

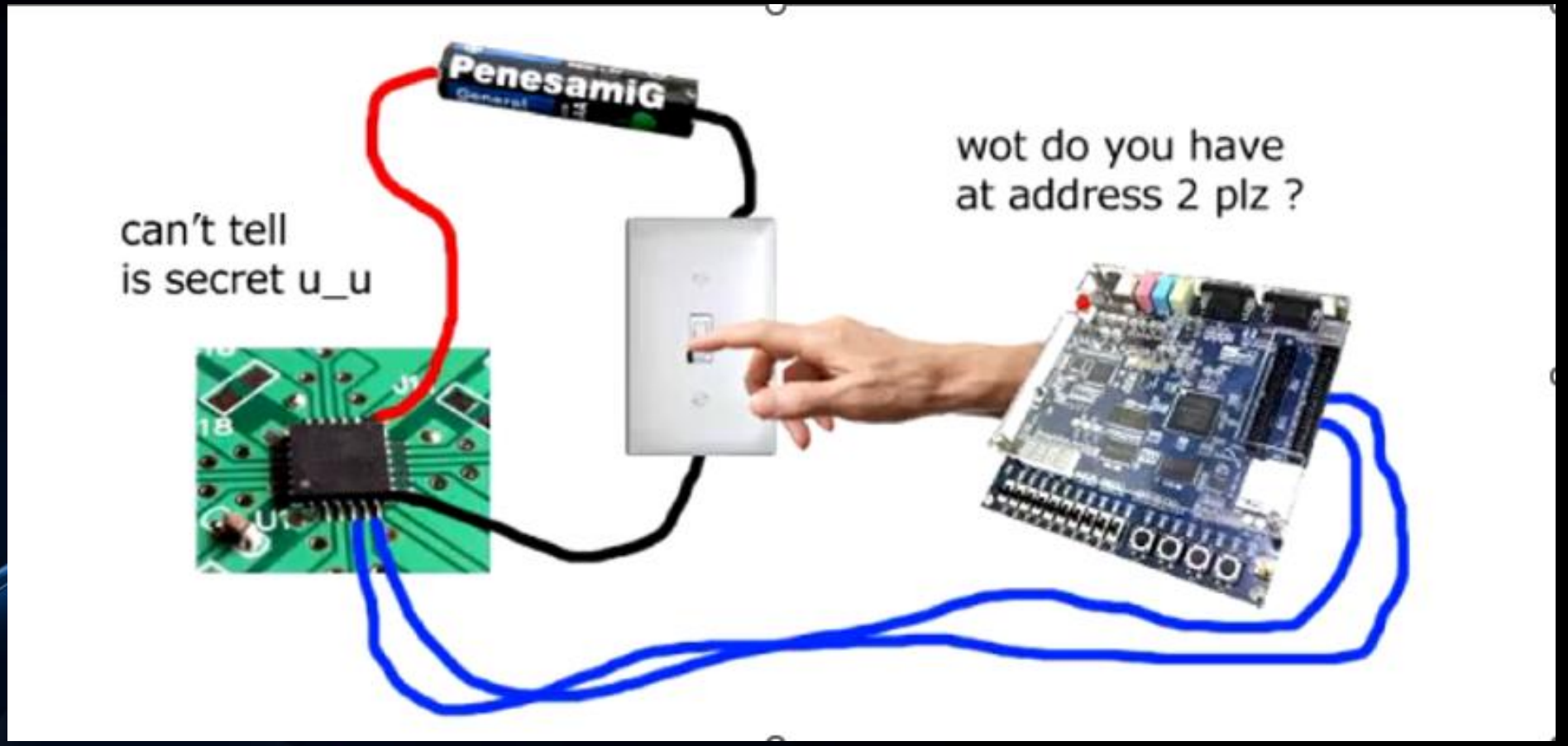




EMFIF2

Electro Magnetic Fault Injection Fuzzing Framework

Luca Bongiorni
Cybersecurity Lab Director & CPSO
ZTE Italia


PenesamiG
wot do you have
at address 2 plz ?
can't tell
is secret u_u

ZTE

Our Journey Today

1. The Problem: The High Barrier to Entry
2. Our Solution: Introducing EMFIF2
3. Under the Hood: The Architecture
4. Visualizing Vulnerability: Heat Maps
5. Demos & A Key Discovery
6. Finding: Global vs. Local Variables
7. The Impact: What's Next?

ZTE

1. Proprietary & Costly Tools 💰
2. Lack of Standardized Frameworks 👩‍🔬
3. Manual & Time-Intensive Processes ⏳
4. No Systematic "Fuzzing" Approach 🎰

ZTE

Our Journey Today

1. The Problem: The High Barrier to Entry
2. **Our Solution: Introducing EMFIF2**
3. Under the Hood: The Architecture
4. Visualizing Vulnerability: Heat Maps
5. Demos & A Key Discovery
6. Finding: Global vs. Local Variables
7. The Impact: What's Next?

ZTE

EMFIF2 Private

Watch 0

main 1 Branch 0 Tags Go to file t Add file Code

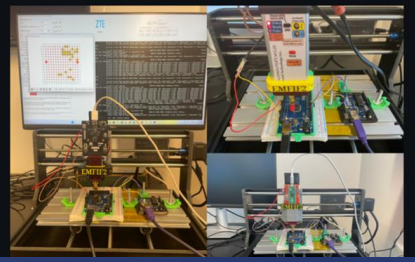
ZTE-Cybersecurity-Lab adding STL files for 3D Printed Holders 2508924 · 3 weeks ago 8 Commits

3D_Printed_Parts	adding STL files for 3D Printed Holders	3 weeks ago
attacks_type	initial commit	5 months ago
controllers	initial commit	5 months ago
devices	initial commit	5 months ago
gui	initial commit	5 months ago
heatmap	initial commit	5 months ago
images	Add files via upload	2 months ago
test	initial commit	5 months ago
LICENSE	Initial commit	5 months ago
README.md	Update README.md	2 months ago
constant.py	initial commit	5 months ago
logger.py	initial commit	5 months ago
main.py	initial commit	5 months ago
requirements.txt	initial commit	5 months ago
utils.py	initial commit	5 months ago

README MIT license

EMFIF2: Electro Magnetic Fault Injection Fuzzing Framework

> Empower your hardware security research through automated EM fault injection fuzzing.



🚀 Getting Started

```
# 1. Clone the repository
git clone https://github.com/ZTE-Cybersecurity-Lab/emfif2.git

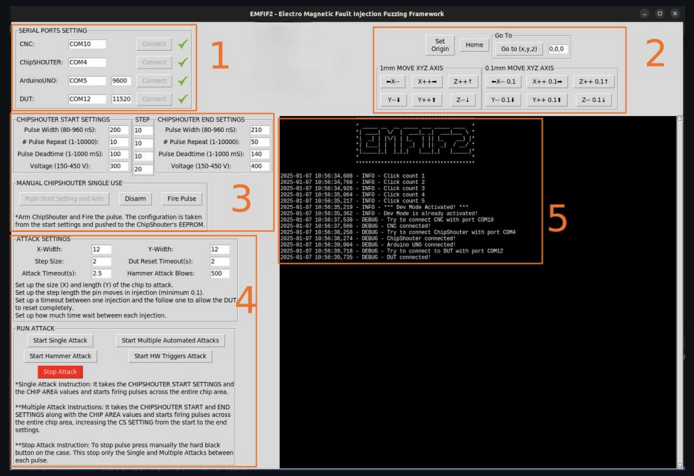
# 2. (Optional) Create a virtual environment
create virtual enviroment with [pyenv](https://realpython.com/intro-to-pyenv/)

# 3. Install dependencies
pip install -r requirements.txt

# 4. Launch the GUI
python main.py

# 5. Connect all USB devices to the PC
```

🖥️ GUI Overview



The EMFIF2 GUI consists of 5 main sections:

1. **Serial Port Setup** – Connect to CNC, ChipShouter/PicoEMP/Faultycat, Arduino, and DUT by selecting the correct COM ports.
2. **CNC Control** – Move the CNC in fine/coarse steps, recalibrate, and home.
3. **ChipShouter Settings** – Configure pulse width, voltage, delay, repetitions, etc.
4. **Attack Settings** – Define board size, CNC step size, DUT timeout, attack timeout, and hammer blows.
5. **Attack Execution** – Choose one of four attack types and begin fuzzing.

Overview

In an era where embedded systems are increasingly integral to critical infrastructure, the need for robust hardware security has never been more paramount. However, conventional security testing methods often overlook physical attack vectors, particularly fault injection attacks. This presentation introduces EMFIF2, an advanced Electro Magnetic Fault Injection Fuzzing Framework designed to expose vulnerabilities in hardware through targeted electromagnetic pulse (EMP) attacks.

EMFIF2 leverages the precision of CNC machines combined with an EMP generator to automate fault injection, enabling researchers and engineers to conduct sophisticated hardware attacks with minimal manual intervention. The framework offers a modular and customizable approach to fuzzing, allowing users to script complex fault scenarios and observe device responses in real-time.

Whether you are a security researcher, embedded developer, or hardware enthusiast, EMFIF2 provides a powerful toolset for advancing your understanding of fault injection attacks and fortifying your hardware against them.

🔧 Hardware Setup

Compatible Components

- CNC 3018
- ChipShouter or PicoEMP or FaultyCat
- Device Under Test (a.k.a. DUT)

Configuration Examples

Depending of what is the DUT other hardware configurations can be quite different:

Example 1

- Arduino UNO (used for reset signal)
- LCD screen (optional, for DUT feedback)
- DUT

> All devices are connected via USB (COM ports). Arduino UNO triggers DUT reset. LCD connected to DUT.

Example 2

- Arduino UNO
- DUT (e.g. Arduino Mega 2560, NRF52, ESP32, STM32, etc.)

> Arduino UNO triggers reset. ChipShouter's impulse is sent to DUT.

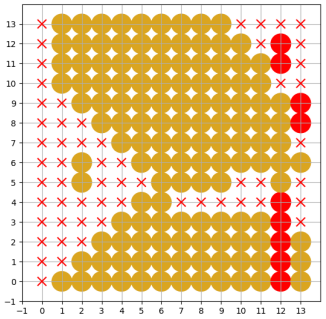
ZTE

Our Journey Today

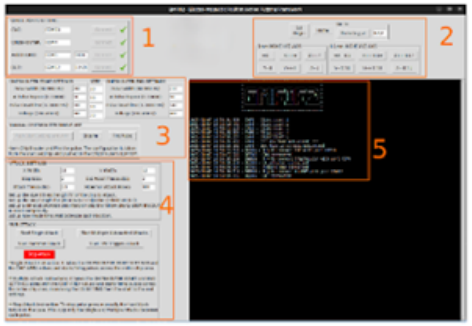
1. The Problem: The High Barrier to Entry
2. Our Solution: Introducing EMFIF2
3. **Under the Hood: The Architecture**
4. Visualizing Vulnerability: Heat Maps
5. Demos & A Key Discovery
6. Finding: Global vs. Local Variables
7. The Impact: What's Next?

