Detecting Algorithmically Generated Domains using CNNs

By: Matt Jansen

Background / Motivation

- The problem:
  - Malware (e.g. viruses, worms, etc.) plagues the internet
  - Colonial Pipeline hit by ransomware this past month, had to halt operations briefly [2]
  - In the past decade, malware has been shown to implement Domain Generation Algorithms (DGAs) - generating algorithms on the fly to avoid detection
    - Example benign domain: www.google.com
    - Example DGA domain: yrugfqkarqitoir[.]ru
- The solution:
  - Use CNNs to detect algorithmically-generated domain names
- Overview
  - Review of the Invincea model for detecting malicious strings (filenames, URLs, registry keys)
  - Describe experimental setup, and datasets used
  - Show results of experimentation, and conclusions that can be drawn
  - Future work

Related Work - Various ML/DL Models

- Machine learning (lexical features):
  - Random Forest [6,12]
  - Unsupervised clustering [7]
- Deep learning:
  - Invincea (parallel CNN layers) [4]
  - Endgame (single LSTM layer) [8]
  - NYU (stacked CNN layers) [9]
  - CMU (forward LSTM layer + backward LSTM layer) [10]
  - MIT (stacked CNN layers + single LSTM layer) [11]

- In 2017, researchers as Invincea, Inc. had released a whitepaper describing a deep learning architecture for detecting malicious strings
- In 2018, the Invincea Model was re-implemented specifically for the purpose of detecting DGA domains in [3] and [5]

Technical Approach

- Invincea architecture [3,4]

- Used Adam optimizer, starting with LR=0.01
- Character embedding layer
  - Convert input into matrix, where each character is mapped to a 1x32 vector (of floats)
- Convolutional (feature-detection) layer
  - Apply multiple kernel convolutions
    - 1024 filters: 256 filters for each conv. kernel size: 2,3,4,5
  - Layer normalization, sum pooling and 50% dropout after each conv. layer
- Two hidden layers, each with 1024 nodes
- Single output layer with sigmoid activation, binary cross-entropy loss

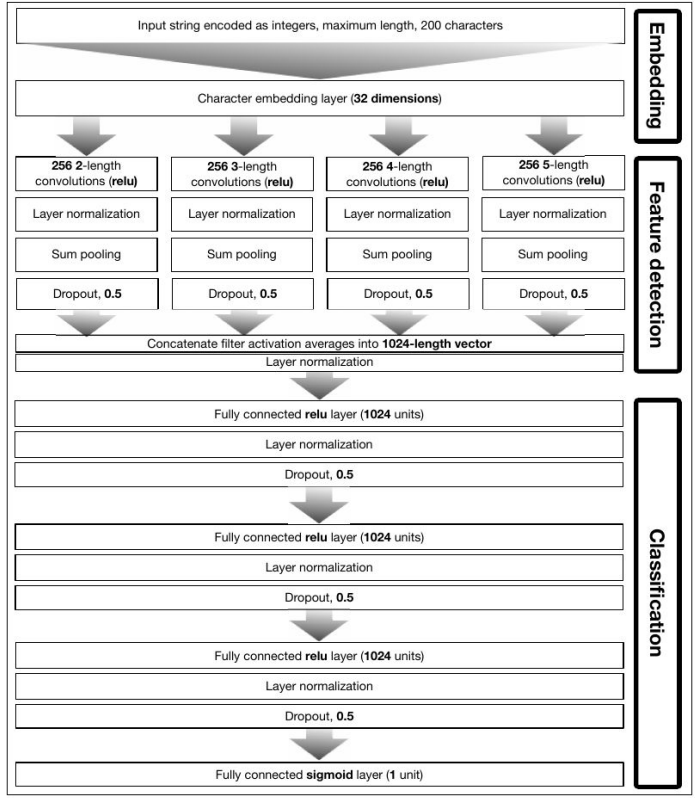


Fig. 2. The neural network architecture of our CNN model.

