Cyber Shield:
An approach to defeat malware in edge computers using hardware diversity.

Authors: Colter Barney, Tristan Running Crane, Clemente Izurieta, & Brock LaMeres
BU / Org: Montana State University
POC Email: lameres@montana.edu

Raytheon Sponsor: Jay Lala, Sr. Principal Engineering Fellow
BU / Org: RMD
Email: jay_lala@raytheon.com

Date: 11/3/2022

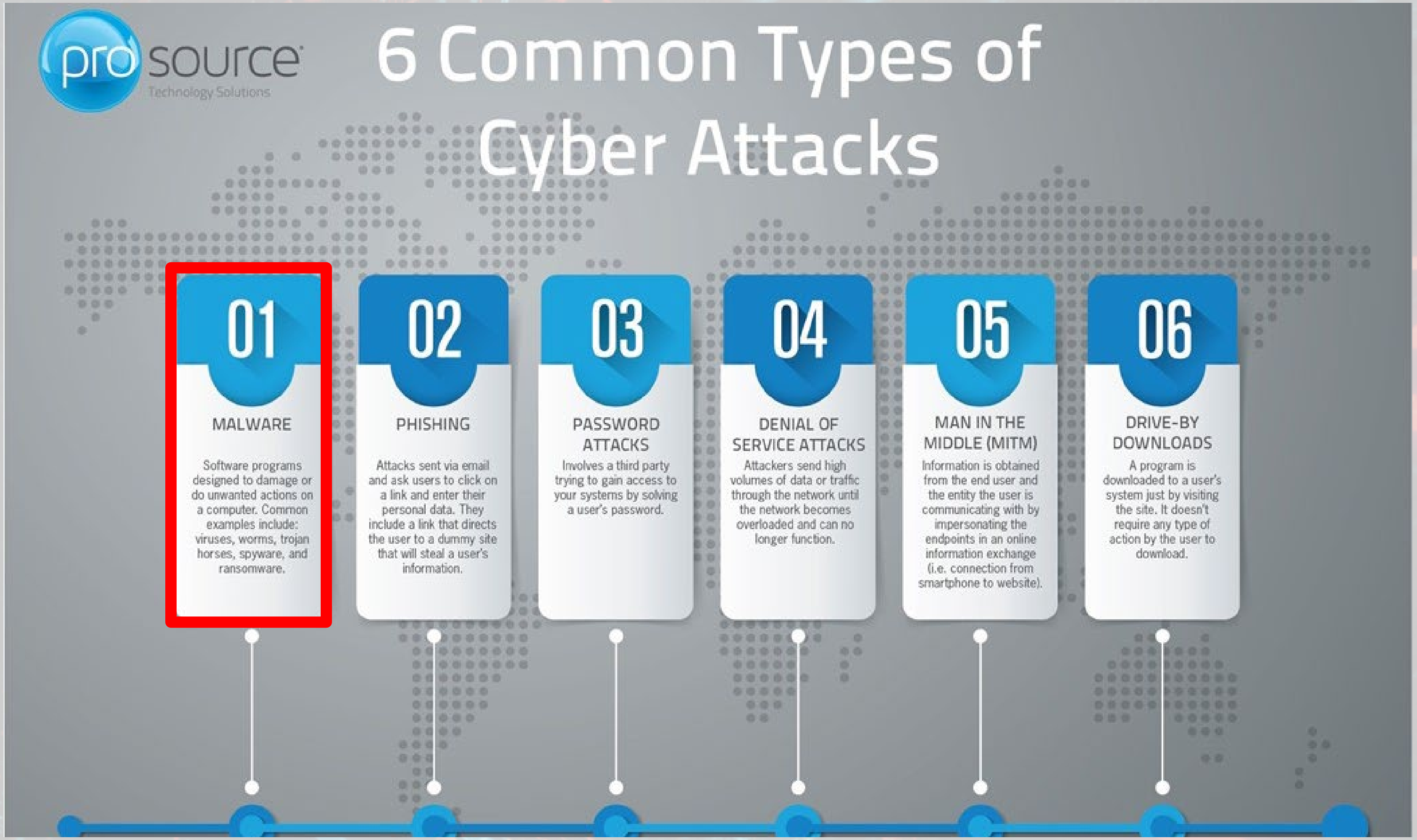
Raytheon Technologies

RTX | 2022 Cyber Summit

Types of Cybersecurity Attacks

Malware Focus

- **Inserting malicious code into the computer's program memory and tricking the processor into executing it.**



Background

The Malware Cybersecurity Challenge

- The nation's cyber infrastructure consists of a massive number of identical computer systems.
- This homogeneity is advantageous because a single piece of software can be deployed across millions of systems to increase capacity.

However, this gives an attacker a significant advantage in terms of effort relative to system defenders by re-using their attack across numerous systems.

Raytheon Technologies

RTX 2022 Cyber Summit

The attacker’s advantages become greater as we move to Embedded Computing.

Personal Computers

- 400 Million sold in 2018

Smart Phones

- 1.5 Billion sold in 2018

Embedded Computers

- 25 Billion Computers

Embedded Computers need Protection from Cyber Attacks as well

Our Approach

Hardware Diversity

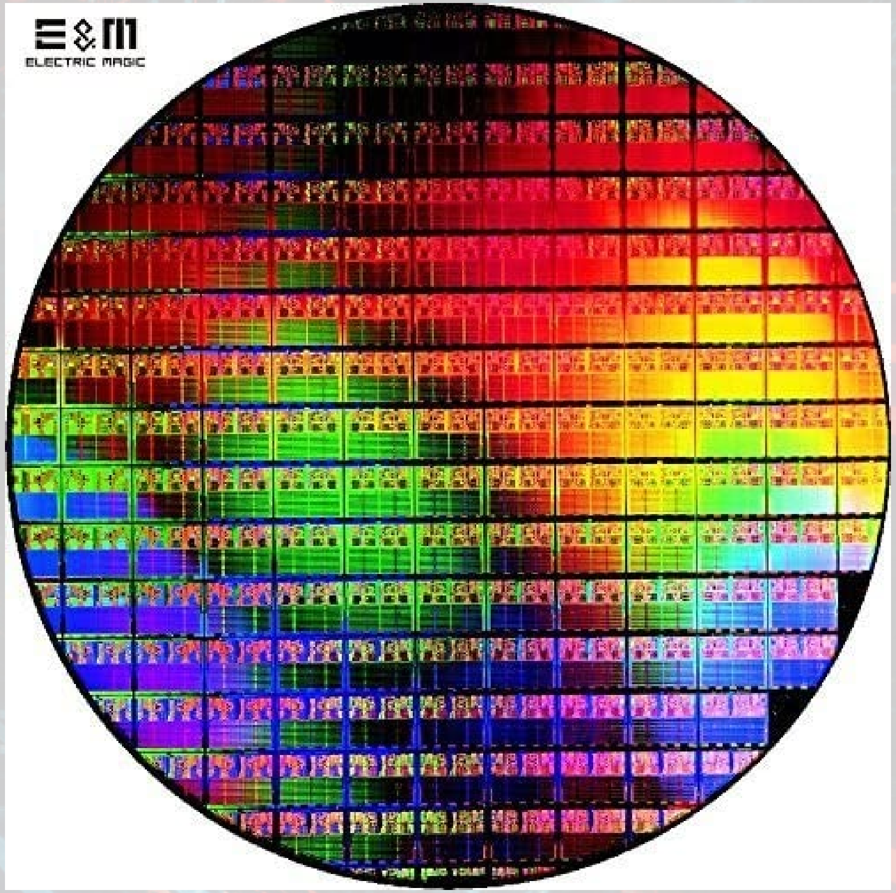
- Homogenous hardware give attackers of embedded systems advantages when injecting code.