ZTE

Framework Architecture



GUI Interface



EMFI Device

Trigger

Support Board

Control

Device Under Test (DUT)

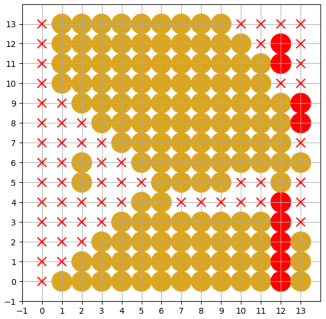
Connections

CNC (3-axis stage)

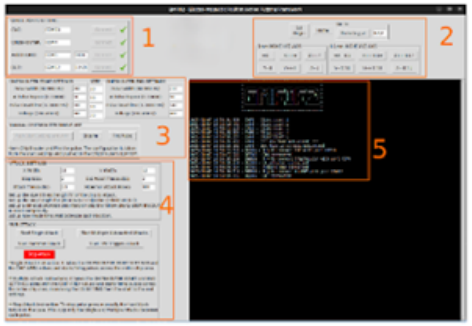
1

ZTE

Framework Architecture



GUI Interface



EMFI Device

Trigger

2

Support Board

Control

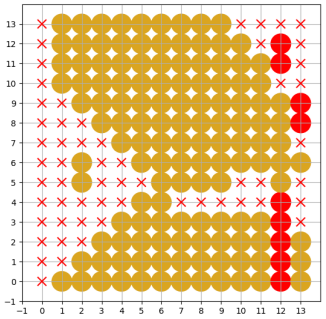
Device Under Test (DUT)

Connections

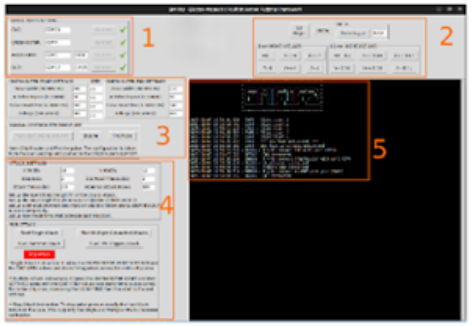
CNC (3-axis stage)

ZTE Cyber Security Lab

Framework Architecture



GUI Interface



EMFI Device

3

Trigger

Support Board

Control

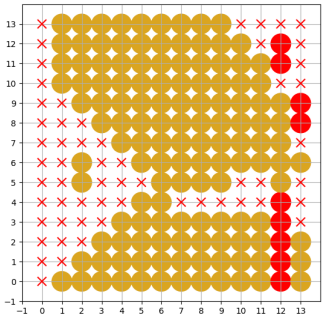
Device Under Test (DUT)

Connections

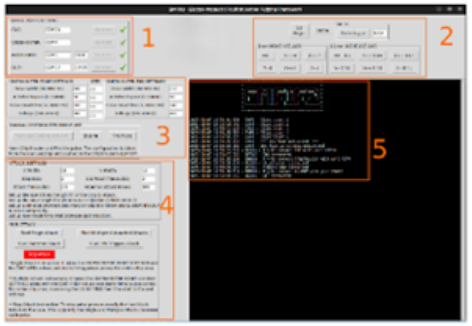
CNC (3-axis stage)

ZTE

Framework Architecture



GUI Interface



4

EMFI Device

Trigger

Support Board

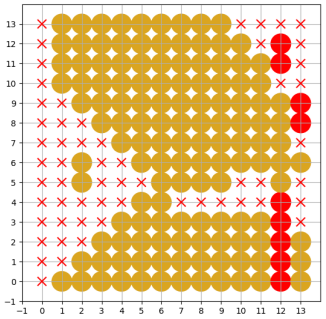
Control

Device Under Test (DUT)

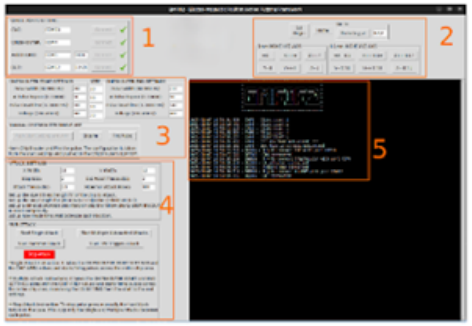
Connections

CNC (3-axis stage)

Framework Architecture



GUI
Interface



EMFI
Device

Trigger

Support Board

5

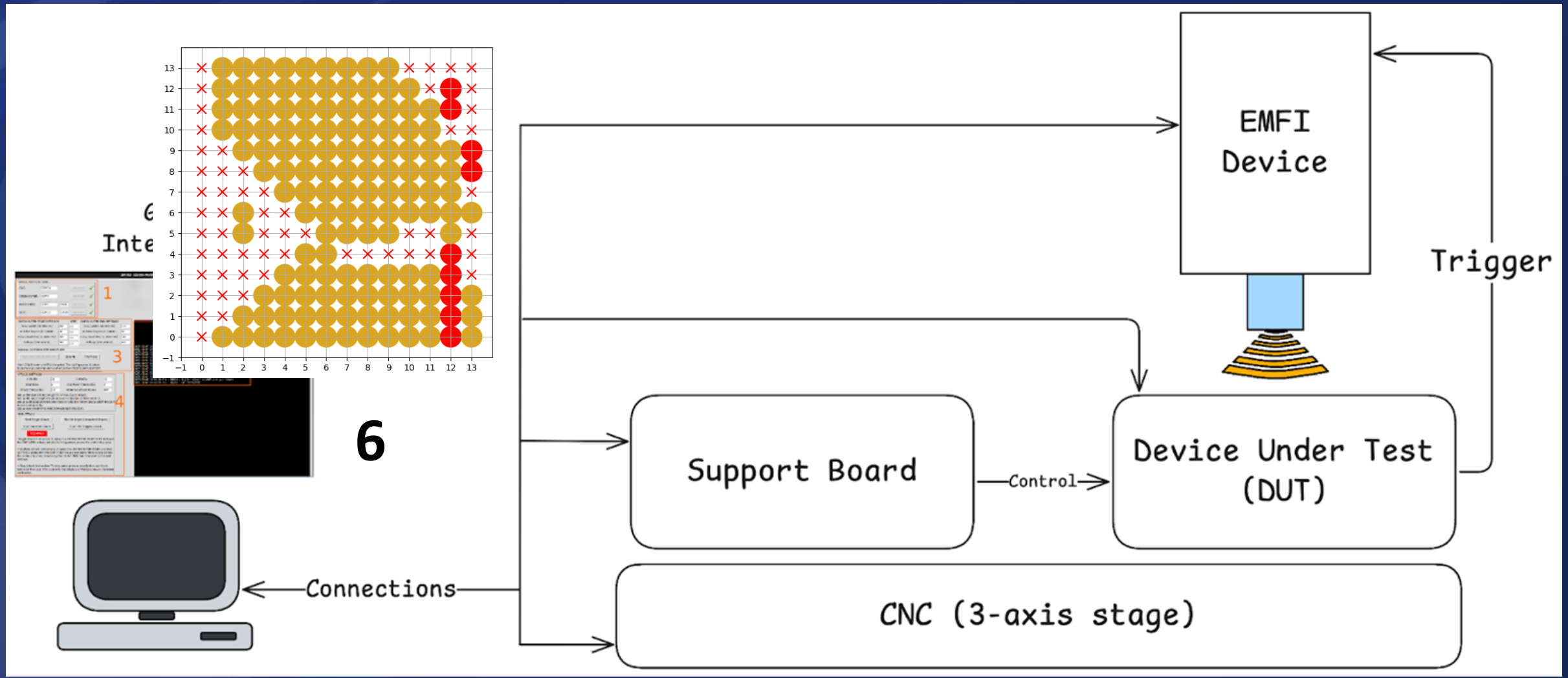
Control

Device Under Test
(DUT)

Connections

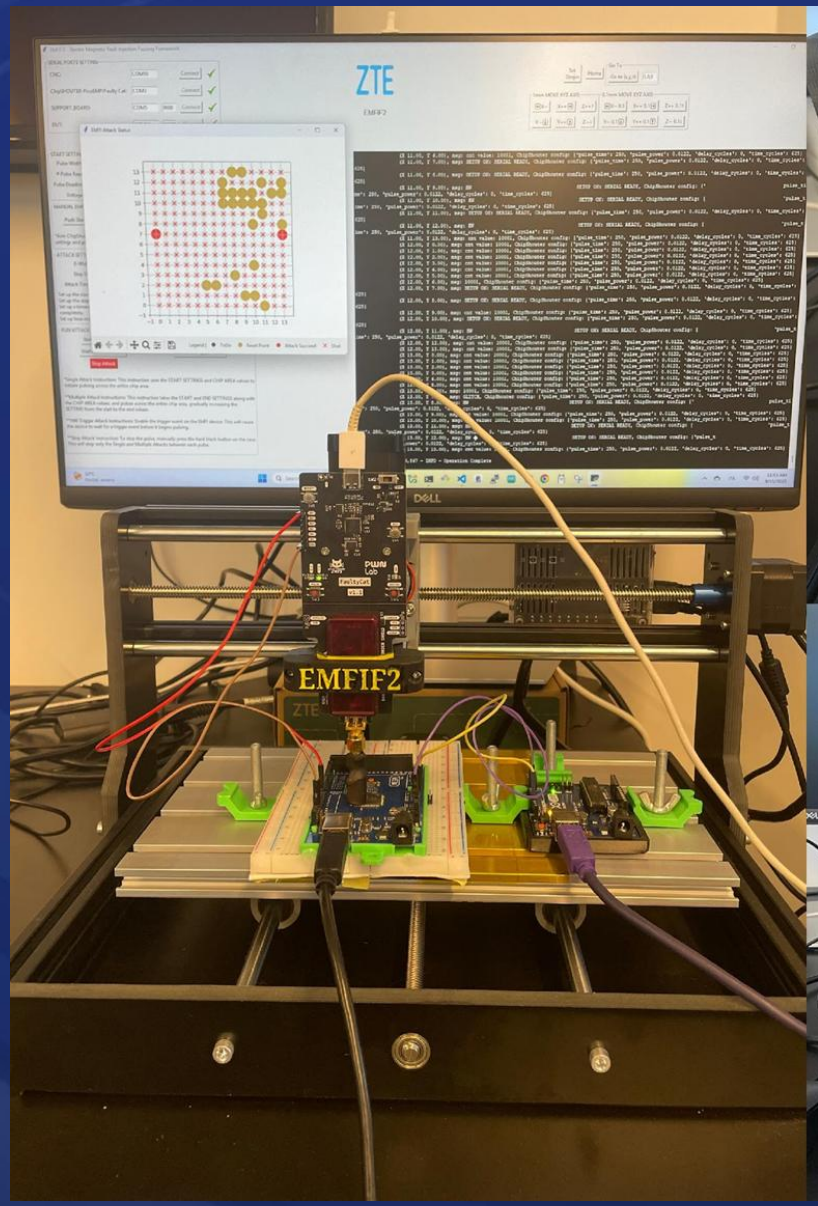
CNC (3-axis stage)

