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

NODENS

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.
    - Chosen as it could be ported between Windows and Linux systems
- Produces 64 features as raw output
- Reduced down to 22 used for classification
- During initial training a Legitimate label was appended to each process, to allow supervised training of the classifier(s)
- For each entry the process Name is used as the index

Proposed Method

- Features used during training

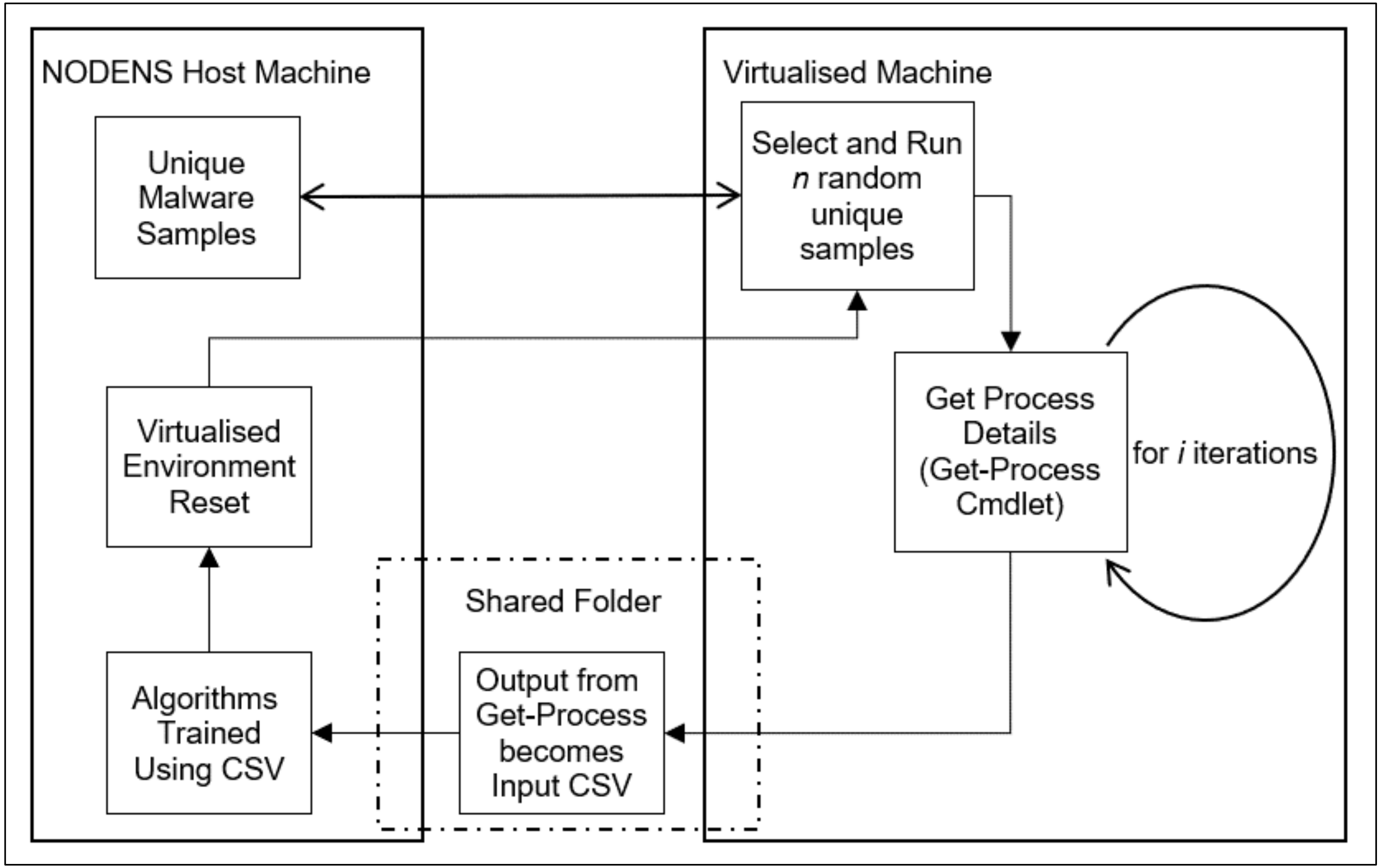
1. Handles
2. Path
3. Company
4. Description
5. Product
6. HasExited
7. Handle
8. HandleCount
9. NonpagedSystemMemorySize64
10. PagedMemorySize64
11. PagedSystemMemorySize64
12. PeakPagedSystemMemorySize64
13. PeakWorkingSet64
14. PeakVirtualMemorySize64
15. PrivateMemorySize64
16. PrivilegedProcessorTime
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
GNB
DecisionTree
KNearestNeighbour
AdaBoost
SVC
GradientBoosting
LogisiticRegression
OneClassSVM

