Efficient and Interpretable Real-Time Malware Detection Using Random-Forest NODENS

UWE Bristol
University of the West of England

ALAN MILLS
ALAN2.MILLS@LIVE.UWE.AC.UK

THEODOROS SPYRIDOPOULOS
THEO.SPYRIDOPOULOS@UWE.AC.UK

PHIL LEGG
PHIL.LEGG@UWE.AC.UK
Scope

• Current State of Play
• Project Overview - NODENS
• Proposed Method
• Dataset
• Results
• Interpretability
• Further Work
Current State of Play

- Machine learning used in lots of proof-of-concept models or as augmentation
- Use of Machine-Learning that incorporate existing tools
  - Cuckoo, Sandbox, Anubis, HookMe
- High accuracy, but incur a time penalty
- Computationally expensive
- Little work on the interpretability of decisions
Project Overview – NODENS

- Malware detection system using Machine Learning
- Identify malware using ‘process signatures’
- Lightweight – can be deployed from a Pi (Tested on a Pi 2B)
- Interpretable output – without sacrificing speed or accuracy
- Average detection speed of 3 – 8 seconds
- Use of re-fitting and end user input
Proposed Method

- PowerShell was used to collect process data from the target VM
- Produces 64 features as raw output
- Reduced down to 22 used for classification

1. Handles
2. Path
3. Company
4. Description
5. Product
6. HasExited
7. Handle
8. HandleCount
9. NonpagedSystemMemorySize64
10. PagedMemorySize64
11. PagedSystemMemorySize64
12. PeakPagedSystemMemorySize64
13. PeakWorkingSet64
14. PeakVirtualMemorySize64
15. PrivateMemorySize64
16. PrivilegedProcessorTime
17. ProcessorAffinity
18. Responding
19. TotalProcessorTime
20. UserProcessorTime
21. VirtualMemorySize64
22. WorkingSet64
Proposed Method

- Multiple algorithms were tested against a pool of 55 malware samples
- $n$ samples were randomly selected and run 10 times
- Each time the virtual environment was reset to a clean default state
- Features were captured, manually labelled and tested against:

  - Random-Forest
  - KNearestNeighbour
  - GradientBoosting
  - GNB
  - AdaBoost
  - LogisticRegression
  - DecisionTree
  - SVC
  - OneClassSVM
Proposed Method
Dataset

- A total of 146 malware samples overall (all from OS repositories)
- A total of 1,048,575 processes

<table>
<thead>
<tr>
<th>Process Classification</th>
<th>Number</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malware</td>
<td>95,191</td>
<td>9%</td>
</tr>
<tr>
<td>Benignware</td>
<td>953,384</td>
<td>91%</td>
</tr>
</tbody>
</table>

- Malware processes were all PE32 (.exe)
- Benignware included
  - Background Processes
  - Third party software
  - Portable Apps

- Refitting
  - New process data was appended to the training dataset
  - This included benignware processes from the same period
  - As a result the dataset was continually expanding
Dataset – Ransomware

- Dedicated ransomware test was conducted
- 10 unique samples of ransomware
- On average detection was within 9 seconds
- Two outliers
  1. 96 seconds
  2. 30 seconds
Interpretability
Interpretability

CSV output

DOT output

JSON output
Results

• The use of multiple memory features lends weight to assessments regarding malwares unique memory footprint

• These features are used with low frequency other features are favoured

• This is assessed to be due to some ‘easy win’ metrics
  • Malware which deletes it’s own path
  • Malware which injects itself into another process

<table>
<thead>
<tr>
<th>Root Node Feature</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Affinity</td>
<td>20%</td>
</tr>
<tr>
<td>Total Processor Time</td>
<td>16%</td>
</tr>
<tr>
<td>User Processor Time</td>
<td>16%</td>
</tr>
<tr>
<td>Handle</td>
<td>13%</td>
</tr>
<tr>
<td>Path</td>
<td>12%</td>
</tr>
<tr>
<td>Product</td>
<td>10%</td>
</tr>
<tr>
<td>Privileged Processor Time</td>
<td>3%</td>
</tr>
<tr>
<td>Peak Virtual Memory Size64</td>
<td>2%</td>
</tr>
<tr>
<td>Paged System Memory Size64*</td>
<td>&lt;=2%</td>
</tr>
<tr>
<td>Virtual Memory Size64</td>
<td>1%</td>
</tr>
<tr>
<td>Handle Count*</td>
<td>&lt;=1%</td>
</tr>
<tr>
<td>Handles</td>
<td>&lt; 1%</td>
</tr>
<tr>
<td>Working Set 64</td>
<td>&lt; 1%</td>
</tr>
</tbody>
</table>

* Appeared twice, with different threshold values
Further Work

• Increased sample size
  • Further sample testing
  • Bulk data
• Environmentally Aware malware
  • Virtually hardened system
  • Physical machine testing
• More robust processing system
  • Improve or remove shared folder system
Any Questions?

ALAN MILLS
ALAN2.MILLS@LIVE.UWE.AC.UK
• Features used during training

1. Handles  9. NonpagedSystemMemorySize64  17. ProcessorAffinity
2. Path  10. PagedMemorySize64  18. Responding
4. Description  12. PeakPagedSystemMemorySize64  20. UserProcessorTime
5. Product  13. PeakWorkingSet64  21. VirtualMemorySize64
6. HasExited  14. PeakVirtualMemorySize64  22. WorkingSet64
7. Handle  15. PrivateMemorySize64
8. HandleCount  16. PrivilegedProcessorTime