Representation Learning for Code Malware

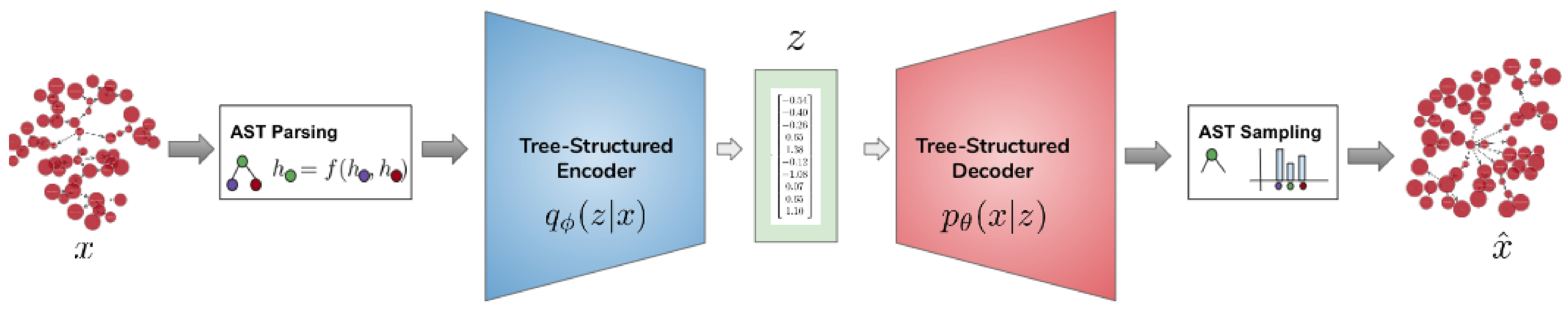
Sanja Simonovikj, Abdullah Al-Dujaili, Shashank Srikant,
Erik Hemberg, Una-May O'Reilly ALFA, CSAIL, MIT

GOAL: Learn a representation for Powershell code malware. Modeled using a Tree-Structured Variational Autoencoder which are robust to program tree and token-level obfuscations

Project Overview

- **PowerShell** – common target for cyberadversaries; can be obfuscated and executed from memory
- **Obfuscations** – different code but same functionality; defeat text-based approaches
- **Abstract Syntax Tree (AST)** – abstracts away code's specific details while retaining control flow and content-related information

