MALWARE DETECTION
BY EATING A WHOLE
EXE

Presented by:
Edward Raff
Jared Sylvester
Robert Brandon
1 November 2017
Malware Detection? Don’t AVs do that?

- Single incidents of malware are now causing millions in damages.
  - Potential impact is growing, see: WannaCry, Petya
  - Lives can be on the line, especially when older hospital infrastructures get infected
- AV products are built around a Signature Based approach
  - Essentially extended RegExs for binaries
  - Do some fancy stuff too, but often not as much
  - Makes the approach reactionary
  - Signatures have high specificity, but low generalization
Sounds like a Standard Classification Problem…

• Machine Learning has enjoyed huge success in recent years at predicting things
  • What is in this picture? (Object Detection)
  • What did you say? (Speech-to-text, Alexa, Siri)
  • What did you mean? (Sentiment Analysis)

• But Malware is more challenging for several reasons

I found this to be a charming adaptation, very lively and full of fun. With the exception of a couple of major errors, the cast is wonderful. I have to echo some of the earlier comments -- Chynna Phillips is horribly miscast as a teenager. At 27, she's just too old (and, yes, it DOES show), and lacks the singing "chops" for Broadway-style music. Vanessa Williams is a decent-enough singer and, for a non-dancer, she's adequate. However, she is NOT Latina, and her character definitely is. She's also very STRIDENT throughout, which gets tiresome. The girls of Sweet Apple's Conrad Birdie fan club really sparkle -- with special kudos to Brigitta Dau and Chiara Zanni. I also enjoyed Tyne Daly's performance, though I'm not generally a fan of her work. Finally, the dancing Shriners are a riot, especially the dorky three in the bar. The movie is suitable for the whole family, and I highly recommend it.
Binaries Lack Spatial Consistency

- Jumps and Calls add weird locality
- Spatial correlation ends at function boundaries
  - Except for when it doesn't
- Multiple hierarchies of relationships
  - Basic-block level
  - Function level
  - Function composition into classes

```plaintext
jmp 0x4010eb
push 0x10024b78
lea ecx, dword ptr [esp + 4]
call dword ptr [MFC71.DLL:None]
push ebx
push esi
push edi
push 0x10024c05
lea ecx, dword ptr [esp + 0x14]
call dword ptr [MFC71.DLL:None]
lea ecx, dword ptr [esp + 0x24]
mov ebx, 1
push ecx
mov byte ptr [esp + 0x20], bl
call 0x41f8ec
mov edx, dword ptr [eax]
```
Malware Complicates *Everything*

- Malware may intentionally break rules / format specifications
  - Bug that is part of an exploit
  - Intentionally trying to obfuscate itself
    - Attribution, purpose, that it is even malware
- x86 code gives you the freedom to make your programs, gives malware the freedom to be weird
  - Binaries with no “code”
  - Binaries with only code
  - Binaries within binaries
  - Binaries composed of only the x86 `mov` instruction.
  - Binaries that can detect if they are in a VM
Complication Makes Feature Extraction Difficult

• Simple things like getting values from the PE header are non-trivial
  • We’ve tested multiple libraries with disagreements on header content
  • Windows doesn't even follow the PE-spec
• A number of companies have followed through on this domain-knowledge based path
  • Expensive proprietary feature extraction systems
    • Reverse engineering the windows loader
    • Hooking deep into the OS
    • Enhanced emulated execution
  • Huge amount of effort and person-hours just for features
• What if we want to work for any new format?
A Domain Knowledge Free Approach

- DK-free means we don’t encode any knowledge about the file format in the solution: Looking at raw bytes.
  - Means we are going to be doing static analysis.
- DK-free means we can adapt to new file formats (given data).
  - Build new models for PDFs, RTFs, etc., as they become a problem.
  - Ready to work for any new file format as it arises.
  - Save time on feature extraction, time-to-solution reduced.
- DK-free means we get rid of old problems, but also introduce new ones. That’s what we tackle in this work.
- We think a neural-network based solution is most likely to succeed.
How do we Make a Neural Net Process a Whole Binary?

• Problems:
  • Binaries are variable length
  • Binaries are large
  • Binaries can store many things

• We found that many best-practices in the image domain didn’t translate to our space
  • We needed to make our network shallow instead of deep
  • We needed to use large filter sizes instead of small
  • We needed to be very careful in how we handle variable length

• Memory constraints are the primary bottle neck
  • Modern frameworks were never designed for inputs of 2 million time steps!
  • Just the first convolution uses >40GB of RAM for backpropagation
MalConv Architecture, Part 1

Input (1-2M bytes)
Byte string

Tokenization (non-trainable lookup table)
Integers

Zero padding to batch
max length ~2MB

8-dimensional embedding (trainable lookup table)

\[ E = D_w, \]

1D Convolution
kernel size 500,
stride 500,
128 filters

\[ A = E \ast W + b \]

\[ B = E \ast V + c \]

\[ \sigma \]
MalConv Architecture, Part 2

Gating

\[ G_0 = A \otimes \sigma(B) \]

Temporal max pooling

\[ P = \max_{\text{channels}}(G_L) \]

128-dim FC layer

\[ H_0 = WP \]

Softmax

\[ Y = \text{softmax}(WH_K) \]
Data and Evaluation

- Using two test sets, Groups “A” and “B”
  - Allow us to better test generalization
  - The I.I.D. assumption is strongly violated by malware
    - Cross-Validation will over-estimate your accuracy!
  - Group A is public data, benign comes from Microsoft Windows
  - Group B is private AV data, real-world

- Training, we use two private datasets from our AV partner
  - 400k training set, used in prior work.
  - 2 million training set, over 2 TB in size!
Primary Results

• We have a model and we have data. Now for some results!