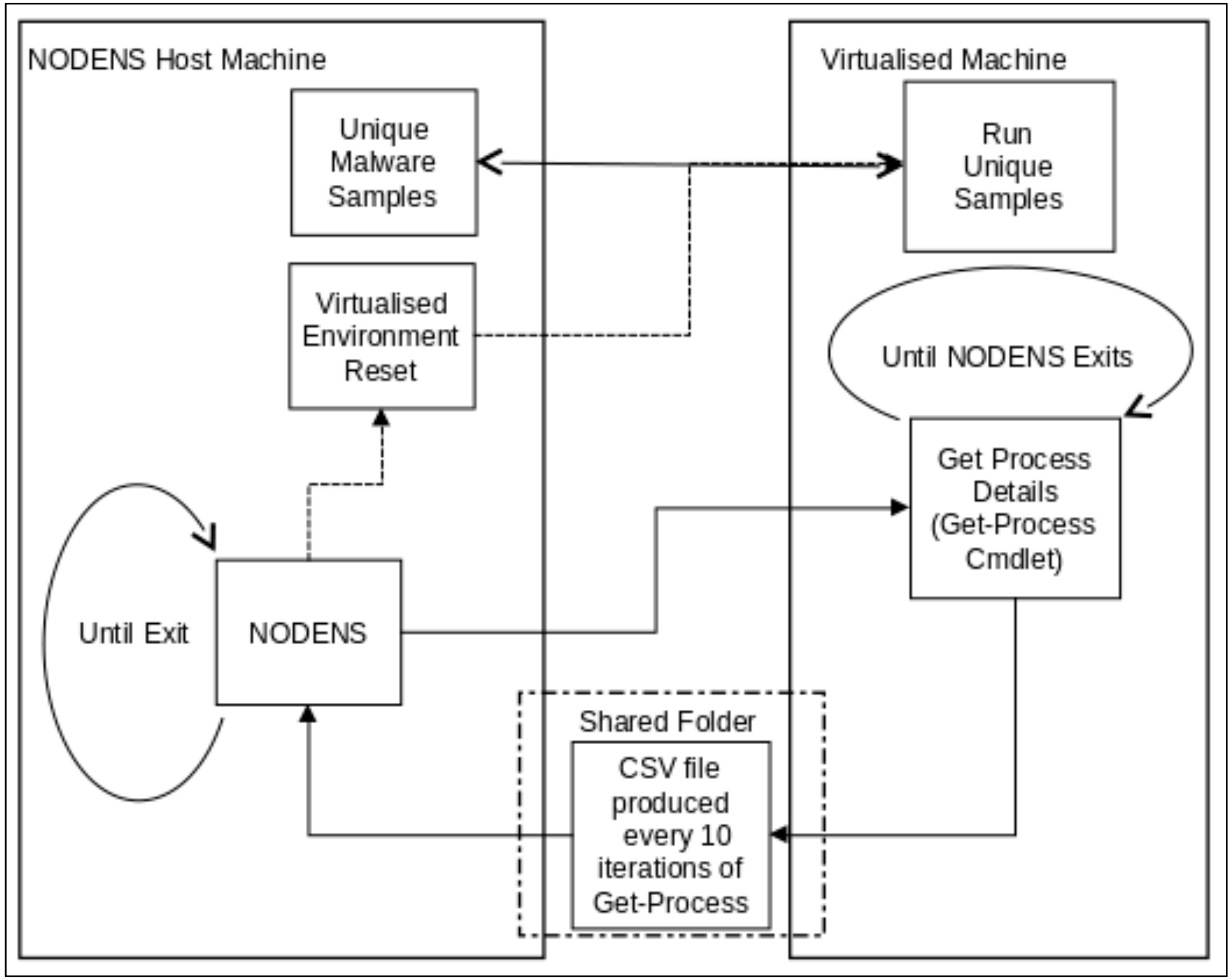
Proposed Method


NODENS Host Machine
Unique Malware Samples
Virtualised Environment Reset
Algorithms Trained Using CSV
Virtualised Machine
Select and Run *n* random unique samples
Get Process Details (Get-Process Cmdlet)
for *i* iterations
Shared Folder
Output from Get-Process becomes Input CSV

Proposed Method

- A Random-Forest classifier was trained on all combined training data
- Live testing started, but with an initial detection delay of 30 seconds
- The classifier and supporting scripts were modified and the delay reduced to 3-8 seconds
- Feature selection was found to negatively effect accuracy, so removed and re-trained

