



Supporting Secure Code for IoT Devices in Mainstream Compiler Tool Chains

IoTsf Conference

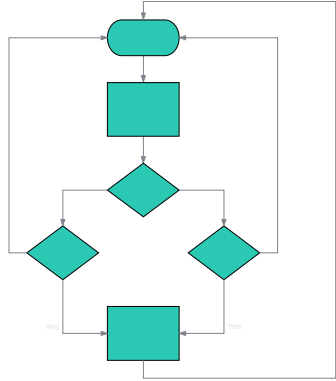
4 December 2018

Jeremy Bennett
Graham Markall
Simon Cook
Paolo Savini
Craig Blackmore
Sam Leonard



Copyright © 2018 Embecosm.
Freely available under a Creative Commons license.

Developing Secure IoT Software



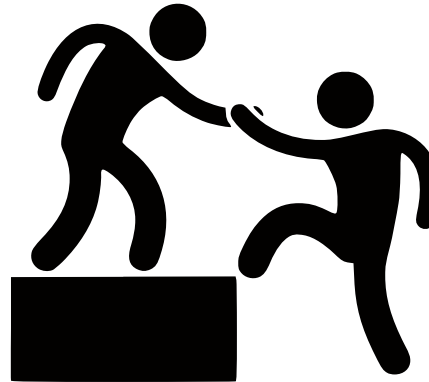
Good software
engineering
practices



Engineering teams
must follow process

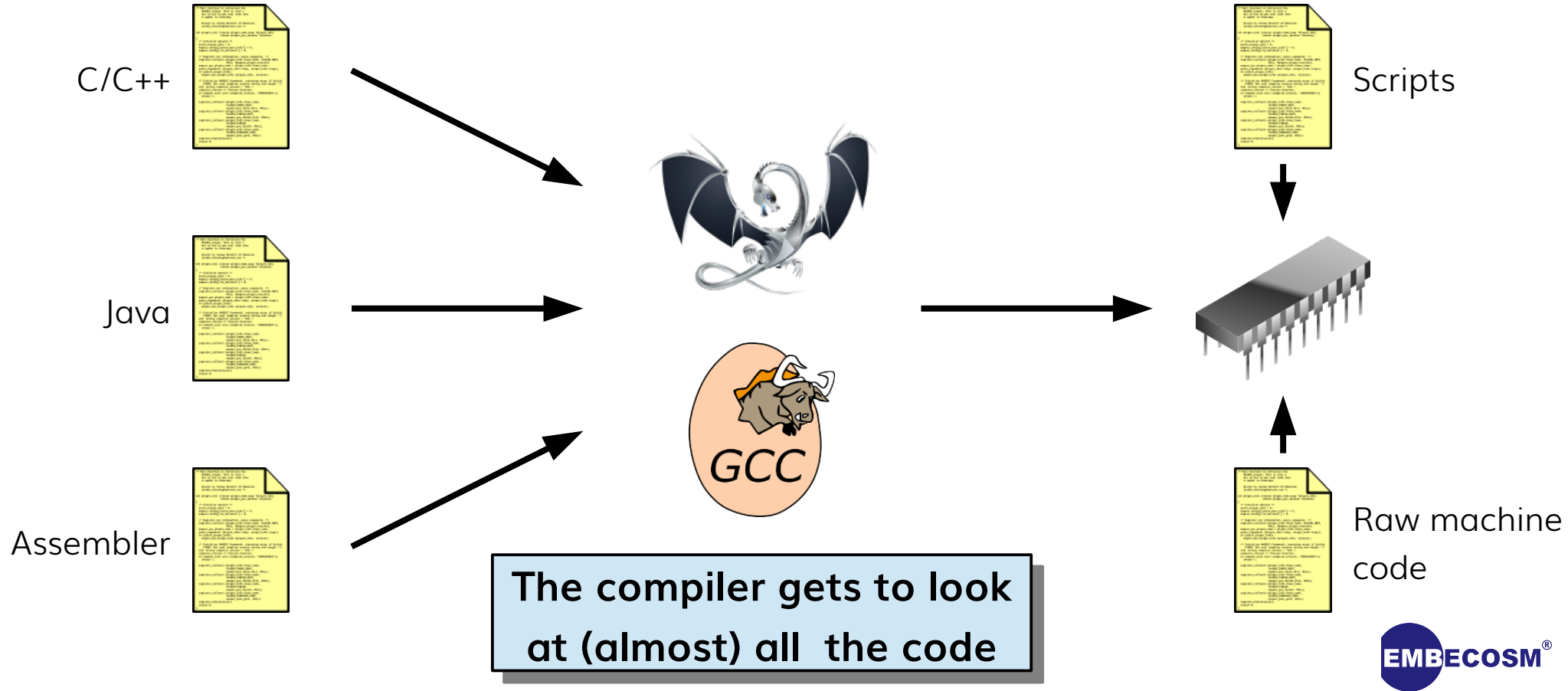


Coding
guidelines



Need to facilitate teams
in following process

Why the Compiler?