- These attacks can be defeated by using Heterogenous hardware, but at the loss of single architecture development.

Our Approach

Hardware Diversity

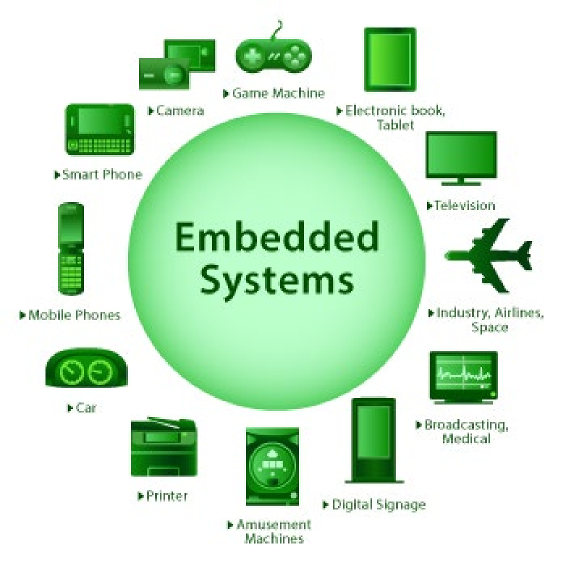
- Hardware is fixed and takes months/years to fabricate.
- There has been some prior work in the area of randomization of instructions sets in Virtual Machines, with promising results.



Our Approach

Embedded Computer Characteristics

- Dedicated software, not general-purpose.
- Smaller (sometimes 8-pin packages)
- Lower Clock Frequencies (1MHz - 16MHz)
- Smaller memories (256k to 1M)
- Often no OS other than real-time scheduler.



Missiles
(Left: RMD SM-6, Right: RMD Patriot)

Radar
(Left: RMD GhostEye® Radar, Right: RMD SPY-6)

Our Approach (FPGAs!)

FPGA Design

- Field Programmable Gate Arrays (FPGAs) allow hardware to be designed using a Hardware Description Language (HDL)

Diversification

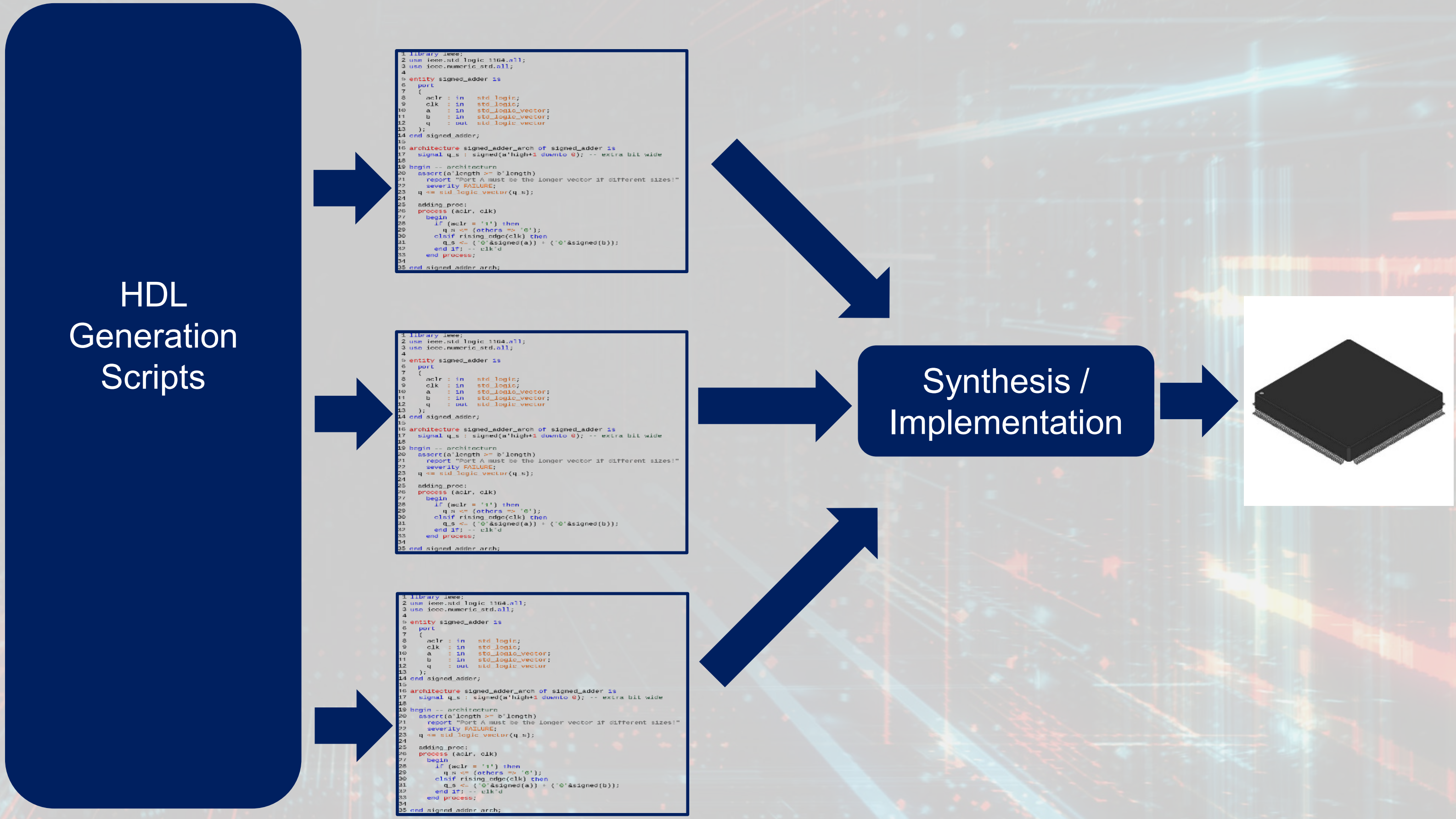
- Once an embedded computer is designed in HDL scripts can be written to create alterations of the computer.

Compile Time Diversification

- The scripts that alter the HDL design can be executed as part of the code generation from a C compiler

Our Approach (Three Cores)

Once we control the HDL generation, we can make modifications to the design & and even replicate it.

