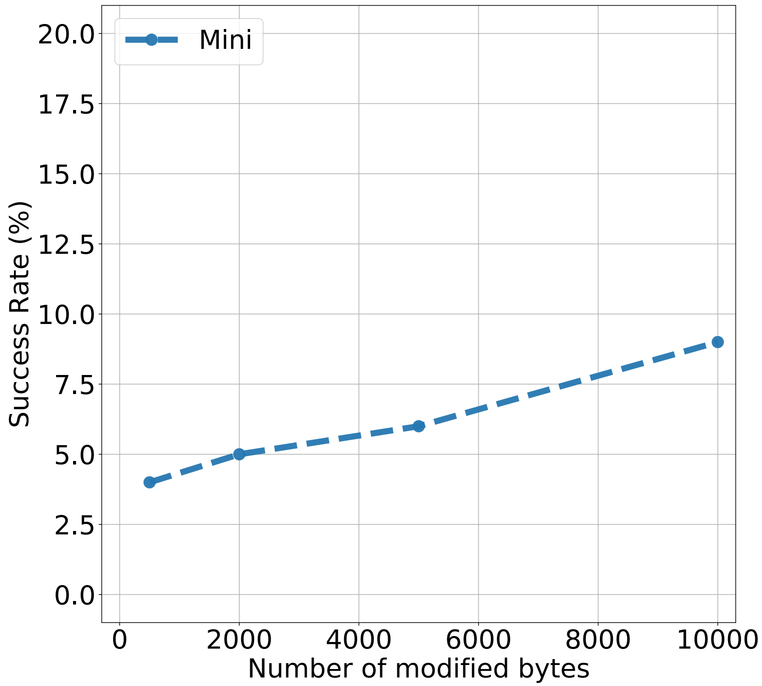
Original Sample

\x90\x00\x03\x00\x00\x04\x1C \x60\xFA\x3B\xC1\x00

Adversarial Noise

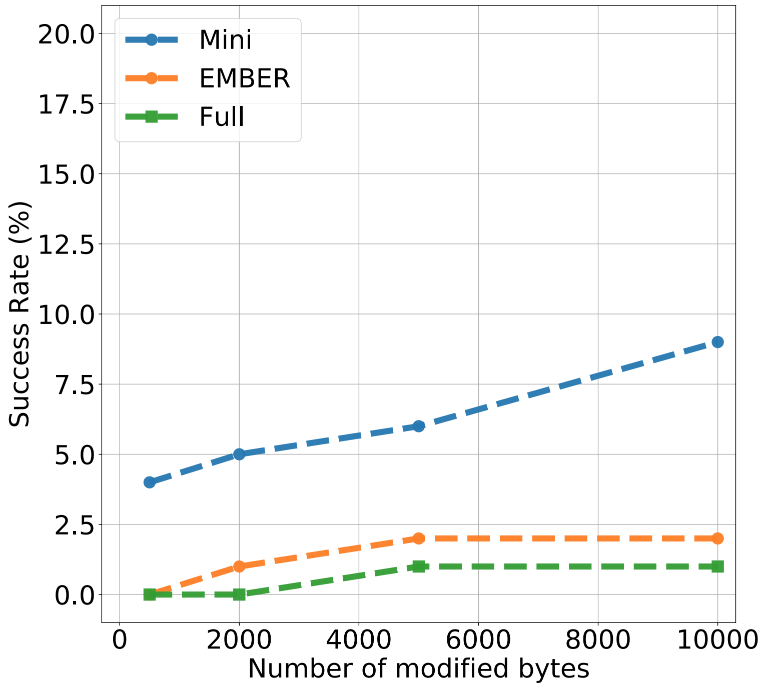
- Adversarial bytes are copied from benign samples correctly classified with high confidence

Benign Append Results



- SR on MINI increases linearly with number of bytes
  - Model overfits benign features due to a small dataset used for training a large capacity network

Benign Append Results



- SR on MINI increases linearly with number of bytes
  - Model overfits benign features due to a small dataset used for training a large capacity network
- EMBER & Full models are robust to the attack
  - Harder to overcome dataset features by appending benign bytes at the end of file

Take-away

Consider dataset biases when drawing conclusions about adversarial attack effectiveness