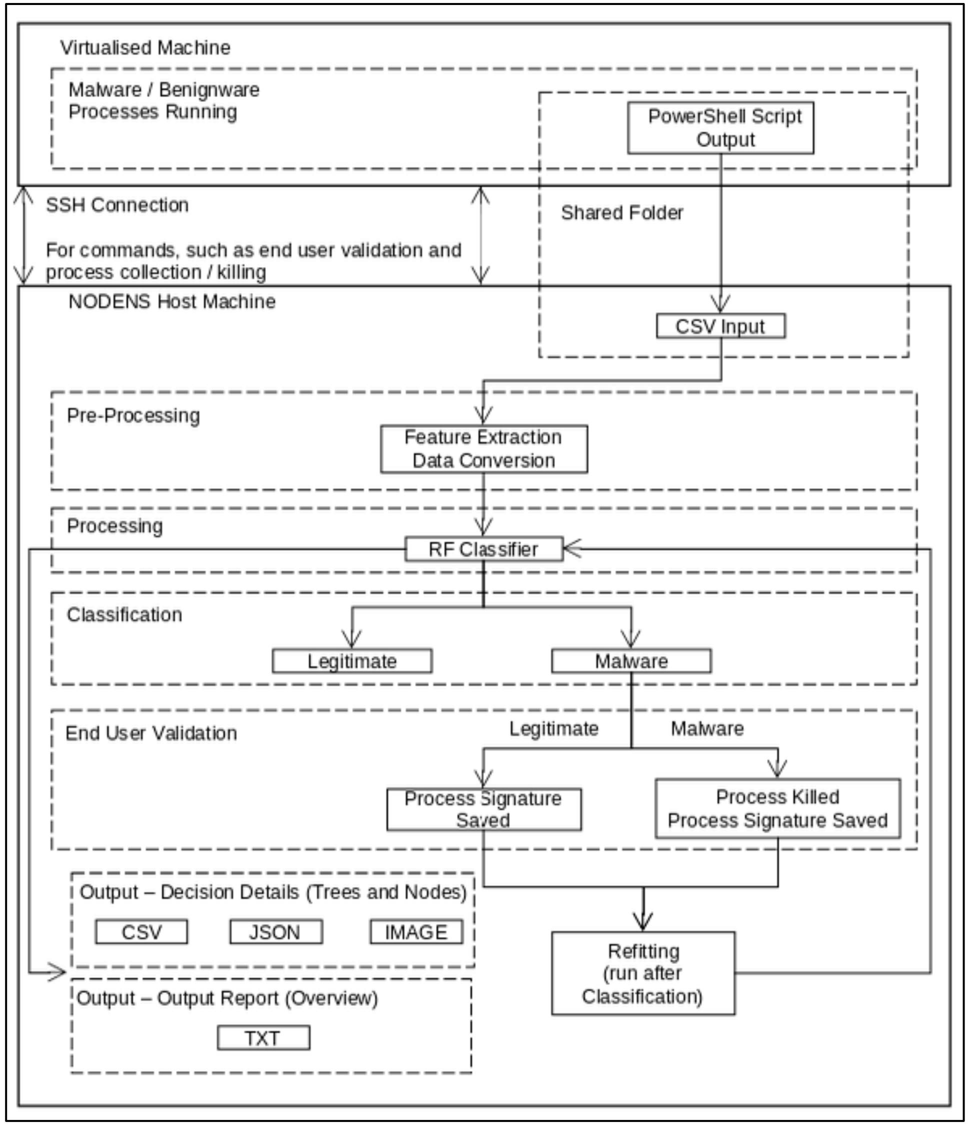
Proposed Method



Proposed Method

- Command line interface to allow for validation or countering of decisions
- Modular ‘plug-in’ scripts
  - Start and Stop data collection and detection
  - Termination of malicious processes
  - Re-fitting of classifier

Proposed Method



Virtualised Machine
Malware / Benignware Processes Running
PowerShell Script Output
SSH Connection
For commands, such as end user validation and process collection / killing
Shared Folder
NODENS Host Machine
CSV Input
Pre-Processing
Feature Extraction Data Conversion
Processing
RF Classifier
Classification
Legitimate
Malware
End User Validation
Legitimate
Malware
Process Signature Saved
Process Killed Process Signature Saved
Output – Decision Details (Trees and Nodes)
CSV
JSON
IMAGE
Output – Output Report (Overview)
TXT
Refitting (run after Classification)

Dataset

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

Process Classification	Number	Percentage
Malware	95,191	9%
Benignware	953,384	91%

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

Dataset

Malware Classification	Number	Percentage
Trojan	47	93%
Ransomware	15	100%
Spyware	15	100%
RAT	7	100%
Bit Coin Miner	3	100%
Process Injector	3	100%
Virus	1	100%

Dataset – Refitting

- Refitting was included to allow NODENS to ‘learn’ from the malware data
- New process data was saved in a .csv and appended to the training dataset
  - This included benignware processes captured within the same time period
- Re-trained using a pickle warm-start
- As a result the training dataset is continually expanding

