

# INTERNSHIP REPORT

## Fault Injection Attacks

Under the supervision of:

**Dr. Faruk Kazi**

Mentor:

**Sushant Mane**

Aryan Bawankar - 211060052

Devang Shinde - 211060028

Vishwajeet Dhamal - 211060003



# Voltage Glitch Attack on Arduino Nano

## Description:

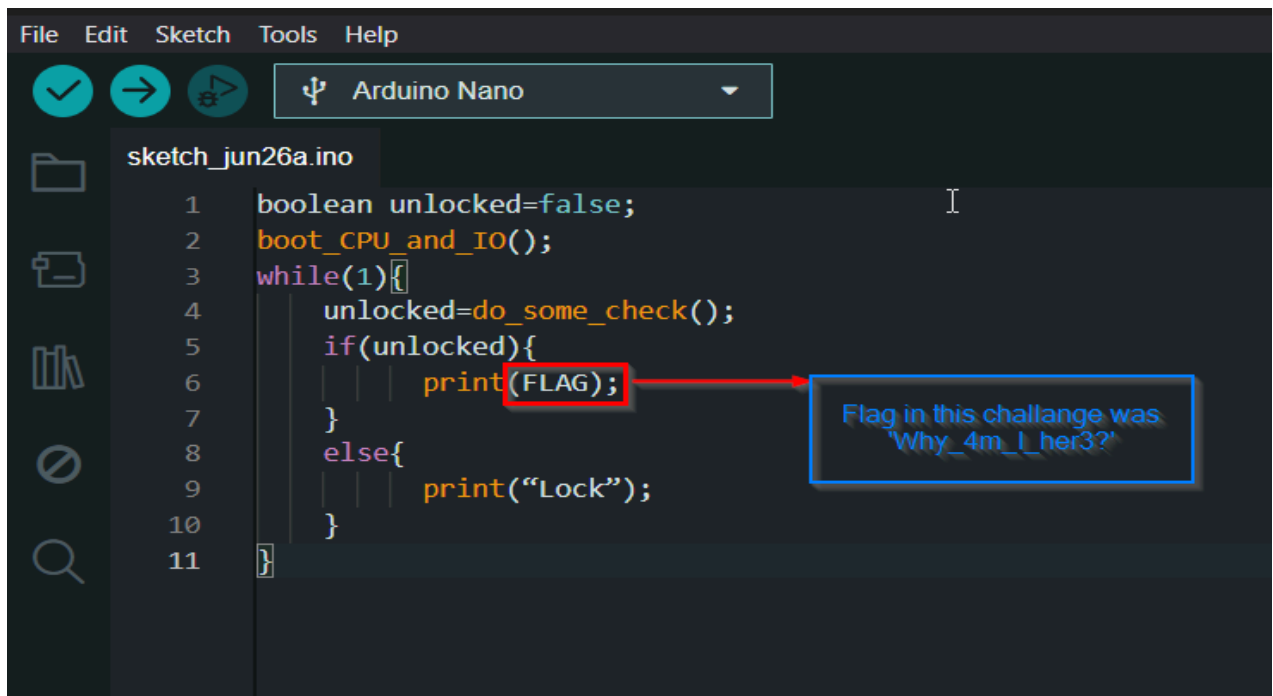
A voltage fault injection attack on an Arduino Nano using ChipWhisperer Lite involves manipulating the voltage supply to induce errors in the microcontroller's operation. By deliberately altering the voltage levels, attackers can disrupt normal execution, potentially causing unintended behaviors such as crashes, data corruption, or even security vulnerabilities. This method is used in security testing to probe the resilience of embedded systems against such attacks and to uncover potential weaknesses in their design or implementation.

By carefully timing voltage spikes or drops, they can bypass a continuous loop running on the Arduino, potentially gaining access to sensitive information such as the flag value.

The challenge (encrypted hex file to be flashed on target device: <https://github.com/Riscure/Rhme-2016/blob/master/challenges/binaries/fiesta/fiesta.hex>) is to try to capture a secret flag hidden in firmware running on Arduino, when we run this firmware we can see an infinite loop printing "LockLockLock..." we can't break an infinite loop using any serial commands or GPIO signals. Instead, we have to inject faults into the CPU.

This attack can be done either using clock glitch or voltage glitch, Usually both types of attacks have to be injected in tens to hundreds of nanoseconds, here clock frequency of target board is 16 Mhz so, we get  $1\text{clk} = 62.5\text{ns}$ .

