Dynamic Kernel Security System

Introduction

The dynamic kernel security system is used to work against kernel source code level attacks or bugs. The system is currently based on the UNIX kernel architecture but will be updated to be used with other kernel architectures soon.
It works on the kernel source code to inspect and see if any third party changes has been done. If the change results in a destructive manner, the algorithm will select the faulty code and provide changes accordingly. It can be set to either automatically apply the changes or ping the developer to review and apply the changes using a scripting language.

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.

Kernel

The kernel is a core component of the operating system which has majority of the control of the system processes, The UNIX kernel has been selected for current use case for its open source nature and wide usage in the research field, reviewing the previous UNIX source code repositories provides us with a robust code base to review and inspect for various kinds of bugs, attacks and code deprecations which restricts the proper functioning of the kernel. Using this we can provide security against small-scale attacks on the kernel, kernel based attacks which manipulate the source code has been recently on the rise so currently even small scale protection will be beneficial for the system. Working on the UNIX source codes provides security to major research firms, servers and some UNIX based work station. Further testing would be done on currently used UNIX based operating systems such as various flavours of BSD and some LINUX based operating systems. The system would be able to fix any deprecation and reinitialise the kernel based on a live-reinitialising method.

Database

Any research study relies heavily on data. To validate a theory, data and efficient data processing are required.
- Processing manual data is one method of data processing: The entire procedure is carried out by hand; Technology is only utilised occasionally.
- Processing of data mechanically: Data is processed without human intervention by automating the process with machines and computers.
- Processing of electronic data: the process of processing business data with automated methods. such as processing huge amounts of stock data.

For both the generation and processing of the data in this research paper, we employ a variety of algorithms. To be precise, we are employing a multiple probability simulation or the Monte Carlo

Method, a mathematical technique that is used to predict the likely outcomes of an uncertain event. Monte Carlo Simulations have been used to evaluate the effects of risk in a variety of real-world contexts, such as artificial intelligence, stock markets, sales forecasting, project management, and pricing, ever since they were first developed.

Working of Machine Learning Model and Algorithm

We employ the Monte Carlo algorithm to generate entropy. After that, we'll use the produced entropy. to construct the dataset that will be used to instruct the Machine Learning model, which will result in the production of two additional datasets. There will be patches for specific errors in another collection of depreciated code, but that is one. The basic operation of the Trained machine learning model is that it reads the kernel source code from the first unix source code that is available in the repo to examine various previous deprecations and how the issue was resolved. In essence, this model will be trained in such a way that it will frequently traverse various unix repos that contain both the problems and their solutions. The new issue that the model discovered and fixed will be automatically saved for future reference. Algorithm and the machine learning model collaborate.
The way it works is that the kernel's algorithm is written in plain old C. It looks at the source code one chunk at a time in increasing bit order up to a maximum buffer size or file size (for smaller files) and compares it to deprecated code snippets in the database of deprecated code. Using the fixed code database, the threat is evaluated and flagged in accordance if it is found in the database. Different levels of threats can be dealt with appropriately by using a script as shown in figure 1. For example, a low-level threat may be fixed automatically, while a high-level threat may be reported to the developer, who will investigate it and, if necessary, apply the patch. The senior developer may be notified if the threat level is unclear, and they will address it as necessary. Beginning with an algorithm that verifies code chunks using an increasing bit approach, the software will inspect each file in the UNIX kernel source directory, More iterations can be used to improve this algorithm. Both C and PDP-11 assembly will work with this approach.