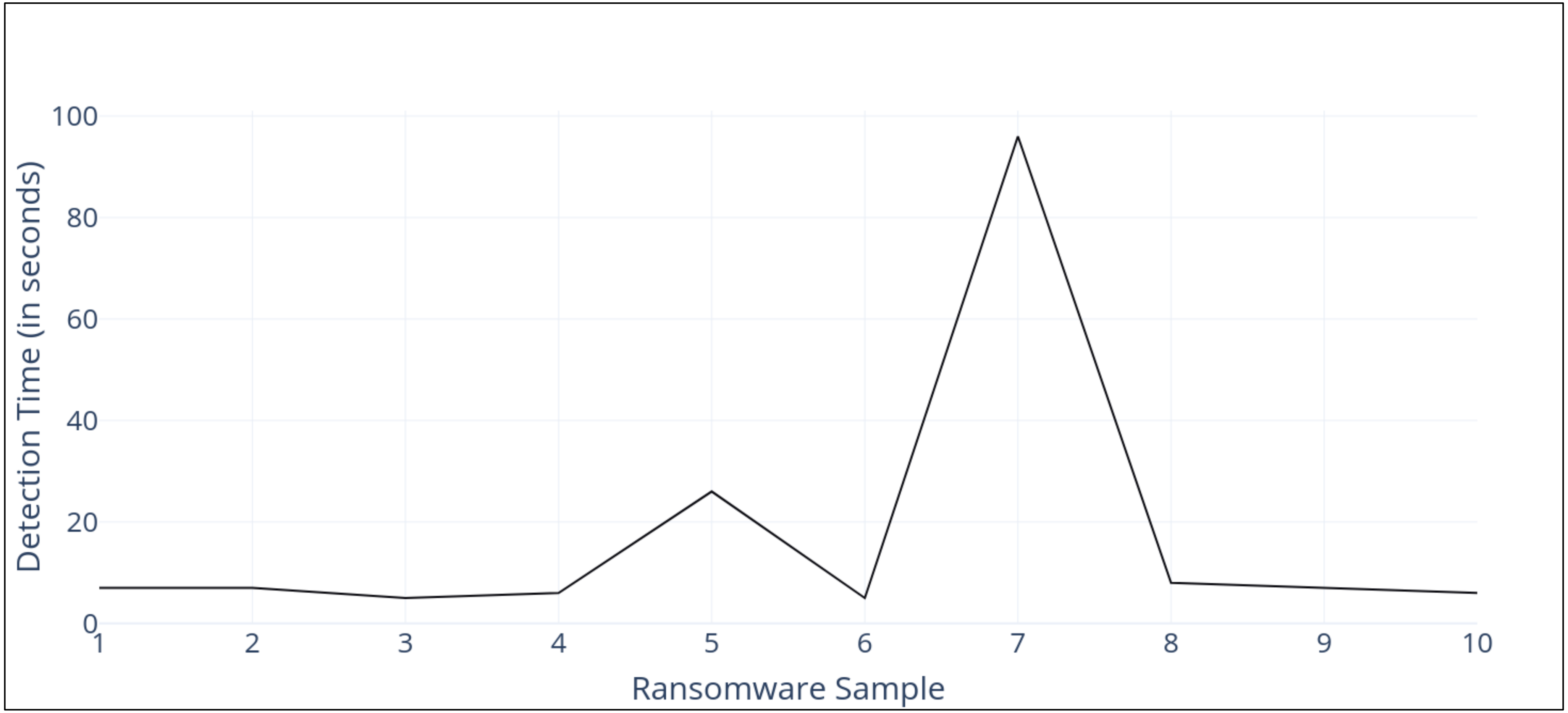
Dataset – Refitting

- Refitting was effective in two ways
    1. It showed that NODENS was able to ‘learn’, having identified 5 samples through refitting
    2. This indicates that (among the samples tested) there is an underlying pattern to behaviour which does indicate a process is malicious

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

Dataset – Ransomware



Dataset – Ransomware

- Ransomware was encrypting the CSV process details
- Each time NODENS was forced to wait for a new file
- More robust design is required

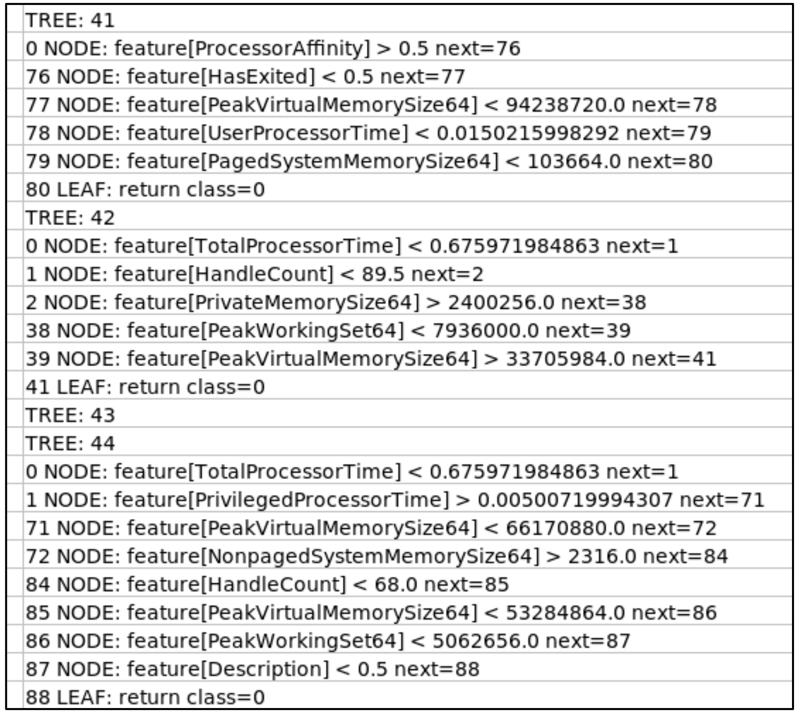
Dataset – Persistence

- In addition to OS malware NODENS was tested against custom malware
- Persistent malware created using msfvenom
- NODENS was able to detect all created malware
- It was unable to defeat persistence
    - Assessed to be linked to a lower memory footprint when re-initialised

Interpretability

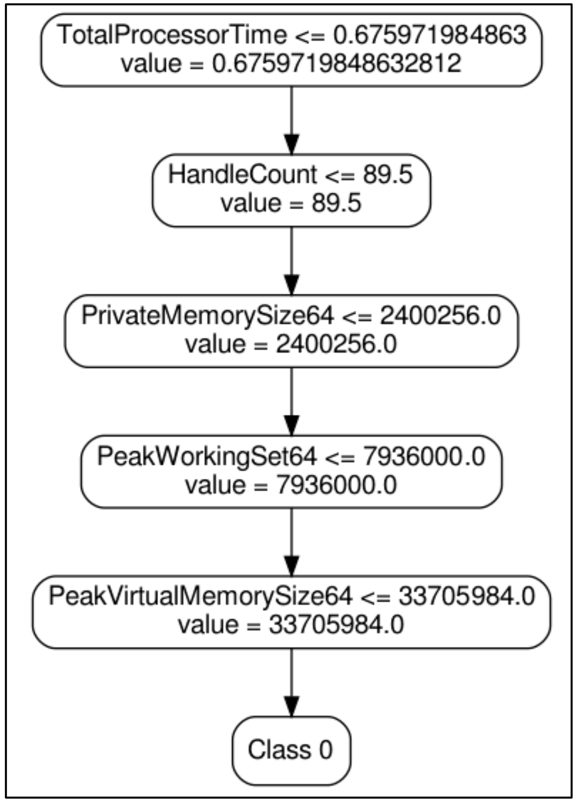
- Initially through manual interrogation of raw CSV process output
    - Removal of feature selection made this un-workable
- Modified to produce multiple output formats at point of decision
    - CSV
    - JSON
    - DOT
    - PNG

