# CWE Detail – CWE-1319

## Description

The device is susceptible to electromagnetic fault injection attacks, causing device internal information to be compromised or security mechanisms to be bypassed.

## Extended Description

Electromagnetic fault injection may allow an attacker to locally and dynamically modify the signals (both internal and external) of an integrated circuit. EM-FI attacks consist of producing a local, transient magnetic field near the device, inducing current in the device wires. A typical EMFI setup is made up of a pulse injection circuit that generates a high current transient in an EMI coil, producing an abrupt magnetic pulse which couples to the target producing faults in the device, which can lead to: Bypassing security mechanisms such as secure JTAG or Secure Boot Leaking device information Modifying program flow Perturbing secure hardware modules (e.g. random number generators)

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2020-27211: Chain: microcontroller system-on-chip uses a register value stored in flash to set product protection state on the memory bus and does not contain protection against fault injection (CWE-1319) which leads to an incorrect initialization of the memory bus (CWE-1419) causing the product to be in an unprotected state.

## Related Attack Patterns (CAPEC)

* CAPEC-624
* CAPEC-625

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: N/A

## Common Consequences

**•** Impact: Modify Memory, Read Memory, Gain Privileges or Assume Identity, Bypass Protection Mechanism, Execute Unauthorized Code or Commands — Notes:

## Potential Mitigations

**•** Architecture and Design: 1. Redundancy - By replicating critical operations and comparing the two outputs can help indicate whether a fault has been injected. 2. Error detection and correction codes - Gay, Mael, et al. proposed a new scheme that not only detects faults injected by a malicious adversary but also automatically corrects single nibble/byte errors introduced by low-multiplicity faults. 3. Fail by default coding - When checking conditions (switch or if) check all possible cases and fail by default because the default case in a switch (or the else part of a cascaded if-else-if construct) is used for dealing with the last possible (and valid) value without checking. This is prone to fault injection because this alternative is easily selected as a result of potential data manipulation [REF-1141]. 4. Random Behavior - adding random delays before critical operations, so that timing is not predictable. 5. Program Flow Integrity Protection - The program flow can be secured by integrating run-time checking aiming at detecting control flow inconsistencies. One such example is tagging the source code to indicate the points not to be bypassed [REF-1147]. 6. Sensors - Usage of sensors can detect variations in voltage and current. 7. Shields - physical barriers to protect the chips from malicious manipulation. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** Colin O'Flynn has demonstrated an attack scenario which uses electro-magnetic glitching during booting to bypass security and gain read access to flash, read and erase access to shadow memory area (where the private password is stored). Most devices in the MPC55xx and MPC56xx series that include the Boot Assist Module (BAM) (a serial or CAN bootloader mode) are susceptible to this attack. In this paper, a GM ECU was used as a real life target. While the success rate appears low (less than 2 percent), in practice a success can be found within 1-5 minutes once the EMFI tool is setup. In a practical scenario, the author showed that success can be achieved within 30-60 minutes from a cold start.

## Notes

**•** Maintenance: This entry is attack-oriented and may require significant modification in future versions, or even deprecation. It is not clear whether there is really a design "mistake" that enables such attacks, so this is not necessarily a weakness and may be more appropriate for CAPEC.