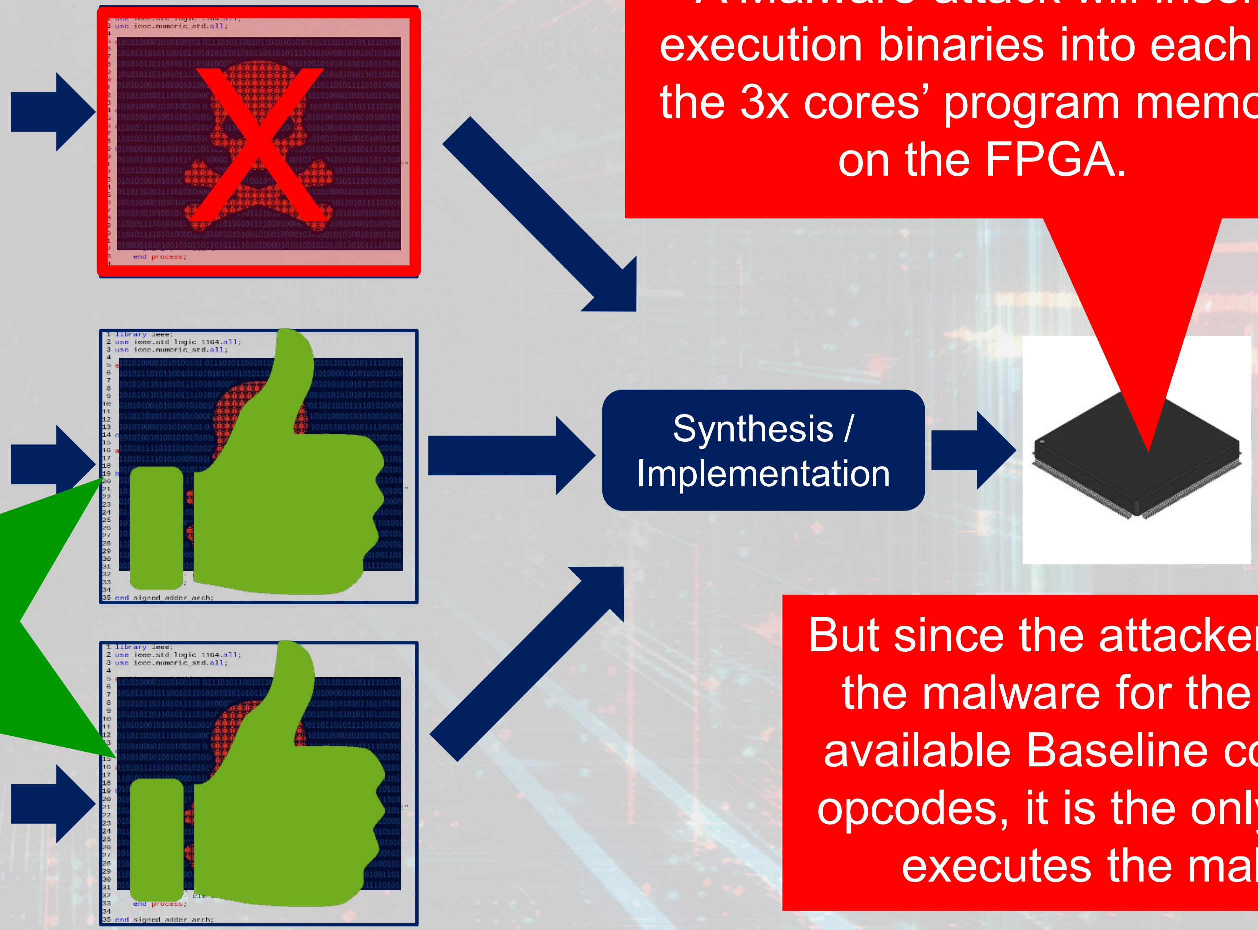


Our Approach (Under Attack)

The three cores share input ports, meaning they cannot be individually targeted

The computers with randomized opcodes don't recognize the malware.

We can either throw an exception or run a pre-defined routine to remove the malware.

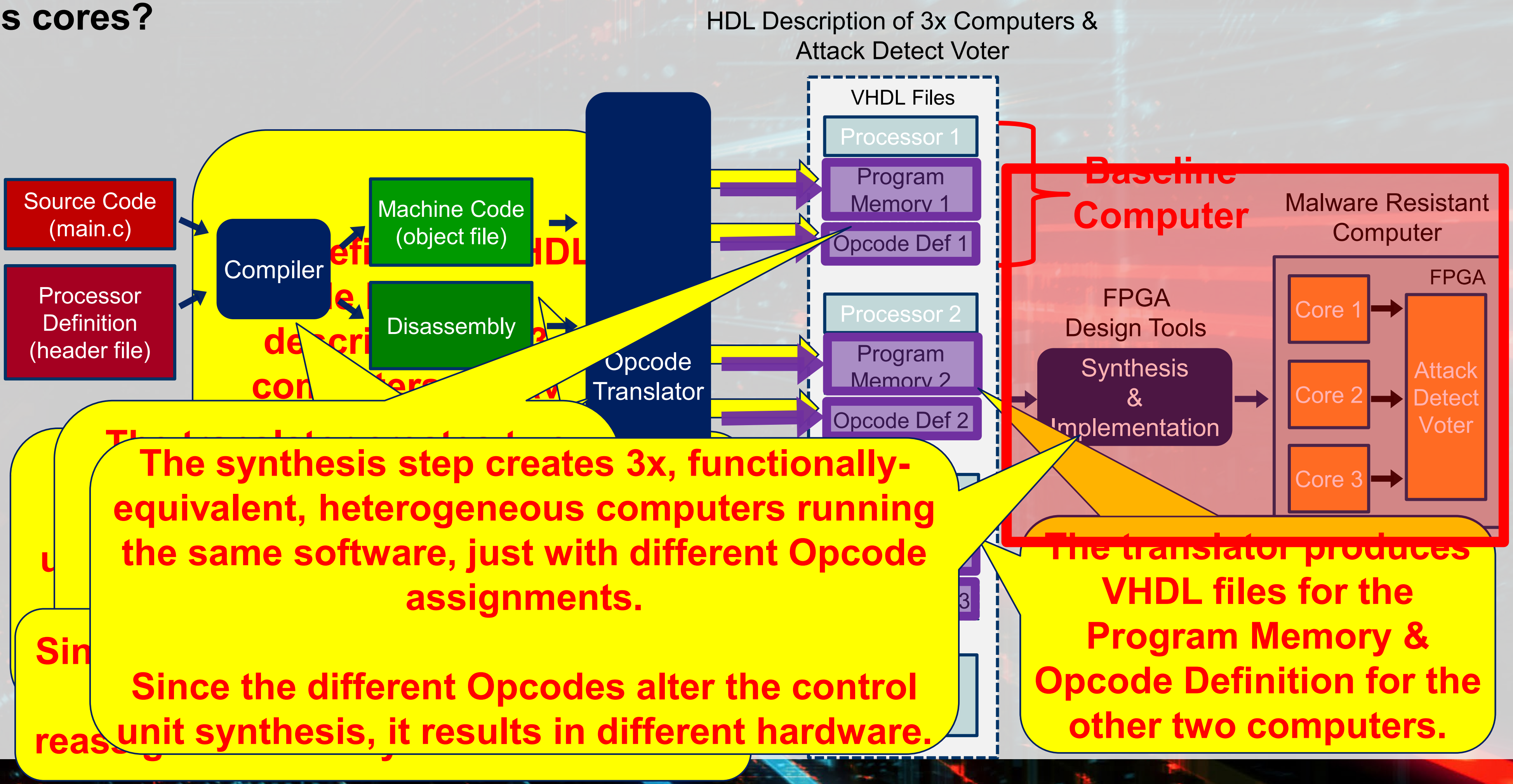


A Malware attack will insert execution binaries into each of the 3x cores' program memory on the FPGA.