Interpretability

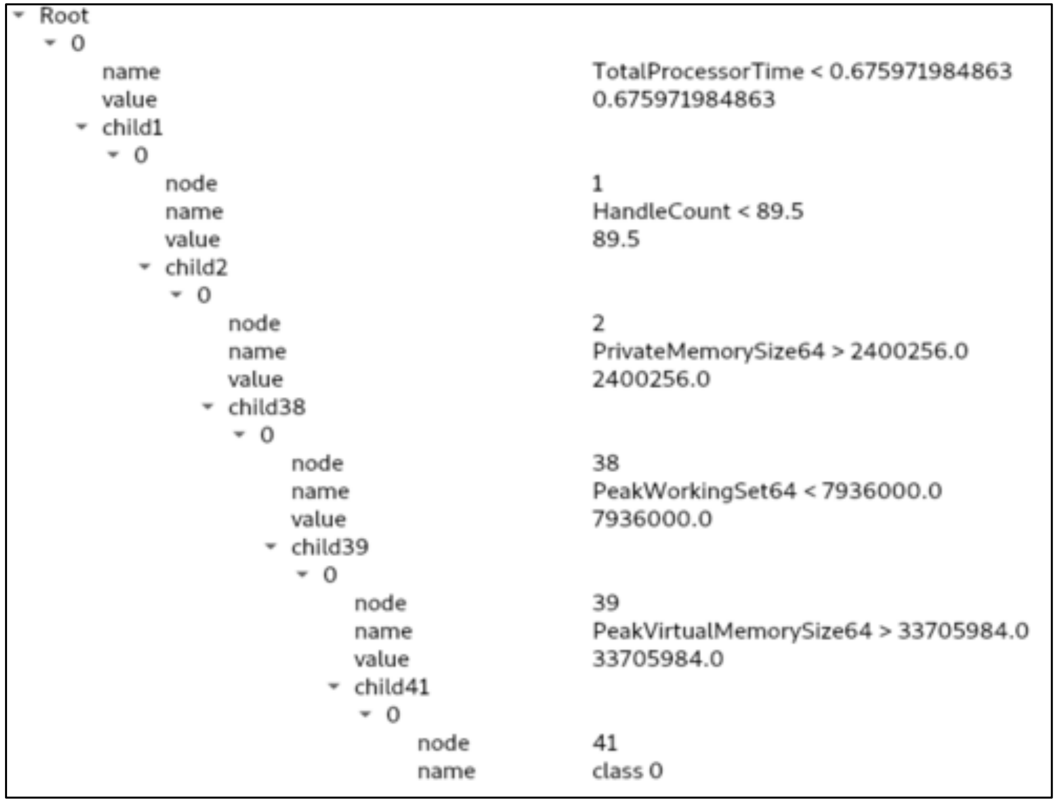


```
TREE: 41
0 NODE: feature[ProcessorAffinity] > 0.5 next=76
76 NODE: feature[HasExited] < 0.5 next=77
77 NODE: feature[PeakVirtualMemorySize64] < 94238720.0 next=78
78 NODE: feature[UserProcessorTime] < 0.0150215998292 next=79
79 NODE: feature[PagedSystemMemorySize64] < 103664.0 next=80
80 LEAF: return class=0
TREE: 42
0 NODE: feature[TotalProcessorTime] < 0.675971984863 next=1
1 NODE: feature[HandleCount] < 89.5 next=2
2 NODE: feature[PrivateMemorySize64] > 2400256.0 next=38
38 NODE: feature[PeakWorkingSet64] < 7936000.0 next=39
39 NODE: feature[PeakVirtualMemorySize64] > 33705984.0 next=41
41 LEAF: return class=0
TREE: 43
TREE: 44
0 NODE: feature[TotalProcessorTime] < 0.675971984863 next=1
1 NODE: feature[PrivilegedProcessorTime] > 0.00500719994307 next=71
71 NODE: feature[PeakVirtualMemorySize64] < 66170880.0 next=72
72 NODE: feature[NonpagedSystemMemorySize64] > 2316.0 next=84
84 NODE: feature[HandleCount] < 68.0 next=85
85 NODE: feature[PeakVirtualMemorySize64] < 53284864.0 next=86
86 NODE: feature[PeakWorkingSet64] < 5062656.0 next=87
87 NODE: feature[Description] < 0.5 next=88
88 LEAF: return class=0
```

CSV output


TotalProcessorTime <= 0.675971984863
value = 0.6759719848632812
HandleCount <= 89.5
value = 89.5
PrivateMemorySize64 <= 2400256.0
value = 2400256.0
PeakWorkingSet64 <= 7936000.0
value = 7936000.0
PeakVirtualMemorySize64 <= 33705984.0
value = 33705984.0
Class 0