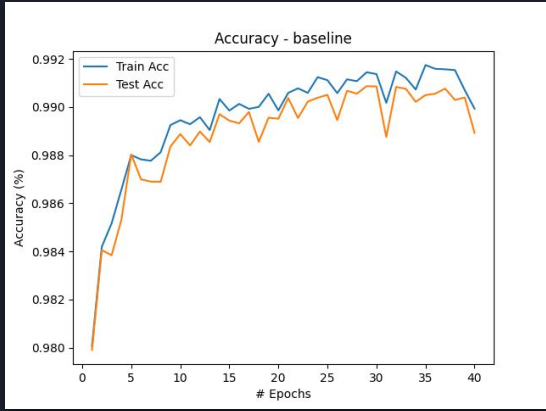
Experiment Setup

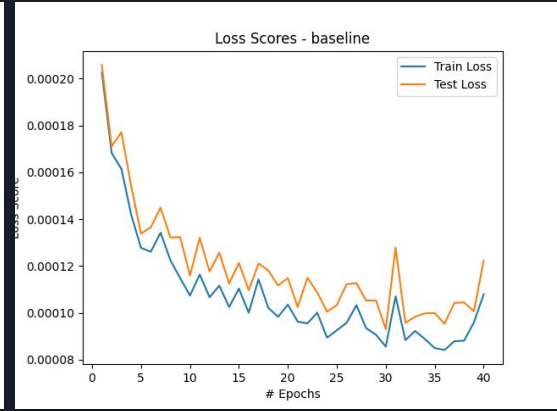
- Baseline model has the following features:
  - Batch size of 256 = 128 benign domains, 128 malicious domains (randomly shuffled)
  - Each character mapped to a 1x32 vector
  - Each convolutional layer output has 256-dimensions
  - Each fully-connected linear layer has 1024 nodes

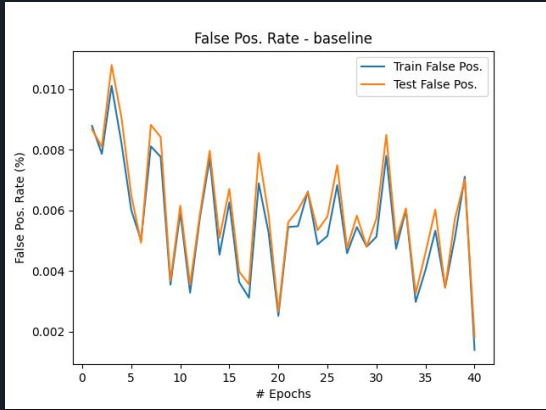
- I altered the model slightly in each of the above variables, including batch size, character embedding size, convolutional output, and number of linear nodes
  - During each experiment, I kept track of accuracy, loss scores, and false positive/negative rate

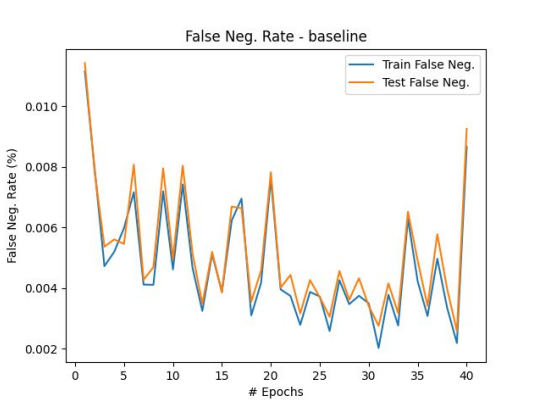
- Datasets
  - Benign domains: Cisco Umbrella 1 million (1 million domains)
  - DGA domains: Netlab-360 (1.4 million domains)

Experimental Results - Baseline







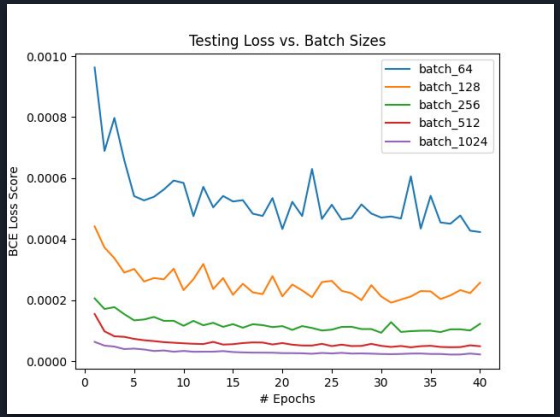


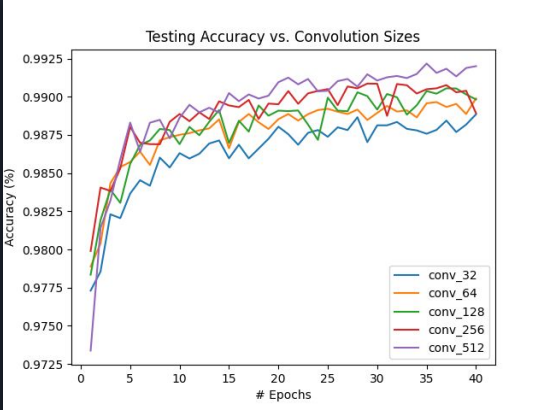
Experimental Results - Alterations

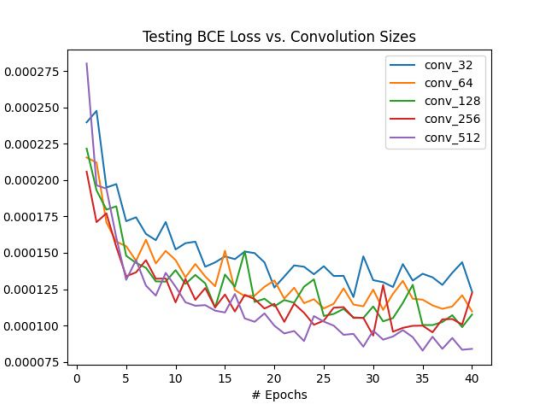
- Observations from altering the model
  - Observations are only for results on testing dataset