FGSM Append Attack

\x90\x00\x03\x00\x00\x04\x1C

Original Sample

↓

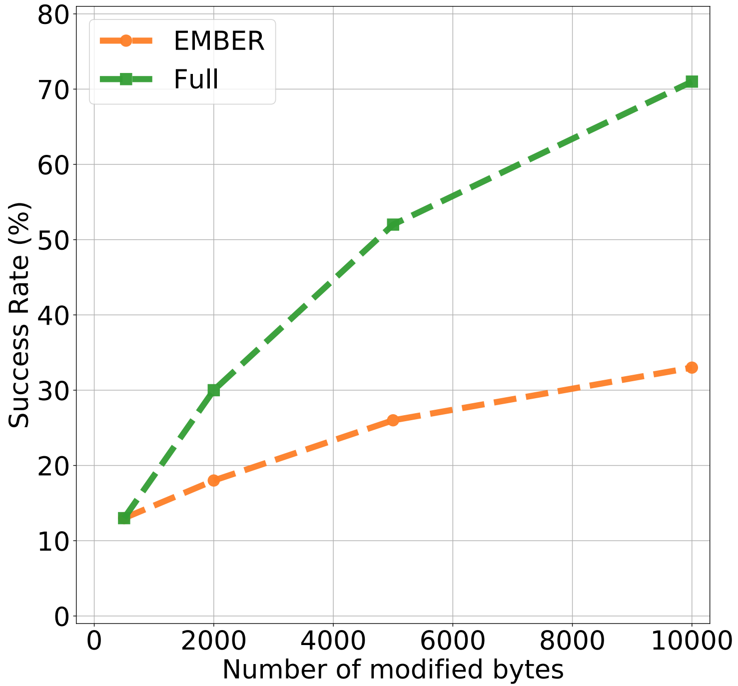
\x90\x00\x03\x00\x00\x04\x1C | $\mathbf{Sign}(\nabla_x J(\theta, x, y))$

Adversarial Noise

- Adversarial embeddings are generated using the **single-step** Fast Gradient Sign Method **[Goodfellow+, 2015]**
- Adversarial bytes are chosen as the L2 closest values in the embedding space

