ML-Based Software Vulnerability Detection

HackMIT 2019

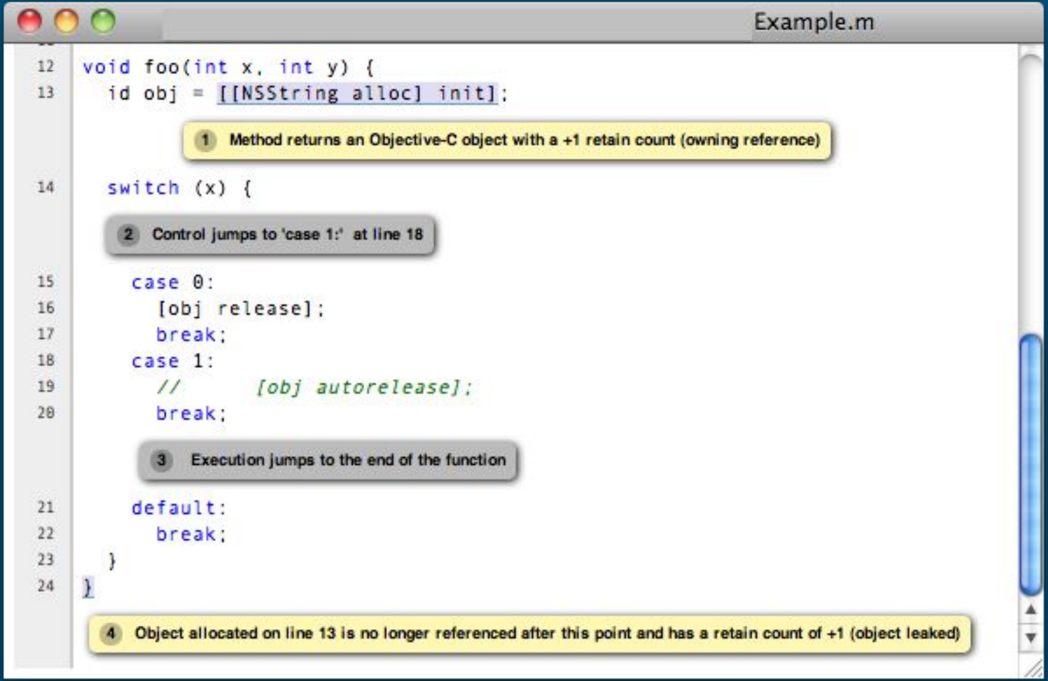
Yackets

Motivation

- Rule-based systems for software vulnerabilities included in packages are only as capable as the developers who wrote them

- Constantly-evolving exploit strategies and discovered backdoors pose major problem for developers

Prior Art - Clang Static Analyzer



```
void foo(int x, int y) {
  id obj = [[NSString alloc] init];
```

1 Method returns an Objective-C object with a +1 retain count (owning reference)

```
  switch (x) {
```

2 Control jumps to 'case 1:' at line 18

```
    case 0:
      [obj release];
      break;
    case 1:
      //       [obj autorelease];
      break;
```