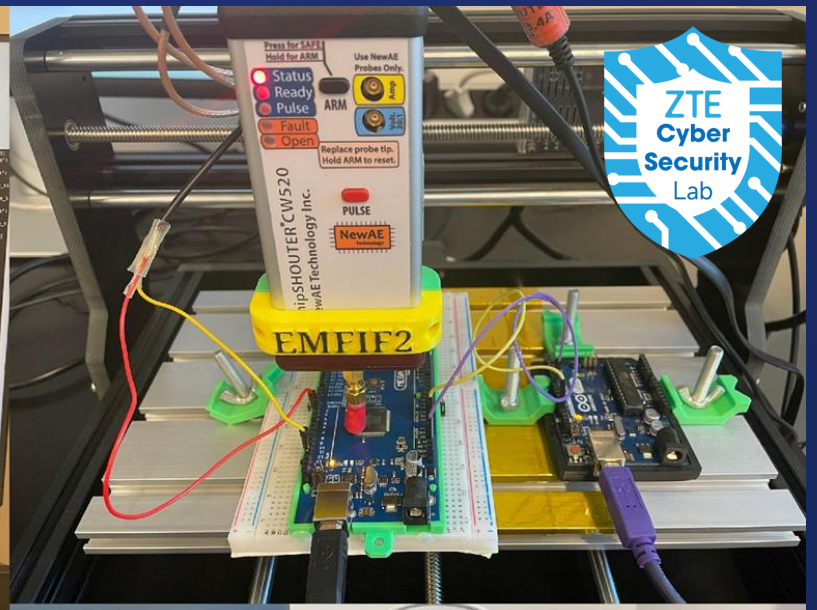
Framework Architecture

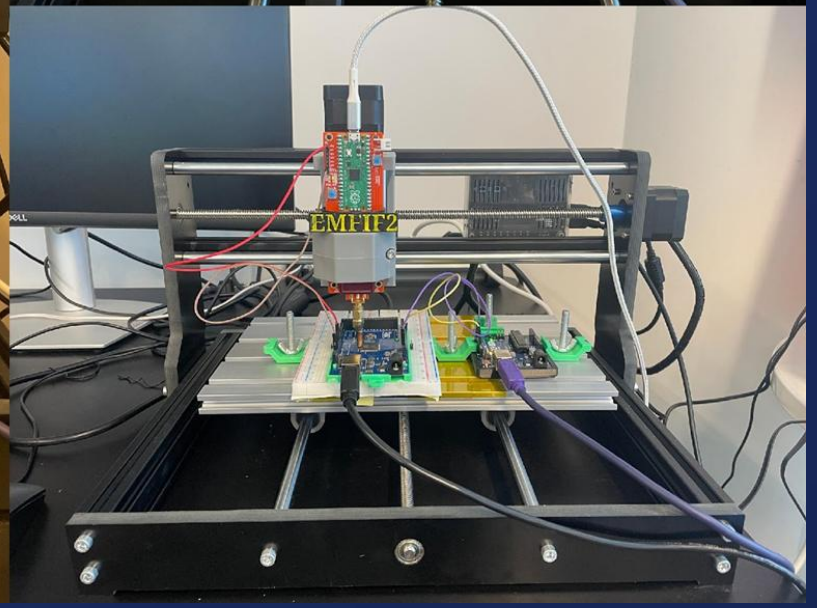


ZTE

Device-Agnostic Approach







FaultyCat

ChipShouter

NewAE Technology

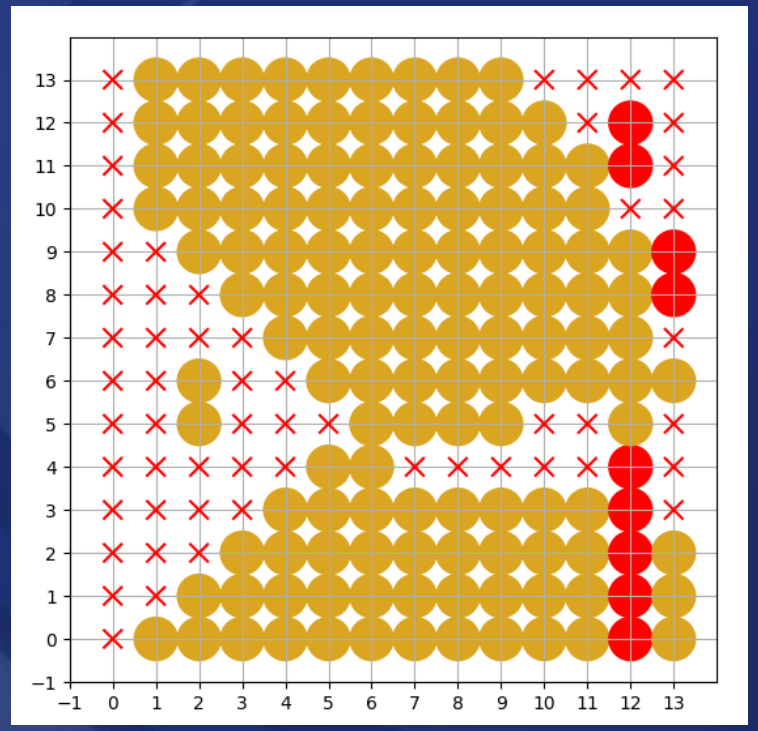
PicoEMP

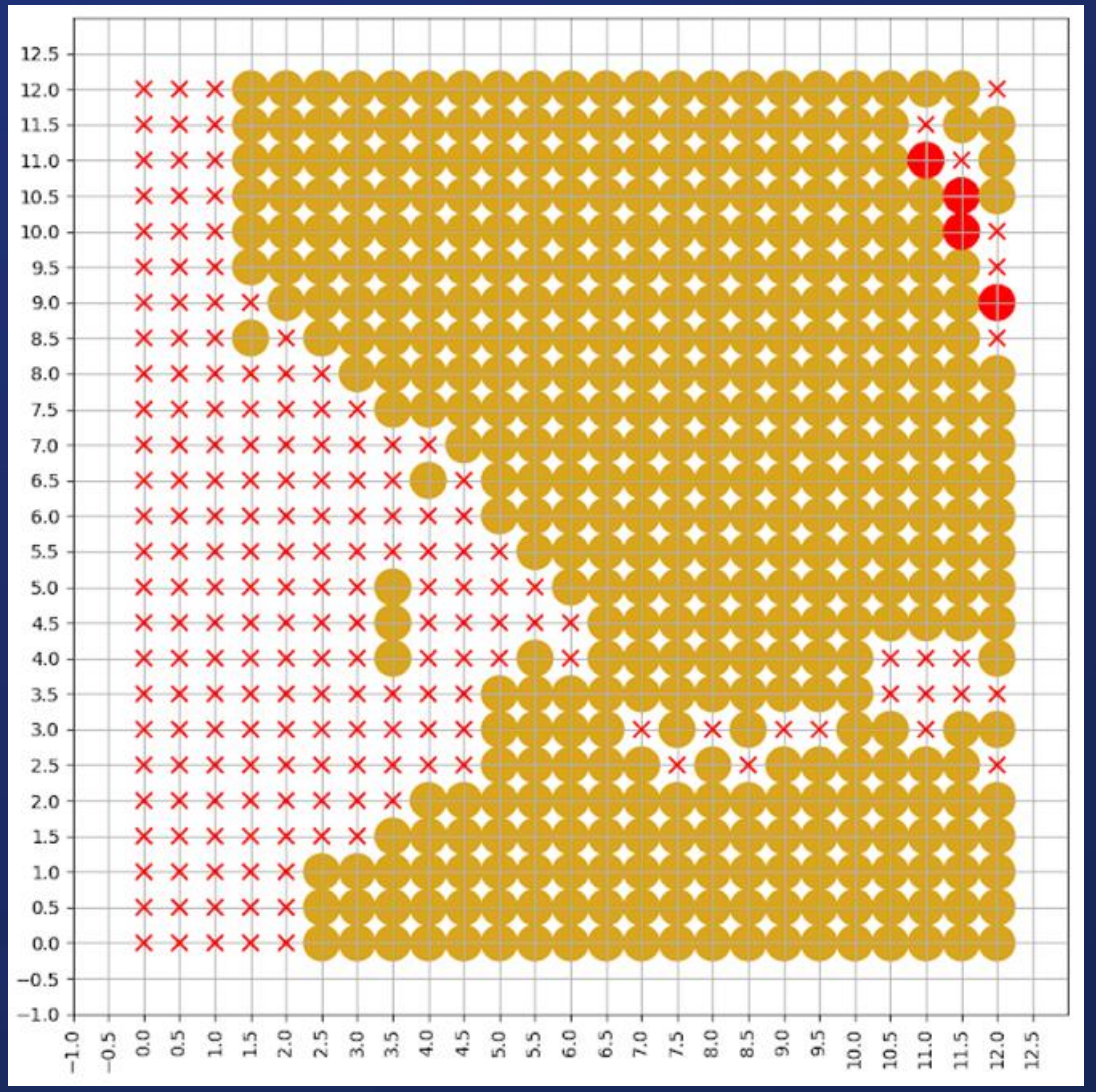
ZTE

Our Journey Today

1. The Problem: The High Barrier to Entry
2. Our Solution: Introducing EMFIF2
3. Under the Hood: The Architecture
4. **Visualizing Vulnerability: Heat Maps**
5. Demos & A Key Discovery
6. Finding: Global vs. Local Variables
7. The Impact: What's Next?

