# AVANT: An Automated Vulnerability Analysis Tool for Assessing Security Vulnerabilities in Processor Architectures

Strahinja Trecakov*, Satyajayant Misra*, Abdel-Hameed Badawy*, and Jaime Acosta+
*New Mexico State University
+U.S. Army Research Laboratory, White Sands Missile Range, NM
{trecakov, misra, badawy}@nmsu.edu, jaime.c.acosta.civ@mail.mil

## I. INTRODUCTION

Vulnerabilities can be present and triggered by any layer of a computer system (Figure 1). However, each vulnerability has a certain risk level depending on what can be compromised.

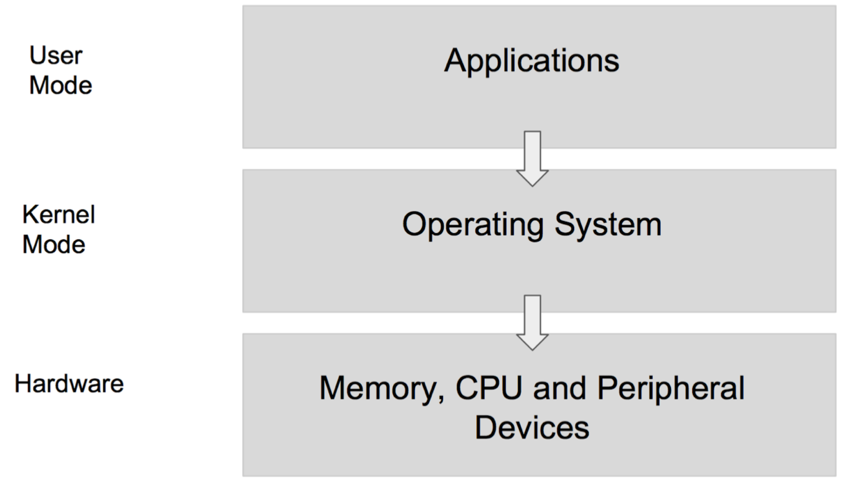
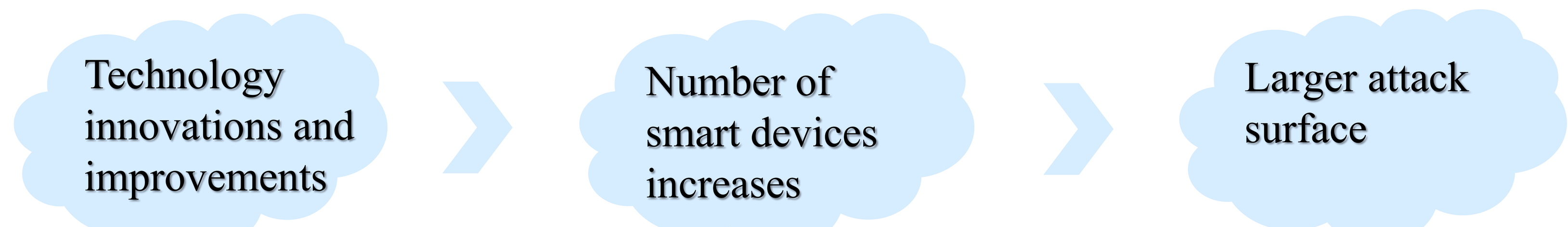



Figure 1: Typical representation of layers in computer system.

We present Automated Vulnerability Analysis Tool (AVANT), a vulnerability checking tool that is architecture-agnostic and reports vulnerabilities found for binaries in our test suite.

