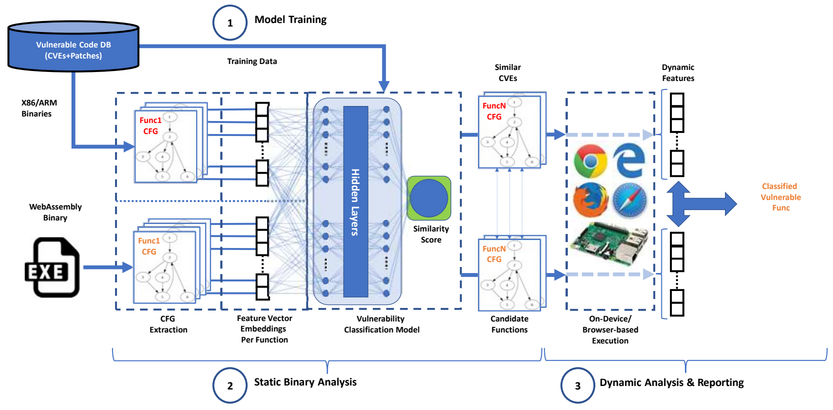
# A framework to automatically detect **vulnerabilities** in **WASM** binaries based on know vulnerabilities by combining deep learning-based **static** binary analysis with **dynamic** binary analysis

## Known Vulnerability Detection for WebAssembly Binaries

Pengfei Sun, Luis Garcia, Yi Han, Saman Zonouz, Yao Zhao
**F5 Networks, USC ISI and Rutgers University**

### WASP Overview



## Main Idea

- Deep learning is used to train the vulnerability detector;
- The vulnerability detector is used to statically analyze the target WASM binary;
  - ➢ WASP leverages a trained multilayer perceptron (MLP) followed by the cross-entropy between the softmax outputs to determine if the two are similar.
- The identified vulnerable subroutines are run for in-depth dynamic analysis and verification of the existence of a vulnerability.
  - ➢ WASPleverages IDA Pro andWasabi to run the CVE vulnerable function binary as wellas the target Wasm function binary on identical input values

## Problem Setting and Challenges

- CVE-2018-14550
  - ➢ get_token() in libpng
- The belong figure highlights the syntactic differences between the x86 binary assembly code representation and WASM assembly code representation.

```
020B2:  push  rbp
020B3:  mov   rbp, rsp
020B6:  sub   rsp, 20h
020BA:  mov   [rbp-18h], rdi
020BE:  mov   [rbp-20h], rsi
020C2:  mov   dword ptr [rbp-8], 0
020C9:  mov   rax, [rbp-18h]
020CD:  mov   rdi, rax        ; stream
020D0:  call  _fgetc
020D5:  mov   [rbp-4], eax
020D8:  cmp   dword ptr [rbp-4], 23h
020DC:  jnz   short loc_20FF
020DE:  mov   rax, [rbp-18h]
020E2:  mov   rdi, rax        ; stream
020E5:  call  _fgetc
020EA:  mov   [rbp-4], eax
020ED:  cmp   dword ptr [rbp-4], 0Ah
020F1:  jz    short loc_20FF
020F3:  cmp   dword ptr [rbp-4], 0Dh
020F7:  jz    short loc_20FF
020F9:  cmp   dword ptr [rbp-4], 0FFFFFFFFh
020FD:  jnz   short loc_20DE
020FF:  cmp   dword ptr [rbp-4], 0FFFFFFFFh
02103:  jz    short loc_2159
02105:  mov   eax, [rbp-8]
…
```

x86 binary

```
002369:  local[0..1] type=i32
00236b:  loop
00236d:    block
00236f:     local.get 0
002371:      call 355
002374:      local.tee 2
002376:      i32.const 35
002378:      i32.ne
002379:      br_if 0
00237b:      loop
00237d:        block
00237f:         local.get 0
002381:          call 355
002384:          local.tee 2
002386:          i32.const 4294967286
002388:          i32.add
002389:          br_table 2 1 1 2 0
002390:        end
002391:        local.get 2
002393:        i32.const 4294967295
002395:        i32.ne
002396:        br_if 0
002398:      end
002399:    end
00239a:    local.get 2
…
```

wasm binary

## Preliminary Result and Future Work

- Six different CVEs have been evaluated. WASP can identify the correct matches among the top 3 ranked outcomes 100% of the time
- Evaluating WASP on the aforementioned large dataset of real-world WASM binaries to characterize vulnerabilities in the wild.

Take a picture to download our previous work