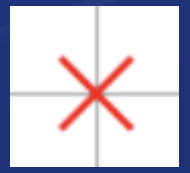
ZTE

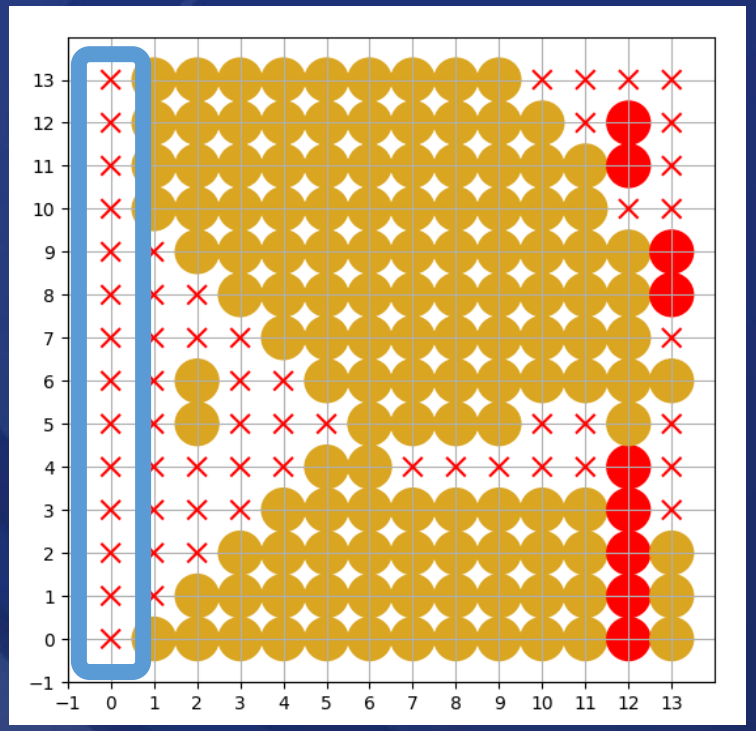


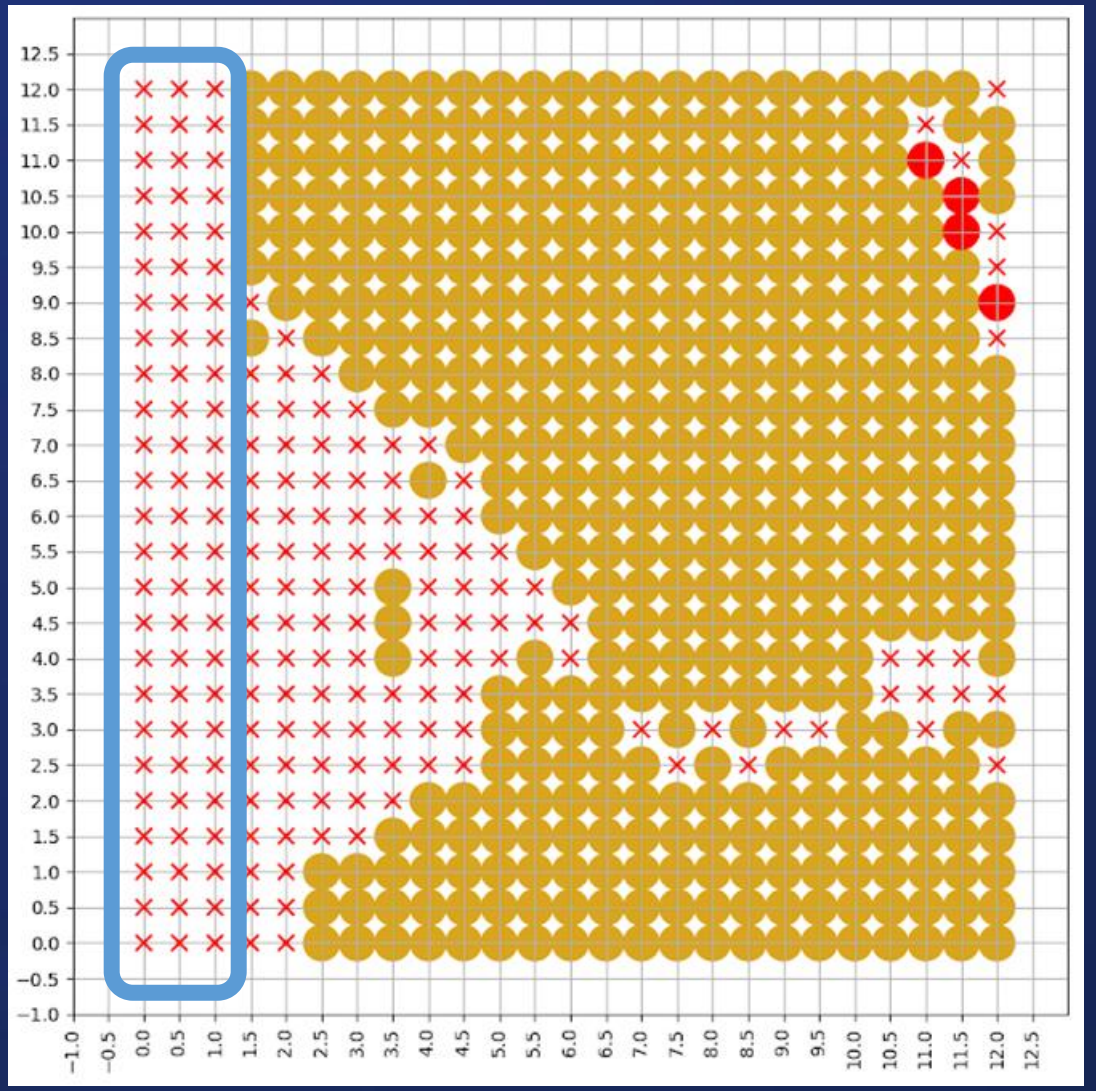


ZTE Cyber Security Lab

ZTE



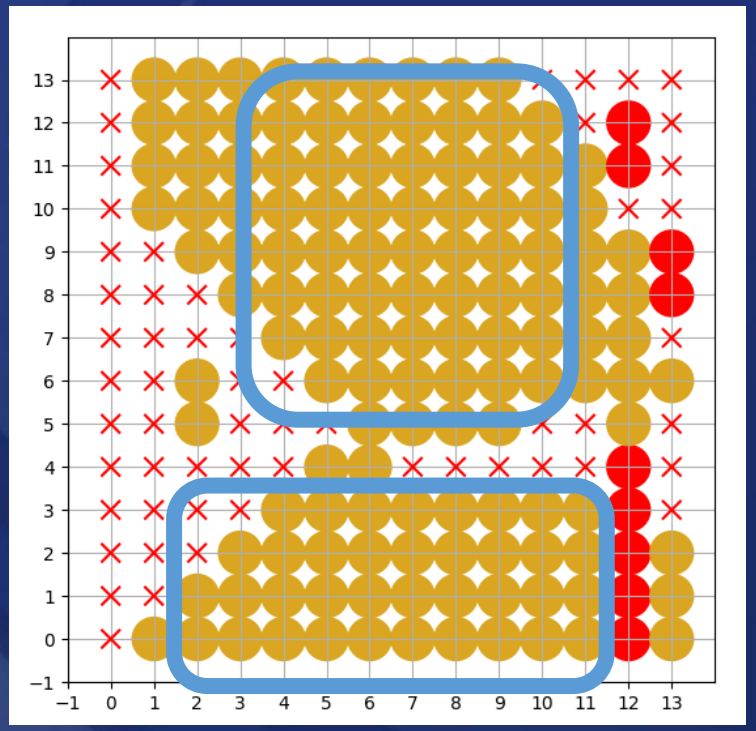


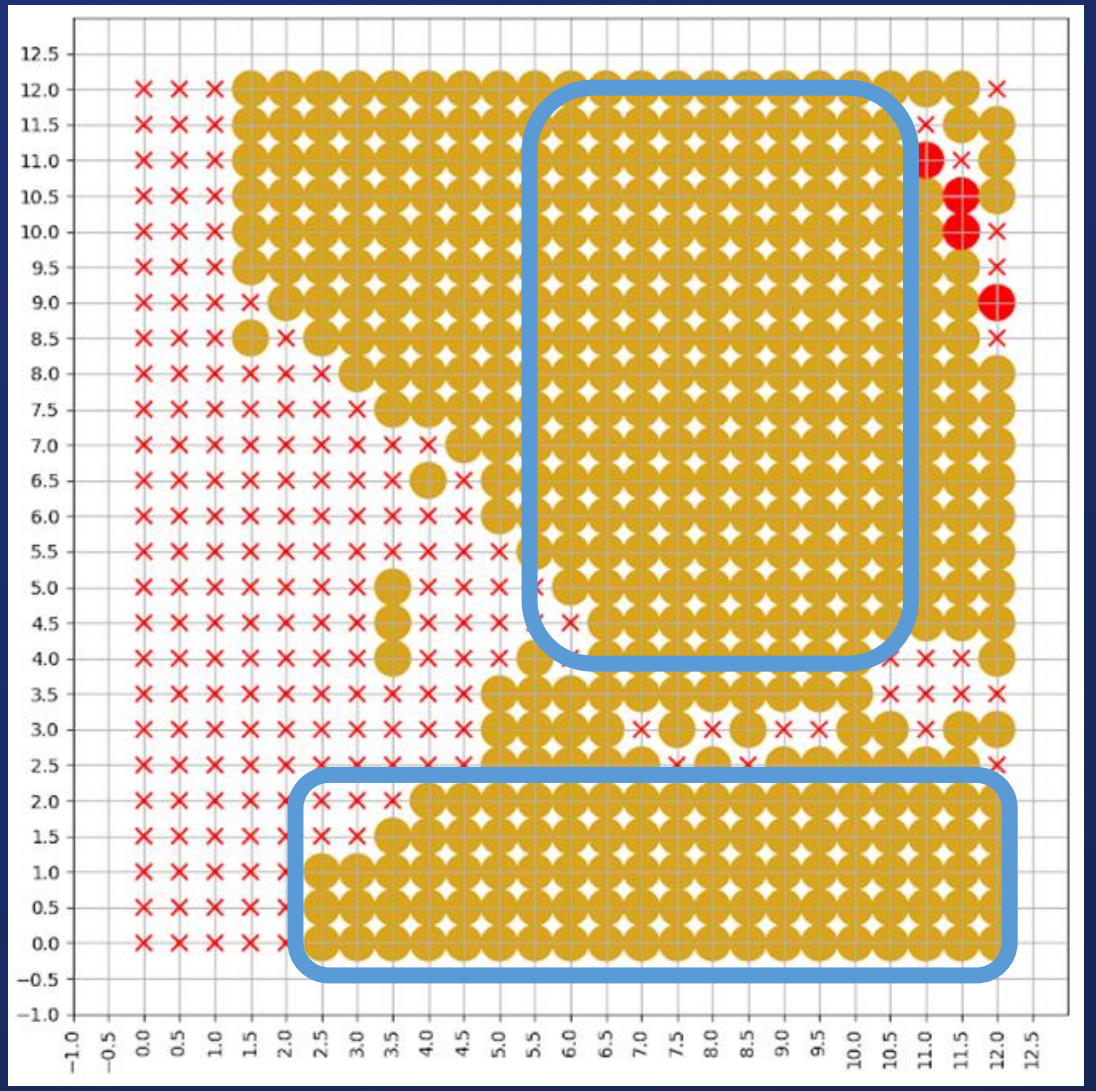


ZTE Cyber Security Lab

ZTE



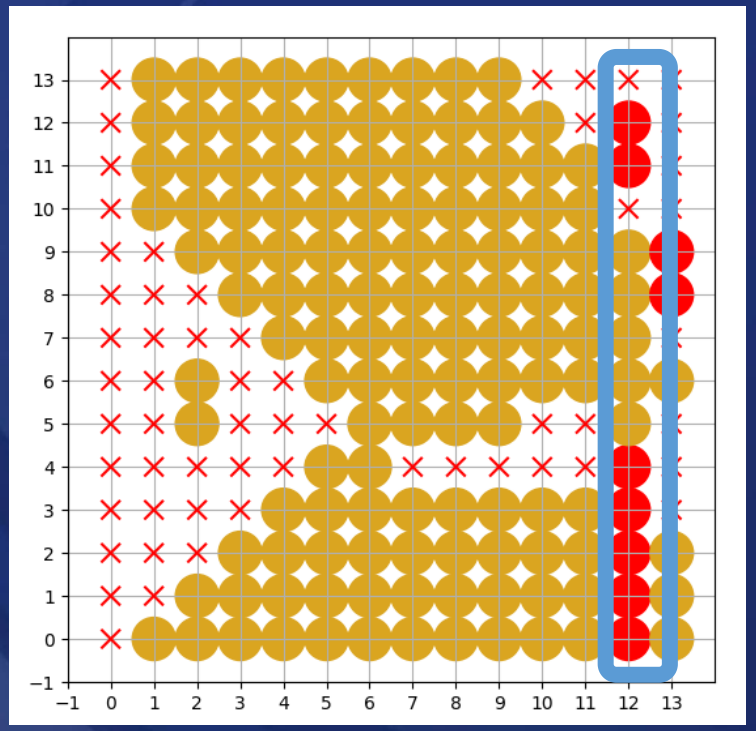


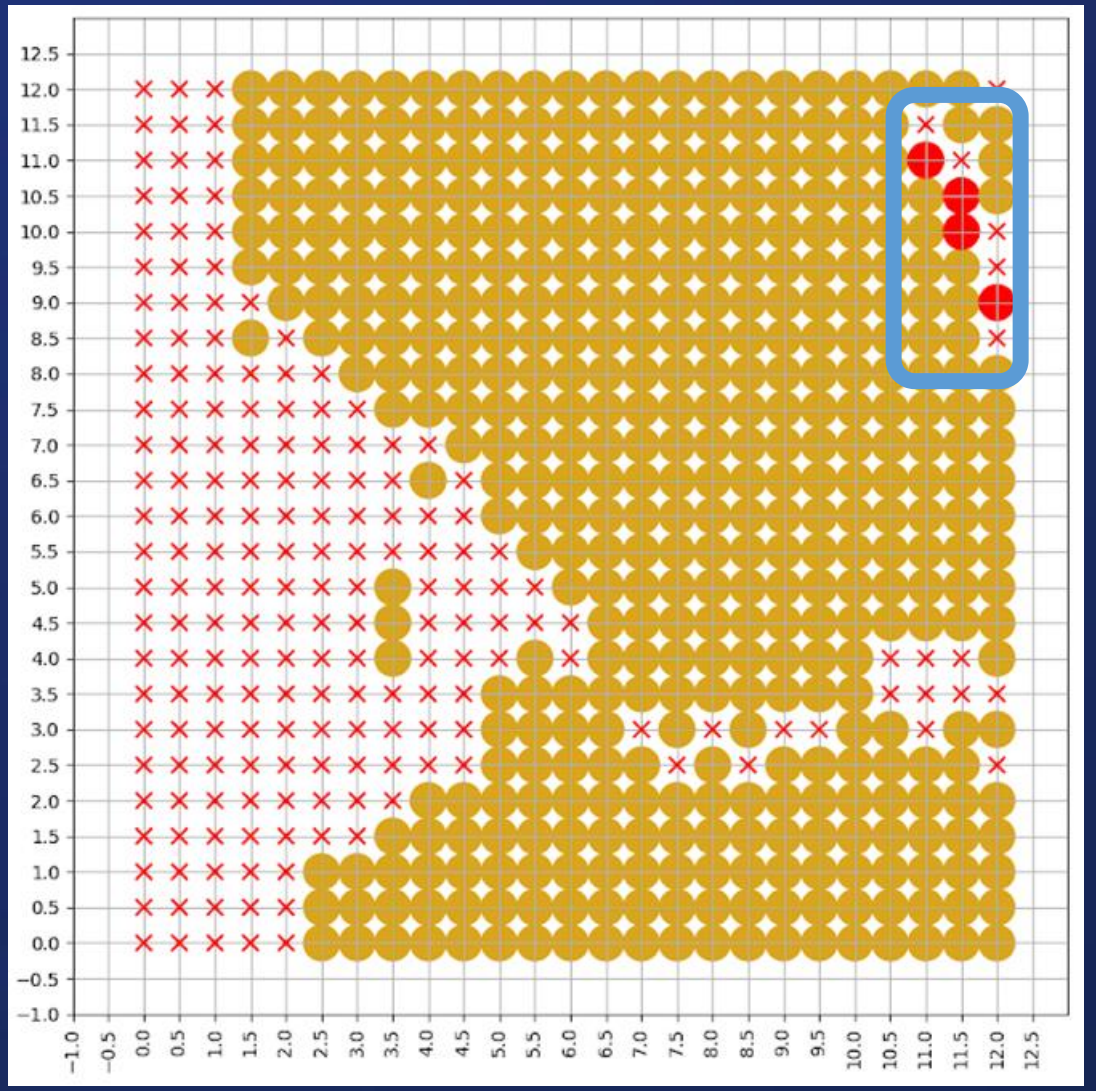


ZTE Cyber Security Lab

ZTE

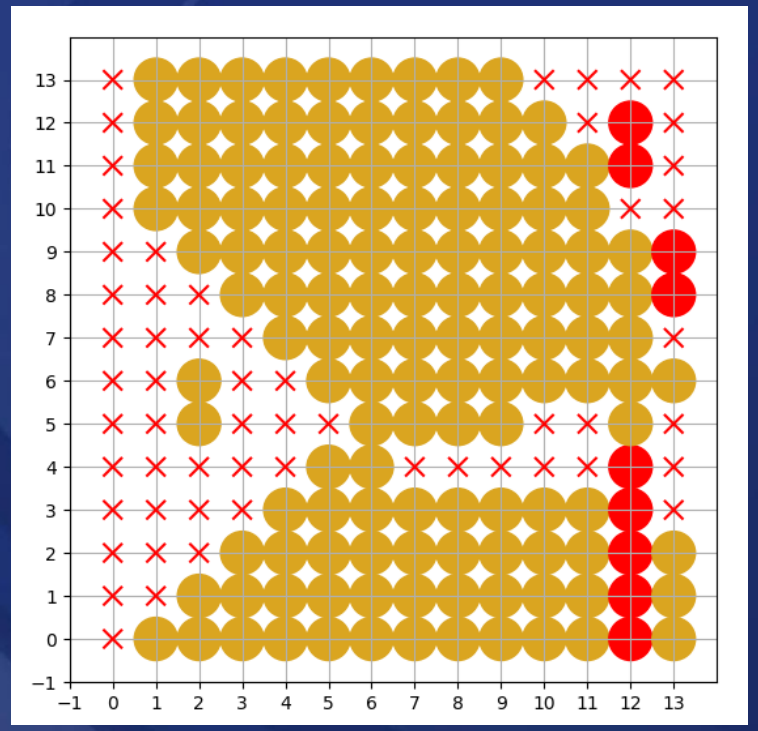


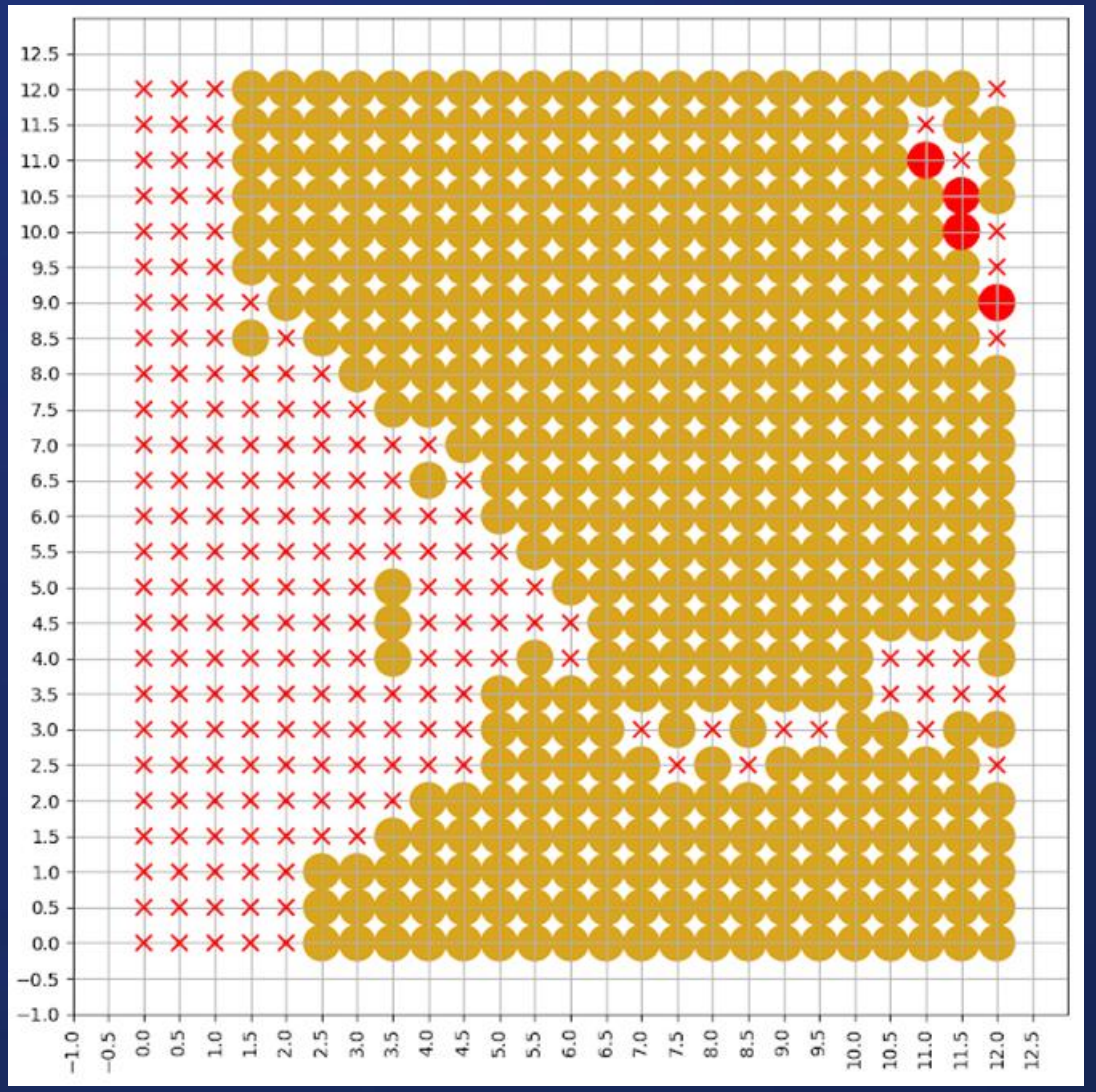




ZTE Cyber Security Lab

ZTE





ZTE Cyber Security Lab

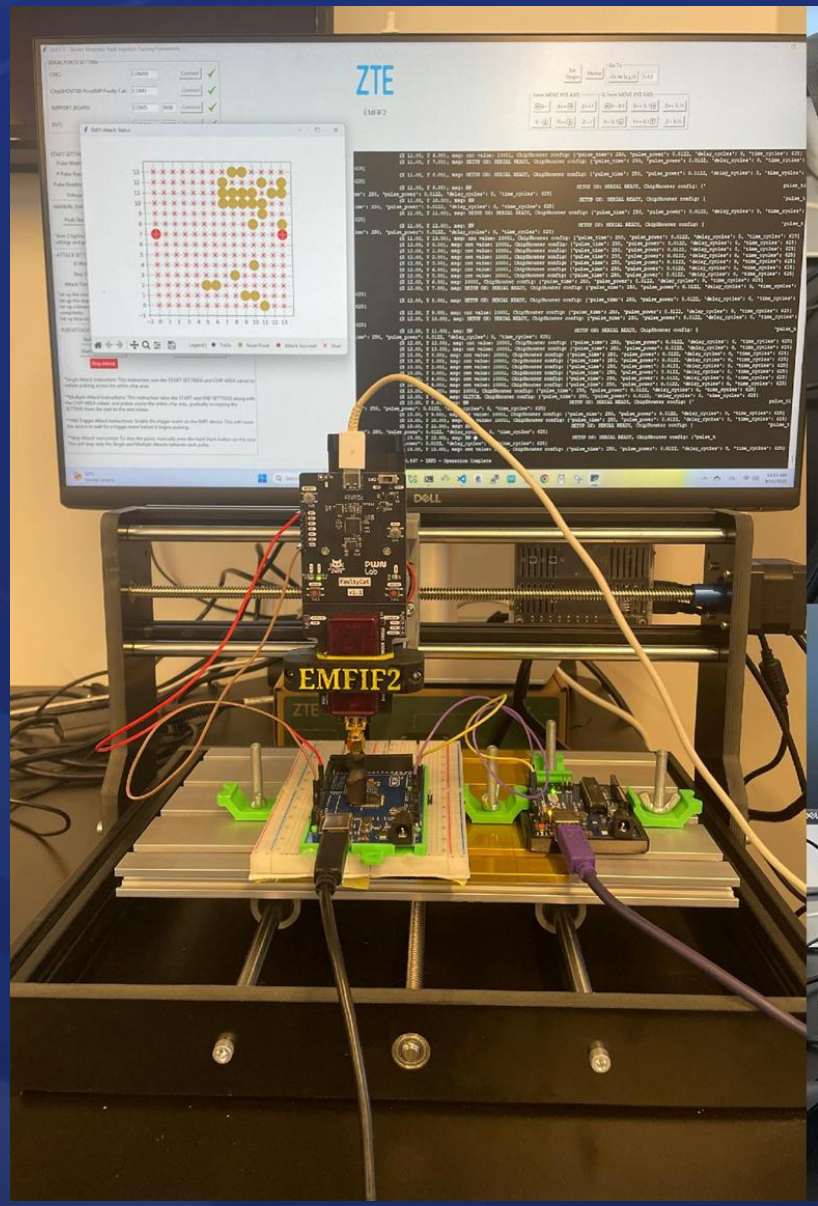
ZTE

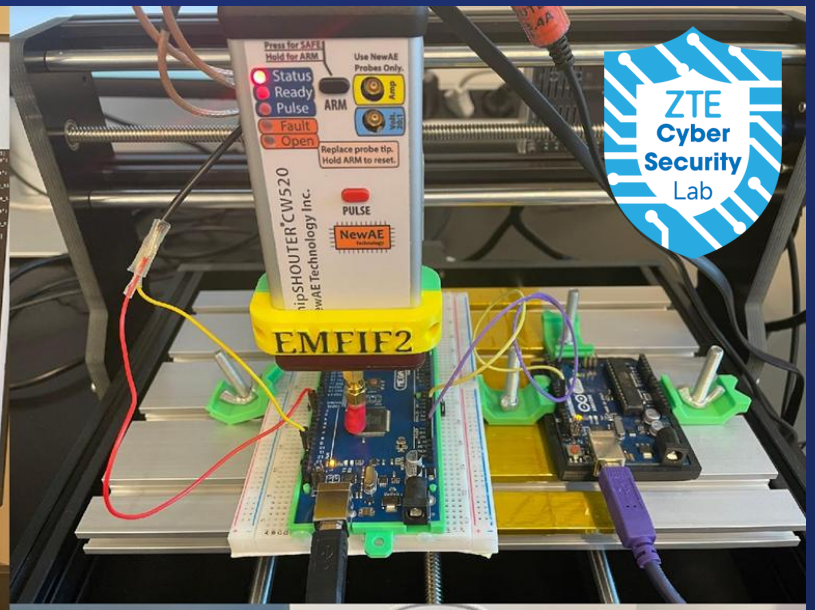
Our Journey Today

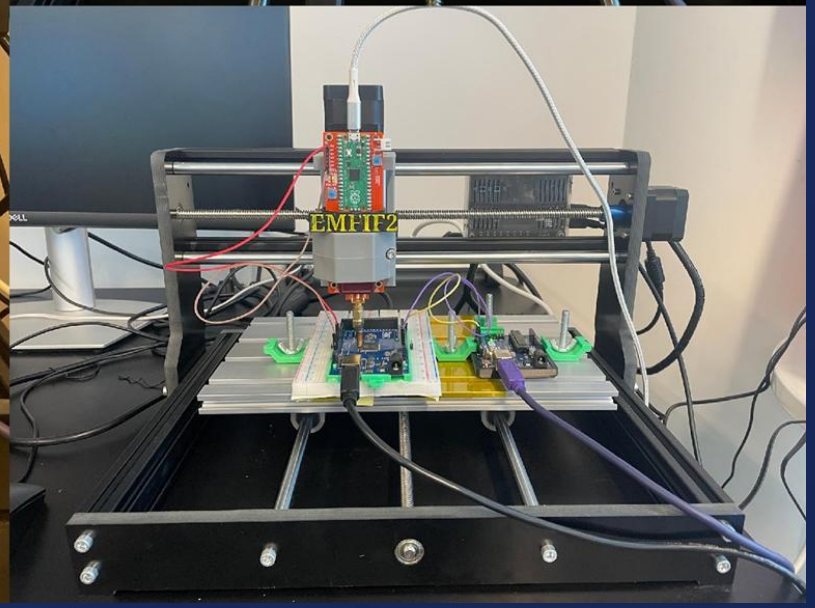
1. The Problem: The High Barrier to Entry
2. Our Solution: Introducing EMFIF2
3. Under the Hood: The Architecture
4. Visualizing Vulnerability: Heat Maps
5. **Demos & A Key Discovery**
6. Finding: Global vs. Local Variables
7. The Impact: What's Next?

ZTE