How the Compiler Can Help

Warning of bad practice



Advising the programmer when code appears to follow bad practice

Providing heavy lifting



Automating complex tasks to make them easier for the programmer

Innovate UK



Academic / Industrial Context

- LADA: Leakage Aware Design Automation
 - Prof Elisabeth Oswald
 - Dr Dan Page
- Customers' Secure Processors
 - Using LLVM to Guarantee Program Integrity



University of
BRISTOL

The slide is titled "Encoding a Function" and features the EMBECOSM logo in the top left. It displays LLVM assembly code for a function named "foo". The code includes instructions like "lsl", "andi", "add", and "jmp" with register and immediate values. To the right of the assembly code, there are memory addresses in hexadecimal. A video inset on the right shows a man, Simon Cook, speaking at a podium. Below the video, the text "Using LLVM to guarantee program integrity" is displayed, along with the LLVM.org logo.

```
int foo(int x, int y) { return (4*x) + (y&5); }
```

Instruction	Registers/Values	Address
lsl	\$r10, \$r2, 2	019a 4000
andi	\$r13, \$r3, 5	5d87 4002
add	\$r2, \$r13, \$r10	aa82 0900
jmp	\$r0	0050

Instruction	Registers/Values	Address
lsl	$e(i_1, S_0) \rightarrow E_1$	0001 0203
andi	$e(i_2, S_1) \rightarrow E_2$	0405 0607
add	$e(i_3, S_2) \rightarrow E_3$	0809 0a0b
jmp	$e(i_4, S_3) \rightarrow E_4$	0c0d

SIMON COOK

Using LLVM to guarantee program integrity

LLVM.org

Techniques

- **Stack / Register Erase:** Protecting secrets
- **Bit-slicing:** Side-channel resistance
- **Control Flow Balancing:** Side-channel resistance
- **Sensitive Control Flow:** Side-channel warnings
- **Bit-splitting:** Hardware snooping attack resistance
- **Defensive stores:** Glitch attack resistance

Example: CERT MEM03-C.

"Clear sensitive information stored in reusable resources"

Today's Techniques

- `explicit_bzero` (BSD, Glibc)
- `memset_s` (C11 standard)
- `SecureZeroMemory` (MS Windows)
- Finalizers, Limited Private types (Ada)

Today's Problems

- **Ephemeral storage**
 - (e.g. stack) only Ada addresses this
- **Explicit application**
 - must hand apply to all relevant variables

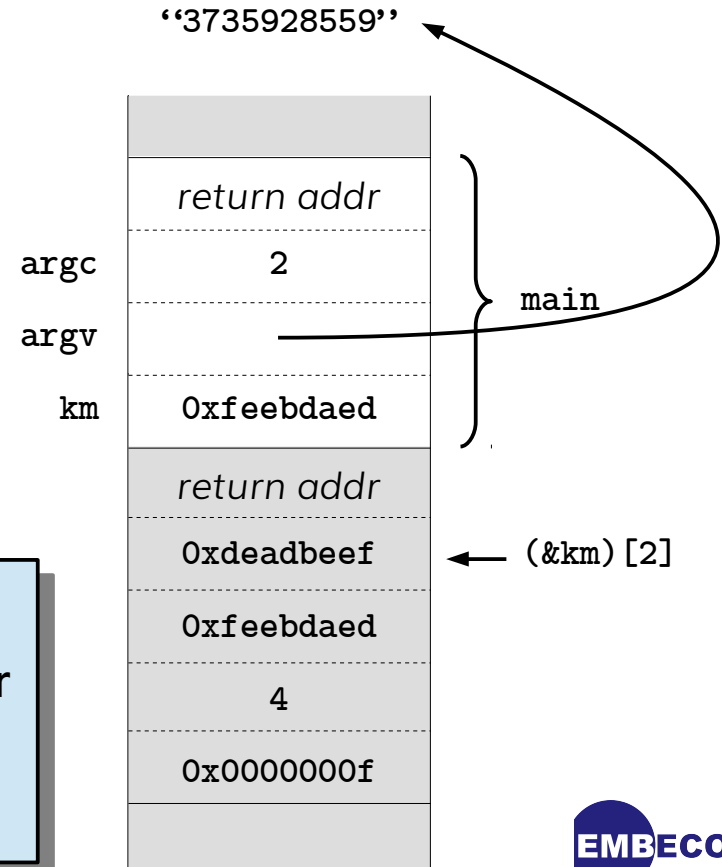
Problem: Critical Data on the Stack

```
int mangle (uint32_t k)
{
    uint32_t res = 0;
    int i;

    for (i = 0; i < 8; i++)
    {
        uint32_t b = k >> (i * 4) & 0xf;
        res |= b << ((7 - i) * 4);
    }
    return res;
}

int main (int argc,
          char *argv[])
{
    uint32_t km;
    km = mangle (atoi (argv[1]));
    return (&km) [2];
}
```