But since the attacker compiled the malware for the publicly-available Baseline computer's opcodes, it is the only one that executes the malware.

Our Approach

But how do we map the original source code opcode assignments used by the compiler into the two heterogenous cores?



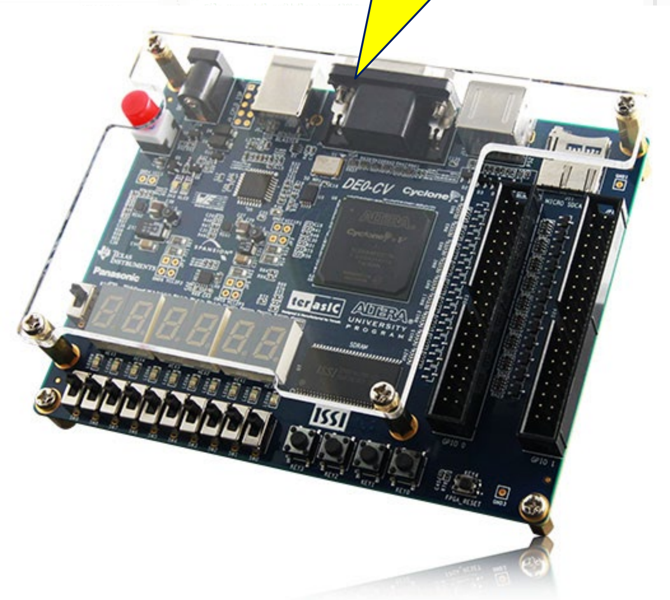
Testbed for Demonstration

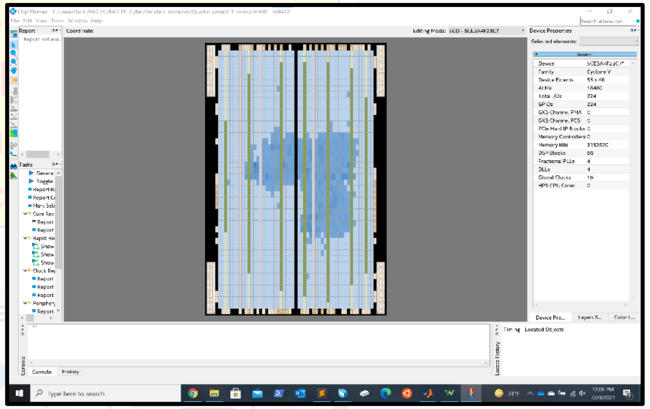
We built a fully functional MSP430 in

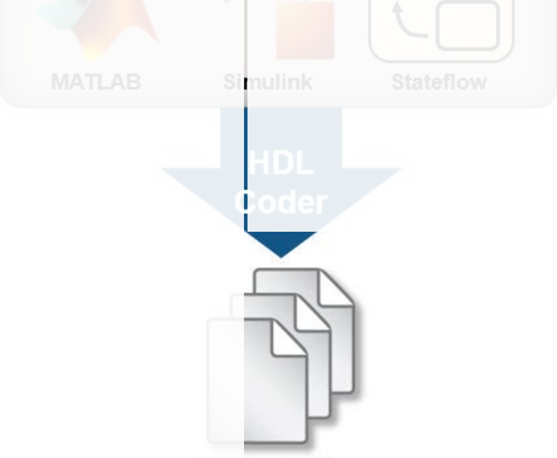
Standard Eclipse Programming Environment

We used Matlab Simulink

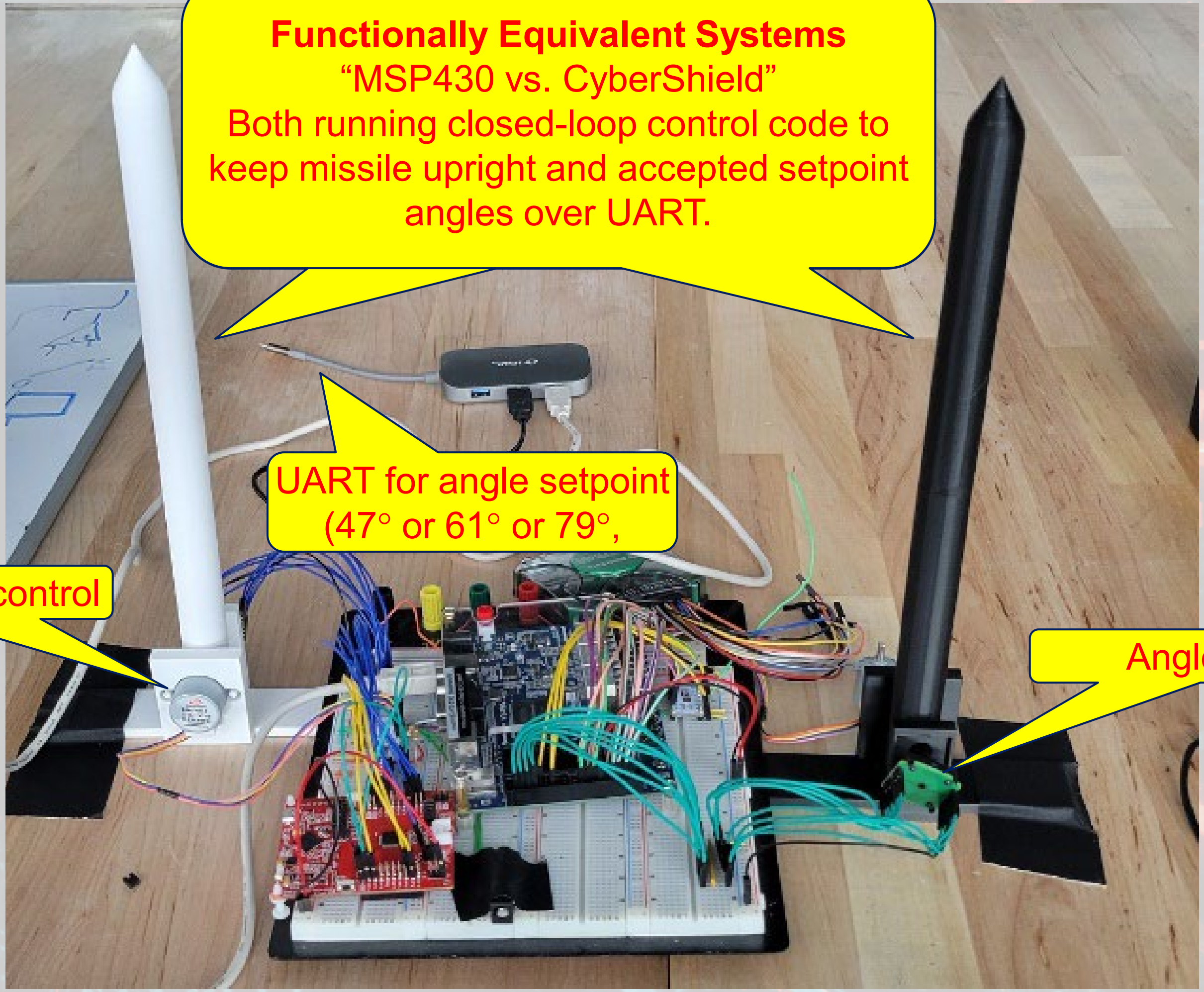
We used the DE0-CV FPGA board with an intel Cyclone V FPGA.



