Memory Vulnerabilities in Binary Code

PhD Student: Graham Peden; Supervisor: Prof Awais Rashid: Sponsor: Toshiba UK

Bristol Cyber Security Group, Woodland Road, Merchant Venturers Building, Bristol BS8 1UB, UK

University of BRISTOL
Bristol Cyber Security Group

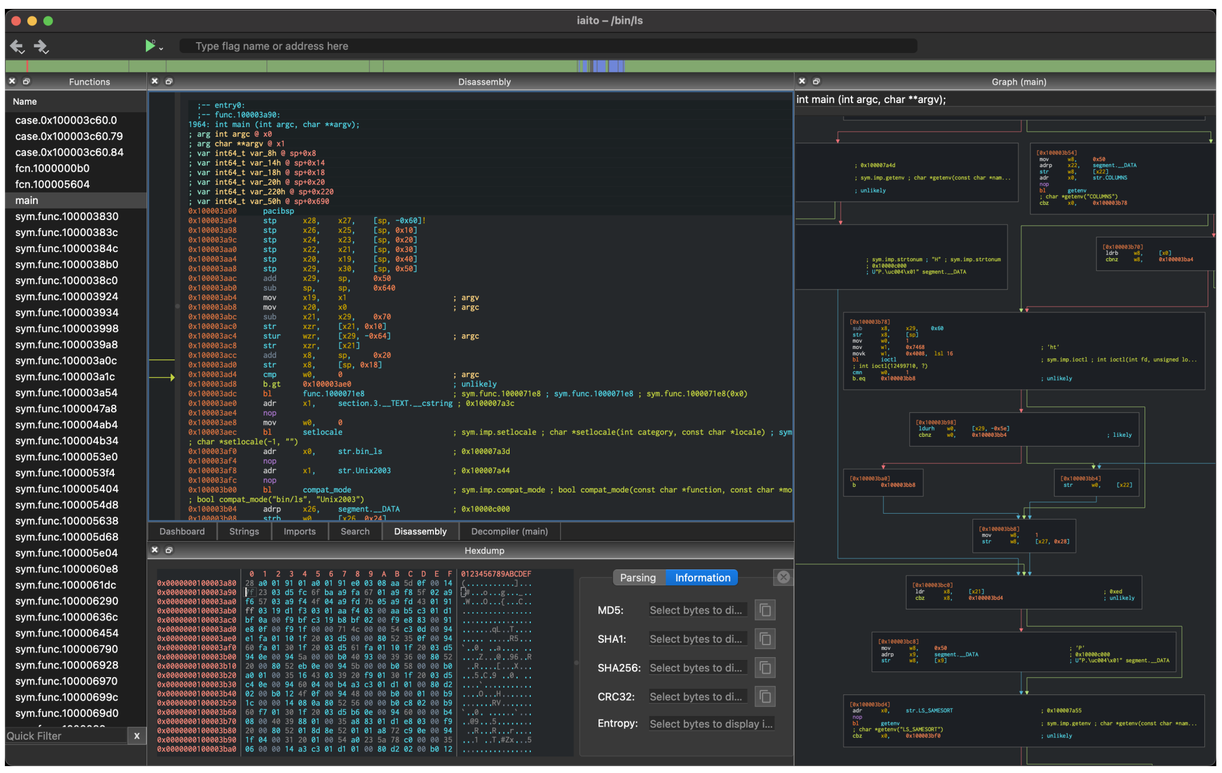
The Problem

- Modern languages and techniques help to avoid many coding errors associated with the C language.
- Firmware written in C is found in legacy systems and is still used in developing embedded code for IoT devices and industrial control systems.
- Bugs can easily occur when allocating and using memory resources, and these are frequently difficult to find!
- Typical memory bugs include writing beyond the end of a buffer's allocated size, or using memory after it is freed.

```
40      if(STATIC_CONST_TRUE)
41      {
42          /* FLAW: Set a pointer to a buffer that does not
43           * string copies in the sinks  */
44          data = dataBadBuffer;
45          data[0] = L'\0'; /* null terminate */
46      }
47      {
48          wchar_t source[10+1] = SRC_STRING;
49          size_t i, sourceLen;
50          sourceLen = wcslen(source);
51          /* Copy length + 1 to include NUL terminator from
52          /* POTENTIAL FLAW: data may not have enough space
53          for (i = 0; i < sourceLen + 1; i++)
54          {
55              data[i] = source[i];
```