No mechanism
for programmer
to access and
clear the stack

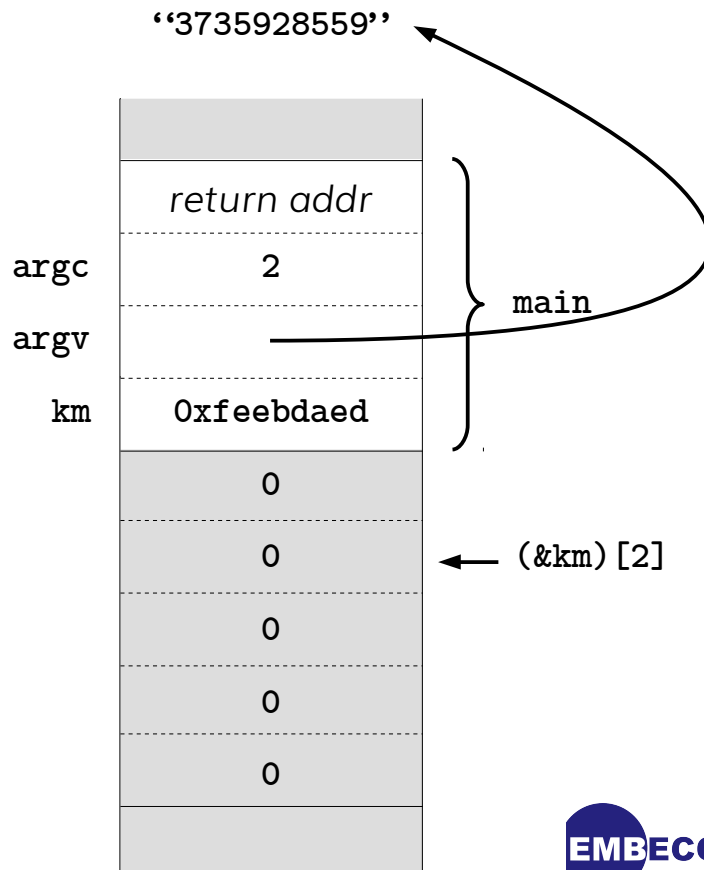


Solution: erase_stack Attribute

```
int mangle (uint32_t k)
__attribute__((erase_stack))
{
    uint32_t res = 0;
    int i;

    for (i = 0; i < 8; i++)
    {
        uint32_t b = k >> (i * 4) & 0xf;
        res |= b << ((7 - i) * 4);
    }
    return res;
}

int main (int argc,
          char *argv[])
{
    uint32_t km;
    km = mangle (atoi (argv[1]));
    return (&km)[2];
}
```



Demo: Exploit Defeated By erase_stack

<https://www.youtube.com/watch?v=B-ZGXn4urj4>


Takeaways

- Developing secure IoT software requires good practice
- Good practice is supported by good tooling
- Example in this case: compiler-assisted security techniques

Takeaways



Developing
secure IoT
software
requires good
practice



The compiler
tool chain is a
great place
for security
tooling



Good practice
is supported
by good
tooling



Thank You

www.embecosm.com

**Jeremy Bennett
Graham Markall
Simon Cook
Paolo Savini
Craig Blackmore
Sam Leonard**



Copyright © 2018 Embecosm.
Freely available under a Creative Commons license.