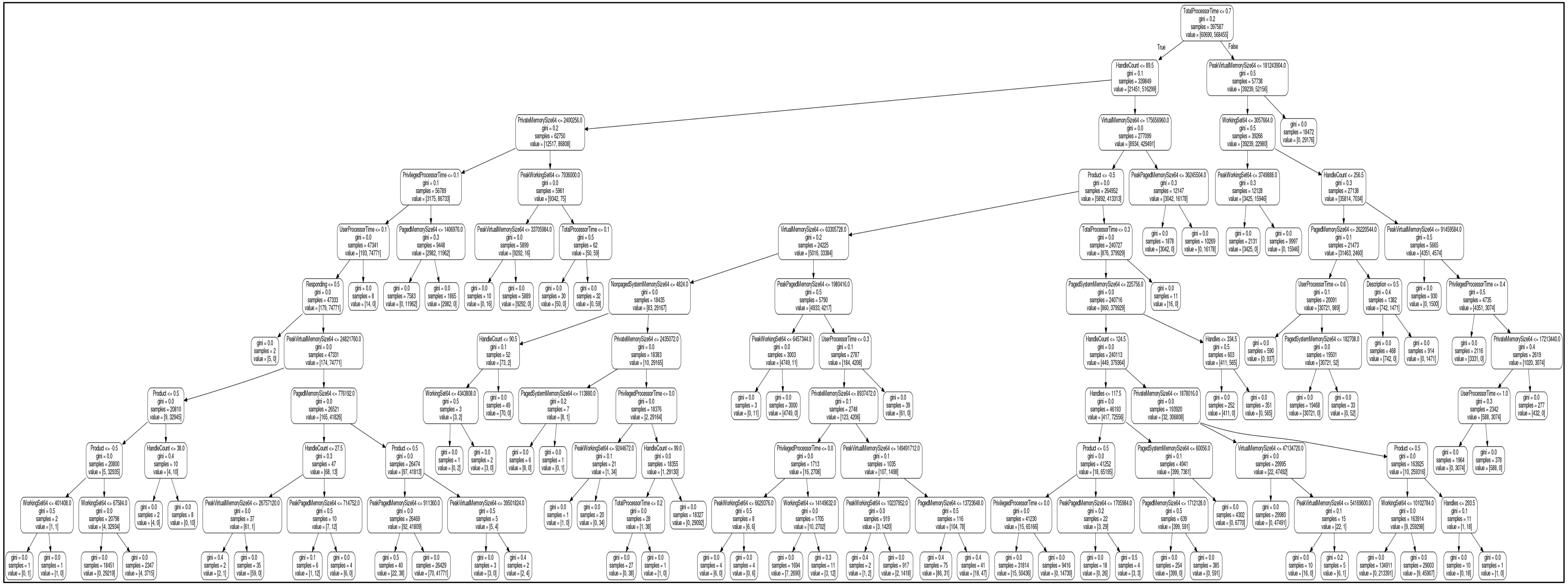
DOT output



```
▾ Root
  ▾ 0
      name                    TotalProcessorTime < 0.675971984863
      value                   0.675971984863
    ▾ child1
      ▾ 0
          node                1
          name                HandleCount < 89.5
          value               89.5
        ▾ child2
          ▾ 0
              node            2
              name            PrivateMemorySize64 > 2400256.0
              value           2400256.0
            ▾ child38
              ▾ 0
                  node        38
                  name        PeakWorkingSet64 < 7936000.0
                  value       7936000.0
                ▾ child39
                  ▾ 0
                      node    39
                      name    PeakVirtualMemorySize64 > 33705984.0
                      value   33705984.0
                    ▾ child41
                      ▾ 0
                          node 41
                          name class 0
```

