# EMFI for Safety-Critical Testing of Automotive Systems[*]

Colin O'Flynn

NewAE Technology Inc, Halifax, Canada
colin@oflynn.com

**Abstract.** Electromagnetic Fault Injection (EMFI) is a well known method of introducing faults for security analysis of digital devices. Such faults can be seen as analogous to the faults which are known to naturally occur in digital devices, a known problem with designing safety-critical systems.
Numerous standards have been developed for safety-critical systems, including the development of standards for increasing the rate of naturally occurring faults using particle sources. In this work, we demonstrate that desktop EMFI tooling can be used to accomplish similar testing, but with more control, effectively speeding up the evaluation process.
We demonstrate that using EMFI tooling for safety evaluation allows us to recreate a highly publicized safety issue present in an automotive ECU – one that could not easily be recreated previously with other techniques.

**Keywords:** electromagnetic fault injection · safety testing · security evaluation

## 1 Introduction

Fault injection allows an attacker to modify the operation of a device under test. These attacks can include simple control-flow changes allowing the bypass of a secure boot process, along with differential fault analysis attacks that allow recovery of secret cryptographic material [4][5].

In order to understand the attacker capabilities, a defender typically assumes some *fault model*, which shows what an attacker is capable of performing. These may be simple – such as assuming an attacker is capable of an "instruction skip", or may be more complicated, such as an attacker being able to flip a single or multiple bits. This may include targeting specific bits as part of a control flow hijack [14] or other advanced attack.

These faults are introduced by various methods [2] – manipulating the external clock or voltage supply is a simple method of introducing faults, but other methods exist, including optical faults via flash tubes or lasers, or electromagnetic faults[13].

[*] This is an authors copy of the paper presented at The 2021 Fault Diagnosis and Tolerance in Cryptography (FDTC) Workshop, and is identical in content to the IEEE Xplore version, but with larger images.

These attacker models typically assume that the attacker is triggering the fault at a specific instance in time. This capability of triggering a fault at a specific moment in time defines the difference between faults with a specific security implication and those that might occur naturally due to failures or errors in the computer system.

Random faults in computer systems are a well-known problem, with solutions such as error correcting code memory historically being used in systems which require better long-term stability. With the long history of safety-critical design processes, we can explore the fault models used by safety-critical systems to better understand where safety and security fault models overlap.

This work will particularly look at a well-known example of an automotive product which faced concern around safety-critical design failures. Despite considerable efforts, this failure has previously been only partially captured with classic safety-critical design evaluation tools. By applying tools typically used in security analysis, we will demonstrate how this failure can be reproduced.

The specific contributions of this work are:

1. Linking fault injection techniques commonly used in security evaluation to automotive safety testing.
2. Using EMFI for static corruption of SRAM in a similar manner to known industry standards.
3. Recreation of a safety-critical bug using EMFI in a "black-box" environment.

The paper will start with an introduction to standards used in safety-critical systems in Section 2, including a discussion of how those have previously been reported in similar security-focused papers. Based on previous work in both the security and safety domains, a specific link will be shown to commonly used metrics for evaluation of SRAM memories for corruption will be shown in Section 3. We will then demonstrate an "attack" on a production ECU to recreate the safety-critical bug in Section 4, before discussing conclusions and future opportunities in Section 5.

## 2 ISO 26262-11 Fault and Failure Modes

Safety-critical design of digital systems is well known in many industrial systems. This paper focuses on automotive systems, and in particular the ISO 26262 series of standards that is an adaption of IEC 61508 to automotive systems.

ISO 26262 contains several parts (each of which is a separate purchase from ISO). Of particular interest is Part 11 ("Guidelines on application of ISO 26262 to semiconductors"), Part 5 ("Product development at the hardware level"), and Part 6 ("Product development at the software level").

ISO 26262-11 defines example fault and failure mode for various types of digital devices. Three of these have direct parallels to security fault models: "Fault models of non-memory digital components (5.1.2)", "Detailed fault models of memories (5.1.3)", and "Failure modes of digital components (5.1.4)". The following information in the remainder of this section is defined in ISO 26262-11, but is recreated here for quick reference.