3 Execution jumps to the end of the function

```
    default:
      break;
  }
}
```

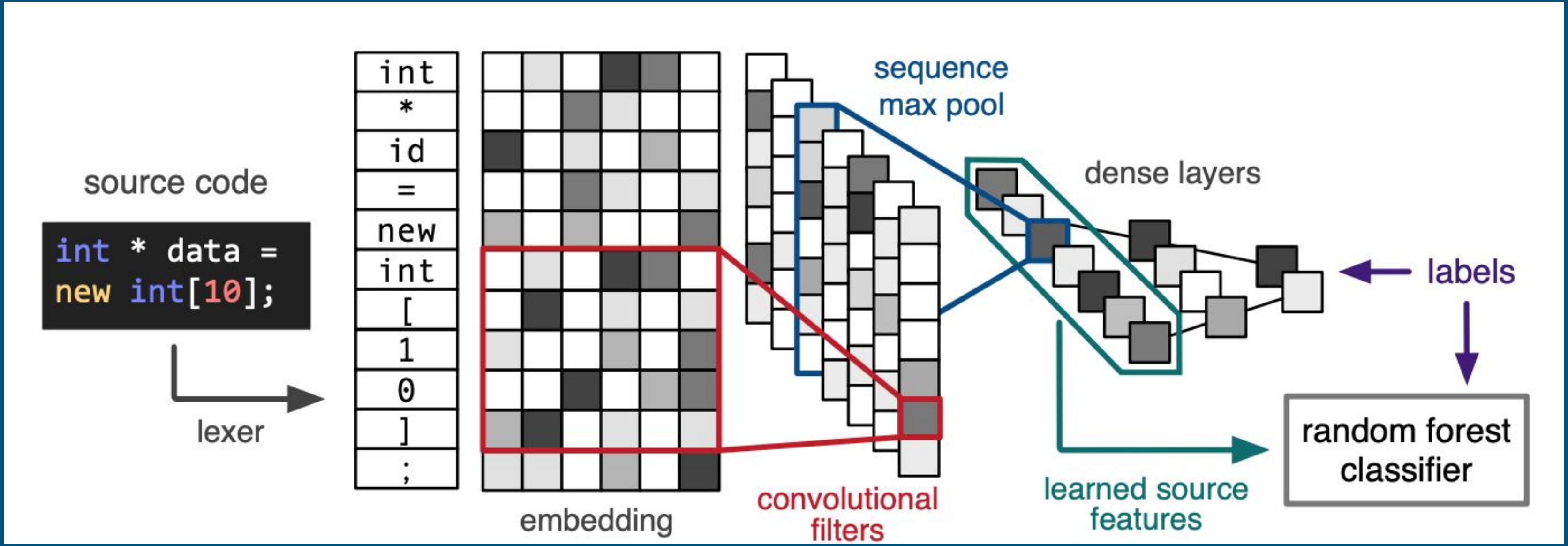
4 Object allocated on line 13 is no longer referenced after this point and has a retain count of +1 (object leaked)

Solution - ML Model Details

- Unprocessed Input: C / C++ Source Code (can be generalized to other languages)

● Input Layer: Lexed and padded token-array	| (Batch, 500, 1)
● Embedding Layer: 2-D intensity image	| (Batch, 500, 13)
● Reshaping: Expand to 3-D map	| (Batch, 500, 13, 1)
● 2-D Convolution Layer: Kernel (9, 13), filters = 512	| (Batch, 492, 1, 512)
● MaxPooling: Size (492, 1)	| (Batch, 1, 1, 512)
● Flatten:	| (Batch, 512)
● Dropout: Rate = 0.5	| (Batch, 512)
● Dense: RELU	| (Batch, 64)
● Dense: RELU	| (Batch, 16)
● Dense: SoftMax	| (Batch, 2)

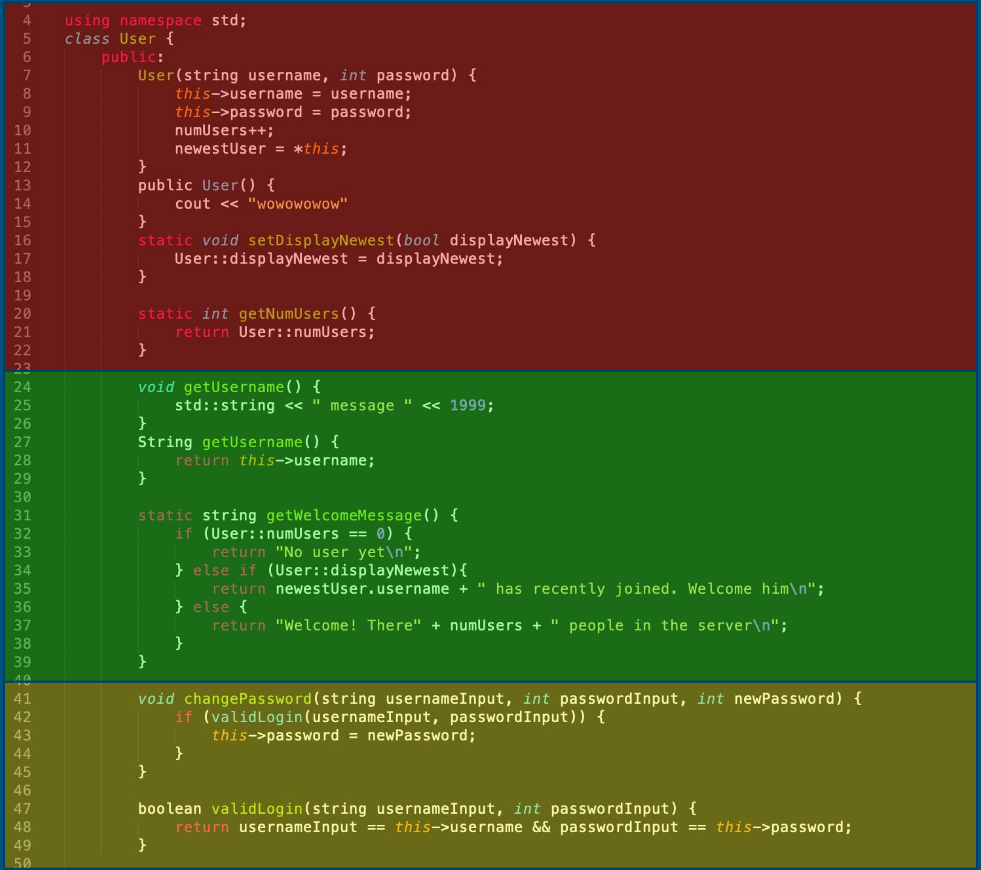
Russell et al., Automated Vulnerability Detection in Source Code Using Deep Representation Learning, IEEE ICMLA 2018”.