- Batch size
  - Only loss scores impacted (possibly due to averaging)
- Size of character embedding vector
  - No notable change for accuracy, loss scores, or false positive/negatives rates
- Convolutional output size
  - Larger convolution output entailed higher accuracy and lower loss scores
  - No notable change from false positive/negative rates
- Number of nodes in a single linear layer
  - No notable change for accuracy, loss scores, or false positive/negatives rates

Experimental Results - Alterations







Experimental Results - Alterations

- Observations from altering the model
  - Observations are only for results on testing dataset

- Batch size
  - Only loss scores impacted (possibly due to averaging)
- Size of character embedding vector
  - No notable change for accuracy, loss scores, or false positive/negatives rates
- Convolutional output size
  - Larger convolution output entailed higher accuracy and lower loss scores
  - No notable change from false positive/negative rates
- Number of nodes in a single linear layer
  - No notable change for accuracy, loss scores, or false positive/negatives rates

Experimental Results - Conclusions

- Observations from altering the model
  - With respect to the baseline model, you can decrease the size of the embedding for each character (from a 1x32 vector to a 1x8 vector) without having a negative impact on the accuracy, loss scores, or false positive/negative rates.
  - With respect to the baseline model, you can decrease the number of nodes in a single linear layer (from 1024 nodes to 256 nodes) without having a negative impact on the accuracy, loss scores, or false positive/negative rates.

- Why are these observations important?
  - A need for speed…
    - Cybersecurity analysts may have to sift through tens/hundreds of gigabytes of logs
    - Network security engineers may design NIDS to classify DNS traffic in real-time

Future Work

- Continuing model alterations
  - What else can be changed in order to minimize model?
  - Time needed to evaluate single domain in smaller models
- Better benign datasets
  - During initial presentation: question about benign dataset not being realistic enough
  - Hopefully find a way to obtain (free/open source) datasets of benign network traffic from residential, business, etc. sources
- Create evaluation tool
  - During incident response / hunting for malware on the network, use the tool against captured DNS traffic to assist in correlating malicious DNS traffic back to an infected computer

Citations

[1]https://purplesec.us/resources/cyber-security-statistics/ransomware/#:~:text=Ransomware%20has%20become%20a%20popular,years%20growing%20350%25%20in%202018.&text=81%25%20of%20cyber%20security%20experts,social%20actions%2C%20such%20as%20phishing.

[2] https://www.npr.org/2021/05/08/995040240/cybersecurity-attack-shuts-down-a-top-u-s-gasoline-pipeline

[3] Yu, Bin, et al. "Character level based detection of DGA domain names." 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018.

[4] Saxe, Joshua, and Konstantin Berlin. "eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys." arXiv preprint arXiv:1702.08568 (2017).

[5] Choudhary, Chhaya, et al. "Algorithmically generated domain detection and malware family classification." International Symposium on Security in Computing and Communication. Springer, Singapore, 2018.

[6] B. Yu, D. L. Gray, J. Pan, M. D. Cock and A. C. A. Nascimento, "Inline DGA Detection with Deep Networks," 2017 IEEE International Conference on Data Mining Workshops (ICDMW), 2017, pp. 683-692, doi: 10.1109/ICDMW.2017.96.

[7] Antonakakis, Manos, et al. "From throw-away traffic to bots: Detecting the rise of DGA-based malware." 21st {USENIX} Security Symposium ({USENIX} Security 12). 2012.

[8] Woodbridge, Jonathan, et al. "Predicting domain generation algorithms with long short-term memory networks." arXiv preprint arXiv:1611.00791 (2016).

[9] Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." arXiv preprint arXiv:1509.01626 (2015).

[10] Dhingra, Bhuwan, et al. "Tweet2vec: Character-based distributed representations for social media." arXiv preprint arXiv:1605.03481 (2016).

[11] Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2Vec: Learning Tweet Embeddings Using Character-level CNN-LSTM Encoder-Decoder. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16). Association for Computing Machinery, New York, NY, USA, 1041–1044.

[12] Pereira, Mayana, et al. "Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic." International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, Cham, 2018.

Questions?

https://www.cttsonline.com/files/2019/06/Raised-Hands-and-Questions-01-300x175.png