26262-11 Section 5.1.2 defines transient faults for non-memory digital components as in Table 1. From this definition we can see the Single Event Transient (SET) causes various types of upsets depending on the structure the transient interacts with. Note that this definition for non-memory digital components may still include storage such as a register in a CPU, but later in 26262-11 Section 5.1.3 stand-alone memory devices (such as FLASH or SRAM memories) are covered with other fault modes such as "stuck-at-0" faults.

Transient faults in memory such as SRAM are well-known to follow the general format of Table 1. The one major addition is that a Single Event Latchup (SEL) fault model, which is normally detected by a constant higher current consumption that lasts until the target device is power cycled. Such evaluations from a "safety" perspective have been performed on different devices using laser fault injection techniques[12], including work on newer devices such as Kintex-7 FPGAs[10].

Table 1: ISO 26262-11 Fault Modes

| **FMx** | **Example** |
|---|---|
| Single Event Transient SET | A momentary voltage excursion (e.g. a voltage spike) at a node in an integrated circuit caused by the passage of a single energetic particle. |
| Single Event Upset SEU | A soft error caused by the signal induced by the passage of a single energetic particle. |
| Single Bit Upset SBU | A single storage location upset from a single event. |
| Multiple Cell Upset MCU | A single event that induces several bits in an IC to fail at the same time. The error bits are usually, but not always, physically adjacent |
| Multiple Bit Upset MBU | Two or more single-event-induced bit errors occurring in the same nibble, byte, or word. |

A distinction is made between the fault mode and failure mode in these safety standards. The given fault may cause one of the failure modes (FM) listed in Table 2. These apply to general digital devices, which perform a given *function*. This *function* changes depending on the device, and area of device under consideration. A CPU could have an overall function of executing an instruction, but specific functions such as the interrupt handler or internal memory access functions. A failure of a given function using the failure modes from Table 2 gives us function-specific failures.

Table 2: ISO 26262-11 Failure Modes

| **FMx** | **Failure Mode** | **Example** |
|---|---|---|
| FM1 | Omission | Function not delivered when needed |
| FM2 | Commission | Function executed when not needed |
| FM3 | Timing | Function delivered with incorrect timing |
| FM4 | Value | Function provides incorrect output |

As an example, a CPU would have the function of "Execute given instruction flow according to given Instruction Set Architecture." Taking the failure modes from Table 2, the failures from Table 3 could occur.

Table 3: Failure Modes applied to CPU Instruction Flow

| **FMx** | **Result** |
|---|---|
| FM1 | Given instruction flow(s) not executed (total omission) |
| FM1.1 | .. due to program counter hang up |
| FMl.2 | .. due to instruction fetch hang up |
| FM2 | Un-intended instruction(s) flow executed |
| FM3 | Incorrect instruction flow timing (too early /late) |
| FM4 | Incorrect instruction flow result |

These failures follow well-known instruction fault models typically found in security-oriented fault injection. The typical 'instruction skip' fault model, for example, is covered in FM1, but could also be part of FM2 and FM3. Taking the incorrect branch could be seen as FM4, and an instruction mutation is covered by FM2.

This mapping is particularly useful when comparing tools and techniques for developers working with safety tooling. While the fact that security fault injection has specific timing means safety-critical systems cannot be assumed to be robust against a motivated attacker [16], this demonstrates that the fundamental fault models themselves are well aligned. We will now investigate how tooling typically targeting security analysis can be used as part of safety critical failure emulation.

## 3 Electromagnetic Fault Injection

The objective of electromagnetic fault injection (EMFI) is to ultimately inject a voltage onto the structure of the die itself – this can cause both "soft-error" faults (such as bit flips in a register or SRAM), or temporary errors in reading voltage levels (a SET). The fact that a strong field has the ability to corrupt data (without damaging the device) has been known since at least 1970[11].

While EMFI is routinely used for security testing purposes, the process of fault injection in safety testing more commonly follows the JESD89A standard (a reference for testing a device against “soft errors”), which uses an alpha particle source to accelerate the rate of soft errors [6]. As these alpha particle sources do not easily fit on a desktop, using standard EMFI tooling to achieve similar results would be valuable in practice. We will next demonstrate that static RAM memory corruption can be recreated with desktop EMFI equipment.

### 3.1 Memory Corruption with EMFI

Previous work on EMFI has suggested that errors occur only during transfers, and static corruptions were not observed [8]. This would be an important difference, as most safety-critical testing for memory devices assumes a charged particle is causing single or multi bit failures. To help validate the link between safety and security testing, we will demonstrate that static bit flips are possible using EMFI, and this aligns with assumptions widely used as part of safety-critical testing.

