CWE Detection Analysis

Mahavir

Common Weakness Enumeration

- List of common software weaknesses
- Unified terminology provides common ground for discussions regarding software security
- Allows software vendors to quantify the security characteristics of their software in a uniform manner

Problem Statement

- Software vulnerabilities exist both in public and proprietary code
- Vulnerabilities can lead to system compromise, risk of exploit, result in system compromise
- Existing program analysis tools detect limited subset of possible errors based on predefined rules
- Automated Vulnerability Detection

Manual CWE analysis:

For manual extraction of CWE vulnerabilities we used:

1. CppCheck
2. Flawfinder

Result from flawfinder

```
Flawfinder version 2.0.11, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining CWE-20/src/test1.c
Examining CWE-20/src/test2.c

FINAL RESULTS:


ANALYSIS SUMMARY:

No hits found.
Lines analyzed = 58 in approximately 0.04 seconds (1512 lines/second)
Physical Source Lines of Code (SLOC) = 39
Hits@level = [0]  10 [1]   0 [2]   0 [3]   0 [4]   0 [5]   0
Hits@level+ = [0+]  10 [1+]   0 [2+]   0 [3+]   0 [4+]   0 [5+]   0
Hits/KSLOC@level+ = [0+] 256.41 [1+]   0 [2+]   0 [3+]   0 [4+]   0 [5+]   0
Minimum risk level = 1
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
(torch1py3.5) ml@ml:~/projects/cwe_checker/C-C-_CWE/Ctests$
```

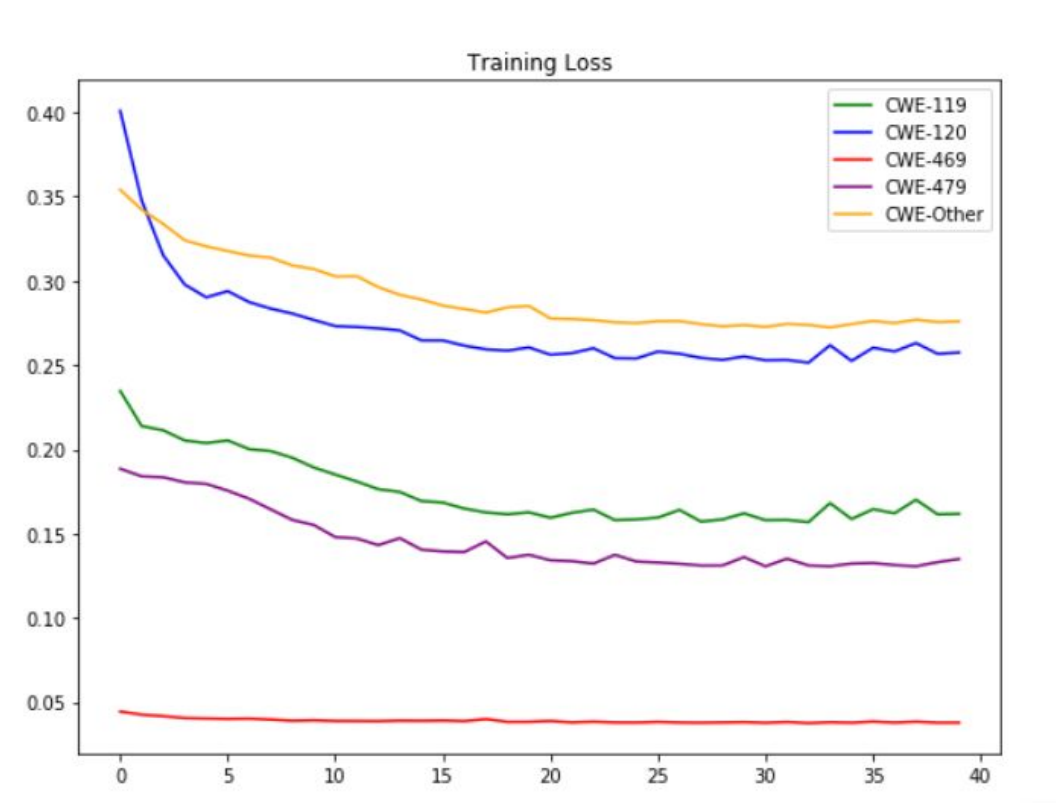
Result from flawfinder

```
(torch1py3.5) ml@ml:~/projects/cwe_checker/C-C-_CWE/Ctests$ flawfinder CWE-401
Flawfinder version 2.0.11, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining CWE-401/src/test1.c

FINAL RESULTS:

CWE-401/src/test1.c:10:  [1] (buffer) read:
  Check buffer boundaries if used in a loop including recursive loops
  (CWE-120, CWE-20).

ANALYSIS SUMMARY:

Hits = 1
Lines analyzed = 20 in approximately 0.00 seconds (4636 lines/second)
Physical Source Lines of Code (SLOC) = 18
Hits@level = [0]   0 [1]   1 [2]   0 [3]   0 [4]   0 [5]   0
Hits@level+ = [0+]   1 [1+]   1 [2+]   0 [3+]   0 [4+]   0 [5+]   0
Hits/KSLOC@level+ = [0+] 55.5556 [1+] 55.5556 [2+]   0 [3+]   0 [4+]   0 [5+]   0
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
```