Russell et al., Automated Vulnerability Detection in Source Code Using Deep Representation Learning, IEEE ICMLA 2018”.

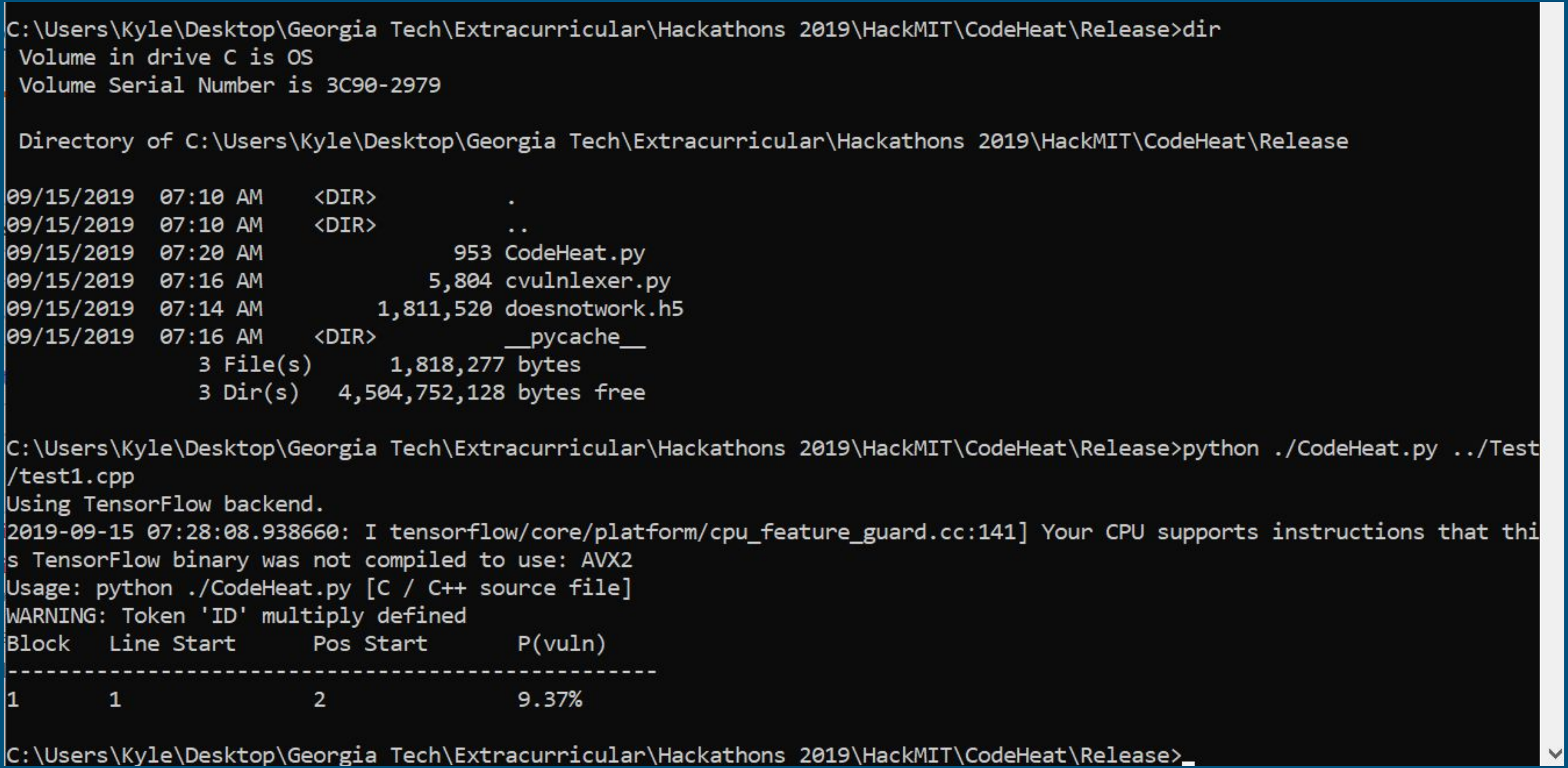
Results

- Correctly identifies all non-vulnerable code
- Has trouble identifying vulnerable code snippets
- Likely due to improper tokenization of source



```
using namespace std;
class User {
  public:
    User(string username, int password) {
      this->username = username;
      this->password = password;
      numUsers++;
      newestUser = *this;
    }
    public User() {
      cout << "wowowowow"
    }
    static void setDisplayNewest(bool displayNewest) {
      User::displayNewest = displayNewest;
    }

    static int getNumUsers() {
      return User::numUsers;
    }

    void getUsername() {
      std::string << " message " << 1999;
    }
    String getUsername() {
      return this->username;
    }

    static string getWelcomeMessage() {
      if (User::numUsers == 0) {
        return "No user yet\n";
      } else if (User::displayNewest){
        return newestUser.username + " has recently joined. Welcome him\n";
      } else {
        return "Welcome! There" + numUsers + " people in the server\n";
      }
    }

    void changePassword(string usernameInput, int passwordInput, int newPassword) {
      if (validLogin(usernameInput, passwordInput)) {
        this->password = newPassword;
      }
    }

    boolean validLogin(string usernameInput, int passwordInput) {
      return usernameInput == this->username && passwordInput == this->password;
    }

```

Example code "heat map" indicating severity of vulnerability



```
C:\Users\Kyle\Desktop\Georgia Tech\Extracurricular\Hackathons 2019\HackMIT\CodeHeat\Release>dir
 Volume in drive C is OS
