MAXS: Scaling Malware Execution with Sequential Multi-Hypothesis Testing

Authors: Phani Vadrevu and Roberto Perdisci

Presented by: Ashwag Altayyar

CISC850
Cyber Analytics
Bare-metal Analysis Environments

• Forcing the malware sample to run on a native system.

• Incurring a high hardware costs.

• Therefore, limiting the number of malware samples.
Problem statement

• Malware analysis environments execute each sample blindly

• Most new malware is repackaged previously analyzed malware.
Resource savings vs Information loss

- Increasing the number of malware samples.
- Reducing the amount of execution time.
- Saving information.
- Reducing the number of malware samples.
- Reducing execution time.
- Losing information.

• Increasing the number of malware samples.
• Reducing the amount of execution time.
• Minimizing the risk of information loss.
MAXS (Malware Analysis eXecution Scaler)

A novel probabilistic multi-hypothesis testing framework for scaling execution in malware analysis environments, including bare-metal execution environments.
Goals and Benefits:

• Increasing the capacity of malware analysis environments by reducing the execution time for each sample.

• Minimizing the information loss.
• MAXS provides a new probabilistic decision framework.

• Every time a new event is observed:
  1- The probability that the sample belongs to a previously learned malware family.

  2- The probability that the sample will generate previously unseen malware behaviors.
MAXS FRAMEWORK

1- A learning phase

2- An operational phase
Learning Phase

(a) Learning phase: malware family discovery and family behavior profile extraction.

- Measuring the similarity by computing the Jaccard index.
- Using DBSCAN clustering algorithm (Density-based spatial clustering of applications with noise).
Operational Phase

(b) Operational phase: overview of sequential tests applied to new malware samples.

Figure 1: MAXS framework.

main parameters to examine the Probabilities

- $\beta$: Threshold to examine the probability ($P_f$)
- $\gamma$: Threshold to examine the probability ($P_b$)
EVALUATION

Goal:

- Decreasing the execution time while minimizing the information loss

Dataset:

- Two large collections of malware execution traces obtained from two different production-level analysis environments (SA, SB)
- 1,251,865 malware samples from SA, and 400,041 from SB

<table>
<thead>
<tr>
<th>dataset</th>
<th>prefixed run time</th>
<th>collection days</th>
<th>avg. samples / day</th>
<th>avg. samples with DNS queries / day</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_A$</td>
<td>240s</td>
<td>77</td>
<td>16,258</td>
<td>15,431</td>
</tr>
<tr>
<td>$M_B$</td>
<td>360s</td>
<td>6</td>
<td>66,674</td>
<td>62,063</td>
</tr>
</tbody>
</table>

Table 1: Summary of malware dataset properties.
Experiments Setup

• Applying to different types of events:
  - Domain name queries extracted via dynamic analysis
  - Malware information extracted via static analysis

• Measuring time savings and information loss
Experiment 1: Malware Domain Intelligence

- MAXS monitors the sequence of domain name queries
- performed on both datasets MA and MB.
Parameter Selection

\[ \beta = 0.05 \text{ and } \gamma = 0.1, \text{ time savings above 40\% with less than 0.1\% of sample with information loss} \]
Longitudinal Train-Test Experiments

Dataset MA:
- Over three months (July, August, and December 2013)
- Three contiguous days for training and building the family behavior profiles.
- The next day for testing and measuring the time savings and information loss.

Dataset MB:
- Over six days (November 2014)
- One day of malware samples for training and one day for testing.
Longitudinal Train-Test Experiments

Figure 3: Longitudinal Study Experiment

<table>
<thead>
<tr>
<th>dataset</th>
<th>median time savings</th>
<th>median domain-based information loss</th>
<th>median samples responsible for loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA</td>
<td>42.2%</td>
<td>0.25%</td>
<td>0.07%</td>
</tr>
<tr>
<td>MB</td>
<td>45.5%</td>
<td>0.08%</td>
<td>0.03%</td>
</tr>
</tbody>
</table>
Summary of Result for Longitudinal Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>samples</th>
<th>samples assigned to a family</th>
<th>avg. family assignment time</th>
<th>avg. stop time</th>
<th>median time savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_A$</td>
<td>15,431</td>
<td>9,201</td>
<td>24.4s</td>
<td>69.6s</td>
<td>42.2%</td>
</tr>
<tr>
<td>$M_B$</td>
<td>62,063</td>
<td>34,305</td>
<td>28.3s</td>
<td>50.4s</td>
<td>45.5%</td>
</tr>
</tbody>
</table>

Table 2: Summary of results for longitudinal study experiments.

<table>
<thead>
<tr>
<th>dataset</th>
<th>prefixed run time</th>
<th>collection days</th>
<th>avg. samples / day</th>
<th>avg. samples with DNS queries / day</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_A$</td>
<td>240s</td>
<td>77</td>
<td>16,258</td>
<td>15,431</td>
</tr>
<tr>
<td>$M_B$</td>
<td>360s</td>
<td>6</td>
<td>66,674</td>
<td>62,063</td>
</tr>
</tbody>
</table>

Table 1: Summary of malware dataset properties.
Experiment 2: Leveraging Static Analysis Information

• Clustering the malware samples based on static analysis features and building family behavior profiles.

• Testing a new sample to decide whether it should be executed or not
The Result of Applying MAXS on Static Analysis Information

<table>
<thead>
<tr>
<th>Information</th>
<th>Time Saving %</th>
<th>Domain Loss %</th>
<th># Domains Lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static+Network</td>
<td>50.93 %</td>
<td>0.3 %</td>
<td>114</td>
</tr>
<tr>
<td>Static</td>
<td>37.16 %</td>
<td>0.22 %</td>
<td>82</td>
</tr>
<tr>
<td>Network</td>
<td>22.01 %</td>
<td>0.08 %</td>
<td>32</td>
</tr>
<tr>
<td>Network</td>
<td>45.5 %</td>
<td>0.08 %</td>
<td>35</td>
</tr>
</tbody>
</table>

Table 3: Average results for a “cascade” decision process using both static analysis information and network events.
Combining Static and Dynamic Analysis

• Applying MAXS on static analysis information

• For every malware sample executed in the first step, apply MAXS over the network events

<table>
<thead>
<tr>
<th>Information</th>
<th>Time Saving %</th>
<th>Domain Loss %</th>
<th># Domains Lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static+Network</td>
<td>50.93 %</td>
<td>0.3 %</td>
<td>114</td>
</tr>
<tr>
<td>Static</td>
<td>37.16 %</td>
<td>0.22 %</td>
<td>82</td>
</tr>
<tr>
<td>+Network</td>
<td>22.01 %</td>
<td>0.08 %</td>
<td>32</td>
</tr>
<tr>
<td>Network</td>
<td>45.5 %</td>
<td>0.08 %</td>
<td>35</td>
</tr>
</tbody>
</table>

Table 3: Average results for a “cascade” decision process using both static analysis information and network events.
Conclusion

The experimental results show that:

• Reduce malware execution time in average by up to 50%, with less than 0.3% information loss.

• Lower the cost of bare-metal analysis environments.