An example of pseudocode on Arduino nano can be described in following format, which basically has an infinite while loop



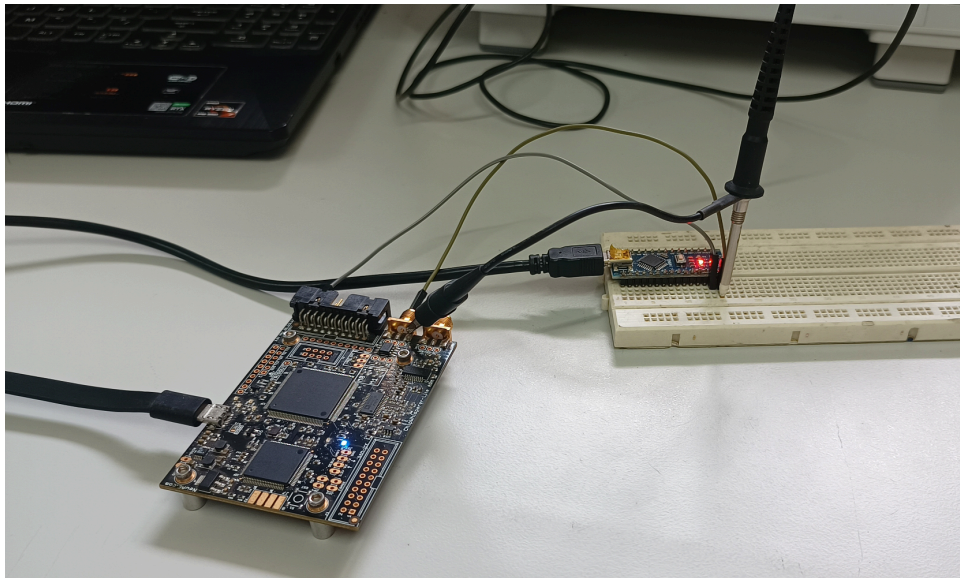
```
File Edit Sketch Tools Help
Arduino Nano
sketch_jun26a.ino
1 boolean unlocked=false;
2 boot_CPU_and_IO();
3 while(1){
4     unlocked=do_some_check();
5     if(unlocked){
6         print(FLAG);
7     }
8     else{
9         print("Lock");
10    }
11 }
```

Flag in this challenge was "Why\_4m\_I\_her3?"

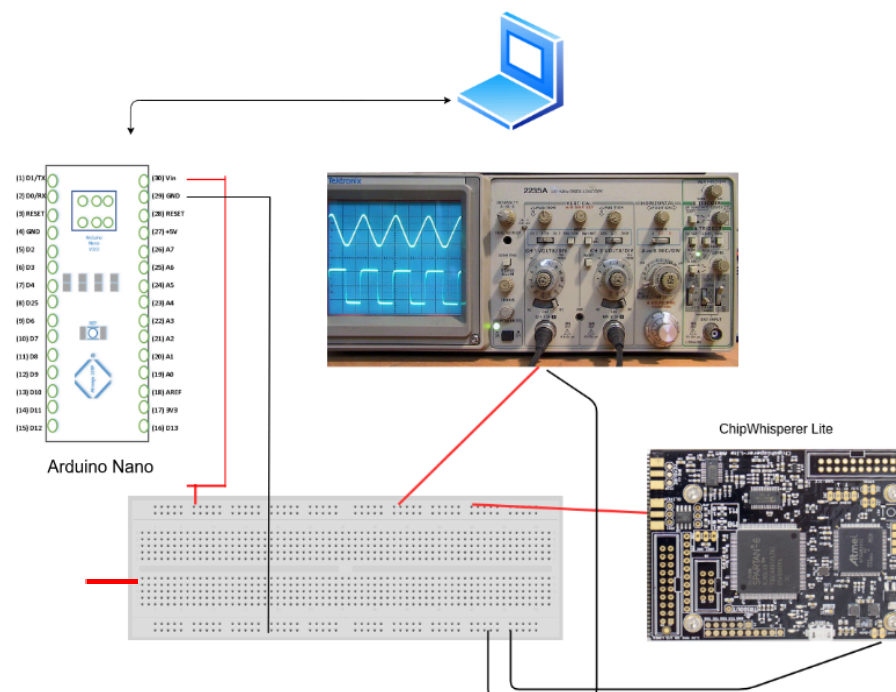
## Hardware Setup:

ChipWhisperer Lite, Arduino Nano, breadboard, jumper cables.