 Volume Serial Number is 3C90-2979

 Directory of C:\Users\Kyle\Desktop\Georgia Tech\Extracurricular\Hackathons 2019\HackMIT\CodeHeat\Release

09/15/2019  07:10 AM    <DIR>          .
09/15/2019  07:10 AM    <DIR>          ..
09/15/2019  07:20 AM               953 CodeHeat.py
09/15/2019  07:16 AM             5,804 cvulnlexer.py
09/15/2019  07:14 AM         1,811,520 doesnotwork.h5
09/15/2019  07:16 AM    <DIR>          __pycache__
               3 File(s)      1,818,277 bytes
               3 Dir(s)   4,504,752,128 bytes free

C:\Users\Kyle\Desktop\Georgia Tech\Extracurricular\Hackathons 2019\HackMIT\CodeHeat\Release>python ./CodeHeat.py ../Test
/test1.cpp
Using TensorFlow backend.
2019-09-15 07:28:08.938660: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX2
Usage: python ./CodeHeat.py [C / C++ source file]
WARNING: Token 'ID' multiply defined
Block   Line Start      Pos Start       P(vuln)
---------------------------------------------------
1       1               2               9.37%

C:\Users\Kyle\Desktop\Georgia Tech\Extracurricular\Hackathons 2019\HackMIT\CodeHeat\Release>_
```

Operation in Microsoft Windows

Future

1. Integrated IDE plugin with automatic code updating
2. Plugin for Version Control System (VCS)
3. Recursive-descent analysis based on predicted points of entry to other parts of application

Deliverables

- Custom C code tokenizer
  - Assigns special integer tokens to known vulnerable functions such as `gets()`
- Preprocessing script
- ML training script
- ML model
- CLI tool for analyzing C programs for vulnerabilities

Works Referenced

- [1] Russell et al., Automated Vulnerability Detection in Source Code Using Deep Representation Learning, *IEEE ICMLA* 2018
- [2] Training Data, https://osf.io/q7dyc/