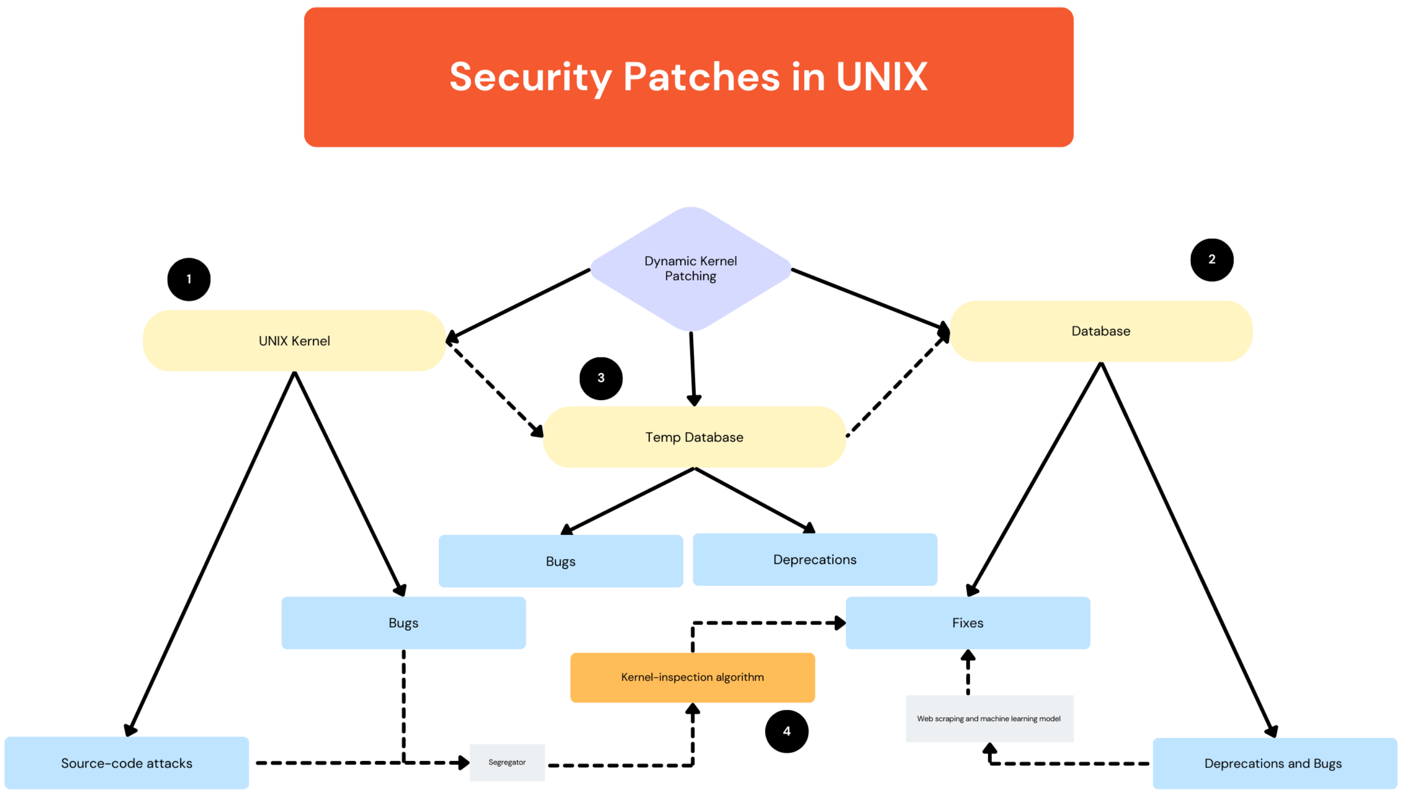

Figure 1.
(Explanation of notations are marked below)

	DESCRIPTION
[1] UNIX KERNEL	The most used research based open source operating system so testing and working on it is better for now
[2] DATABASE	The database consists of the deprecated code and fixed alternatives
[3] TEMP DATABASE	Temporary premade dataset, compiled and worked on to train the machine learning model
[4] KERNEL INSPECTION ALGORITHM	The algorithm made in vanilla C to check the source code in an increasing bit format to inspect the code and release fixes

Figure 2.

Conclusion

The system will analyse any third-party manipulation of the kernel source code and automatically give the patches using machine learning models and the kernel level algorithm. This will be included into the operating system's base layer to offer security against such assaults from the ground up.