The testing will be done using an off-the-shelf target board called the Ballistic Gel, with details of the board available[1]. This target has a large 32 Mbit SRAM chip, which can be used to understand how bit flips vary with different settings. By downloading a pattern to the device, injecting a fault, and seeing where the pattern flipped bits we can get an idea of the sort of effects that are possible. The physical board is shown mounted on a test platform in Figure 1.

When performing EMFI injection, we have several characteristics we can adjust. The first is the design of the injector itself, for which protocols designed to help compare across devices have been proposed [15]. In this work we will use a single EMFI platform, the ChipSHOUTER. We will compare the effect of changing the specifics of the injection tip (coil), along with the charge voltage.

### 3.2 Affect of Injection Tip Dimensions and Voltages

The included injection tips vary with the size of the ferrite core, along with the direction of the winding around the core: two 4mm tips and two 1mm tips, each with both “clockwise” (CW) and “counter-clockwise” (CCW) versions. It is expected that swapping the positive and negative pulse polarity is equivalent to changing the winding direction. However, for safety reasons changing the winding direction ensures the highest voltage is not at the most exposed end of the winding tip. Examples of the various coils is shown in Figure 2.

Comparing the effect of a changing charge voltage shows an obvious link between pulse voltage and number of bytes corrupted. Increasing the charge voltage increases the number of bytes flipped.

[1] Schematic and other details are posted at https://github.com/newaetech/ChipSHOUTER-ballisticgel

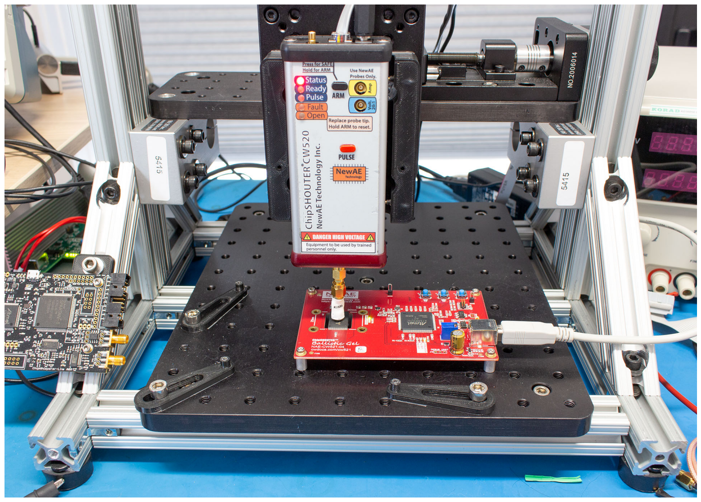


Fig. 1: Ballistic Gel mounted on XYZ table with EMFI Tool

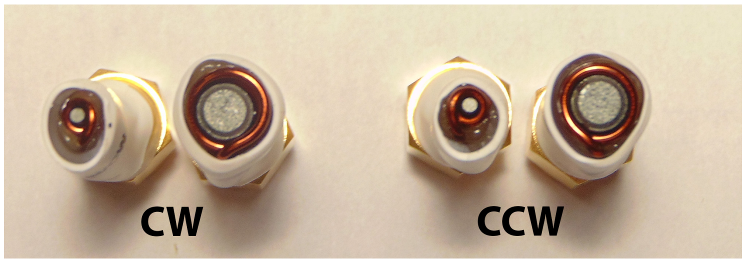


Fig. 2: Showing the wire wrapping direction around the injection tip (coil).

With the 4mm probe tip, the clockwise (CW) and counter clockwise (CCW) winding direction does not make a notable difference in the number of bytes corrupted per fault injection attempt. This may result in some biasing of bit-set or bit-reset faults, but typically for safety testing this is less of a concern.

The 1mm cores appear to have different results than the 4mm core. Here it appears that the CW and CCW tips have a different response. Most likely, this is due to physical construction tolerance – the tips themselves are slightly different lengths, making them different heights above the chip surface. Note this does show that a small number of byte errors are possible. The 1mm CW tip in this configuration showed 1 – 10 byte errors for charge voltages below 280V.