## II. MOTIVATION




Below is an example of a buffer overflow vulnerability:

```
char pass[] = "abcd";
int validate_user() {
  char buff[5];
  printf("Enter your password:\n -> ");
  gets(buff);
  return !strcmp(buff, pass);
}
```

Figure 2: CWE-120 Buffer Overflow based example.

```
0x7fffffffd9b0:0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00
0x7fffffffd9b8:0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00
0x7fffffffd9c0:0xe0 0xd9 0xff 0xff 0xff 0x7f
    0x00 0x00
0x7fffffffd9c8:0xb7 0x06 0x40 0x00 0x00 0x00
    0x00 0x00
```

Figure 3: Representation of the stack pointer before inputted password.

```
0x7fffffffd9b0:0x41 0x41 0x41 0x41 0x41 0x41
    0x00 0x00
0x7fffffffd9b8:0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00
0x7fffffffd9c0:0xe0 0xd9 0xff 0xff 0xff 0x7f
    0x00 0x00
0x7fffffffd9c8:0xb7 0x06 0x40 0x00 0x00 0x00
    0x00 0x00
```

Figure 4: Representation of the stack pointer after inputted ”AAAAAA” as a password.

In this buffer overflow the part of the input that does not fit in the buffer keeps being written in adjacent memory address.

Meltdown and Spectre attacks showed us that a bug in Intel chips allows access to higher parts of computer's memory.

## III. DESIGN MODEL

CWE 121 – Stack-based Buffer Overflow
CWE 122 – Heap-based Buffer Overflow
CWE 124 – Buffer Underwrite
CWE 126 – Buffer Over-read
CWE 127 – Buffer Under-read
CWE 190 – Integer Overflow
CWE 191 – Integer Underflow

Test cases: From NSA's Juliet Test Suite.

Modified and tested on Windows 7, Ubuntu 16.04, High Sierra 10.13.2.

AVANT

A simple vulnerability tool: Compiles vulnerability test suite, runs the analysis using Address Sanitizer tool and reports found vulnerabilities.

## IV. EXPERIMENTAL SETUP

**100+ Test cases**

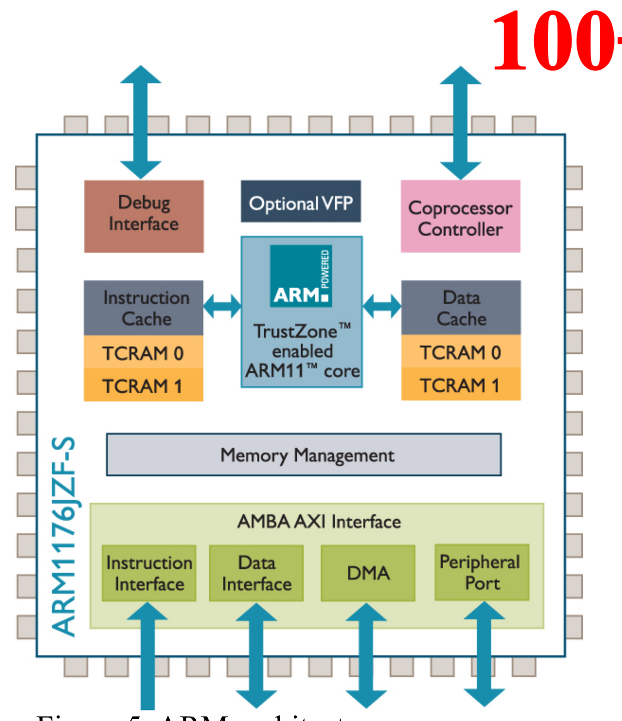


Figure 5: ARM architecture.

