Pink Lemur: Convolutional Neural Network based DNS Tunnelling Detection

David Rodriguez Andrea Kaiser Dejan Donin Brian Somers
Cisco Systems, Inc.
{davrodr3,ascarfo,ddonin,bsomers}@cisco.com

February 7th, 2025

Introduction

- Pink-Lemur is a convolutional neural network trained to identify string encodings associated with data-exfiltration techniques in DNS.
- Uses a character embedding table and bottleneck convolutional network architecture.
- Achieves efficient and accurate technique to distinguish exfiltration and domain name labels prevalent in DNS.
- Achieves low false-positive rates.
- Utilizes a custom tensor library developed in C translated from PyTorch models
- Running on the edge in DNS resolvers requiring fast and scalable performance
- Classifies traffic in real-time.

Introduction (cont')

- DNS exfiltration is achieved by encoding a document in the labels of a domain name. Sometimes the label is encoded in *base8*, *base16*, *base32*.
- Information theory techniques such as *entropy* and *perplexity* have been used successfully on large strings
- However DNS names pose a particular problem since many names are short and don't follow typical language patterns.
- Additionally, detecting *exfiltration* in DNS is a high-risk problem: if wrong, internet resources are unreachable.
- By using a CNN model, we can curate training sets to carve out lexical characteristics that are unique to DNS names, in addition to managing heuristics.
- Thus, we tradeoff writing heuristics to curating training sets and retraining a CNN model in the event that model efficacy requires improvement.

DNS Resolution

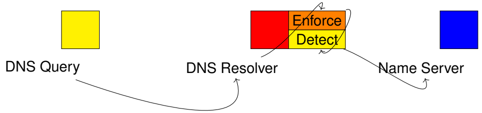


Figure 1: DNS Resolution Path through the Umbrella Resolvers.

Why a CNN?

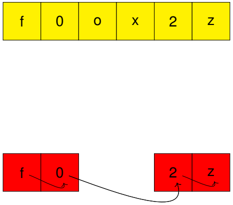


Figure 2: A CNN enables us to expand the window of character associations without having to leverage a transformer architecture.

CNN Architecture

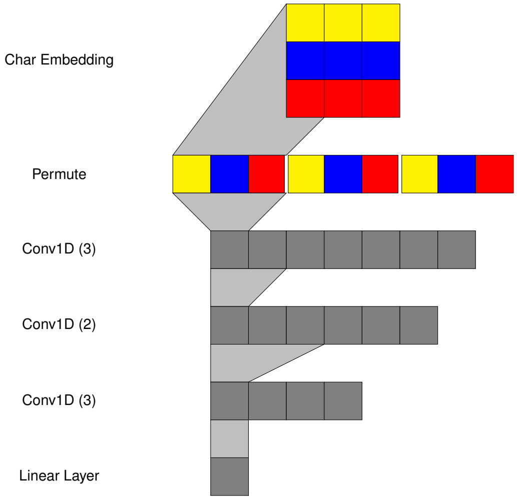


Figure 3: Pink-lemur convolutional neural network architecture.

- Two experiments were conducted on internal datasets gathered from different portions of Umbrella DNS traffic: the *Brew* and *SignatureDB* datasets. These two datasets have ground truth labels of more than 10K+ domains each.
- The results indicate that *pinklemur-0.3* performed best on the *Brew* dataset but the *pinklemur-0.1* performed best on the *SignatureDB* dataset. An interesting observation, is that adding source code encodings did not improve the model, which might be due to the number of possible domains that were derived from source code in the test sets.

Pink-Lemur Regression Tests

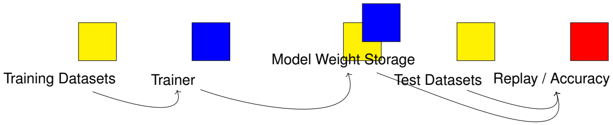


Figure 4: Model automation retrains models and exports weights. Each new model is then regressed across a holdout dataset to capture model drift and accuracy.

Model	F1	TP	FP	TN	FN	Unk
pinklemur-0.3	0.940	0.95	0.07	0.93	0.05	0.18
pinklemur-0.1	0.932	0.89	0.02	0.98	0.11	0.10
mrsmarkov-1.0	0.927	0.94	0.09	0.91	0.06	0.43
pinklemur-0.4	0.897	0.92	0.13	0.87	0.08	0.29
pinklemur-0.2	0.896	0.87	0.07	0.93	0.13	0.15

Table 1: Brew Dataset Model Leaderboard ranked by F1-statistic.