## Hardware Photo



## Schematics:



## Steps to recreate:

**Make connections according to the Schematics.**

**Importing Libraries:** `chipwhisperer` for hardware interaction and `time` for managing timing delays.

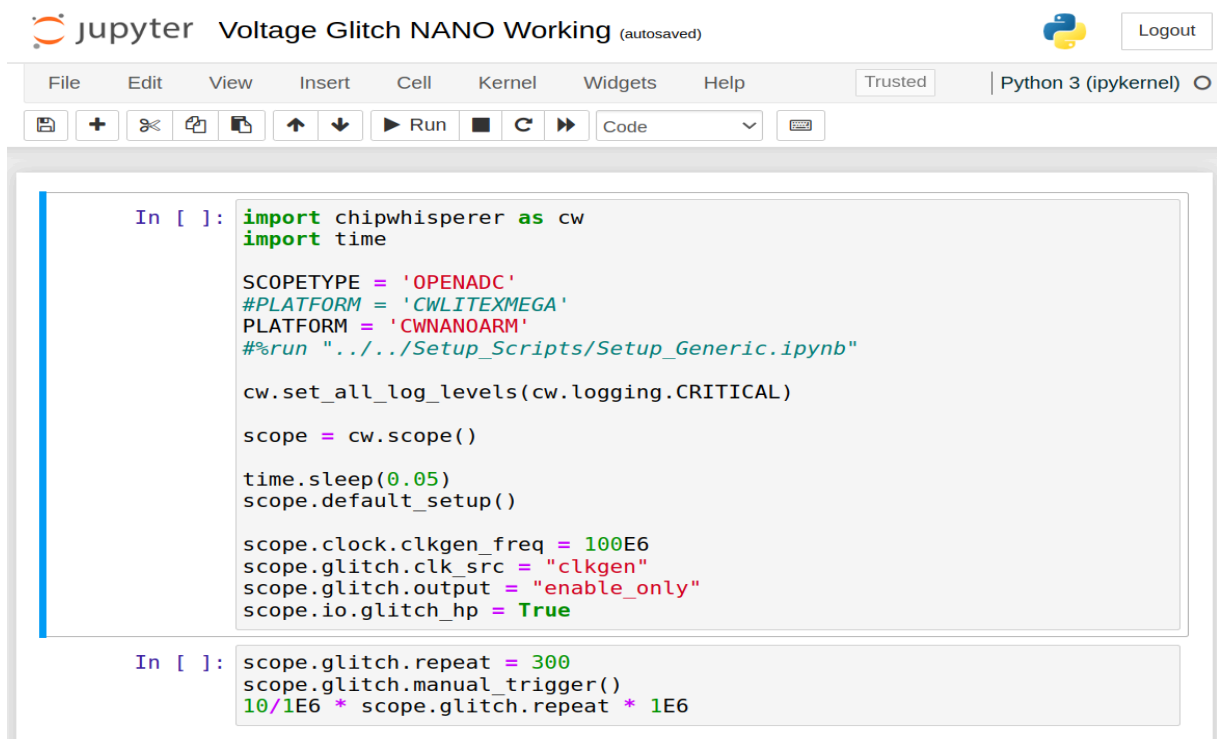
**Setting Up ChipWhisperer:** It initializes the ChipWhisperer platform for a specific target device (`CWNANOARM` in this case) using the `scope()` function.

**Configuration:** Various parameters are configured for the glitching attack:

- `scope.clock.clkgen_freq = 100E6`: Sets the clock frequency of the target to 100 MHz.
- `scope.glitch.clk_src = "clkgen"`
- `scope.glitch.output = "enable_only"`
- `scope.io.glitch_hp = True`: Enables glitch high-pass filtering to better target specific glitch pulses.
- `scope.glitch.repeat = 300`: Sets the number of glitch attempts to 300.

**Triggering the Pulse:** by `scope.glitch.manual_trigger()`

## Code:



```
In [ ]: import chipwhisperer as cw
import time

SCOPETYPE = 'OPENADC'
#PLATFORM = 'CWLITEXMEGA'
PLATFORM = 'CWNANOARM'
#%run "../Setup_Scripts/Setup_Generic.ipynb"

cw.set_all_log_levels(cw.logging.CRITICAL)

scope = cw.scope()

time.sleep(0.05)
scope.default_setup()

scope.clock.clkgen_freq = 100E6
scope.glitch.clk_src = "clkgen"
scope.glitch.output = "enable_only"
scope.io.glitch_hp = True

In [ ]: scope.glitch.repeat = 300
scope.glitch.manual_trigger()
10/1E6 * scope.glitch.repeat * 1E6
```

In this case, the glitch is repeated 300 times, we get  $1 \times 10^{-8} \times 300 = 0.000003 \text{ s} = 3 \mu\text{s}$ .

### Observation & Conclusion;