FGSM Append Results

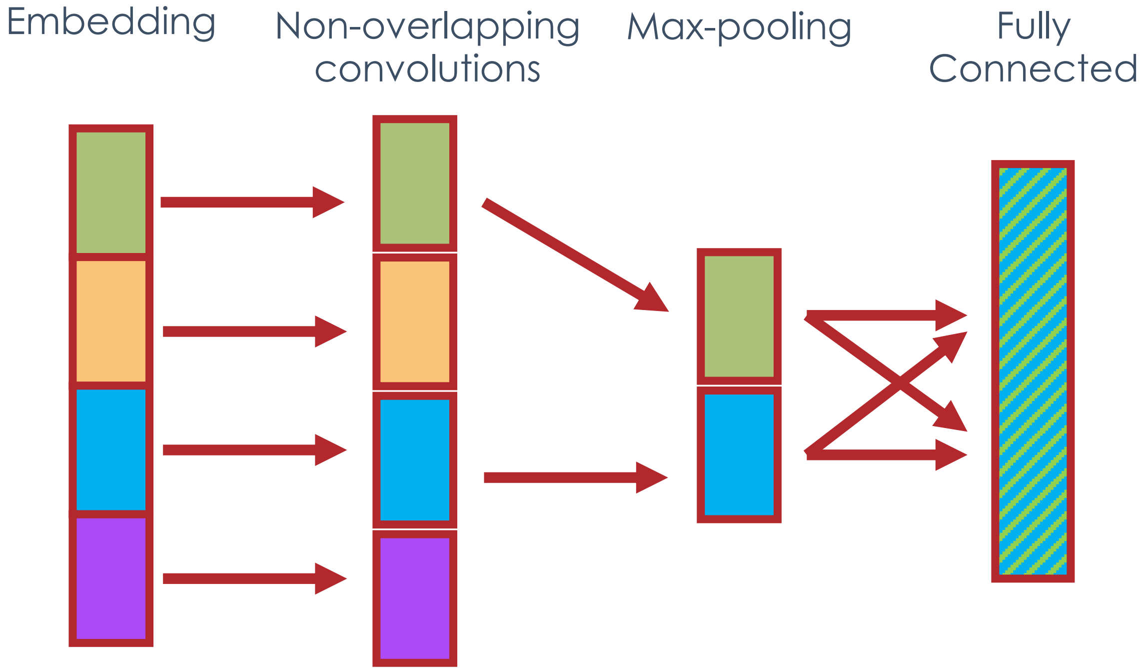
Upper bound attack performance



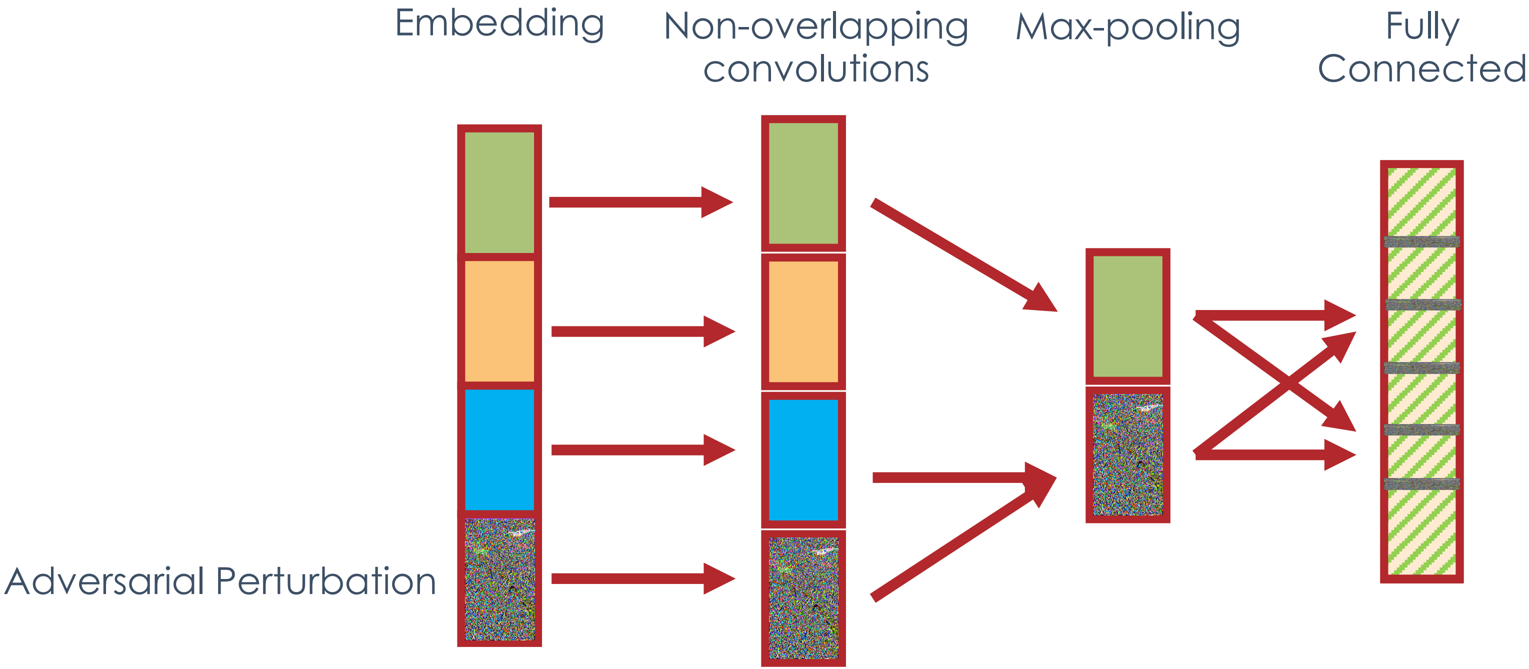
- Larger training set leads to more vulnerable model
  - Full model encodes more sequential features
- High Success Rate highlights model vulnerability
  - Ample opportunity to evade MalConv

Why is attack so effective?

