Predicting Vulnerable Software Components via Text Mining

JARGON

CYCLOMATIC COMPLEXITY

This is a software metric used to indicate the complexity of the software.

The cyclomatic complexity of a section of source code is the number of linearly independent path within it.

CODE CHURN

Added, deleted or modified lines from one version to another version.

DISCRETIZATION

It refers to the technique of converting numeric values to nominal ranges.

MOTIVATION:

- V&V {Verification and Validation} adds up to 30% to the development costs and even after that you can't be 100% sure, if your software is secure.

IMPORTANT POINTS:

- Portion of the codes which are changed frequently are more likely to have security bugs or vulnerabilities.
- Use of certain libraries makes the software more vulnerable.
- **Bag-of-words** representation model is used.
- we analyzed analysed 20 "apps" for the Android OS platform and followed their evolution over time. In total, we analyzed 182 releases, spanning a period of two years.
- It has been shown that this classifier model which uses term frequency provides results with good precision and recall.
    - Precision : Probability that classification of a file as vulnerable is correct.
    - Recall : Probability that the classification finds a file which is known to be vulnerable.

VULNERABILITY PREDICTION MODEL

5 dimensions

- type of prediction feature used.
- source of the vulnerability data.
- type of prediction technique.
- applications used for validation.
- available performance indicator.

STATIC CODE ANALYSIS AND VULNERABILITIES

- Warnings generated by SCA are strongly correlated with that generated by NVD.

RESEARCH METHODOLOGY

- Our main goal is to build a binary classifier to predict if a software component is vulnerable or not.

```
vulnerable if | 1 if number of warnings > 0,
              | 0 otherwise:
```

```
vulnerable = f(term frequencies)
```

PERFORMANCE PREDICTOR

```
Precision P = TP/(TP + FP)
Recall R = TP/(TP + FN)
Fall Out O = FP/(FP + TN)
```

NOTE: Our benchmark to judge a high quality model is 80 percent for both precision and recall.

SELECTION OF APPLICATIONS

- F-Droid Repository.
- As different languages use different keywords and naming conventions, we focused our text mining analysis on applications written in Java, which is the standard programming language used for Android application development.

- Applications must be at least 1000 lines of code.

DEPENDENT VARIABLE

- HP Fortify SCA was used as a static analysis tool.
- Each file was scanned and marked as **V** or NV based on the above tool.
- This tool also tells us the type of vulnerability & severity of vulnerability on a scale from 1 - 5.
- Scanning an application is a time consuming task.

INDEPENDENT VARIABLES

- Each Java file is tokenized as a vector of terms and frequency of each term in the file was counted.
- Delimiters = space, java punctuation characters, mathematical and logical operators.

MACHINE LEARNING TECHNIQUES

- 5 ML techniques were explored in this study
  - Random Forest
  - Naive Bayes
  - Decision Tree
  - K-nearest Neighbors
  - Support Vector Machines
- Only Random Forest and Naive Bayes were used because they provided the best results. This doesn't mean that other techniques are bad, It just means that more tuning of parameters are required in those learners.
- 100 random trees were generated.
- It was noted that Discretization of features produced better results.