## 3.3 Effect of Height

The physical height of the injection tip would also be expected to affect the number of bit flips. In the following example we will keep the charge voltage fixed at 200 V, and use the 1 mm CCW tip. The tip will be raised off the surface

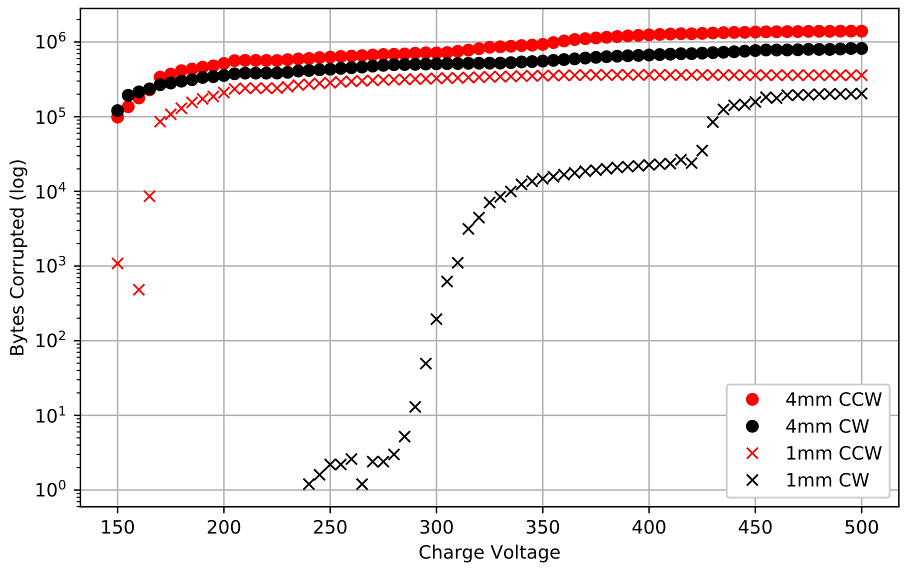


Fig. 3: Comparison of charge voltage and coils

of the chip, and the fault repeated 100 times. This allows us to evaluate the repeatability of height changes. The results are shown in Figure 4.

Moving the probe off the surface showed a reduction in faults, but as demonstrated in this diagram, there is still considerable randomness to this process. It suggests to be biased towards zero as the probe moves away, but with occasional large number of flips for the same position.

## 3.4 XY Scanning Location

Not only does scanning the EMFI location over the chip surface affect where the faults are injected, it can affect the number due to different features in the device. As a simple example, scanning our same fault over the SRAM top gives us the results from Figure 5. Note that multiple areas have a 1-byte fault, some areas have more than 1-byte fault (but not the large number of faults seen previously), and other areas have the large number of faults.

This demonstrates that there are several different variables that affect the types of faults generated by an EMFI platform. This can be used to target different types of faults required for safety testing, such as faults affecting both a small and large portion of the device at once.

## 4 Safety Testing with EMFI

As shown in the previous section, EMFI is capable of injecting faults similar to those that systems designed to operate in safety-critical environments should be

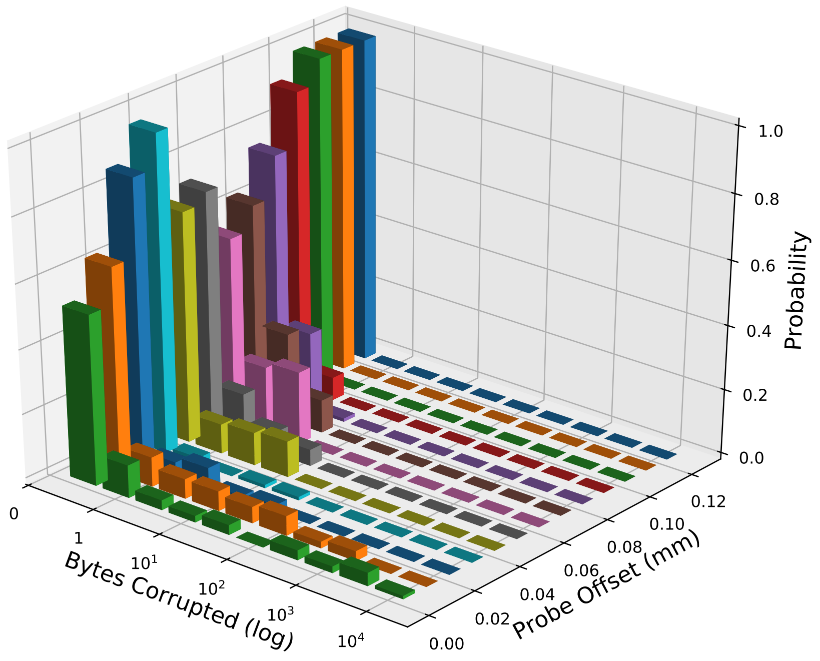


Fig. 4: Changing the distance from the SRAM surface causes a differing number of faults to occur.