• 1) How accurate is MalConv?
  • Is it better than what we could do before?

• 2) What does MalConv learn?
  • Does it learn more than what prior results did?

• 3) What have we learned?
  • A lot of ML practice does not easily transfer to this new domain!
MalConv Results

<table>
<thead>
<tr>
<th>Test Set</th>
<th>MalConv Accuracy</th>
<th>MalConv AUC</th>
<th>Byte n-grams Accuracy</th>
<th>Byte n-grams AUC</th>
<th>PE-Header Network Accuracy</th>
<th>PE-Header Network AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A</td>
<td>88.1</td>
<td>98.5</td>
<td>87.0</td>
<td>98.4</td>
<td>90.8</td>
<td>97.7</td>
</tr>
<tr>
<td>Group B</td>
<td>89.6</td>
<td>95.8</td>
<td>92.5</td>
<td>97.9</td>
<td>83.7</td>
<td>91.4</td>
</tr>
</tbody>
</table>

- Trained on 400,000 binaries
- Evaluated on two datasets
- MalConv has best holistic performance
  - Outperformed our prior work looking at just the PE-Header
  - Smallest gap between two test sets, indicates robustness to features
## MalConv Results

<table>
<thead>
<tr>
<th>Test Set</th>
<th>MalConv Accuracy</th>
<th>MalConv AUC</th>
<th>Byte n-grams Accuracy</th>
<th>Byte n-grams AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A</td>
<td>94.0</td>
<td>98.1</td>
<td>82.6</td>
<td>93.4</td>
</tr>
<tr>
<td>Group B</td>
<td>90.9</td>
<td>98.2</td>
<td>91.6</td>
<td>97.0</td>
</tr>
</tbody>
</table>

- Trained on a larger corpus of 2 million binaries
- Took *a month* on a DGX-1
- N-grams took one month to count using 12 servers.
- MalConv performance improved, Byte n-grams *decreased*
  - MalConv still has growth on the learning curve
  - N-grams are overfitting
What is MalConv Learning?

- Our prior work has found that byte n-grams really only learn the PE-Header.
  - We expect PE-Header to make a big portion of any model, because it’s the easiest to learn.
- Because MalConv has temporal max-pooling, we can look back and see which areas of the binary will respond.
  - Produces a sparse set of 128 regions each of 500 bytes per binary.
- Using tools to parse the PE-Header, we can look at what sections the blocks were found in.
  - Gives us an idea about the type of features it is learning.
What is MalConv Learning?

- Blocks can indicate they were used to recognize benign-ness or maliciousness.
  - The PE-Header makes up ~60% of regions used. PE-Header properties are a strong indicator of maliciousness to domain experts.
  - Lots of new regions we weren’t learning from before!
- UPX1 for both benign and malicious is interesting.
  - UPX is a packer, and many models degrade to saying packers are always malicious.
- Significant use of resource and code sections
  - Strong indication that we are learning to extract far more information than previous approaches.

<table>
<thead>
<tr>
<th>Section</th>
<th>Total</th>
<th>PE-Header</th>
<th>.rsrc</th>
<th>.text</th>
<th>UPX1</th>
<th>CODE</th>
<th>.data</th>
<th>.rdata</th>
<th>.reloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malicious</td>
<td>26,232</td>
<td>15,871</td>
<td>3,315</td>
<td>2,878</td>
<td>697</td>
<td>615</td>
<td>669</td>
<td>383</td>
<td>214</td>
</tr>
<tr>
<td>Benign</td>
<td>19,290</td>
<td>11,183</td>
<td>2,653</td>
<td>2,414</td>
<td>596</td>
<td>505</td>
<td>423</td>
<td>243</td>
<td>77</td>
</tr>
</tbody>
</table>
What Didn’t Work: BatchNorm

• Sacrilege warning: BatchNorm doesn’t always work.
• Issue with data modality. Every pixel in an image is a pixel. Meaning doesn’t change.
• Byte meaning is *context sensitive*
• When we trained with BatchNorm, models failed to ever learn.
  • Training accuracy would reach 60% at best.
  • Testing would be 50% random guessing.
  • Happened with *every* architecture we tested.
The Failure of BatchNorm

![Graph showing the distribution of standardized output values for different layers, comparing Res5c, Res3b3, Iv4 Conv1, and MalConv with a normal distribution N(0, 1).]
Questions?

Edward Raff
Raff_Edward@bah.com
@EdwardRaffML

Dr. Jared Sylvester
Sylvester_Jared@bah.com
@jsylvest

Dr. Robert Brandon
Brandon_Robert@bah.com
@Phreaksh0

“Malware Detection by Eating a Whole EXE”
https://arxiv.org/abs/1710.09435