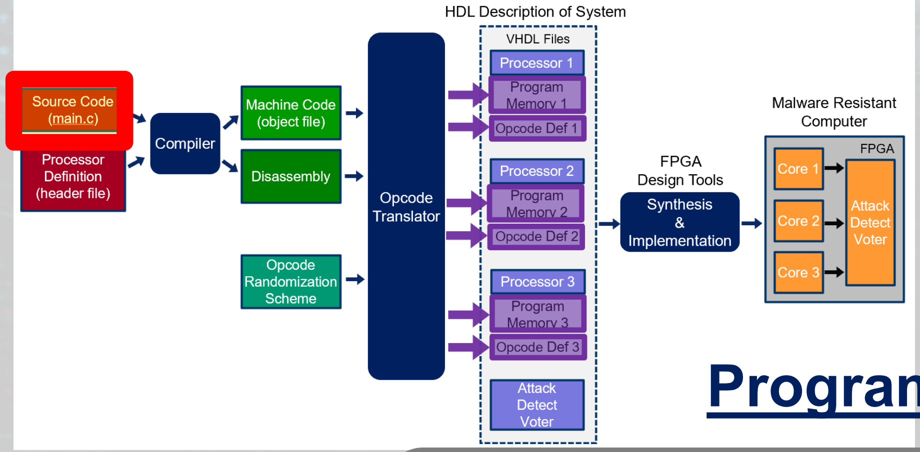




Demonstration Under Attack



Demonstration Under Attack

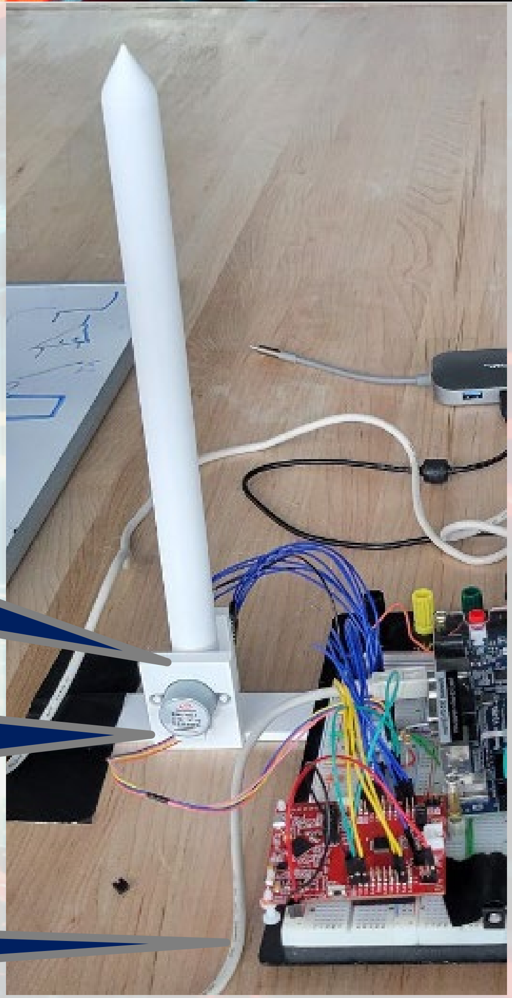


Program Description

The computer periodically sends the stepper motor its setpoint angle. The send frequency is dictated by a timer that triggers and interrupt.

The computer continuously reads the actual angle of the missile from the sensor and compares it to the setpoints. It adjusts motor accordingly.

New setpoints are received asynchronously from a user over UART. A Rx on the UART link triggers an IRQ.