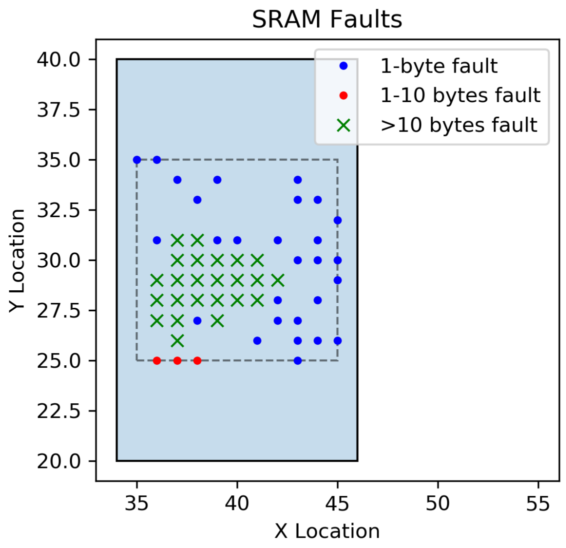


Fig. 5: SRAM Fault Results (200V), light blue is SRAM chip top, dashed area shows scan zone.

protected against. The focus of this work on automotive systems allows us to compare the results of systems with issues suspected of being caused by random bit flips.

The example target will be an ECU from a 2006 model year vehicle. This relatively old ECU was chosen due to existing public work discussing the potential for software failures to occur without tripping the expected fail-safe behaviour[7]. As most production ECUs would not be available for code inspection, the existence of expert witness testimony is particularly interesting as it allows us to understand what potential flaws the expert witness believed were present in the system[3].

Based on this expert witness testimony, certain memory corruption events would not be detected by the failsafe logic used in the ECU. This would allow unintended vehicle operation, and in particular the decoupling of the vehicle throttle from the requested user input. Notably this appeared to show a throttle getting "stuck" at one position, but not actually going to a full open position. This was validated with a test platform, where specific bits were flipped in a full ECU on a running vehicle. The testimony in the trial indicated that drivers who experienced the potential flaw first-hand described a throttle going to a full open (wide open throttle) position before getting stuck [1]. This suggested a different root cause than identified by code analysis, but as noted by the expert witness it would be difficult to cover all potential flaws.

More recent work on these devices has shown that voltage fluctuations would temporarily cause a wide open throttle, but this did not cause a stuck throttle [9]. This does however suggest some combination of a voltage fluctuation with memory corruption could cause the overall error condition, but leaves questions about the practicality of this – could the type of corruption from random particles or EMFI account for both of these?

Using EMFI, we can demonstrate a combination of these failure modes that result in the combination of both a throttle going to wide open along with sticking at one position. This will be done in a black box fashion, without requiring any modifications to the actual ECU or sensors. However, the authors must emphasize that the following safety testing is not performed on a full system – that is, the safety testing is not showing overall system failure, as other failsafes are not investigated in this example. More advanced analysis can be performed if the tester does have knowledge of the system under test, for example it is possible to compare memory and register dumps in order to understand where corruption is being inserted.

### 4.1 Test Bench

In order to perform a safety check, we need to operate the system inside of some normal bounds. In this case, a very simplistic engine/car simulator is built around the ECU to allow the system to operate normally.

This test bench is shown in Figure 6, which contains the following items:

1. The main ECU board we are testing.
2. The physical throttle body, wired to the ECU as normal.
3. The accelerator pedal sensor, wired to the ECU as normal.
4. An ignition switch (switches power on/off as a key would).
5. A simulator to provide cam & crank signals to the ECU.
6. A standard diagnostics reader to monitor the ECU datastream.
7. Probes to monitor the signal driving the throttle body.
8. A ChipSHOUTER to provide EMFI tooling.
9. A XYZ stage to allow scanning of the EMFI tool.