JSON output

Interpretability


TotalProcessorTime <= 0.7 gini = 0.2 samples = 397587 value = [60690, 568455]
True
False
HandleCount <= 89.5 gini = 0.1 samples = 339849 value = [21451, 516299]
PeakVirtualMemorySize64 <= 181243904.0 gini = 0.5 samples = 57738 value = [39239, 52156]
PrivateMemorySize64 <= 2400256.0 gini = 0.2 samples = 62750 value = [12517, 86808]
VirtualMemorySize64 <= 175656960.0 gini = 0.0 samples = 277099 value = [8934, 429491]
WorkingSet64 <= 3057664.0 gini = 0.5 samples = 39266 value = [39239, 22980]
gini = 0.0 samples = 18472 value = [0, 29176]
PrivilegedProcessorTime <= 0.1 gini = 0.1 samples = 56789 value = [3175, 86733]
PeakWorkingSet64 <= 7936000.0 gini = 0.0 samples = 5961 value = [9342, 75]
Product <= -0.5 gini = 0.0 samples = 264952 value = [5892, 413313]
PeakPagedMemorySize64 <= 36245504.0 gini = 0.3 samples = 12147 value = [3042, 16178]
PeakWorkingSet64 <= 3749888.0 gini = 0.3 samples = 12128 value = [3425, 15946]
HandleCount <= 256.5 gini = 0.3 samples = 27138 value = [35814, 7034]
UserProcessorTime <= 0.1 gini = 0.0 samples = 47341 value = [193, 74771]
PagedMemorySize64 <= 1406976.0 gini = 0.3 samples = 9448 value = [2982, 11962]
PeakVirtualMemorySize64 <= 33705984.0 gini = 0.0 samples = 5899 value = [9292, 16]
TotalProcessorTime <= 0.1 gini = 0.5 samples = 62 value = [50, 59]
VirtualMemorySize64 <= 63305728.0 gini = 0.2 samples = 24225 value = [5016, 33384]
TotalProcessorTime <= 0.3 gini = 0.0 samples = 240727 value = [876, 379929]
gini = 0.0 samples = 1878 value = [3042, 0]
gini = 0.0 samples = 10269 value = [0, 16178]
gini = 0.0 samples = 2131 value = [3425, 0]
gini = 0.0 samples = 9997 value = [0, 15946]
PagedMemorySize64 <= 26220544.0 gini = 0.1 samples = 21473 value = [31463, 2460]
PeakVirtualMemorySize64 <= 91459584.0 gini = 0.5 samples = 5665 value = [4351, 4574]
Responding <= 0.5 gini = 0.0 samples = 47333 value = [179, 74771]
gini = 0.0 samples = 8 value = [14, 0]
gini = 0.0 samples = 7583 value = [0, 11962]
gini = 0.0 samples = 1865 value = [2982, 0]
gini = 0.0 samples = 10 value = [0, 16]
gini = 0.0 samples = 5889 value = [9292, 0]
gini = 0.0 samples = 30 value = [50, 0]
gini = 0.0 samples = 32 value = [0, 59]
NonpagedSystemMemorySize64 <= 4824.0 gini = 0.0 samples = 18435 value = [83, 29167]
PeakPagedMemorySize64 <= 1980416.0 gini = 0.5 samples = 5790 value = [4933, 4217]
PagedSystemMemorySize64 <= 225756.0 gini = 0.0 samples = 240716 value = [860, 379929]
gini = 0.0 samples = 11 value = [16, 0]
UserProcessorTime <= 0.6 gini = 0.1 samples = 20091 value = [30721, 989]
Description <= 0.5 gini = 0.4 samples = 1382 value = [742, 1471]
gini = 0.0 samples = 930 value = [0, 1500]
PrivilegedProcessorTime <= 0.4 gini = 0.5 samples = 4735 value = [4351, 3074]
gini = 0.0 samples = 2 value = [5, 0]
PeakVirtualMemorySize64 <= 24821760.0 gini = 0.0 samples = 47331 value = [174, 74771]
HandleCount <= 90.5 gini = 0.1 samples = 52 value = [73, 2]
PrivateMemorySize64 <= 2435072.0 gini = 0.0 samples = 18383 value = [10, 29165]
PeakWorkingSet64 <= 6457344.0 gini = 0.0 samples = 3003 value = [4749, 11]
UserProcessorTime <= 0.3 gini = 0.1 samples = 2787 value = [184, 4206]
HandleCount <= 124.5 gini = 0.0 samples = 240113 value = [449, 379364]
Handles <= 334.5 gini = 0.5 samples = 603 value = [411, 565]
gini = 0.0 samples = 590 value = [0, 937]
PagedSystemMemorySize64 <= 182708.0 gini = 0.0 samples = 19501 value = [30721, 52]
gini = 0.0 samples = 468 value = [742, 0]
gini = 0.0 samples = 914 value = [0, 1471]
gini = 0.0 samples = 2116 value = [3331, 0]
PrivateMemorySize64 <= 17213440.0 gini = 0.4 samples = 2619 value = [1020, 3074]
Product <= 0.5 gini = 0.0 samples = 20810 value = [9, 32945]
PagedMemorySize64 <= 776192.0 gini = 0.0 samples = 26521 value = [165, 41826]
WorkingSet64 <= 4343808.0 gini = 0.5 samples = 3 value = [3, 2]
gini = 0.0 samples = 49 value = [70, 0]
PagedSystemMemorySize64 <= 113880.0 gini = 0.2 samples = 7 value = [8, 1]
PrivilegedProcessorTime <= 0.0 gini = 0.0 samples = 18376 value = [2, 29164]
gini = 0.0 samples = 3 value = [0, 11]
gini = 0.0 samples = 3000 value = [4749, 0]
PrivateMemorySize64 <= 8937472.0 gini = 0.1 samples = 2748 value = [123, 4206]
gini = 0.0 samples = 39 value = [61, 0]
Handles <= 117.5 gini = 0.0 samples = 46193 value = [417, 72556]
PrivateMemorySize64 <= 1878016.0 gini = 0.0 samples = 193920 value = [32, 306808]
gini = 0.0 samples = 252 value = [411, 0]
gini = 0.0 samples = 351 value = [0, 565]
gini = 0.0 samples = 19468 value = [30721, 0]
gini = 0.0 samples = 33 value = [0, 52]
UserProcessorTime <= 1.0 gini = 0.3 samples = 2342 value = [588, 3074]
gini = 0.0 samples = 277 value = [432, 0]
Product <= -0.5 gini = 0.0 samples = 20800 value = [5, 32935]
HandleCount <= 38.0 gini = 0.4 samples = 10 value = [4, 10]
HandleCount <= 27.5 gini = 0.3 samples = 47 value = [68, 13]
Product <= 0.5 gini = 0.0 samples = 26474 value = [97, 41813]
gini = 0.0 samples = 1 value = [0, 2]
gini = 0.0 samples = 2 value = [3, 0]
gini = 0.0 samples = 6 value = [8, 0]
gini = 0.0 samples = 1 value = [0, 1]
PeakWorkingSet64 <= 9244672.0 gini = 0.1 samples = 21 value = [1, 34]
HandleCount <= 99.0 gini = 0.0 samples = 18355 value = [1, 29130]
PrivilegedProcessorTime <= 0.0 gini = 0.0 samples = 1713 value = [16, 2708]
PeakVirtualMemorySize64 <= 149491712.0 gini = 0.1 samples = 1035 value = [107, 1498]
Product <= 0.5 gini = 0.0 samples = 41252 value = [18, 65195]
PagedSystemMemorySize64 <= 60056.0 gini = 0.1 samples = 4941 value = [399, 7361]
VirtualMemorySize64 <= 47134720.0 gini = 0.0 samples = 29995 value = [22, 47492]
Product <= 0.5 gini = 0.0 samples = 163925 value = [10, 259316]
gini = 0.0 samples = 1964 value = [0, 3074]
gini = 0.0 samples = 378 value = [588, 0]
WorkingSet64 <= 401408.0 gini = 0.5 samples = 2 value = [1, 1]
WorkingSet64 <= 67584.0 gini = 0.0 samples = 20798 value = [4, 32934]
gini = 0.0 samples = 2 value = [4, 0]
gini = 0.0 samples = 8 value = [0, 10]
PeakVirtualMemorySize64 <= 26757120.0 gini = 0.0 samples = 37 value = [61, 1]
PeakPagedMemorySize64 <= 714752.0 gini = 0.5 samples = 10 value = [7, 12]
PeakPagedMemorySize64 <= 911360.0 gini = 0.0 samples = 26469 value = [92, 41809]
PeakVirtualMemorySize64 <= 39501824.0 gini = 0.5 samples = 5 value = [5, 4]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 20 value = [0, 34]
TotalProcessorTime <= 0.2 gini = 0.0 samples = 28 value = [1, 38]
gini = 0.0 samples = 18327 value = [0, 29092]
PeakWorkingSet64 <= 6629376.0 gini = 0.5 samples = 8 value = [6, 6]
WorkingSet64 <= 14149632.0 gini = 0.0 samples = 1705 value = [10, 2702]
PeakWorkingSet64 <= 10237952.0 gini = 0.0 samples = 919 value = [3, 1420]
PagedMemorySize64 <= 13723648.0 gini = 0.5 samples = 116 value = [104, 78]
PrivilegedProcessorTime <= 0.0 gini = 0.0 samples = 41230 value = [15, 65166]
PeakPagedMemorySize64 <= 1705984.0 gini = 0.2 samples = 22 value = [3, 29]
PagedMemorySize64 <= 1712128.0 gini = 0.5 samples = 639 value = [399, 591]
gini = 0.0 samples = 4302 value = [0, 6770]
gini = 0.0 samples = 29980 value = [0, 47491]
PeakVirtualMemorySize64 <= 54169600.0 gini = 0.1 samples = 15 value = [22, 1]
WorkingSet64 <= 10102784.0 gini = 0.0 samples = 163914 value = [9, 259298]
Handles <= 293.5 gini = 0.1 samples = 11 value = [1, 18]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 18451 value = [0, 29219]
gini = 0.0 samples = 2347 value = [4, 3715]
gini = 0.4 samples = 2 value = [2, 1]
gini = 0.0 samples = 35 value = [59, 0]
gini = 0.1 samples = 6 value = [1, 12]
gini = 0.0 samples = 4 value = [6, 0]
gini = 0.5 samples = 40 value = [22, 38]
gini = 0.0 samples = 26429 value = [70, 41771]
gini = 0.0 samples = 3 value = [3, 0]
gini = 0.4 samples = 2 value = [2, 4]
gini = 0.0 samples = 27 value = [0, 38]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 4 value = [6, 0]
gini = 0.0 samples = 4 value = [0, 6]
gini = 0.0 samples = 1694 value = [7, 2690]
gini = 0.3 samples = 11 value = [3, 12]
gini = 0.4 samples = 2 value = [1, 2]
gini = 0.0 samples = 917 value = [2, 1418]
gini = 0.4 samples = 75 value = [86, 31]
gini = 0.4 samples = 41 value = [18, 47]
gini = 0.0 samples = 31814 value = [15, 50436]
gini = 0.0 samples = 9416 value = [0, 14730]
gini = 0.0 samples = 18 value = [0, 26]
gini = 0.5 samples = 4 value = [3, 3]
gini = 0.0 samples = 254 value = [399, 0]
gini = 0.0 samples = 385 value = [0, 591]
gini = 0.0 samples = 10 value = [16, 0]
gini = 0.2 samples = 5 value = [6, 1]
gini = 0.0 samples = 134911 value = [0, 213391]
gini = 0.0 samples = 29003 value = [9, 45907]
gini = 0.0 samples = 10 value = [0, 18]
gini = 0.0 samples = 1 value = [1, 0]