EMFIF2 Vs MEGA2560







EMFIF2 - Electro Magnetic Fault Injection Fuzzing Framework

SERIAL PORTS SETTING

CNC:	COM10		Connect ✓
ChipSHOUTER-PicoEMP/Faulty Cat:	COM20		Connect ✓
SUPPORT_BOARD:	COM5	9600	Connect ✓
DUT:	COM12	11520	Connect ✓

ZTE

EMFIF2

Set Origin | Home | Go To: Go to (x,y,z) 0,0,0

1mm MOVE XYZ AXIS: X-- | X++ | Z++↑ | Y-- | Y++ | Z--↓

0.1mm MOVE XYZ AXIS: X-- 0.1 | X++ 0.1 | Z++ 0.1↑ | Y-- 0.1 | Y++ 0.1 | Z-- 0.1↓

START SETTINGS / STEP / END SETTINGS

	START	STEP	END
Pulse Width (80-960 nS):	250	10	210
# Pulse Repeat (1-10000):			
Pulse Deadtime (1-1000 mS):			
Voltage (0.0122):	0.0122	0	0.0122

MANUAL EMFI DEVICE SINGLE USE

Push Start Setting and Arm | Disarm | Fire Pulse

*Arm ChipShouter and Fire the pulse. The configuration is taken from the start settings and pushed to the ChipShouter's EEPROM.

ATTACK SETTINGS

X-Width:	13	Y-Width:	13
Step Size:	1	Dut Reset Timeout(s):	2
Attack Timeout(s):	1	Hammer Attack Blows:	500