The current setup does not automatically monitor the outputs, but uses a human-in-the-loop to observe out of specification modes (such as the throttle position not matching the commanded position).

### 4.2 EMFI Results

While operating the device while performing EMFI, several failure modes were noted, including:

1. ECU resets and continues to operate.
2. ECU enters a fail-safe mode, such as reduced throttle opening.
3. The drive signals for the throttle body motor stopped operating normally.

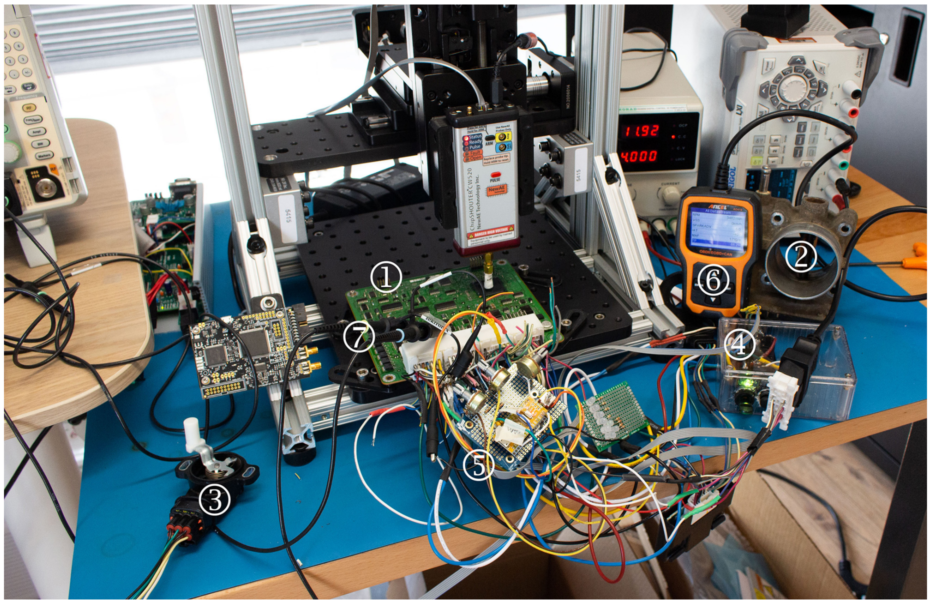


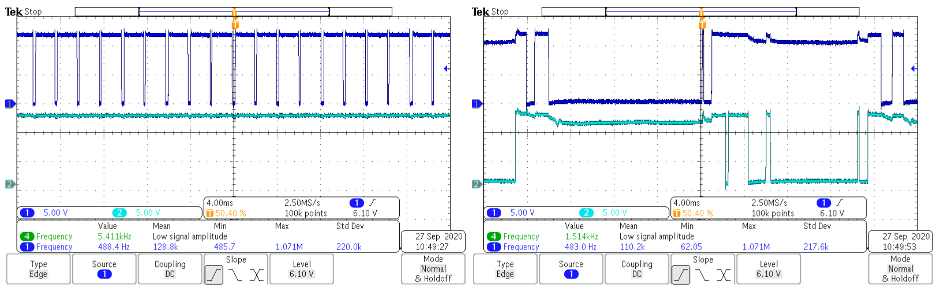
Fig. 6: The test bench showing: ① the ECU under test, ② the throttle body, ③ the position sensor, ④ the ignition switch, ⑤ sensor simulator, ⑥ OBD-II reader, and ⑦ scope probes on PWM signal.

The final failure mode is of the most interest, as it appeared that the control loop was otherwise closed (the position followed the pedal in general). The motor is driven with pulse width modulation (PWM) signals, with a comparison of the two PWM modes shown in Figure 7.

While the throttle body appeared to be maintaining the requested position, the throttle body now had a noticeable “chatter”. In addition, the power consumption jumped from the normal 1.6A to 3 – 5A (the current draw became less constant). It is assumed this was linked to the incorrect PWM waveform. This sudden jump in current consumption results in additional voltage drops on the ECU power rail, which also aligns with the previous work demonstrating wide open throttle condition during voltage fluctuations [9].

**Stuck Throttle Results** Once in the incorrect PWM mode, the throttle would eventually stick either fully closed or fully open, and the accelerator pedal sensor changes no longer have any impact on the throttle position. In this case the current draw increased further, and viewing the PWM waveforms it was clear the output signal was now constant. Once this mode was entered, a power cycle was required to exit this mode.