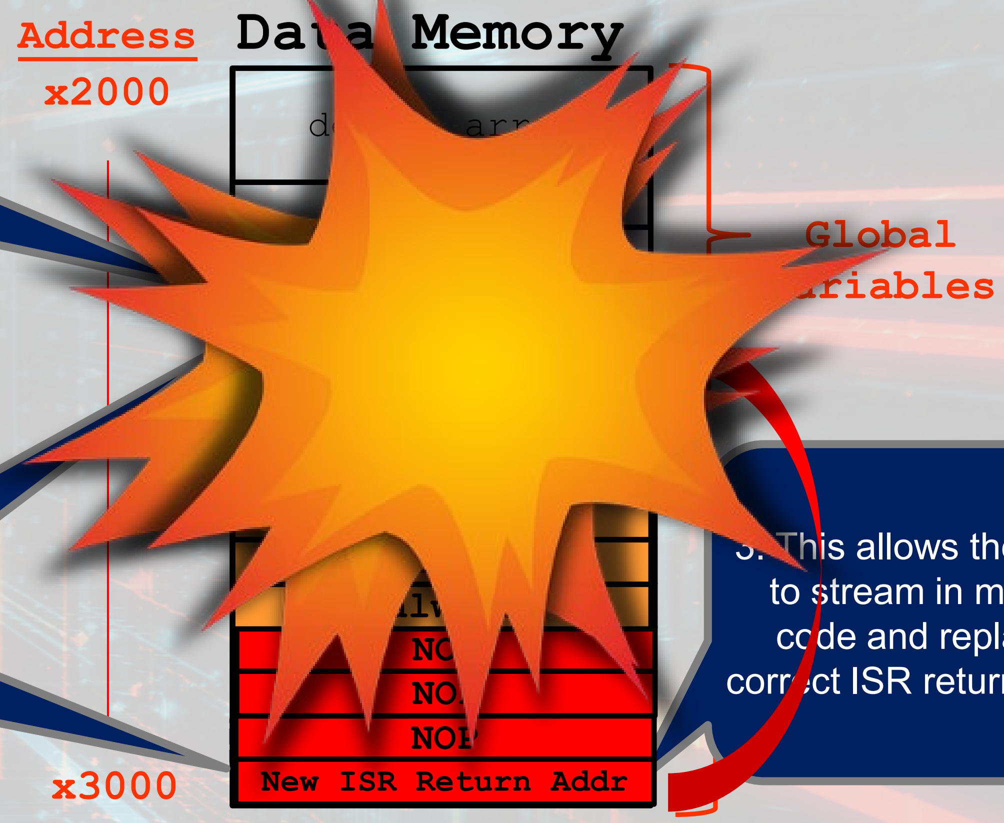
Demonstration Under Attack

```c
while(1){
    for(index=0xFFFF;index!=0;index--){
        _NOP();
    }
    temp = RXBUF[0];
    if(temp == '1'){
        set_angle = 47;
    }else if (temp=='2'){
        set_angle = 79;
    }else{
        set_angle = 61;
    }
    if(rx_index == 1){
        rx_index=0;
    }
    temp = decode_array[P1IN];

    if(temp<set_angle){
        P2OUT &=~(BIT2); //enable stepper motor
//      P2OUT |=(BIT1); //set direction
        P2OUT &=~(BIT5); //set direction
        temp = set_angle-temp;
    }else if (temp>set_angle){
        P2OUT &=~(BIT2); //enable stepper motor
//        P2OUT &=~(BIT1); //set direction
        P2OUT |=(BIT5); //set direction
        temp = temp-set_angle;
    }else{
        P2OUT |=BIT2; //Disable stepper motor
    }
    frequency = 4000 - 63*(temp);
  }
}

#pragma vector = TIMER0_B0_VECTOR;
interrupt void Timer_ISR(){
    TB0CCR0+=frequency;
    P2OUT ^=BIT4;
    //frequency+=1;
    TB0CCTL0 &=~ CCIFG;
//    TB0CCTL0

}

// Service UART
#pragma vector = EUSCI_A1_VECTOR
__interrupt void ISR_EUSCI_A1(void) {
    RXBUF[rx_index++] = UCA1RXBUF;
    UCA1IFG &= ~UCRXIFG;
}
```

2. But the developer introduced a vulnerability by adding a delay loop in the main program to allow the UART to complete before resetting the input buffer size back to 0.

1. When user sends new setpoint over UART, an IRQ triggers, stacks return address, and retrieves new value for RXBUF.

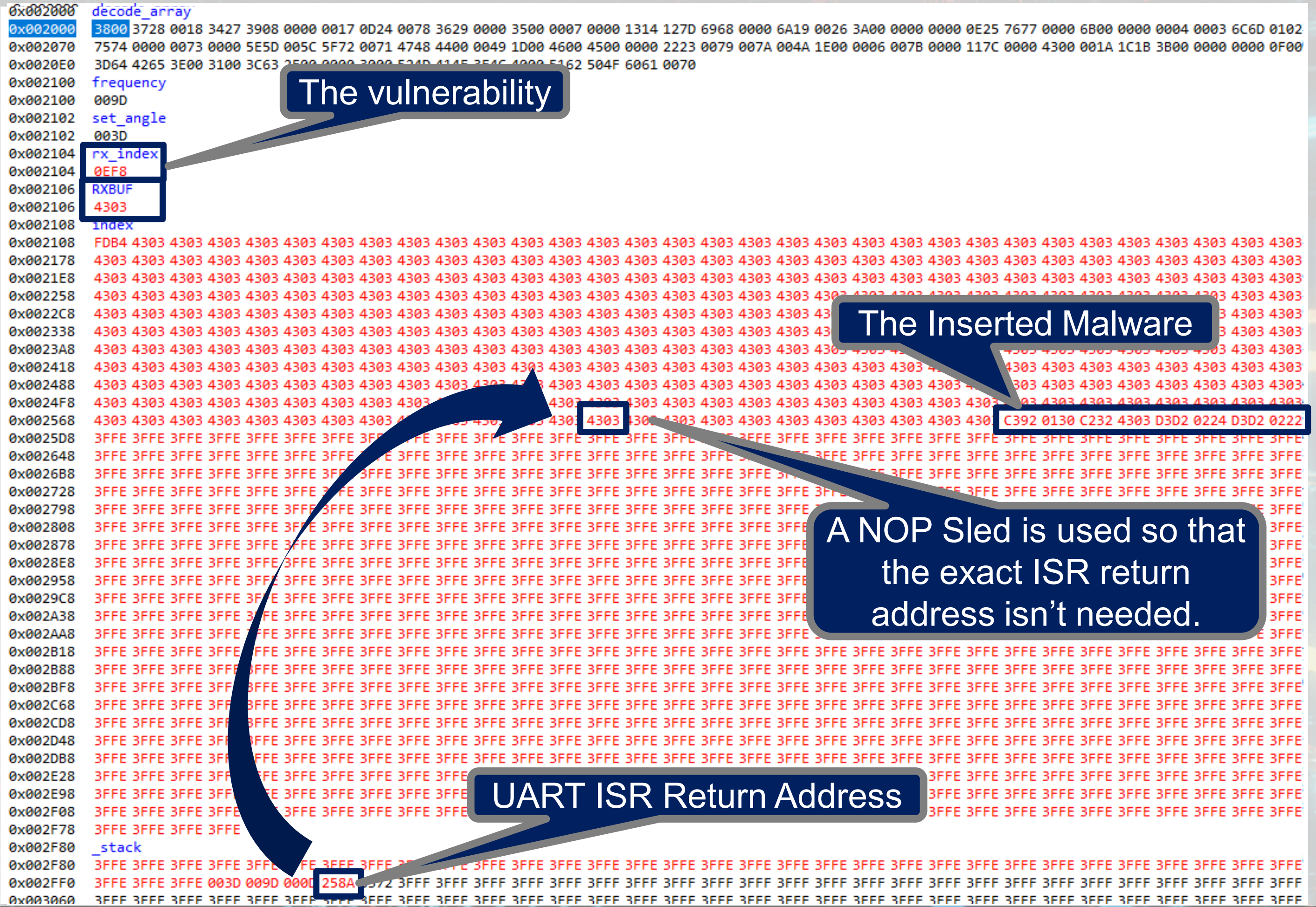
Program Vulnerabilities (Classic Buffer Overflow Attack)

