Automatic techniques for identification of cryptographic code

Data Science Institute
COLUMBIA UNIVERSITY

Detecting cryptography in projects
There is a variety of cryptography libraries that implement secure cryptography algorithms such as OpenSSL. However, some open-source projects implement their own cryptography code that can potentially be insecure. Detecting those custom implementations is of paramount importance for using those projects.

Cryptography code uses ciphers to encrypt/decrypt inputs, using keys and transformations on the input (bitwise operations, permutations etc…).

Figure 1. Cryptography for encrypting/decrypting a file

Collecting the dataset
The problem was narrowed down to file-level binary classification for C/C++.
V1: files from crypto competition submissions, crypto libraries (positive examples) and files from algorithmic competitions and randomly selected github repositories (negative examples). V1 has a few important shortcomings, including:

- easily separable classes,
- lack of diversity and coverage

V2: contains OS code, hashing, signal processing, networks, bitwise based operations, ML and heavily math-based non-crypto examples - with an emphasis on keeping ambiguous files in the mix instead of discarding them.

Figure 3. Sample data

Models
Two different approaches to code processing - how to capture signal from code?:

- **Model A** - using hand-crafted features and metadata features with various counts of code elements (loops, bitwise operations, type declarations, imports…)
- **Model B** - Processing the code as a text input by using an embedding to turn the code into a fixed length vector

Benchmark: pattern matching on code as text (using generic terms like “crypt” as well as calls to crypto libraries and mentions of registered crypto algorithms and protocols) - using Wind-River’s crypto-detector¹

Figure 4. Architecture of the embedding-based model

Results
Classification results from our models:

The benchmark already performs well on our dataset, and Model B only improves slightly on it.

Limitations:

- Dataset: the model performance is bound by the dataset quality
- Hard cases: hashing functions that are very similar to cryptography code

Potential improvements:

- Collect files from more varied sources
- Build syntactic trees to capture the semantics of the code before embedding

Reference

¹Wind-river: cryptography detection tool: [https://github.com/Wind-River/crypto-detector](https://github.com/Wind-River/crypto-detector)

Data Science Capstone Project with Amazon