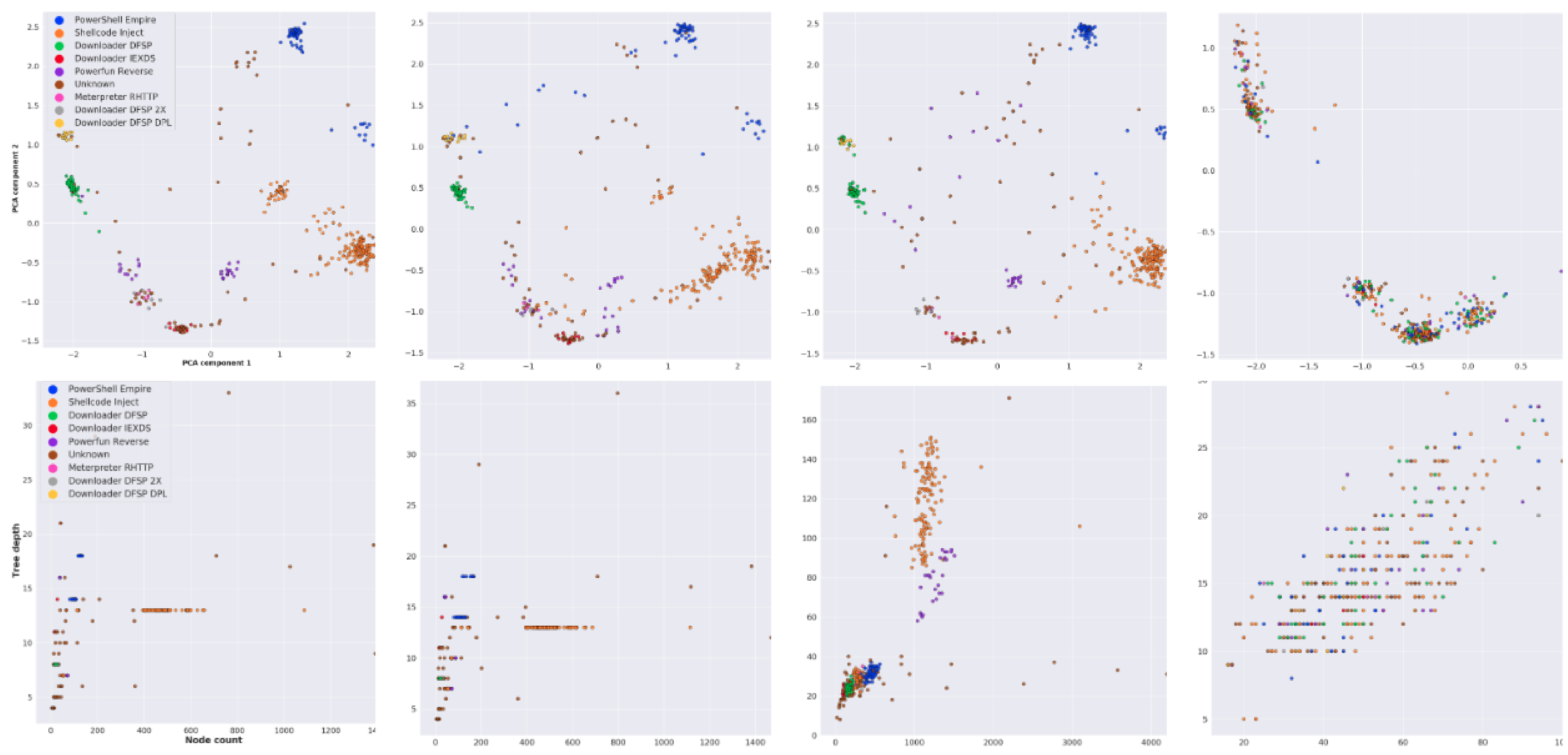
```
# NATURAL
$ipInfo = ifconfig | Select-String 'inet'

# AST Obfuscation
Set-Variable -Name ipInfo -Value (ifconfig | Select-String 'inet')

# TOKEN Obfuscation
${iPi`N`FO} = &("{1}{0}"-f 'onfig','ifc') | .("{0}{1}{2}{3}"
-f 'Select-S','tri','n','g') ("{0}{1}" -f 'in','et')

# STRING Obfuscation
('nmr'+'i'+'pIn'+'fo'+' ='+' if'+'conf'+'ig w9K Se'+'lect-Str'+
'ing Y'+'rb'+'inet'+'Yrb').REPlace(([cHAr]89+[cHAr]114+[cHAr]98),
[STriNG][cHAr]39).REPlace('w9K','|').REPlace('nmr','$') |& (IEX)
```



- **Variational Autoencoder (VAE)** – generative unsupervised method that can be used to learn representation for program trees