Model	F1	TP	FP	TN	FN	Unk
pinklemur-0.1	0.991	1.00	0.02	0.98	0.00	0.44
pinklemur-0.3	0.932	1.00	0.15	0.85	0.00	0.64
pinklemur-0.2	0.912	0.95	0.13	0.87	0.05	0.41
pinklemur-0.4	0.883	0.93	0.18	0.82	0.07	0.55
mrsmarkov-1.0	0.881	0.99	0.26	0.74	0.01	0.72

Table 2: SignatureDB Model Leaderboard ranked by F1-statistic.

Implementation Challenges

- Target implementation are the edge DNS resolver fleet hosts performing thousands of query resolutions and policy applications per second
- In the worst case, to ensure high efficacy, every DNS query would need to go through DNS tunnelling classification algorithm
- Goal is to quickly detect DNS tunnelling exfiltration query attempts, isolate them, flag them to the user and pass that information to the network monitoring infrastructure.

Pre-filtering Approach

One obvious approach to improving the implementation performance is the introduction of pre-filtering of DNS query domains which reduces the number of calls to the CNN-based classification algorithms. In our implementation we used multiple passes of such pre-filtering to reduce the load on the CNN-based algorithm itself.

- Does not perform DNS tunnelling detection for users that do not explicitly configure such detections.
- Quickly isolate and flag DNS queries that contain domains with known exfiltration vectors.
- Domains on pre-defined safe exclusion lists are not passed to the real-time DNS tunnelling detection algorithm.

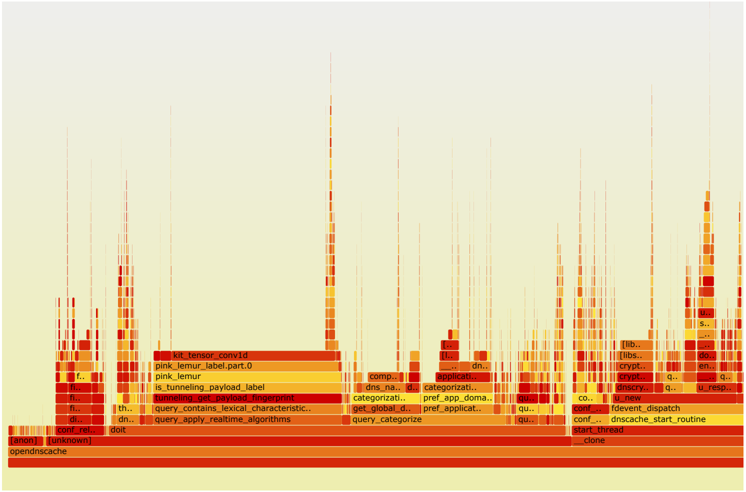
Performance Optimization

The most critical performance bottleneck of the feed forward implementation of pink-lemur CNN is the execution of 1D-convolution window on the permuted input.

- Our initial straightforward port of the PyTorch code was focused on maintaining equivalence with PyTorch results and was not optimized. We noticed that 1D-convolution window operation consumed more than 60% of the total processor usage.
- Analyzing the 1D-convolution code we identified that it is possible to compute ahead common parameters that are used over and over inside nested loops of the 1D-convolution function.
- These changes allowed for more efficient indexing operation of tensors that store inputs and outputs of the 1D-Convolution operations.
- In addition, we embedded these pre-calculated parameters in the tensor structure so that they are more easily passed along between the stages of CNN operation.

Optimization Results

Figure shows the flamegraph of the processor usage after these optimization steps.



Conclusion

- In this paper we discussed *pink-lemur* and a CNN architecture capable of detecting DNS tunneling and exfiltration attempts in DNS names.
- We experimented with different training sets to identify how the CNN architecture is capable of retaining differences in character combinations for different encodings seen in movies, images, and documents.
- The *pink-lemur* architecture has since been ported into C code to achieve maximum efficiency in the edge DNS resolvers and is running in realtime.
- The operation of the 1D-convolution window was optimized and this led to the implementation where DNS tunnelling detection requests can be classified in less than 1ms with minimal impact on DNS query resolution performance.

Future Work

Future discussions and points for consideration are:

- Further optimization of the CNN model.
- Integration with other network monitoring tools.
- Exploration of additional pre-filtering techniques.