Architectural Weakness in MalConv



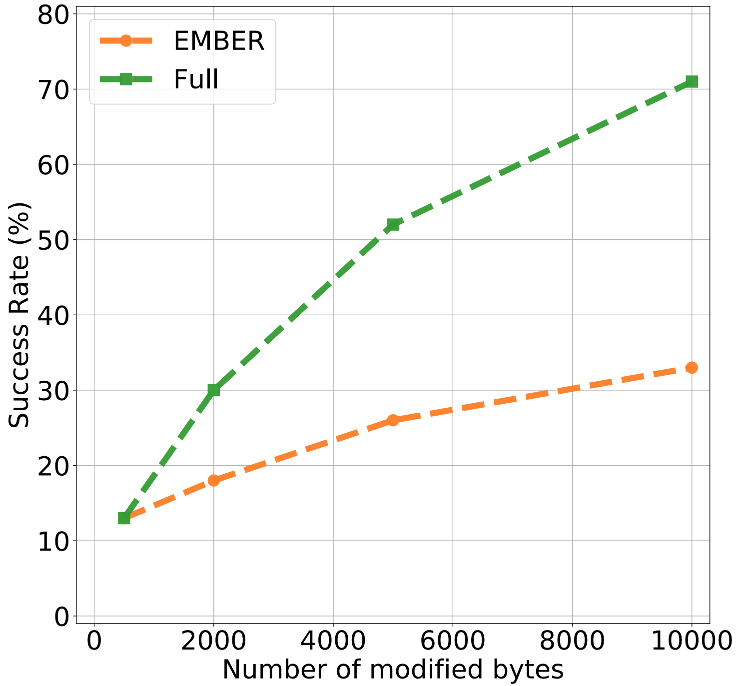
Architectural Weakness in MalConv



MalConv does not encode positional features

Architectural Weakness in MalConv

Upper bound attack performance



- Larger training set leads to more vulnerable model
  - Full model encodes more sequential features
- High Success Rate highlights model vulnerability
  - Ample opportunity to evade MalConv

Take-away

Architectural choices may introduce vulnerabilities against adversarial attacks

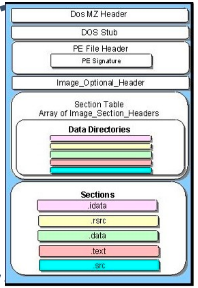
Can we leverage program semantics in attacks?

Outline

- Malware detectors based on deep learning
- Domain challenges for evasion
- Append Attack
- **Slack Attacks**

Finding Slack Regions

\x90\x00\x03\x00\x00\x04\x1C



Section Header

Section 1

Section 2

...

Header contains pointers to sections of executable

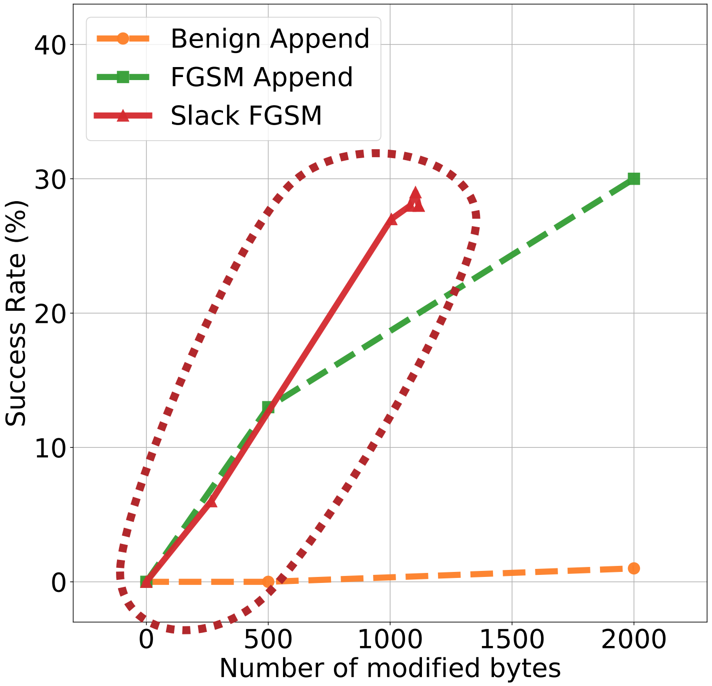
Each Section has **RawSize** (size in PE file) and **VirtualSize** (size when loaded into memory)

The compiler may set **VirtualSize** smaller than **RawSize**

We could use slack regions to inject adversarial noise since they are not mapped to memory

Slack Attack Results

Effectiveness of Slack FGSM on FULL



- Slack FGSM outperforms append strategies at smaller number of modified bytes
  - Attack uses contextual byte information about feature importance
  - But there is a limited number of slack bytes available

Take-away

Reasoning about program semantics helps improve attack effectiveness

Lessons Learned

- Training set matters when testing robustness against adversarial examples
  – Small dataset gives skewed estimates about attack success rates
- Architectural decisions should consider potential effect of adversarial examples
  – Models that do not encode positional information can be easily bypassed
- Semantics is important for improving attack effectiveness
  – Reasoning about feature importance helps exploit higher-level learned ones

Thank you!

Octavian Suciu
osuciu.com
osuciu@umiacs.umd.edu

References

- **[Raff+, 2017]** E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, "Malware detection by eating a whole exe,"
- **[Anderson+, 2018]** H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models,"
- **[Szegedy+, 2014]** C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks,"
- **[Papernot+, 2015]** Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A., "The limitations of deep learning in adversarial settings"
- **[Carlini and Wagner, 2017]** N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks,"
- **[Goodfellow+, 2015]** I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples,"
- **[Kolosnjaji+, 2018]** B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables,"