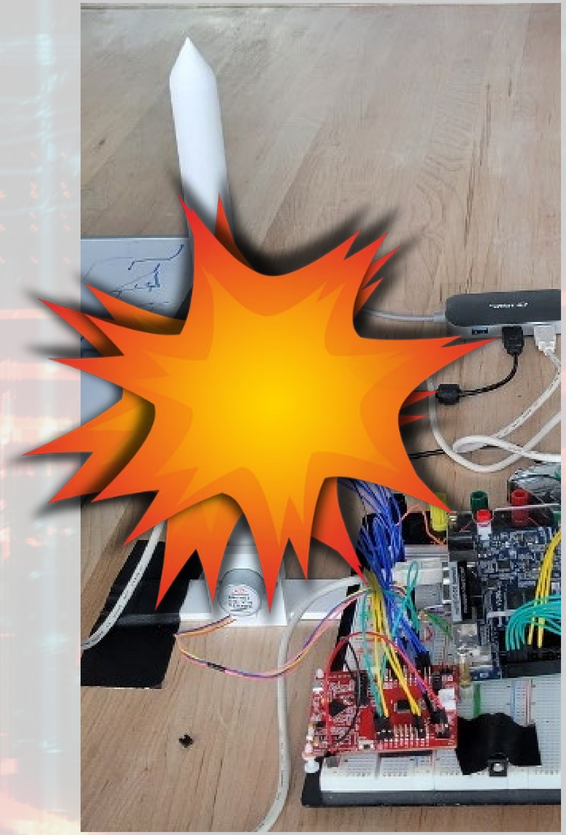


3. This allows the attacker to stream in malicious code and replace the correct ISR return address.

Demonstration Under Attack

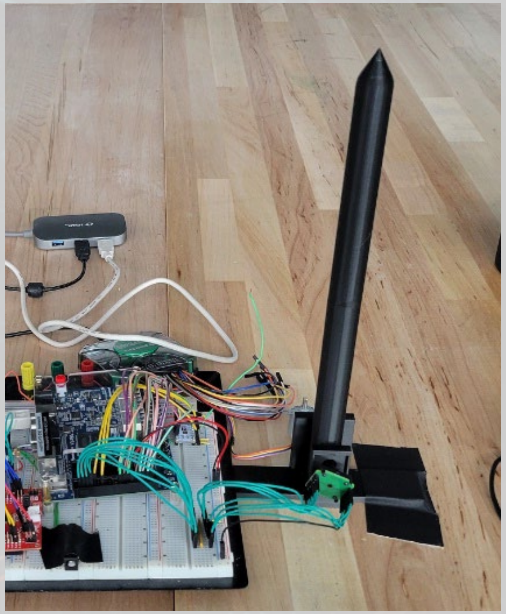
MSP430 Attack – How it looks in data memory…





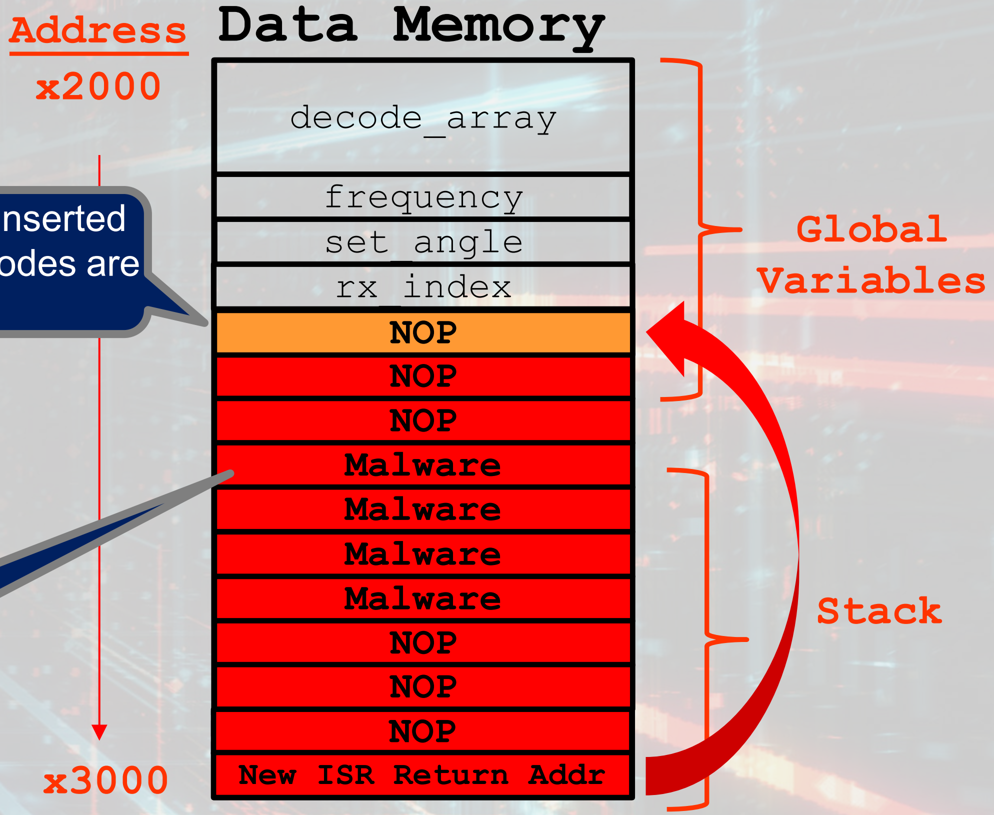
Demonstration Under Attack

The same attack made on our system



But as soon as the starts reading the inserted code in the CPU, it detects that all opcodes are the same!!!

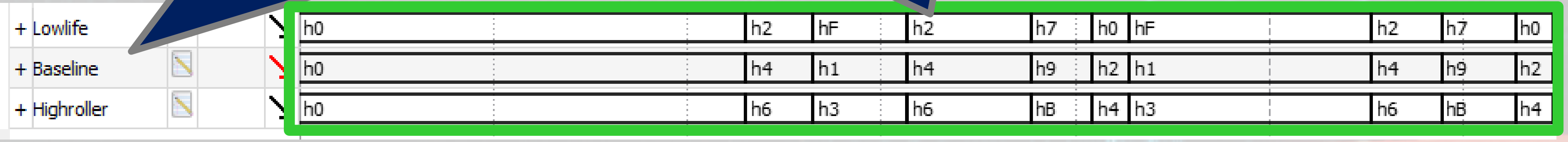
The Malware Still Gets Inserted via Buffer Overflow



Demonstration Under Attack

We can see how CyberShield Responds by Measuring the Instruction Registers in the CPU with a Logic Analyzer.

