Introduction to fault injection

E. Alata

January 12, 2022

When an attacker has physical access to a component, he can perform hardware attacks. In this tutorial, we propose to study a class of hardware attacks based on fault injection. This class of attacks consists of introducing a fault into the system to change its behavior. The objective of this lab is to understand how these attacks work and to propose appropriate security solutions. This lab is divided into several parts. In the first part, we will study a tool allowing to introduce a fault. In a second part, we will characterize, in the broad lines, the effect of this tool on a component. The third part is devoted to the injection of a fault with this tool to extract a secret. The last part proposes to study a second way to introduce a fault in a component.

1 Zapper – diagram

The tool studied in this section will be called a zapper in the following. Its diagram is shown in the figure 1. **This tool must be handled with care.**

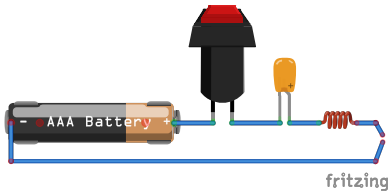

Figure 1: Zapper

S. 1 – Analyze the diagram of the zapper.

S. 2 – Identify, on the prototype, the different elements present on the diagram.

Q. 3 – What happens at the coil level?

Q. 4 – What happens if, during operation, this coil approaches an electronic component?

S. 5 – Deduce the operation of the circuit.

Q. 6 – What are the different behaviors that the zapper can produce?

2 Zapper – characterisation

We will characterize the effect of the zapper in two different ways: with an oscilloscope and by using an arduino.

S. 7 – Position the oscilloscope as shown in the figure 2, at a distance of 10 cm from the zapper.

S. 8 – Produce a pulse with the zapper (a single breakdown), close to the probe of the oscilloscope.

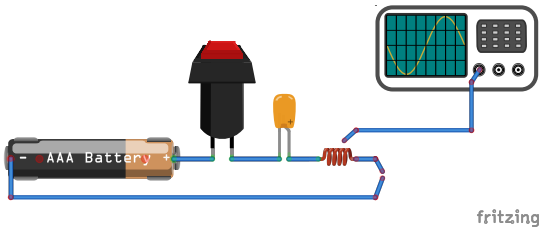

Figure 2: Zapper

S. 9 – Identify the total duration of this pulse and its shape, with the oscilloscope.

S. 10 – Identify, on the oscilloscope, the most significant part of the pulse.

S. 11 – Repeat the experiment by moving the zapper closer to the oscilloscope.

S. 12 – Get the operating frequency of an arduino.

S. 13 – Discuss the distance of the zapper with the arduino to cause the shortest possible fault.

In the following, we will use an arduino programmed with the following code:

```
int main (void) { cpu_init();

  // setup serial
  uart_init();

  // set pin 5 of PORTB for output
  DDRB |= _BV(DDB5);

  // set pin 5 high to turn led on */
  PORTB |= _BV(PORTB5);

  uart_putstr("\n\r");
  uint16_t a = 0;
  uint16_t b = 0;
  uint16_t c = 0;
  uint16_t d = 0;
  while(1) {
    a++; a++; a++; a++; a++; a++; a++; a++;
    a++; a++; a++; a++; a++; a++; a++; a++;
    a++; a++; a++; a++; a++; a++; a++; a++;
    a++; a++; a++; a++; a++; a++; a++; a++;

    b++; b++; b++; b++; b++; b++; b++; b++;
    b++; b++; b++; b++; b++; b++; b++; b++;
    b++; b++; b++; b++; b++; b++; b++; b++;
    b++; b++; b++; b++; b++; b++; b++; b++;

    c++; c++; c++; c++; c++; c++; c++; c++;
    c++; c++; c++; c++; c++; c++; c++; c++;
    c++; c++; c++; c++; c++; c++; c++; c++;
    c++; c++; c++; c++; c++; c++; c++; c++;

    d++; d++; d++; d++; d++; d++; d++; d++;
    d++; d++; d++; d++; d++; d++; d++; d++;
    d++; d++; d++; d++; d++; d++; d++; d++;
    d++; d++; d++; d++; d++; d++; d++; d++;

    if (a > 0xf000) {
      a = 0;
      b = 0;
      c = 0;
      d = 0;
      uart_putstr("x\r\n");
    }
    if (a != b || b != c || c != d) {
      uart_putint(a);
      uart_putchar(':');
      uart_putint(b);
      uart_putchar(':');
      uart_putint(c);
      uart_putchar(':');
      uart_putint(d);
      uart_putchar(':');
      uart_putchar('\n');
      uart_putchar('\r');
    }
  }
}
```

