Presentation.py VulDetective.py

```
1
2
3   CSCS 2022 ‘Conference’ {
4
5     [VulDetective]
6
7     ##Towards a Block-Level ML-Based Python
8       Vulnerability Detection Tool
9     ##Machine Learning Based Vulnerability Detection
10    University of Szeged :
11          #Peter Hegedus
12          #Amirreza-Bagheri
13  }
14
```

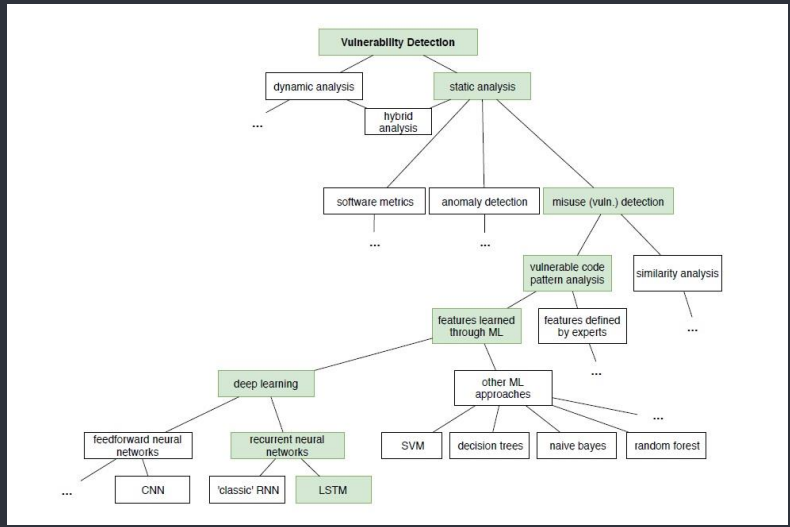
Introduction; {

‘Creating safe, dependable, and secure software is no easy undertaking.’

<p Software architects and engineers' oversights and errors can quickly lead to software vulnerabilities, with serious repercussions >

</p>

}



01 {

[Dataset Collecting]

< We need to use a data set of real source codes for our training for this purpose we used Github API with more than 127 million repositories>

}

Not Vulnerable{

< In general, we'll require non-vulnerable data for two purposes: first, we'll need a validated not-vulnerable data set to train our embedding layer with, and second, we'll need data for model training >

}

Vulnerable{

< We need it for the susceptible data, and we need to gather as much as possible for the primary model training to cover all distinct patterns of vulnerable source code >

}

02 {

[Filtering Dataset]

< We need to filter our data because we couldn't use any relevant filters when we got it from Github >

}

Filtering ‘Dataset’ {

- **Step 01** Only files with a character count of fewer than 10,000 were evaluated.
- **Step 02** Commits that altered or removed a file in more than ten separate places were removed.
- **Step 03** Remove any projects that act as indicators of security problems or exploits.
- **Step 04** Read commit messages to aid in the filtration of specific vulnerabilities.

}

03 {

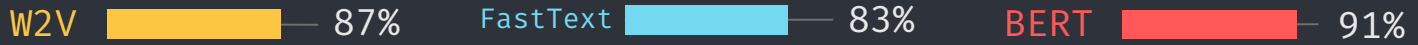
[Embedding Layer]

< We look into various embedding layers to see which one is ideal for conveying our model requirements >

}

Source code ‘representative’ {

Embedding layers



< Word2Vec is state of art embedding layer that usually used in NLP models >

< FastText is basically the same as word2vec but it uses different me >

< BERT is a pretrained embedding layer from microsft >

}

04 {

[Labeling]

< We look into various embedding layers to see which one is ideal for conveying our model requirements >

}

Labeling {

The data is tagged using information from the commit context :

Misguided attempt:

* Using only diffs
* subtle errors in creating the data

Processing the data :

* vulnerable and non-vulnerable treated the same

}

05 {

[Training]

< After we labeled our dataset and both embedding layer and model hyperparameters were set we can start training our model >

}

Training process {

Embedding Layer Training

< A substantial training base of clean and working code is necessary to train >

Model Training

< labeled data used for the model training to produce a forecast between 0 and 1>

Different Hyperprameters

< We tested different hyperprameters both for embedding layer and model training>

}

06 {

[Application]

< Finally, once the model has been built, we can use it to detect and estimate the percentage of vulnerability for each token, and we may display the results in text edit mode or as an image output >

}

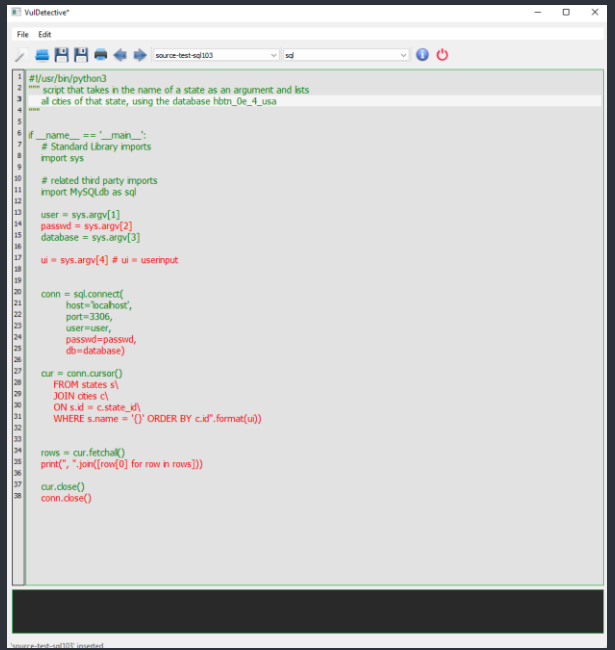
Presentation.py

VulDetective.py

‘VulDetective’ Application {

< The sections of the source code that our model identifies as vulnerable can be seen >

}



Presentation.py

VulDetective.py

ThankYou {

The presented work was carried out within the SETIT Project (2018-1.2.1-NKP-2018-00004). Project no. 2018-1.2.1-NKP-2018-00004 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2018-1.2.1-NKP funding scheme.

}