ARMed Combat: The fight for personal security

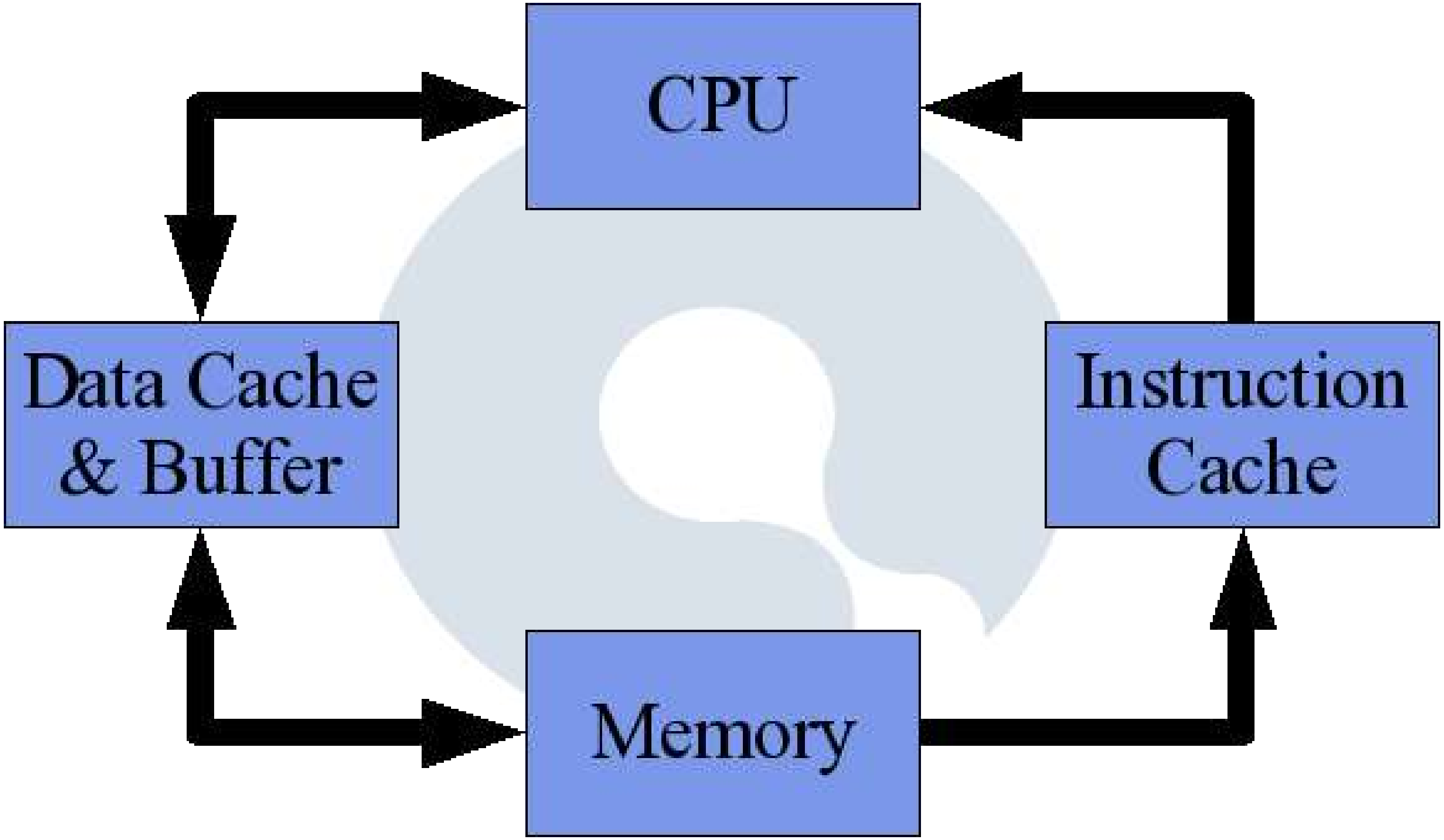
Tim Hurman <tim.hurman@pentest.co.uk>

ARM is…

- RISC based processor
  - Harvard architecture
- 32 bit based instruction set
  - Switchable to Thumb mode (16 bit)
- Separate process privilege levels
- Low power
- VERY common