Q. 14 – When executing this code, what can you say about the values of variables a, b, c and d?

Q. 15 – If you produce an impulse during the execution, what can happen?

Q. 16 – Which parts of the code can be impacted and what can happen?

Connect the arduino to your computer and observe the output using the script `tty.sh`, that you must create:

```
#!/usr/bin/env bash
TTY=$1
stty -F $TTY min 0 cs8 9600 ignbrk -brkint -imaxbel -opost -onlcr -isig -icanon -iexten \
     -echo -echoe -echok -echoctl -echoke noflsh -ixon -crtscts
tail -f $TTY
```

This script expects a parameter corresponding to the `tty` to which the arduino is connected. To identify this `tty`, just execute the command `ls -lrt /dev`. The last file listed should correspond to the arduino. It should be named `/dev/ttyUSB0` or `/dev/ttyACM0`.

S. 17 – Really produce a pulse during the execution of the arduino and analyze the impact according to the position of the zapper in relation to the arduino.

S. 18 – Try to produce a difference of one unit between one of the variables and the others.

3 Zapper – attack

In this part, we propose you to unlock a lock by producing a pulse. This lock is controlled by an arduino, programmed with the following algorithm (**this is a purely fictitious example**):

```
while (1) {
  state = get_pin_states();
  hash = aes_hash(state);
  if (hash != LOCK_HASH) {
    print_secret();
  }
}
```

Q. 19 – Which part of the algorithm takes the longest to execute?

S. 20 – Try to evaluate the impact of a pulse on this algorithm.

S. 21 – Based on the analysis of the previous parts, try to extract the secret.

Q. 22 – To what extent can an attacker use this attack, what are the conditions?

Q. 23 – Are arduino based systems vulnerable?

Q. 24 – How to prevent this attack on these systems?

Q. 25 – Are systems based on critical components vulnerable?

4 Power glitch

In this section, we will control the power supply of the arduino in the previous part, using a malicious arduino. To do this, we have prepared the circuit shown in the figure 3.

S. 26 – Analyze this circuit.

S. 27 – Take a look at the arduino datasheet: datasheet.

S. 28 – Deduce which is the malicious arduino.

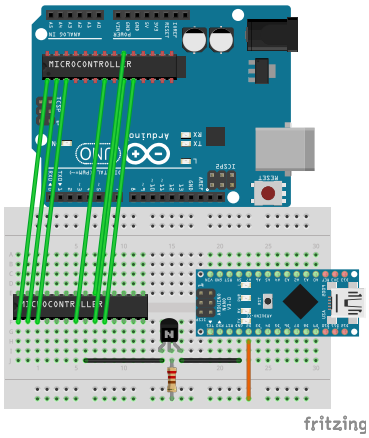

Figure 3: Glitch voltage

Q. 29 – What is the transistor used for?

Q. 30 – On which pins of the legitimate arduino is the malicious arduino connected?

Q. 31 – Why are not all the pins of the legitimate arduino connected?

The algorithm used by the malicious arduino is the following:

```
#define BAUDRATE 9600

int incomingByte = 0;
char b[5];

int powerPin = 2;
int glitchDelay = 0;

void setup() {
  Serial.begin(BAUDRATE);
  Serial.println("Arduino is ready");

  pinMode(powerPin, OUTPUT);

  digitalWrite(powerPin, HIGH);
  delay(5000);

  Serial.println("Gliching is ready");
}

void glitch(){
  int waste = 0;

  digitalWrite(powerPin, LOW);
  for (int i = 0; i<glitchDelay; i++){
    waste++; }
  digitalWrite(powerPin, HIGH);

  glitchDelay +=200;
  Serial.println();
  Serial.print("Glich Delay set to: ");
  Serial.print(glitchDelay);
  Serial.println();
}

void loop() {
  // put your main code here, to run
    repeatedly:

  for (int i = 0; i<200;i++){
      if (Serial.available() > 0) {
        // read the incoming byte:
        incomingByte = Serial.read();
        Serial.print(char(incomingByte));
      }
    }

    delay(1000);
    glitch();
}
```

S. 32 – Identify the part of the code that controls the transistor.

Q. 33 – What happens if this control pin is set to 0?

Q. 34 – What happens if this control pin is set to 1?

Q. 35 – How long does it take to cut the power supply of the legitimate arduino via this control?

Q. 36 – Is this duration constant?

Q. 37 – Why does it vary?

Q. 38 – What can be the impact of a cut of the power supply?

S. 39 – Try the attack.

Q. 40 – What does an attacker need to do in order to carry out this attack in practice?

Q. 41 – How can we protect ourselves from this attack?

S. 42 – Read this research article and identify what this attack did 1903.08102.pdf