Set up the size (X) and length (Y) of the chip to attack.
Set up the step length the pin moves in injection (minimum 0.1).
Set up a timeout between one injection and the follow one to allow the DUT to reset completely.
Set up how much time wait between each injection.

RUN ATTACK

Start Single Attack | Start Multiple Automated Attacks | Start Hammer Attack | Start HW Trigger Attack | Stop Attack

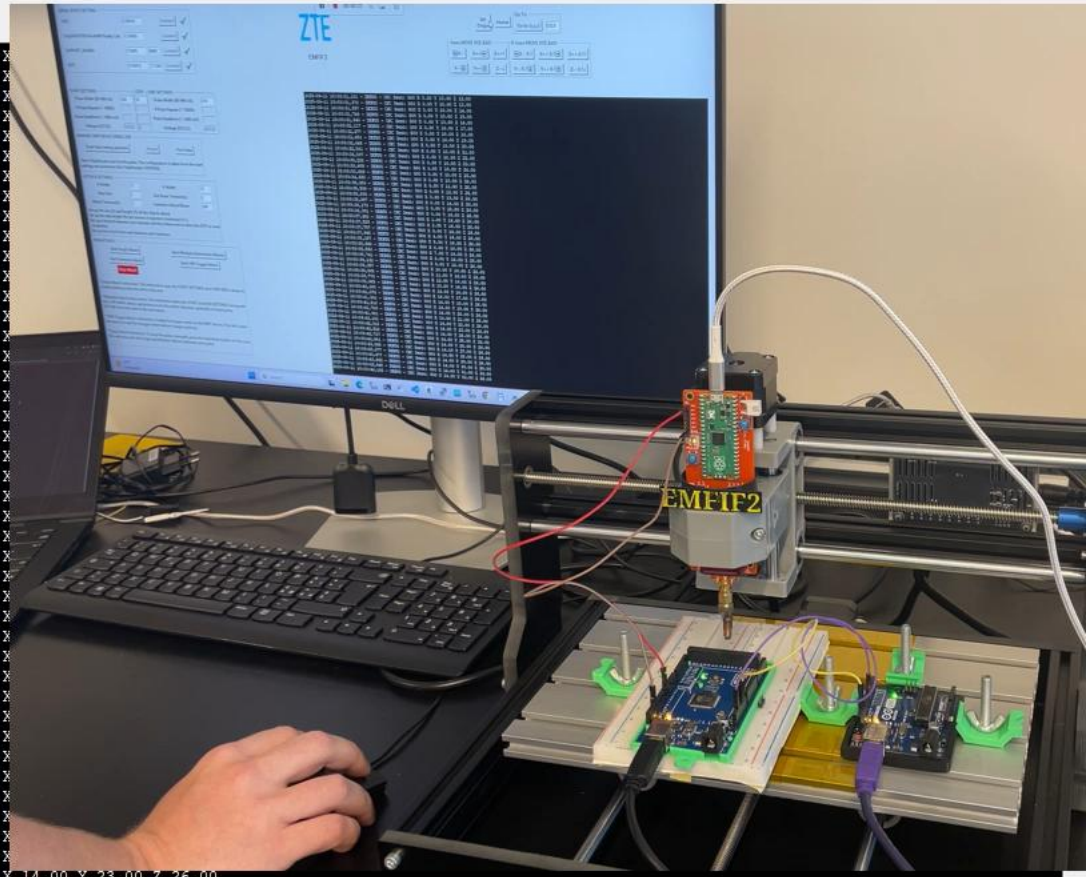
*Single Attack Instruction: This instruction uses the START SETTINGS and CHIP AREA values to initiate pulsing across the entire chip area.

**Multiple Attack Instructions: This instruction takes the START and END SETTINGS along with the CHIP AREA values, and pulses across the entire chip area, gradually increasing the SETTING from the start to the end values.

**HW Trigger Attack Instructions: Enable the trigger event on the EMFI device. This will cause the device to wait for a trigger event before it begins pulsing.