That Harvard thang

- Separate instruction and data buses
- Unsynchronised caches/buffers
- All self modifying code is crippled
- Makes exploits really… painful
  - Return to libc exploits easiest
- Need to ensure caches are in-sync
  - Requires privileged instruction access
  - Trigger a kernel function to sync


CPU
Data Cache & Buffer
Instruction Cache
Memory

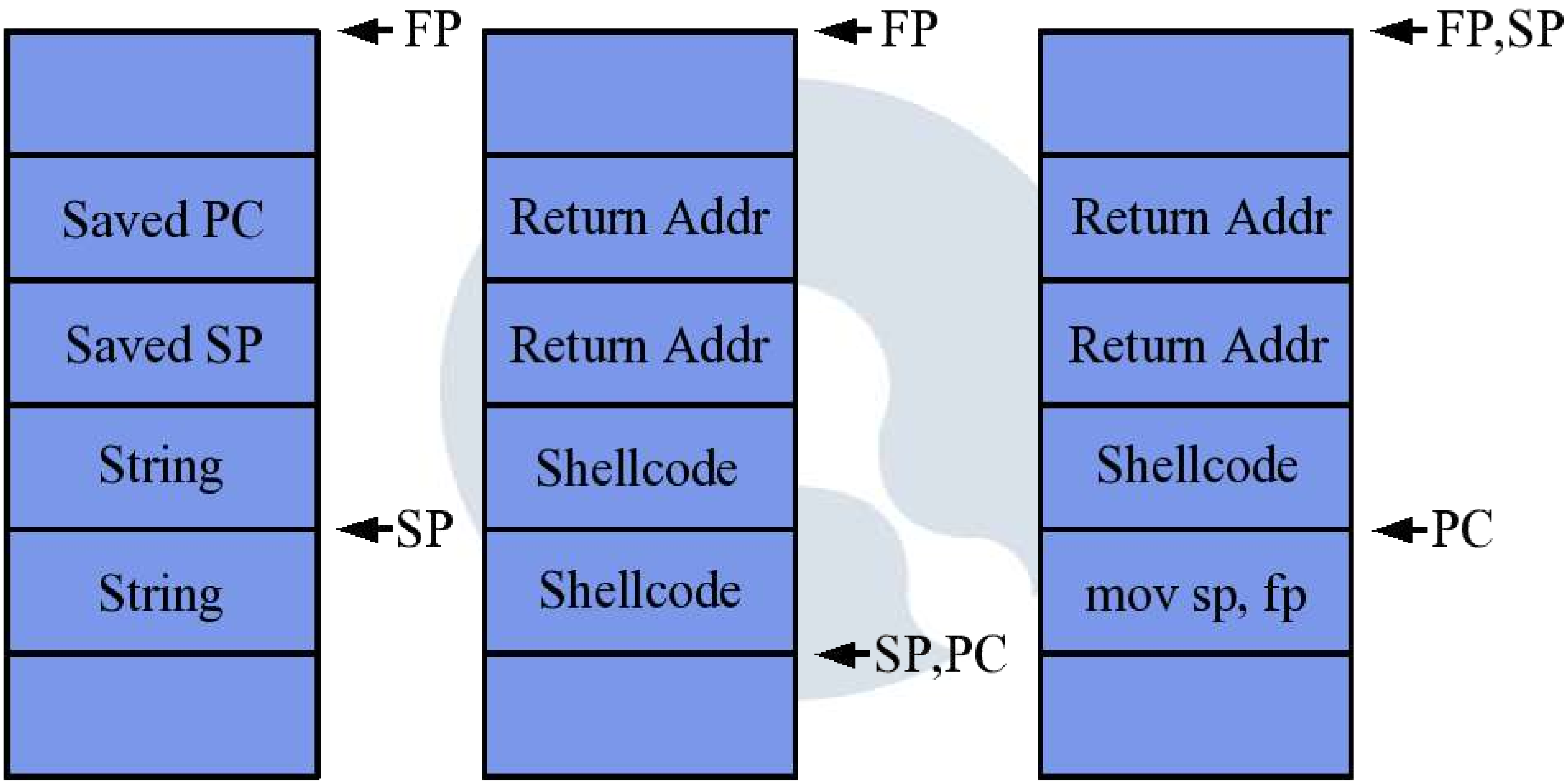
Caches & Buffers

- Need to execute two instructions
  - They require privileged mode
  - User processes should not have this ability
  - Only Linux uses privilege modes
- Trick kernel into flushing caches
  - Still need to execute shellcode prelude

```
mcr p15, 0, r1, c7, c10, 4  // drain the write buffer
mcr p15, 0, r1, c2, c0, 0   // flush instruction cache
```