PNG output

Results

- Binary values
  - Benignware samples were largely True or False for all
  - Malware samples showed a greater variance
- Variable data
  - Benignware processes had on average a higher score
  - Some Malware and Benignware processes within the same ‘score bracket’
  - Malware processes had (on average) higher amounts of private data

Results

- Decision specific data allowed the confirmation of assessments from manual interrogation
- * These features appeared twice, with different threshold values

Root Node Feature	Frequency
Processor Affinity	20%
Total Processor Time	16%
User Processor Time	16%
Handle	13%
Path	12%
Product	10%
Privileged Processor Time	3%
Peak Virtual Memory Size64	2%
Paged System Memory Size64*	<=2%
Virtual Memory Size64	1%
Handle Count*	<=1%
Handles	< 1%
Working Set 64	< 1%

Results

- The highlighted features had previously been identified through feature selection
- This lent weight to previous assessments made during manual interrogation of the data

Root Node Feature	Frequency
Processor Affinity	20%
Total Processor Time	16%
User Processor Time	16%
Handle	13%
Path	12%
Product	10%
Privileged Processor Time	3%
Peak Virtual Memory Size64	2%
Paged System Memory Size64*	<=2%
Virtual Memory Size64	1%
Handle Count*	<=1%
Handles	< 1%
Working Set 64	< 1%

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

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?