AST-Based Deep Learning for Detecting Malicious PowerShell
Gili Rusak¹, Abdullah Al-Dujaili², Una-May O’Reilly²
Stanford University¹, ALFA Group, MIT²

Objective
- Combine static program analysis with deep learning approaches for PowerShell malware detection

Background
- **Introduction**
  - Cyberadversaries use PowerShell (PS) scripts for malicious purposes
  - Previous attempts to use deep learning for PS malware detection used character-level based neural networks [1]

- **Dataset**
  - 4,079 malicious PS scripts annotated and classified based on their family types [2]
  - Example: ShellCode Inject

- **Definitions**
  - Abstract Syntax Tree (AST): tree representation of syntactic structure of script made up of nodes
  - AST Subtree: a non-leaf node and its immediate children

Methods
- **PowerShell Scripts**
  - Extracted to Abstract Syntax Trees (PS AST)
  - Used for Malware Family Classification
  - Used for Learning AST Node Representations

Malware Family Classification
- **Data**
  - Classes: eight different malicious family types
  - Each class has 40 or more examples in dataset
  - Used 70:30 train:test split

- **Experiment**
  - Classify script by family type
  - Technique: RandomForestClassifier
  - Input Features: (PS AST depth, number of nodes)
  - Output: Family Type
  - Weighted classes during training based on number of examples per class due to class imbalance

- **Evaluation**
  - Heatmap for confusion matrix on the held out test set suggests a well-performing model

AST Node Representations
- **Data**
  -Parsed each of 4,079 PS ASTs to its subtrees
  - 62 different AST node types (i.e. ForStatement)

- **Experiment**
  -Learn embedding vector representations of AST nodes based on PS dataset using [3]’s methods
  - Technique: Unsupervised Stochastic Gradient Descent
  - Input: AST Subtrees of PS corpus
  - Output: Optimized vector representation of AST node types
  - Optimized SGD until loss stabilized and tuned hyperparameters

- **Evaluation**
  - Dendrogram of node types and their relationships
  - Promising preliminary results: (TryStatement, CatchClause) and (ForStatement, DoWhile) node types are neighbors
  - Limitations: ForEachStatement and ForStatement node types are not neighbors

Acknowledgements
- This work was supported by the MIT-IBM Watson AI Lab and CSAIL CyberSecurity Initiative. We thank Palo Alto Networks for the dataset.

References

Conclusions and Future Work
- AST-Based Deep learning techniques can be effectively harnessed for malware detection