While in this mode, the ECU continued to provide ignition output signals that responded to changes in the cam & crank signals, and the OBD-II scan tool could continue to provide diagnostics information. A photo of the throttle


Tek Stop

1 5.00 V 2 5.00 V 4.00ms 2.50MS/s 100k points 1 6.10 V T 50.40 %

| | Value | Mean | Min | Max | Std Dev |
|---|---|---|---|---|---|
| 4 Frequency | 5.411kHz | Low signal amplitude | | | |
| 1 Frequency | 488.4 Hz | 128.8k | 485.7 | 1.071M | 220.0k |

27 Sep 2020 10:49:27

Type Edge | Source 1 | Coupling DC | Slope | Level 6.10 V | Mode Normal & Holdoff

Tek Stop

1 5.00 V 2 5.00 V 4.00ms 2.50MS/s 100k points 1 6.10 V T 50.40 %

| | Value | Mean | Min | Max | Std Dev |
|---|---|---|---|---|---|
| 4 Frequency | 1.514kHz | Low signal amplitude | | | |
| 1 Frequency | 483.0 Hz | 110.2k | 62.05 | 1.071M | 217.6k |

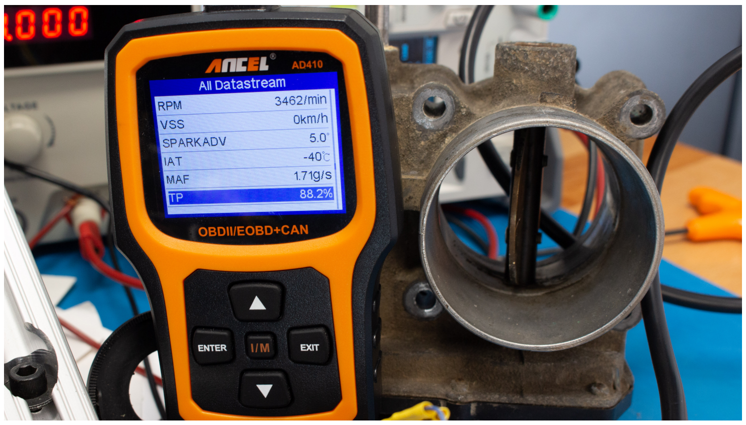
27 Sep 2020 10:49:53

Type Edge | Source 1 | Coupling DC | Slope | Level 6.10 V | Mode Normal & Holdoff

(a) Normal PWM Operation (b) Chattery PWM Operation

Fig. 7: PWM output for throttle body control before and after EMFI insertion.

stuck open is shown in Figure 8. Note the throttle is shown commanded to 88% opening here. During regular operation, the maximum throttle opening on the test bench is only 81%, thus the 88% throttle position commanded here appears to be even beyond regular operating values.



All Datastream

```
RPM        3462/min
VSS           0km/h
SPARKADV        5.0°
IAT            -40℃
MAF          1.71g/s
TP            88.2%
```

Fig. 8: The ECU after an EMFI based fault caused some error where the throttle is commanded to become fully open.

## 5 Conclusions

The possibility of memory corruption or other failures is a well-known problem in safety-critical design, with references to issues such as electromagnetic fields causing memory corruption going back well before the usage of these properties in security evaluations[11]. There is a large body of knowledge around the design of safety-critical systems such as for automotive devices, which often remains split from the knowledge gained and applied with a focus on security analysis of automotive systems.

Previous work has shown that EMFI is useful for security analysis, such as bypassing passwords or performing differential fault analysis. In this work we demonstrate that EMFI can also be used as part of a safety evaluation, by showing that the typical types of errors expected to be created randomly by charged particles can also be generated with EMFI.

In addition, the type of "black box" attack that is commonly applied with security evaluation has also been used to recreate an elusive bug in an automotive ECU. This bug was based on trial testimony [1] of an unintended acceleration case. While expert witnesses in that trial demonstrated that a throttle could stick at wide open, they did not demonstrate how a throttle could go from stuck to wide open [3]. More recent work has shown that voltage fluctuations can cause a momentary throttle opening [9], but it would seem unlikely that a memory error would occur by chance right when a voltage fluctuation occurs.

