NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

School of Computer Science and Engineering
College of Engineering

Pre-Fuzzing Analysis of Embedded System binaries with Ghidra SRE

Student: Ron Ng Jian Ying
Supervisor: Professor Liu Yang

PROBLEM

Only two solutions currently exist in the market for coverage guided fuzzing of embedded systems – inline instrumentation which requires source codes or fuzzing within a virtual machine a la QEMU.

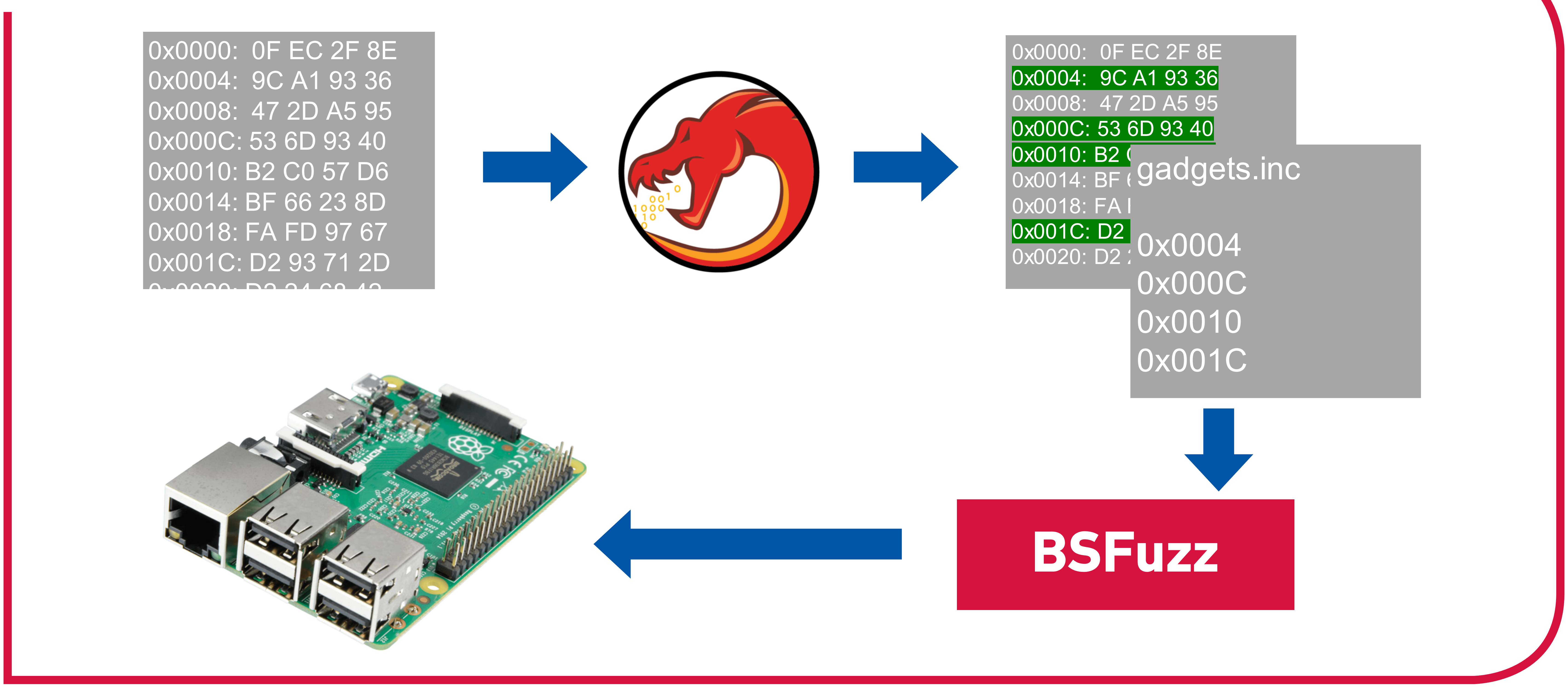
Fuzzing in a virtual machine is slow and inline instrumentation without source code is complex and inefficient.

MOTIVATION

Detour/Hooking instrumentation can be used instead. Suitable portions of instructions in code can be identified as relocatable.

This allows instructions to be copied into a memory segment with instrumentation code added. Original instruction flow remains undisturbed.

Workflow



```
0x0000:  0F EC 2F 8E
0x0004:  9C A1 93 36
0x0008:  47 2D A5 95
0x000C: 53 6D 93 40
0x0010: B2 C0 57 D6
0x0014: BF 66 23 8D
0x0018: FA FD 97 67
0x001C: D2 93 71 2D
```

```
0x0000:  0F EC 2F 8E
0x0004:  9C A1 93 36
0x0008:  47 2D A5 95
0x000C: 53 6D 93 40
0x0010: B2 C
0x0014: BF 6
0x0018: FA F
0x001C: D2
0x0020: D2 3
```

```
gadgets.inc

0x0004
0x000C
0x0010
0x001C
```

SOLUTION

Ghidra was used to analyse binaries and its API used to create a Python script that retrieves a list of compatible address locations for hooking. These are known as gadgets.

The gadget list is then passed on to BSFuzz which is a coverage-guided embedded fuzzer developed by NTU CSL. BSFuzz uses the list of gadget addresses to know which function to hook at runtime.