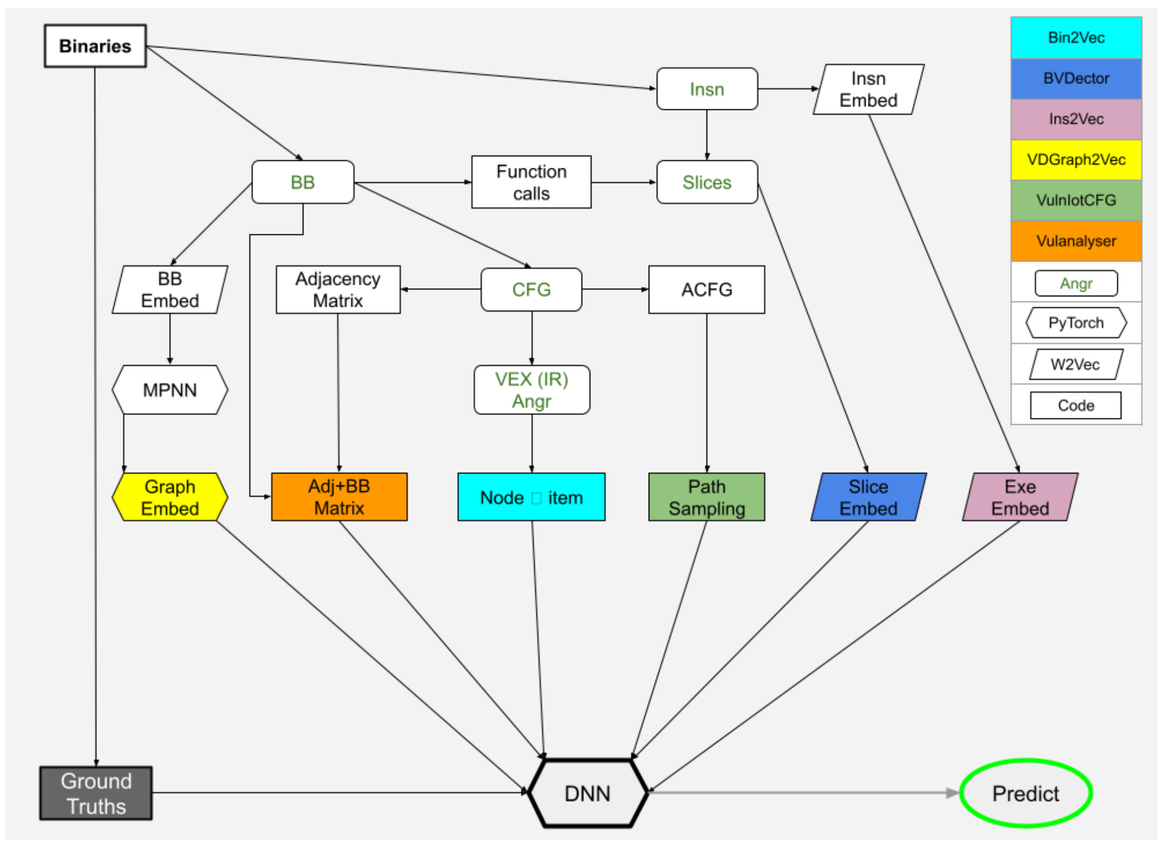
Figure 1: C is still widely used in firmware for IoT and other devices

