Dynamic Kernel Security System

# DATABASE

Using montè carlo simulation, random cyber attack frequencies will be generated to train and test the algorithm. Taking inspiration from the KD-99 dataset we would create our own predicted dataset with some essential variables required by the algorithm, using this we would then create two databases, one which contains snippets of vulnerable code derived by trained tensor flow models and the other database would contain fixes for the deprecated code snippets.

Training the tensorflow algorithm to create databases

The tensor flow algorithm would be trained using previous iterations of deprecated UNIX kernel. The cyber attacks would be predicted using the montè carlo simulation and the model will be trained vigorously using the simulation output.

# KERNEL

Introduction

Using vanilla C, the kernel patching system would be created which reads the file block by block (buffer size 512 or bigger) to check for deprecated code snippets referring to the deprecated code database. If deprecated code is found, the code will be rewritten and reinitialised.

Pathway

The Software will move from the UNIX kernel source directory checking every file with an algorithm which checks chunks of code using an increasing bit method, i.e, first 16 bits of code will be checked, then 32, 64, 128 and 512, it can be increased in further iterations. This approach allows the software to detect different patterns of chunks of code for better detection.

After checking all the files in a directory, it will move on the next directory in the parent path. The algorithm will support both PDP-11 assembly and the C language. Once the kernel is rewritten, using live reinitialising methods the kernel will be reinitialised without the need of rebooting the system.

# Objectives

1. Training a tensor flow model to imitate cyber attacks for testing and training the patching algorithm.
2. Creating databases containing the deprecated code snippets and their fixes which would be accessed by the algorithm.
3. Creating an algorithm which reads blocks of code to check and rewrite the kernel code.