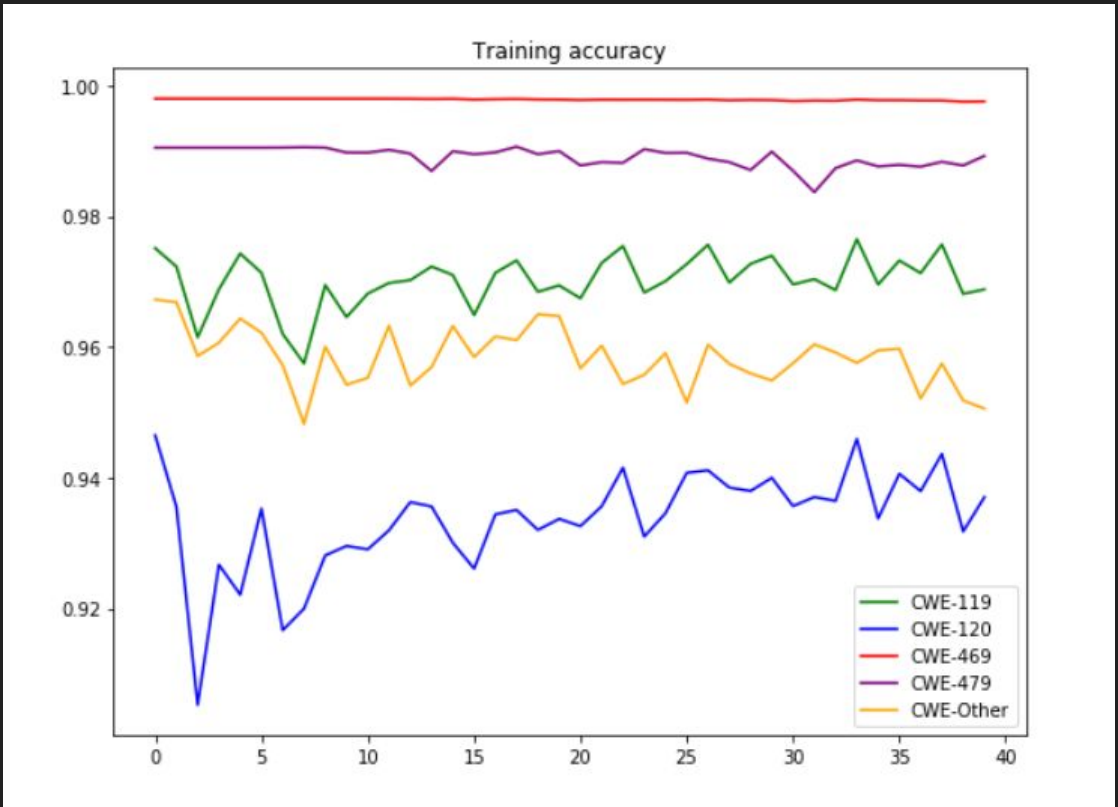
Results from cppcheck

```
(torch1py3.5) ml@ml:~/projects/cwe_checker/C-C-_CWE/Ctests$ cppcheck CWE-401/src/test1.c
Checking CWE-401/src/test1.c...
[CWE-401/src/test1.c:11]: (error) Memory leak: buf
```