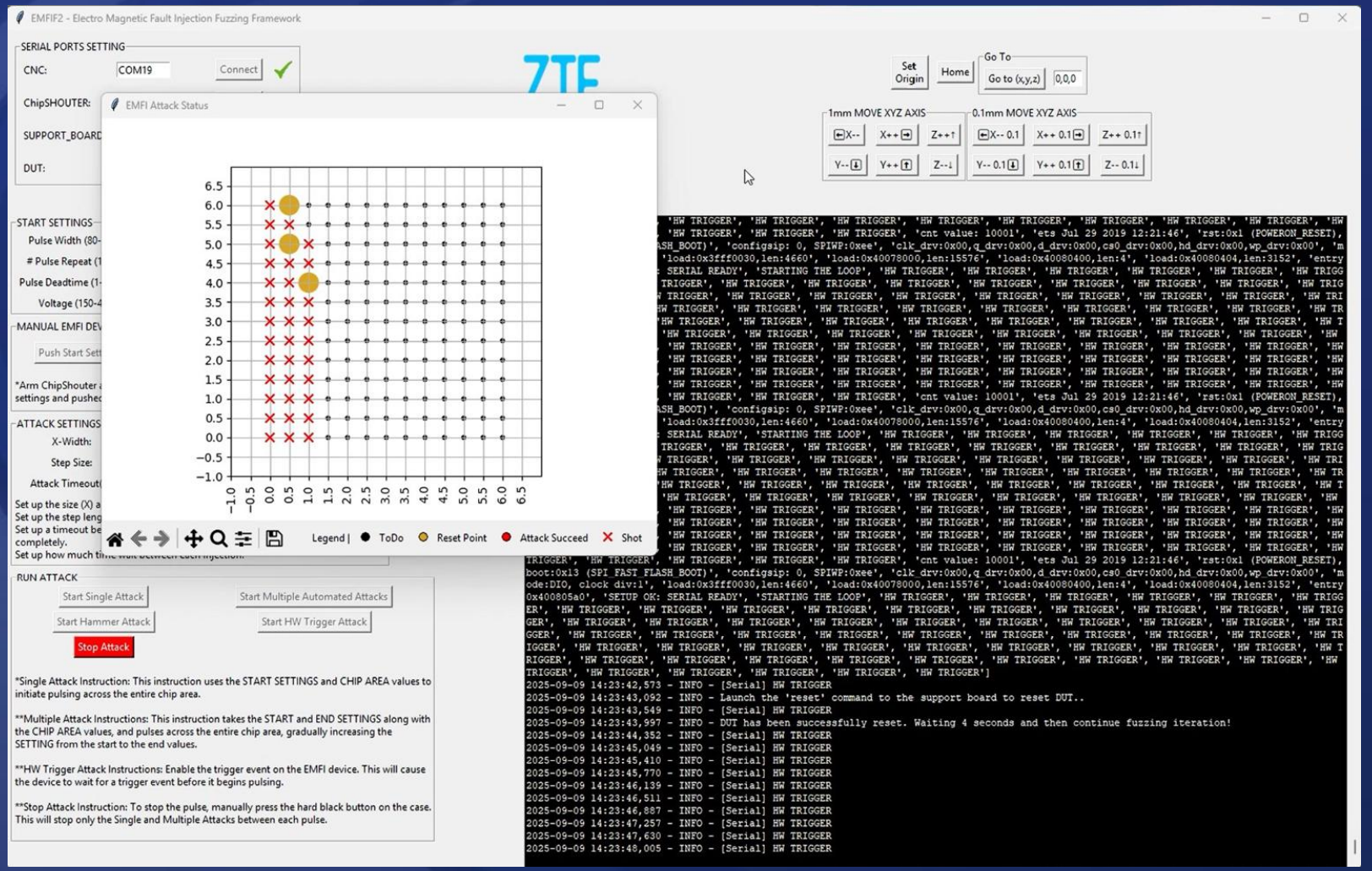
**Stop Attack Instruction: To stop the pulse, manually press the hard black button on the case. This will stop only the Single and Multiple Attacks between each pulse.

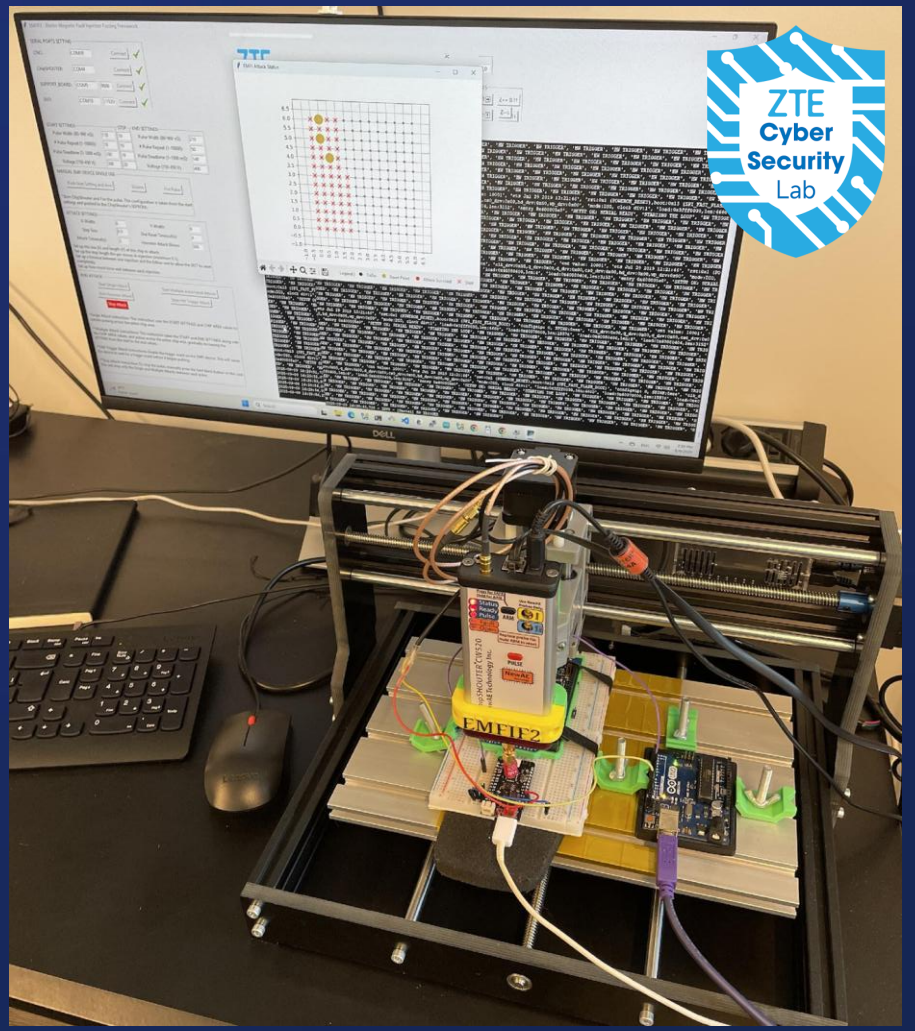
```
2025-09-11 10:03:31,181 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:31,373 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:31,557 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:31,749 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:31,940 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:32,117 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:32,277 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:32,453 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:32,645 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:32,861 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:33,037 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:33,228 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:33,405 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:33,669 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:34,660 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:35,389 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:35,589 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:35,805 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:35,997 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:36,173 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:36,381 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:37,261 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:37,525 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:37,757 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:37,948 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:38,157 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:38,349 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:38,556 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:38,741 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:39,132 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:39,357 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:39,565 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:39,773 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:39,965 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:40,180 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:41,076 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:41,301 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:41,516 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:41,709 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:41,901 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:42,101 - DEBUG - CNC Sent: G00 X
2025-09-11 10:03:42,316 - DEBUG - CNC Sent: G00 X 14.00 Y 23.00 Z 26.00
2025-09-11 10:03:42,524 - DEBUG - CNC Sent: G00 X 14.00 Y 24.00 Z 26.00
2025-09-11 10:03:42,717 - DEBUG - CNC Sent: G00 X 14.00 Y 25.00 Z 26.00
2025-09-11 10:03:42,901 - DEBUG - CNC Sent: G00 X 14.00 Y 26.00 Z 26.00
2025-09-11 10:03:43,085 - DEBUG - CNC Sent: G00 X 14.00 Y 27.00 Z 26.00
2025-09-11 10:03:43,277 - DEBUG - CNC Sent: G00 X 14.00 Y 28.00 Z 26.00
2025-09-11 10:03:43,477 - DEBUG - CNC Sent: G00 X 14.00 Y 29.00 Z 26.00
2025-09-11 10:03:43,692 - DEBUG - CNC Sent: G00 X 14.00 Y 30.00 Z 26.00
2025-09-11 10:03:43,948 - DEBUG - CNC Sent: G00 X 14.00 Y 31.00 Z 26.00
2025-09-11 10:03:44,165 - DEBUG - CNC Sent: G00 X 14.00 Y 32.00 Z 26.00
```



EMFIF2 + PicoEMP Vs MEGA2560

ZTE EMFIF2 Vs ESP32





ZTE

Our Journey Today

1. The Problem: The High Barrier to Entry
2. Our Solution: Introducing EMFIF2
3. Under the Hood: The Architecture
4. Visualizing Vulnerability: Heat Maps
5. Demos & A Key Discovery
6. **Finding: Global vs. Local Variables**
7. The Impact: What's Next?

ZTE

```
for (int i=0; i<10001; i++){
    // each 1000 iterations, trigger the EMFI Device
    if (i!=0 and i%1000 == 0) {
        Serial.println("HW TRIGGER");
        digitalWrite(6, HIGH);
        digitalWrite(6, LOW);
    }
    cnt++;
}

if (cnt != 10001){
    Serial.println("GLITCH");
}
```

ZTE Cyber Security Lab

ZTE

```
for (int i=0; i<10001; i++){
    // each 1000 iterations, trigger the EMFI Device
    if (i!=0 and i%1000 == 0) {
        Serial.println("HW TRIGGER");
        digitalWrite(6, HIGH);
        digitalWrite(6, LOW);
    }
    cnt++;
}

if (cnt != 10001){
    Serial.println("GLITCH");
}
```

ZTE Cyber Security Lab

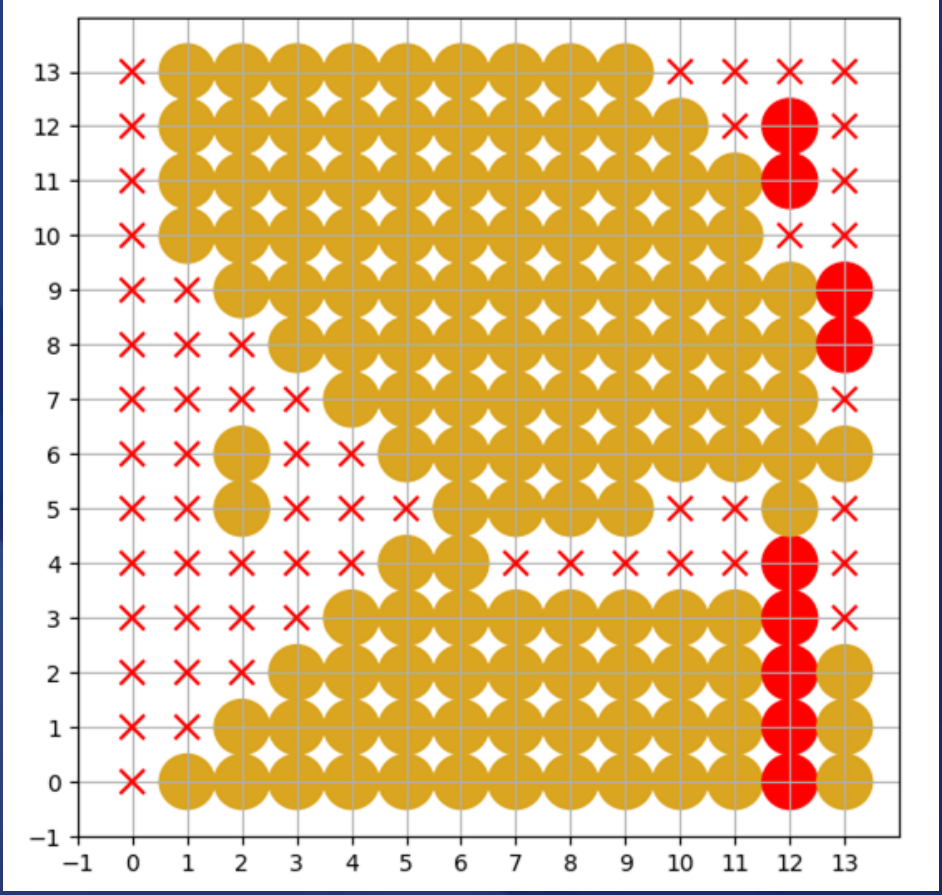
ZTE

```
int cnt = 0; // Global variable declaration
```