Existing Approaches



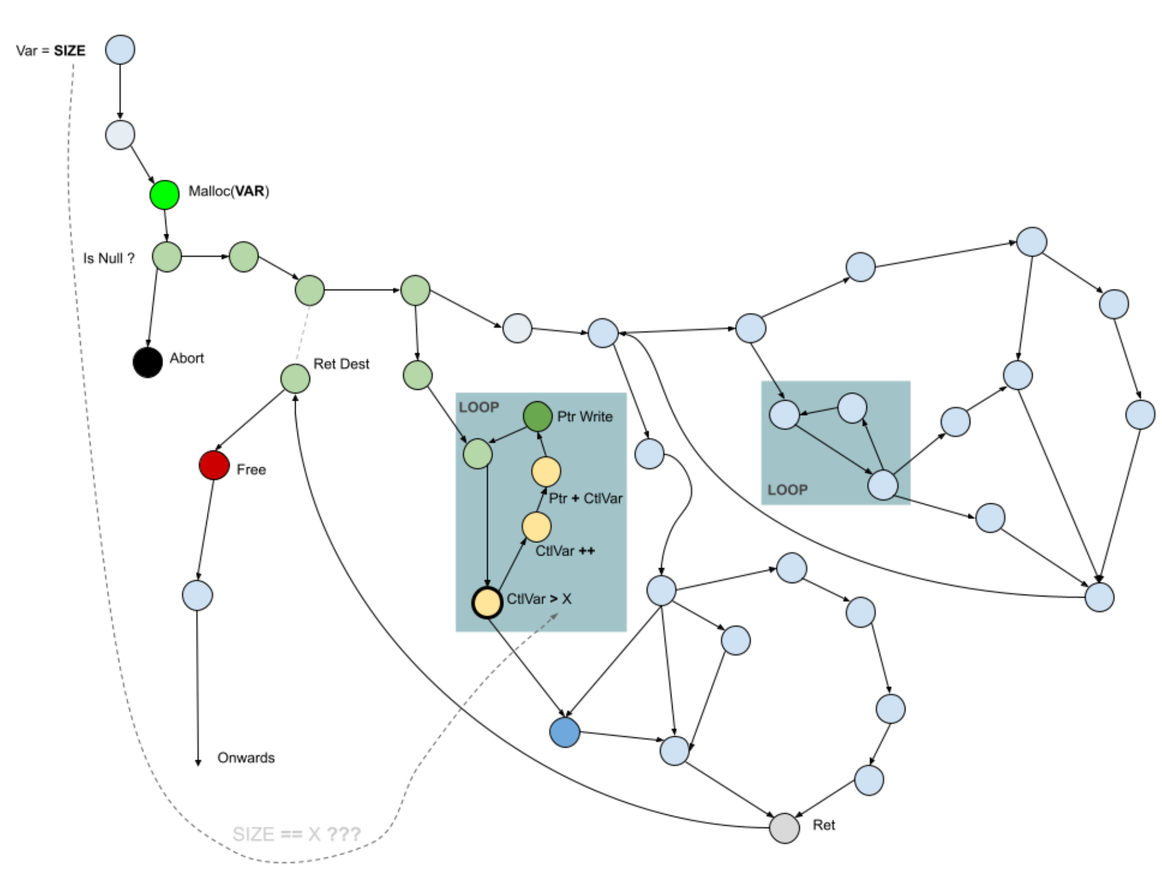
- Often an expert undertakes a detailed study of a program, but automated analysis at scale is desirable.
- Much work has been done analysing the source code of programs, but less for binary code.
- Static analysis scrutinises code without running the program; dynamic analysis executes code to some degree.
- Machine learning techniques have been applied, often splitting programs at jump statements to create basic blocks. These can be transformed into features within a training dataset, with the location of known bugs marked.
- Neural networks and other ML algorithms can predict the presence of bugs in new binary code.

Same Needle; Larger Haystacks



- Reproduce 6 papers: All use Juliet Data Set
- Different feature representations and models
- Embed Juliet functions inside larger, generated binaries
- Increase binary complexity and size with dummy code
- Score each model on original vs new datasets
- **Does accuracy decrease and if so, in what way?**

Call Flow Graphs ==> Data Mining?



- Program Analysis techniques can track data usage back to origin
- Features such as memory allocation, loops and pointer arithmetic are detectable
- Collected features can be used to learn patterns indicating pathologies