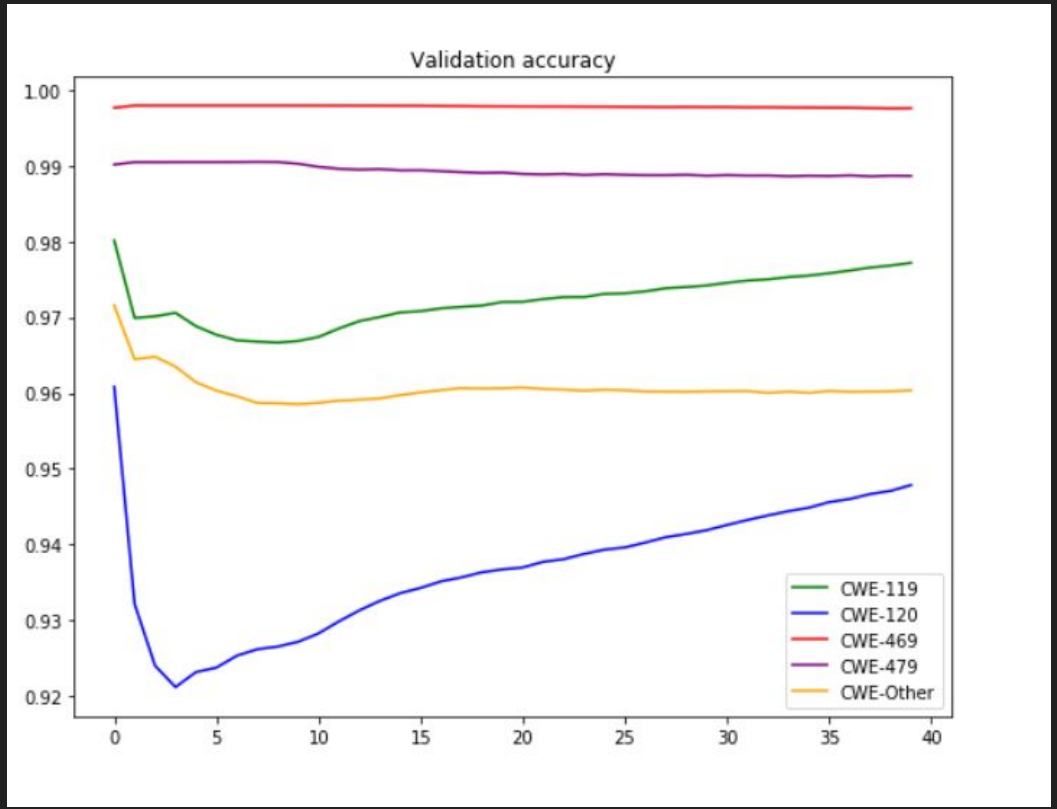
Loss on Training on Dataset



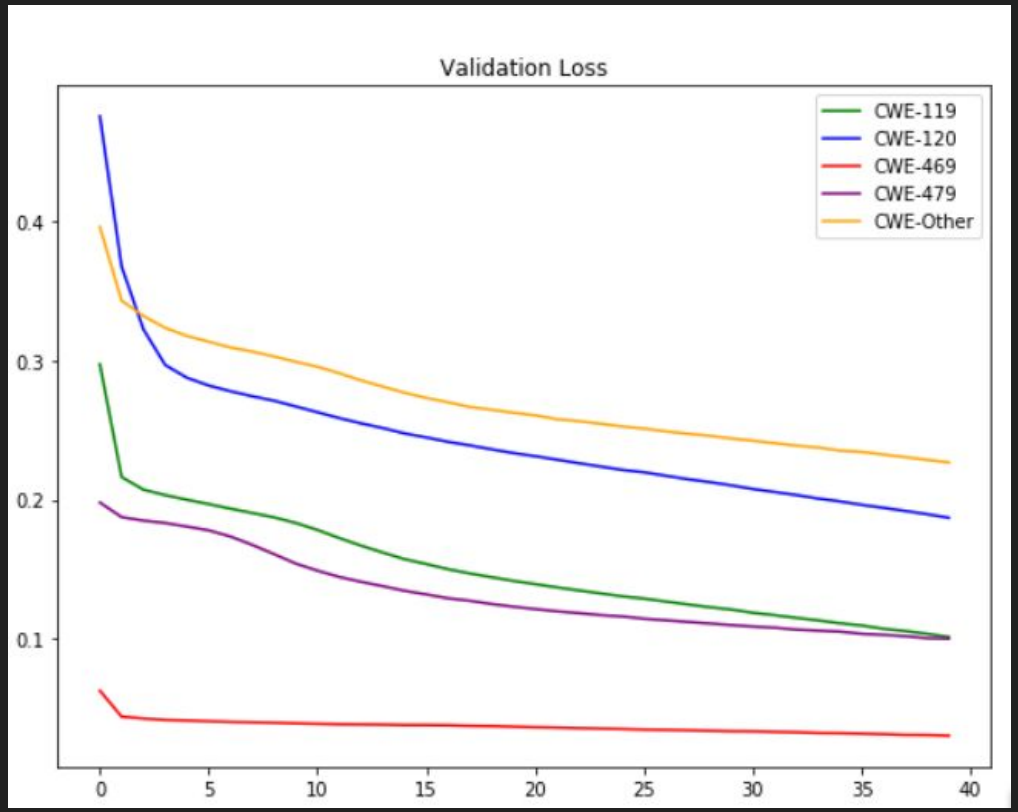
Accuracy on Training Dataset



Accuracy on Validation Dataset



Validation Accuracy



Binary Classification