UNIVERSITY OF HEIDELBERG

Project of the Practical AI Methods and Tools for Programming

**BinaryML:** Classifying Binaries for Malware

and Vulnerability Detection

Team Members:

**Anusha** Chattopadhyay

**Soumili** Samanta

Sri **Pavan** Sesha Sai Rallapalli

**Research papers used:**

* **For Malware detection model:**

1. [Recasting Self-Attention with Holographic Reduced Representations](https://arxiv.org/pdf/2305.19534.pdf)

* **For Vulnerability detection model:**

1. [ROMEO: A Binary Vulnerability Detection Dataset for Exploring Juliet through the Lens of Assembly Language](https://elib.dlr.de/194605/1/2112.06623.pdf)
2. [ILLUMINATI: Towards Explaining Graph Neural Networks for Cybersecurity Analysis](https://arxiv.org/pdf/2303.14836.pdf)
3. **Project Aspects Handled**:

* **Research for Finding Relevant Papers**:

Anusha Chattopadhyay and Soumili Samanta took the lead in researching and identifying various papers related to malware & vulnerability detection techniques, keeping the team informed about the latest advancements in the field.

Below is the detailed breakdown:

* Anusha Chattopadhyay: Explored 11 papers in both the domains
* Soumili Samanta: Explored 13 papers in both the domains
* Sri Pavan Sesha Sai Rallapalli: Explored 6 papers in both the domains
* **Understanding and development of main models:**
* “Recasting self-attention” malware detection model: Sri Pavan Sesha Sai Rallapalli
* “Romeo” vulnerability detection model: Soumili Samanta
* **“**Illuminati” vulnerability detection model: Anusha Chattopadhyay
* **Training:**
* Sri Pavan Sesha Sai Rallapalli handled the training of the "Recasting self-attention..." paper’s malware detection model as well as the vulnerability detection model (cross-domain) using vulnerability dataset.
* Anusha Chattopadhyay carried out the training for vulnerability detection model in “Romeo” paper along with the training of cross-domain malware detection model.
* **Data Loading and Preprocessing:**
* Malware detection model (“Recasting self-attention…” paper):
  1. Soumili Samanta is responsible for the data loader and preprocessing research.
  2. Sri Pavan Sesha Sai Rallapalli managed the data loading process in the malware detection model and also cross domain vulnerability detection model, ensuring that malware and vulnerable samples were correctly processed for training and testing.
* Vulnerability detection model (“Romeo” and “Illuminati” papers):
  1. Anusha Chattopadhyay is responsible for the data loader research, POC and code development for malware dataset in Illuminati.
  2. Sri Pavan Sesha Sai Rallapalli managed the implementation of data loader to process the cross-domain malware data in Romeo vulnerability detection model.
  3. Soumili Samanta worked on the research and implementation of a custom data loader for malware in Romeo model. Also responsible for understanding and managing the original Romeo’s vulnerability data loader.
* **Testing:**
* “Recasting self-attention…” models (original &cross domain): Sri Pavan Sesha Sai Rallapalli.
* “Romeo” models (original & cross domain): Anusha Chattopadhyay
* **Experimentation, Evaluation and Tuning:**
* Sri Pavan Sesha Sai Rallapalli and Soumili Samanta collaborated on fine-tuning hyperparameters, and evaluating the performance of Recasting self-attention’s malware and cross-domain vulnerability detection models.
* Anusha Chattopadhyay experimented with different dataset variations and contributed to tuning the Romeo’s both vulnerability detection model and cross-domain malware detection model for improved accuracy.