ZTE

```
int cnt = 0; // Global variable declaration
```



ZTE Cyber Security Lab

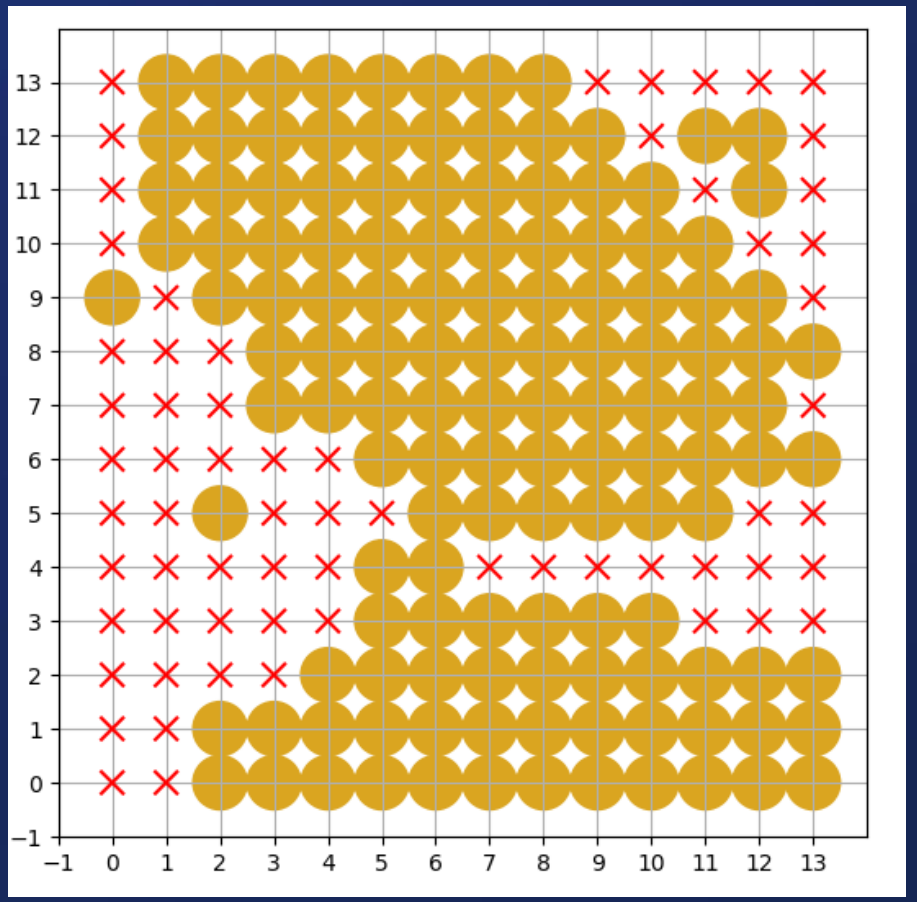
ZTE

```
void loop() {
   // ... serial handling code ...
   int cnt = 0; // Local variable declaration
   // ... counting loop ...
}
```



ZTE

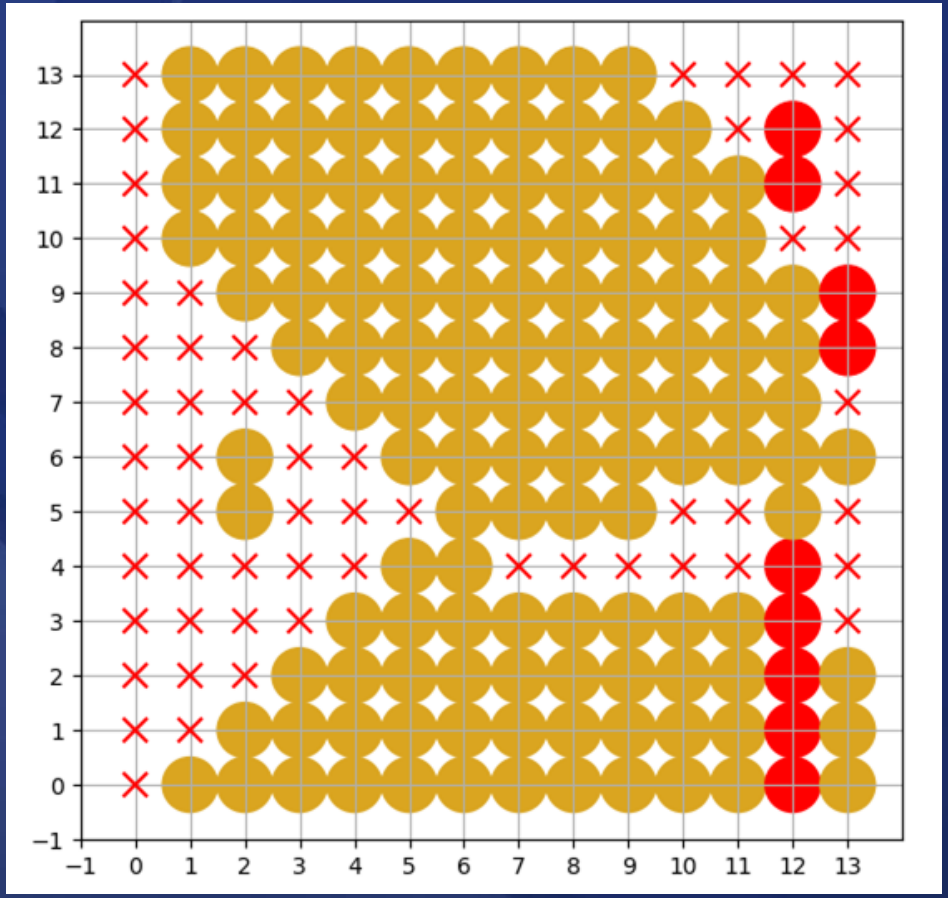
```
void loop() {
   // ... serial handling code ...
   int cnt = 0; // Local variable declaration
   // ... counting loop ...
}
```



ZTE Cyber Security Lab

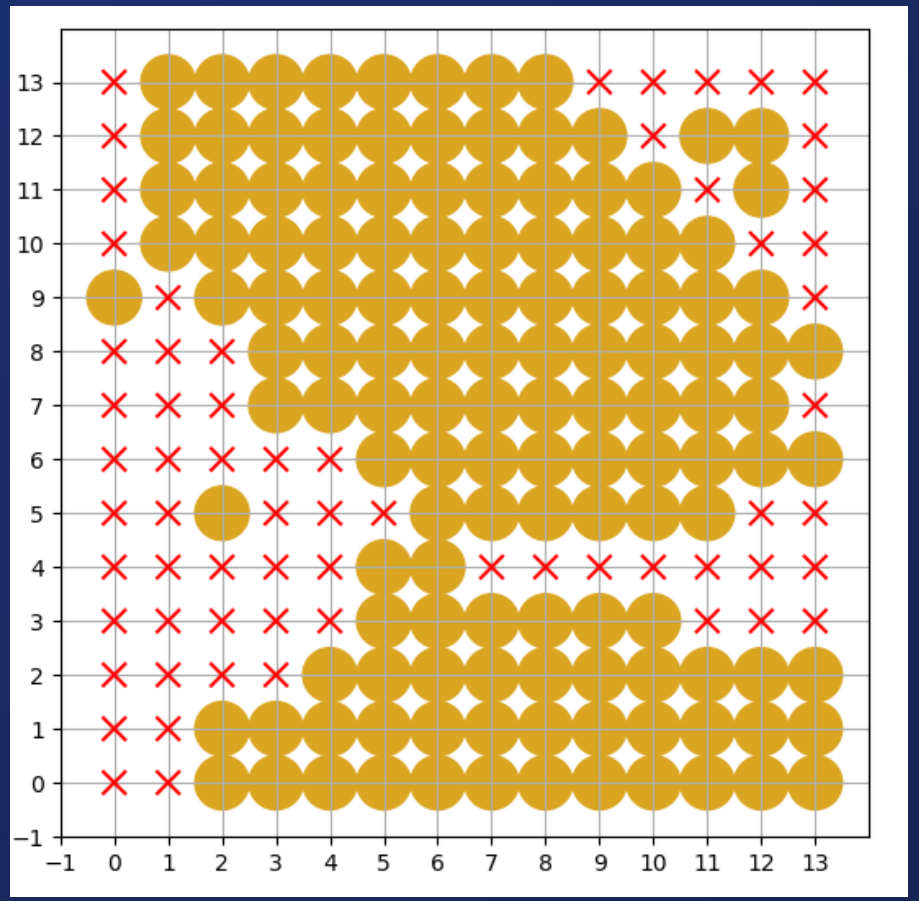
ZTE

```
int cnt = 0; // Global variable declaration
```



Vs

```
void loop() {
   // ... serial handling code ...
   int cnt = 0; // Local variable declaration
   // ... counting loop ...
}
```



ZTE Cyber Security Lab

ZTE

Our Journey Today

1. The Problem: The High Barrier to Entry
2. Our Solution: Introducing EMFIF2
3. Under the Hood: The Architecture
4. Visualizing Vulnerability: Heat Maps
5. Demos & A Key Discovery
6. Finding: Global vs. Local Variables
7. **The Impact: What's Next?**

ZTE

Customizable Glitch Detection Patterns

Current fault detection relies on predefined success criteria. The next enhancement will introduce a **flexible pattern detection system allowing researchers to define custom success indicators based on specific attack scenarios**. The Pattern-Based Detection Engine will implement real-time pattern matching for Device Under Test (DUT) output streams:

- **Text Pattern Matching**: Custom regular expressions and string patterns indicating successful fault injection (error messages, debug output, security bypass indicators)
- **Behavioral Signatures**: Sequential output patterns suggesting privilege escalation or authentication bypass
- **Multi-Modal Detection**: Concurrent monitoring of serial output, timing anomalies, and power consumption

Automated Hardware Trigger Attack Implementation

Current operations require manual parameter configuration for each test iteration. **Automated hardware trigger attacks** will enable systematic **exploration of the entire parameter space with minimal human intervention**.
The automated system will coordinate multi-dimensional attack campaigns:

- **Spatial Coverage**: Systematic exploration of entire chip surface area using configurable grid patterns
- **Parameters Sweeping**: Gradual escalation from START_SETTINGS to END_SETTINGS with configurable step sizes
- **Intelligent Optimization**: Adaptive strategies including grid search, gradient ascent, and Bayesian optimization







EMFIF2

Electro Magnetic Fault Injection Fuzzing Framework

github.com/ZTE-Cybersecurity-Lab/EMFIF2