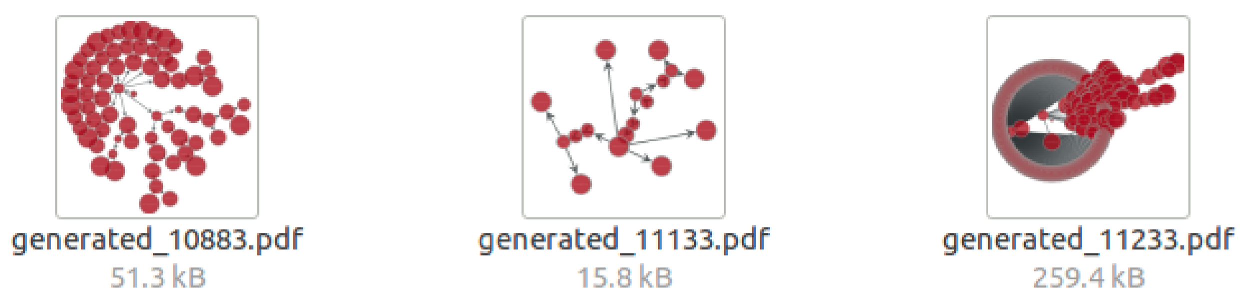
Current progress

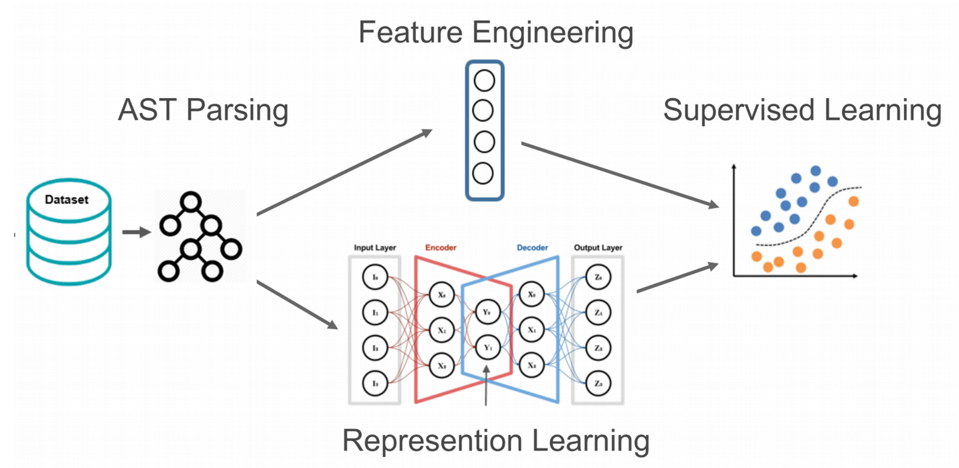
- Three types of obfuscations: *AST*, *TOKEN* and *STRING*; available from online tool *Invoke-Obfuscation*[1]
- Dataset – obtained from Palo Alto Networks[2]; originally 4079 datapoints, 469 after preprocessing
- Train Random Forest $\mathbf{R_B}$ with hand-engineered features
- Train Random Forest $\mathbf{R_E}$ with learned representations from tree-structured VAE
- Compare performance on both natural and obfuscated dataset



(a) Natural (b) AST-obfsucation (c) TOKEN-obfsucation (d) STRING-obfsucation

Samples	R_B accuracy %	R_E accuracy %
Natural Samples	96	90
AST Obfuscation	86	80
TOKEN Obfuscation	15	84
STRING Obfuscation	14	15





Observations

- The learned representations are robust against AST and TOKEN but not STRING obfuscations
- Further investigation lead to the fact that STRING obfuscations transform the code in a very specific manner where the code is converted to a string and is passed to IEX command, similar to the *eval* procedure in most programming languages. This resulted in very similar ASTs of very few nodes, which explains the failure of the STRING obfuscations observed both qualitatively and quantitatively.

Relevant links
1. Daniel Bohannon 2018. Invoke Obfuscation v1.8. https://github.com/danielbohannon/Invoke-Obfuscation
2. Jeff White 2017. Pulling Back the Curtains on Encoded Command PowerShell Attacks. https://researchcenter.paloaltonetworks.com/2017/03/unit42-pulling-back-the-curtains-on-encodedcommand-powershell-attacks

Open questions

- **Dataset** – need for larger, curated, labeled dataset that can be used for PowerShell malware detection and classification
- **AST engineering** – revealed shortcomings when applied to STRING obfuscations
- **De-obfuscation** - ML projects would require data preprocessing component where de-obfuscation might be essential

Future vision

- **Stronger baseline** – define a baseline that uses more complex features
- **Supervised learning** – try out supervised representation learning methods
- **Adversarial learning** – use obfuscated samples during training
- **Other languages** - explore languages other than PowerShell, (C, Python etc)

Firstname Una-May
Lastname O'Reilly | unamay@csail.mit.edu