1. **Quantification of the efforts**

Below table provides an overview of the code files with percentage of efforts per file by participants:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Projects | Files / Authors | Pavan | Soumili | Anusha |
|  |  |  |  |  |
| Malware detection model (Recasting self attention) | dataset.py | 88,71% |  | 11,29% |
| hrrformer\_mgpu.py | 38,07% | 61,93% |  |
| initializations.py | 100,00% |  |  |
|  | predictive\_analysis\_stats.py | 69.8% | 30.2% |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Vulnerability detection model (Romeo) | juliet\_access/init.py | 100,00% |  |  |
| pipeline\_framework/init.py | 39,96% | 60,04% |  |
| pipeline\_steps/evaluation\_pipeline\_steps | 53,43% | 46,57% |  |
| classifier/init.py |  |  | 100,00% |
| pipeline\_steps/juliet\_pipeline\_steps.py | 100,00% |  |  |
| pipeline\_steps/preprocessing\_pipeline\_steps.py | 75,91% | 24,09% |  |
| preprocessing/init.py | 100,00% |  |  |
| run\_pipeline.py | 35,28% |  | 64,72% |
| pipeline\_steps/transformer\_pipeline\_steps.py |  | 100,00% |  |
| preprocessing/loader\_init.py |  | 100,00% |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Vulnerability detection model (Illuminati) | illuminati-changed files/load\_datasets.py |  |  | 100,00% |
| illuminati-changed files/train\_graph.py |  |  | 100,00% |
| illuminati-changed files/MalNet\_Tiny.py |  |  | 100,00% |
|  | illuminati-changed files/encoding.py |  |  | 100,00% |

1. **Percentage of code changes:**

Below table provides an overview of the amount of code changed when compared to the original project. i.e., stats of the extensions of the existing projects.

|  |  |  |
| --- | --- | --- |
| Projects | Files / Authors | Overall code change |
|  |  |  |
| Malware detection model (Recasting self attention) | dataset.py | 31% |
| hrrformer\_mgpu.py | 75,08% |
| initializations.py | 100% (new file) |
|  | predictive\_analysis\_stats.py | 100% (new file) |
|  |  |  |
|  |  |  |
| Vulnerability detection model (Romeo) | juliet\_access/init.py | 45,56% |
| pipeline\_framework/init.py | 1,99% |
| pipeline\_steps/evaluation\_pipeline\_steps | 2,76% |
| classifier/init.py | 6,50% |
| pipeline\_steps/juliet\_pipeline\_steps.py | 5,10% |
| pipeline\_steps/preprocessing\_pipeline\_steps.py | 9,29% |
| preprocessing/init.py | 17,68% |
| run\_pipeline.py | 14,89% |
| pipeline\_steps/transformer\_pipeline\_steps.py | 3,55% |
| preprocessing/loader\_init.py | 100% (new file) |
|  |  |  |
|  |  |  |
| Vulnerability detection model (Illuminati) | illuminati-changed files/load\_datasets.py | 28.70% |
| illuminati-changed files/train\_graph.py | 28.13% |
| illuminati-changed files/MalNet\_Tiny.py | 100,00% |
|  | illuminati-changed files/encoding.py | 100,00% |

1. **Individual code updates:**

Below table provides an overview of the percentage of code changes done by each participant in various files. These stats include the changes that may not in the final project, but were part of our research and experimentation to finalize on the optimal codes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Projects | Files / Authors | Pavan | Soumili | Anusha |
|  |  |  |  |  |
| Malware detection model (Recasting self attention) | dataset.py | 27,50% |  | 3,50% |
| hrrformer\_mgpu.py | 28,57% | 46,51% |  |
| initializations.py | 100% |  |  |
|  | predictive\_analysis\_stats.py | 69.8% | 30.2% |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Vulnerability detection model (Romeo) | juliet\_access/init.py | 45,56% |  |  |
| pipeline\_framework/init.py | 1,99% | 2,99% |  |
| pipeline\_steps/evaluation\_pipeline\_steps | 2,76% | 2,41% |  |
| classifier/init.py |  |  | 6,50% |
| pipeline\_steps/juliet\_pipeline\_steps.py | 5,10% |  |  |
| pipeline\_steps/preprocessing\_pipeline\_steps.py | 9,29% | 2,95% |  |
| preprocessing/init.py | 17,68% |  |  |
| run\_pipeline.py | 5,25% |  | 9,63% |
| pipeline\_steps/transformer\_pipeline\_steps.py |  | 3,55% |  |
| preprocessing/loader\_init.py |  | 100% |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Vulnerability detection model (Illuminati) | illuminati-changed files/load\_datasets.py |  |  | 28,70% |
| illuminati-changed files/train\_graph.py |  |  | 28,13% |
| illuminati-changed files/MalNet\_Tiny.py |  |  | 100,00% |
|  | illuminati-changed files/encoding.py |  |  | 100,00% |

1. **Note**:
2. All the above statistics are calculated based on the number of lines added or updated by each person. Similarly calculated the percentage of code extension which represents amount of code changed compared to original existing repositories.
3. All the diff files are uploaded in “docs” folder of our GitHub repository.