All Opcodes are Different by Design



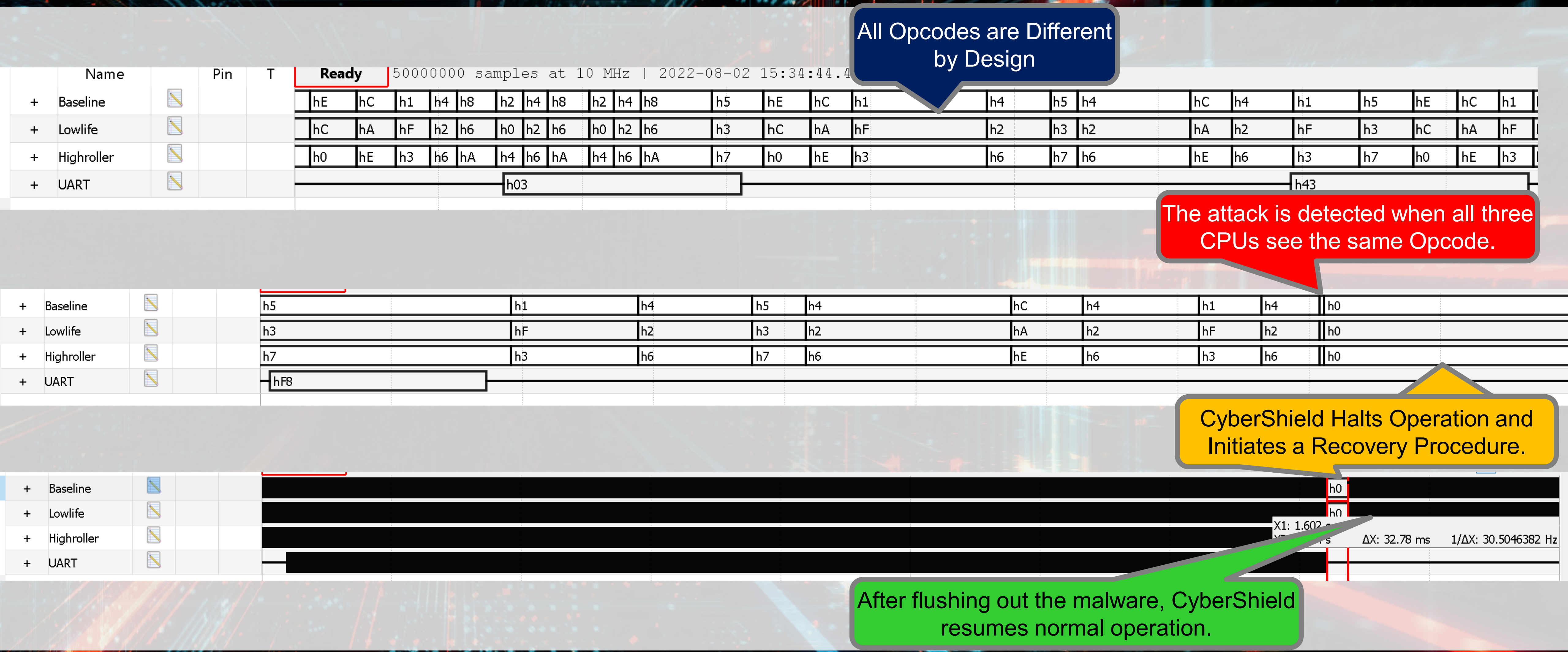
The attack is detected when all three CPUs see the same Opcode.

CyberShield Halts Operation and Initiates a Recovery Procedure.

After flushing out the malware, CyberShield resumes normal operation.

The rapid nature of hardware recovery allows low latency and the ability to operate-through-attack.

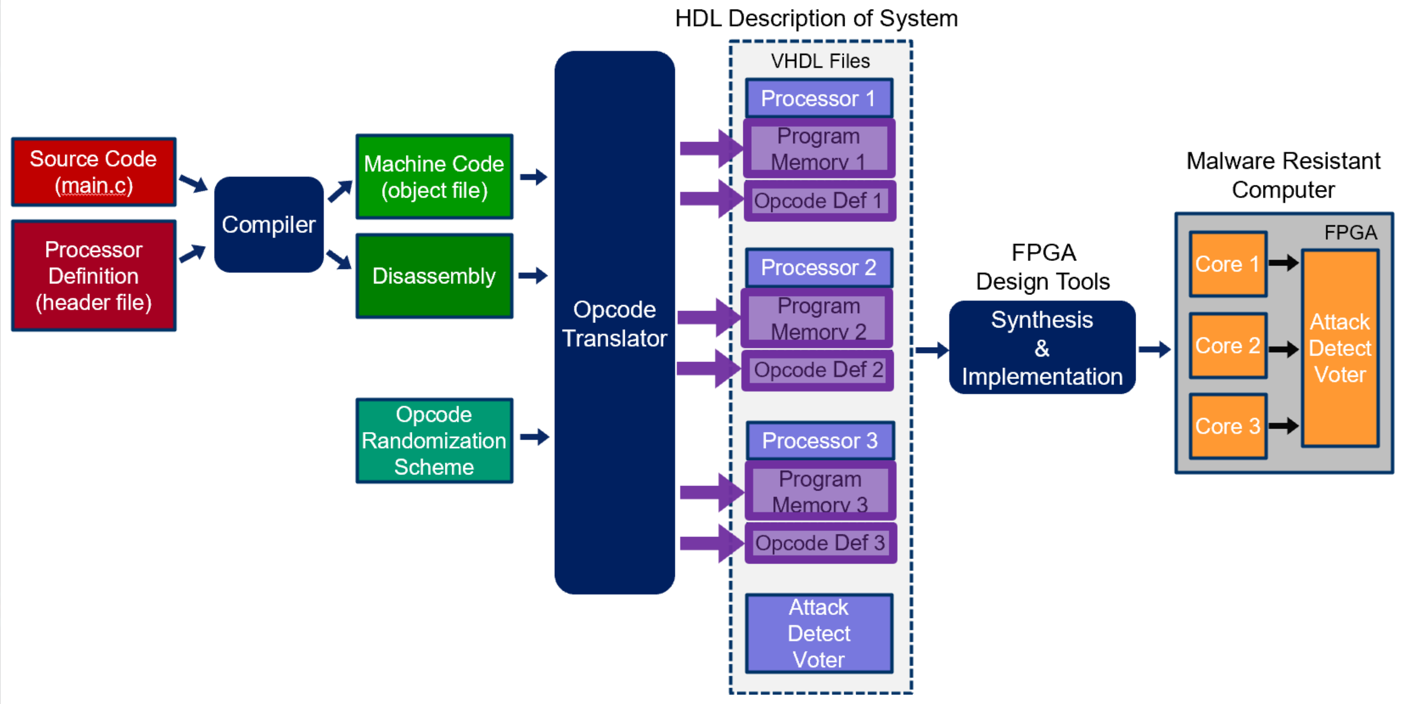
Demonstration Under Attack



Conclusion

- CyberShield is an approach to defeating malware by introducing hardware diversity at the hardware level.
- This is enabled by real-time HDL generation at compile-time.
- A buffer insertion attack was used to test CyberShield.
- CyberShield was able to detect the malware, remove it, and continue operation while an MCU was not.

Questions


HDL Description of System
VHDL Files
Source Code
(main.c)
Processor
Definition
(header file)
Compiler
Machine Code
(object file)
Disassembly
Opcode
Randomization
Scheme
Opcode
Translator
Processor 1
Program
Memory 1
Opcode Def 1
Processor 2
Program
Memory 2
Opcode Def 2
Processor 3
Program
Memory 3
Opcode Def 3
Attack
Detect
Voter
FPGA
Design Tools
Synthesis
&
Implementation
Malware Resistant
Computer
FPGA
Core 1
Core 2
Core 3
Attack
Detect
Voter