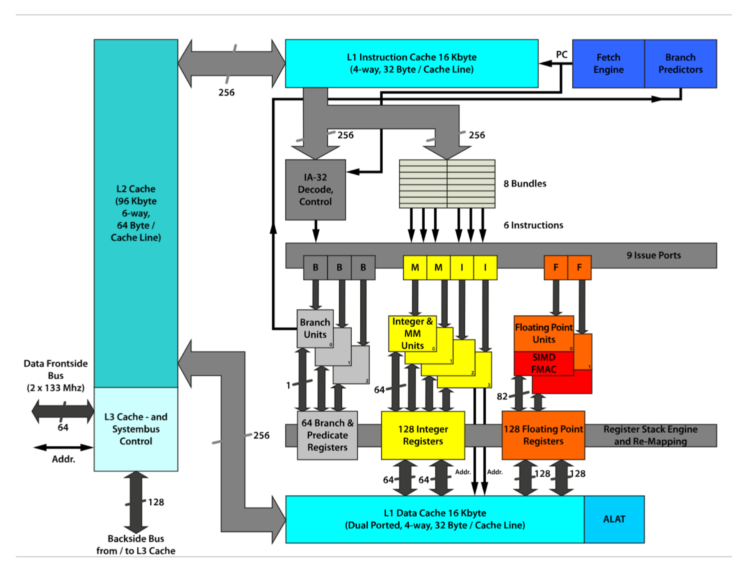


Figure 6: Intel x86_64 architecture.

| | ARM Architecture | Intel x86_64 |
|---|---|---|
| Architecture | armv71 | x86_64 |
| Byte Order: | Little Endian | Little Endian |
| CPU(s): | 4 | 8 |
| On-line CPU(s) list: | 0-3 | 0-7 |
| Thread(s) per core: | 1 | 2 |
| Core(s) per socket: | 4 | 4 |
| Socket(s): | 1 | 1 |
| Model: | 4 | 94 |
| Model name: | ARMv7 Processor rev 4 (v71) | Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz |
| CPU max MHz: | 1200.0000 | 4200.0000 |
| CPU min MHz: | 600.0000 | 800.0000 |
| BogoMIPS: | 38.40 | 8015.91 |