WinCE Stack Silliness

- Shellcode insertion trashes stack
- IP = SP = Jump Address
- Need to fix registers & stack
    - System calls cause freeze – Easy DoS
    - Cannot have SP <= IP <= FP
    - FP = Start of thread stack
    - Addresses > FP may be the stack of another thread


FP
Saved PC
Saved SP
String
SP
String
FP
Return Addr
Return Addr
Shellcode
Shellcode
SP,PC
FP,SP
Return Addr
Return Addr
Shellcode
PC
mov sp, fp

ARM Debugging

- ARM has no hardware debug features
  - Except XScale
- Software debuggers replace opcodes
  - Code segments must be r/w
  - Running code in debugger changes return addresses and lots of other info
  - Code may not even be vulnerable in debuggers
  - Messes with stack and memory

Debugging Gotchas

- Linux – easy, just use local GDB
- WinCE – harder
    - Debuggers use ActiveSync – PPP session
    - Stack overflow locks ActiveSync
    - Debug session fails – neat huh
- Symbian
    - Remote GDB

JTAG

- JTAGs are your friends
  - Remote hardware based debugging
  - Not reliant on sync software
  - Expensive
  - Debugger interfaces expensive and fractured
- Can build or buy GhettoJTAG
- Needs a custom board of soldering to a production board

Cool JTAG-ing

- Every I/O pin is tristate connected
  - Can insert signals to bus of chip without the other being aware
  - Great for reverse engineering
  - Access to complete memory range, including MMAPed IO.
- Trace Buffers can record execution trace
- Access to CPU registers

Dissecting an Exploit

1. Drain the data bus write buffer
2. Repair the stack, or create stack space
3. Decode our shellcode and shift it away from the SP
   - To the heap or further down the stack
4. Drain the data bus write buffer
5. Flush the instruction cache
6. JUMP!

Where do we go!

- WinCE has 32 “slots” which processes run in.
- Process always mapped to its own slot, and when running, to slot 0.
- Neat, no need to worry about the slot.
- Not so neat, slot 0's address is 0x00
- Never fear, ROM is here – System processes always start in the same order

Demo

- Working exploit on an HP iPAQ 5450
- Exploits the vCal parsing engine
- Always on
- Always unauthenticated
- Loads more of these bugs
- Affects all known WinCE devices with the WIDCOMM Bluetooth stack

How about patches?

- Devices run code from Flash & *ROM
  - Read only XIP code
  - Can only patch in Flash/RAM
  - Reverts to original on hard reset
- Complete update is lengthy
  - Not something you want to do often
- Updates often contain new features
  - Manufacturers charge for these

Protection Systems

- Many stack/heap overflow protection mechanisms for x86, why not ARM?
  - Cynically, the devices would fail often
- ARM CPUs have enough power to run protection, why not use it?
  - Want devices to be fast
  - Lack of developer education?
  - Lack of impetus?

Software protection

- Firewalls
  - No network protection
  - Even Linux devices
  - 3rd party implementations
- Anti Virus
  - Only flimsy support
  - Targeted at specific Viruses/Worms

Imagine If…

- Virus infected you PDA
  - Which stores your most confidential info
- You walk into your home/work and sync
  - Creates network connection – unfirewalled
  - Same access as sync computer
- PDA attacks internal network
- Leaks sensitive info to external sites
- PLUS it infects other PDAs

So Basically…

- We have loads of unprotected, vulnerable devices about
- We connect them to our internal networks
- We store our most personal information on them
  - Bank Details, PINs
- Anyone been here before?

Fin!

With Thanks to Mark Rowe
(because he threatened to sack me
if I didn't acknowledge him!)