**Introduction**

- **What is Malware Obfuscation?**
  It is malware modified in order to make it difficult to detect it.
- **Malware camouflage progression:**
  
<table>
<thead>
<tr>
<th></th>
<th>No Stealth</th>
<th>Encrypted</th>
<th>Obfuscogenic</th>
<th>Polymorphic</th>
<th>Metamorphic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>1987</td>
<td>1990</td>
<td>1996 / now</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**O’Kane et al. (2016) achieved 86% DA with VxHeaven, we obtain 7% DA using Naive Bayes (NB)**

**Problem with obfuscated malware**

- An adversary may **obfuscate malware** affecting the accuracy of the approach.
- **Metame** modifies static features of the binary keeping its behaviour.
- **NB** is not able to detect obfuscated malware.

**Initial Framework**

![Initial Framework Diagram]

**Experimental Results**

**Conclusions and Ongoing work**

- This approach obtains better results and describes robustness.
- We are testing this approach with real data.
- To advance this approach could use different obfuscation techniques, other ML algorithms or several adversaries.

**References**


**Virus Total**

<table>
<thead>
<tr>
<th>Method</th>
<th>DA</th>
<th>FPR</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>0.86</td>
<td>0.21</td>
<td>0.03</td>
</tr>
</tbody>
</table>

**VxHeaven**

<table>
<thead>
<tr>
<th>Method</th>
<th>DA</th>
<th>FPR</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>0.93</td>
<td>0.11</td>
<td>0.02</td>
</tr>
</tbody>
</table>

**Problem with obfuscated malware**

- An adversary may **obfuscate malware** affecting the accuracy of the approach.
- **Metame** modifies static features of the binary keeping its behaviour.
- **NB** is not able to detect obfuscated malware.

**Cleo’s problem**

Cleo’s elements are:

- \( p_c(y) \), with \( p_c(M) + p_c(B) = 1 \) and \( p_c(M), p_c(B) \geq 0 \)
- \( p_c(x|y) \)
- \( p_c(x'|a, x) \)
- \( u_c(y_c, y) \)
- \( p_c(a, x, y) \)

Cleo aims at finding the class \( c(x’) \) maximising her expected utility

\[
 c(x’) = \arg\max_{y_c} \left[ u_c(y_c, M)p_c(M) \sum_{x \in x'} p_c(a, x'|x, M)p_c(x|M) + u_c(y_c, B)p_c(x'|B)p_c(B) \right]
\]

where \( p_c(a, x'|x, M) \) models the probability that Alan will perform attack \( a, x’, x' \) transforming \( x \) into \( x’ \).

**Alan’s problem**

Alan’s elements are:

- \( p_a(x'|a, x) \)
- \( u_a(y, y, a) \)
- \( p_a(c(x'|x)) \)

Alan seeks to maximise his expected utility through

\[
 A^*(x, M) = \arg\max_{a} \left( U_a(x, M, a) - U_a(x, B, M, a) \right) P_{a|x} + U_a(B, M, a)
\]

\( P_{a|x} \) could be based on an estimate \( P_{r|x} = M|x' \) with \( r = [0, 1] \). We could make \( P_{a|x} = \beta \epsilon(\delta_1, \delta_2) \).

**AROA**

- The model adopts the **ACRA** approach, Naveiro et al. (2019)
- The problem faced by Alan (the adversary) and Cleo (the classifier) is represented through **Bi-agent influence diagram**
- **Grey nodes** are the adversary decisions.
- **White nodes** are the classifier decisions.
- \( x, x’ \) represent the binary and the binary attacked, respectively.
- \( a, a' \) Alan’s attack chosen.
- \( y_c \) Cleo’s label prediction.
- \( u_c, u_a \) are Cleo’s and Alan’s utilities, respectively.

{alberto.redondo,david.rios}@icmat.es