## V. PRELIMINARY RESULTS

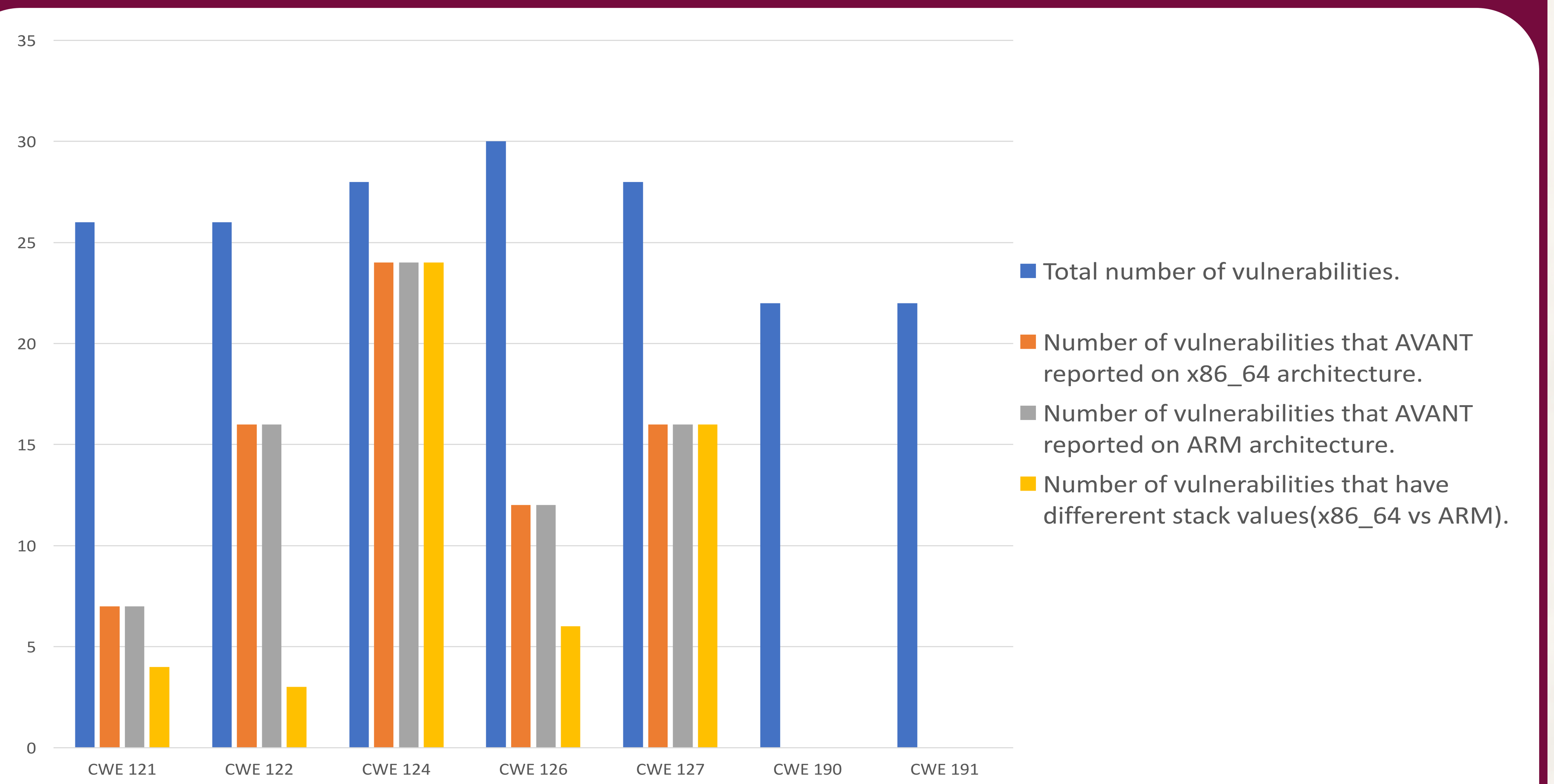



Figure 6: Representation of the total number of test cases ran using AVANT on both x86_64 and ARM architectures and the number of reported vulnerabilities.

```
SUMMARY: AddressSanitizer: stack-buffer-underflow
(/home/strecako/Documents/OSFA-
Benchmarks/testcases/CWE124_Buffer_Underwrite/s01/CWE124_Buffer_Unde
rwrite__char_declare_memcpy_01.out+0x4a2b44)
Shadow bytes around the buggy address:
  0x10002e003f50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002e003f60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002e003f70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002e003f80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002e003f90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x10002e003fa0: 00 00 00 00 f1 f1  f1 [f1]00 00 00 00 00 00 00 00
  0x10002e003fb0: 00 00 00 00 04 f3 f3  f3  f3  f3  f3  f3 00 00 00 00
  0x10002e003fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002e003fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002e003fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002e003ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 7: Stack representation of the CWE124 Buffer Underwrite test case ran on x86_64.

```
SUMMARY: AddressSanitizer: stack-buffer-underflow (/home/pi/OSFA-
compiled/OSFA-
Benchmarks/testcases/CWE124_Buffer_Underwrite/s01/CWE124_Buffer_
Underwrite__char_declare_memcpy_01.out+0xab6db) in __asan_memset
Shadow bytes around the buggy address:
  0x2fd21e40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x2fd21e50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x2fd21e60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x2fd21e70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x2fd21e80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x2fd21e90: 00 00 00 00 00 00 00 00 00 00 00 00 00 f1[f1]00 00
  0x2fd21ea0: 00 00 00 00 00 00 00 00 00 00 04  f3 f3  f3  f3 f3
  0x2fd21eb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x2fd21ec0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x2fd21ed0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x2fd21ee0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 8: Stack representation of the CWE124 Buffer Underwrite test case ran on Raspberry Pi 3.

## VI. CONCLUSION & FUTURE WORK

- These results show us that both Intel x86_64 and ARM architectures report same vulnerabilities; this is good from the security standpoint. The interesting part is that same reported vulnerabilities on two architectures differ in their stack values/addresses which can lead to some other attacks on the security of those architectures.
- The research done in this project will be extended to look more deeply into this problem by expanding the test set in order to identify more vulnerabilities. We are developing a test suite that includes binaries with annotated and categorized vulnerabilities.

## VII. REFERENCE

[1] Trecakov, S., Tran, C., Badawy, H., Siddique, N., Acosta, J., & Misra, S. (2017, October). Can Architecture Design Help Eliminate Some Common Vulnerabilities?. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)* (pp.590-593). IEEE.