In this case, using EMFI we demonstrate that these two events are linked. Introducing memory corruption with EMFI causes the power consumption to become erratic, which in turn causes the voltage fluctuations previously identified. At the same time once the memory is corrupted the ECU appears to frequently crash in a "stuck throttle" condition.

This demonstrates that both safety and security engineering have the ability to learn useful tools and techniques from each other, and the type of black box testing common in security evaluations can be used to find bugs in safety critical systems.

## References

1. Transcript Of Morning Trial Proceedings: Case CJ-2008-7969 (2013), http://safetyresearch.net/Library/Koopman%2010-11-13%20a.m.PDF
2. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The Sorcerer's Apprentice Guide to Fault Attacks. Proceedings of the IEEE **94**(2), 370–382 (Feb 2006). https://doi.org/10.1109/JPROC.2005.862424
3. Barr, M.: 2005 Camry L4 Software Analysis, https://www.safetyresearch.net/Library/BarrSlides_FINAL_SCRUBBED.pdf
4. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Checking Cryptographic Protocols for Faults. In: Fumy, W. (ed.) Advances in Cryptology — EUROCRYPT '97. pp. 37–51. LNCS, Springer, Berlin, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0
5. Dusart, P., Letourneux, G., Vivolo, O.: Differential Fault Analysis on A.E.S. In: Zhou, J., Yung, M., Han, Y. (eds.) Applied Cryptography and Network Security. pp. 293–306. LNCS, Springer, Berlin, Heidelberg (2003)
6. JEDEC, S.S.T.A.: JESD89A: Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices (2006), https://www.jedec.org/system/files/docs/JESD89A.pdf
7. Koopman, P.: A Case Study of Toyota Unintended Acceleration and Software Safety (2014), https://ptolemy.berkeley.edu/projects/chess/pubs/1081/koopman14_toyota_ua_slides.pdf

8. Menu, A., Bhasin, S., Dutertre, J.M., Rigaud, J.B., Danger, J.L.: Precise Spatio-Temporal Electromagnetic Fault Injections on Data Transfers. In: 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 1–8 (Aug 2019). https://doi.org/10.1109/FDTC.2019.00009
9. Park, S., Choi, Y., Choi, W.: Experimental study for the reproduction of sudden unintended acceleration incidents. Forensic Science International **267**, 35–41 (Oct 2016). https://doi.org/10.1016/j.forsciint.2016.08.014, https://www.sciencedirect.com/science/article/pii/S0379073816303449
10. Pechenkin, A.A., Novikov, A.A., Novikova, M.M., Bobrovsky, D.V., Sorokoumov, G.S.: SEL and SEFI discrimination in Kintex-7 using focused laser irradiation. In: 2018 18th European Conference on Radiation and Its Effects on Components and Systems (RADECS). pp. 1–3 (Sep 2018). https://doi.org/10.1109/RADECS45761.2018.9328667, iSSN: 1609-0438
11. Sabo, J.D., Karp, J.A.: Radiation circumvention technique (Nov 1983), https://patents.google.com/patent/US4413327A/en
12. Savchenkov, D.V., Chumakov, A.I., Petrov, A.G., Pechenkin, A.A., Egorov, A.N., Mavritskiy, O.B., Yanenko, A.V.: Study of SEL and SEU in SRAM using different laser techniques. In: 2013 14th European Conference on Radiation and Its Effects on Components and Systems (RADECS). pp. 1–5 (Sep 2013). https://doi.org/10.1109/RADECS.2013.6937411
13. Schmidt, J.M., Hutter, M.: Optical and EM Fault-Attacks on CRT-based RSA : Concrete Results. In: Proceedings of Austrochip 2007, 15th Austrian Workshop on Microelectronics (2007)
14. Timmers, N., Spruyt, A., Witteman, M.: Controlling PC on ARM Using Fault Injection. In: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 25–35 (Aug 2016). https://doi.org/10.1109/FDTC.2016.18
15. Toulemont, J., Ouldei-Tebina, N., Galliere, J.M., Nouet, P., Bourbao, E., Maurine, P.: A Simple Protocol to Compare EMFI Platforms (2020), published: Cryptology ePrint Archive, Report 2020/1277
16. Wiersma, N., Pareja, R.: Safety != Security: On the Resilience of ASIL-D Certified Microcontrollers against Fault Injection Attacks. In: 2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 9–16 (Sep 2017). https://doi.org/10.1109/FDTC.2017.15