BADFET: Defeating Modern Secure Boot Using Second-Order Pulsed Electromagnetic Fault Injection

Ang Cui, Rick Housley
{a, r}@redballoonsecurity.com
Red Balloon Security

Abstract

Numerous Electromagnetic Fault Injection (EMFI) techniques have been used to attack FPGAs, ASICs, cryptographic devices, and microcontrollers. Unlike other classes of fault injection techniques, EMFI-based attacks can, in theory, be carried out non-invasively without requiring physical contact with the victim device. Prior research has demonstrated the viability of EMFI-based attacks against relatively simple, low-frequency, synchronous digital circuits. However, theoretical and practical constraints limit the range, degree of isolation and temporal resolution of existing EM injector hardware. These limitations, combined with the trend towards faster, denser and more complex digital circuits has made the application of many previously proposed EMFI techniques infeasible against modern computers and embedded devices.

This paper makes two contributions. First, we present a novel method of leveraging controlled electromagnetic pulses to attack modern computers using *second-order* effects of induced faults across multiple components of the target computer. Second, we present the design and implementation of BADFET: a low-cost, high-performance pulsed EMFI platform. We aim to share BADFET with the research community in order to democratize future EMFI research. Using these two contributions, we present a reliable and effective attack against a widely used TrustZone-based secure boot implementation on a multi-core 1Ghz+ ARM embedded system. Additionally, we disclose two novel vulnerabilities within a widely used implementation of TrustZone SMC in Appendix A.

1 Introduction

The use of external physical stimuli to induce faults within computing hardware has been well documented.

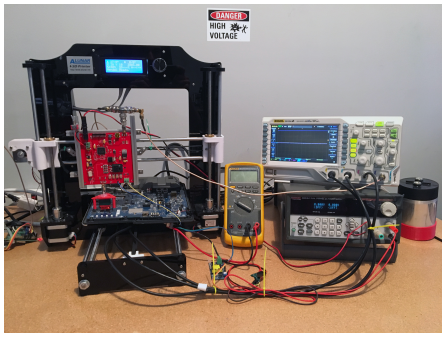

Figure 1: BADFET Platform and Victim Device

Generally, fault injection and glitching attacks aim to cause specific, non-destructive, reliably reproducible fault conditions in the target device in order to manipulate the computational behavior of the device. A wide gamut of fault injection modalities have been explored. Such modalities include the manipulation of system power[6], device clock[11], as well as the introduction of localized thermal variations[8], and optical stimulation[1, 14].

The use of electromagnetic fields to intentionally induce faults within digital computing devices have been well documented. Two general methods of inducing faults with electromagnetic radiation have been proposed: one using harmonic EM radiation, the other using transient EM radiation[13]. Harmonic-based EMFI is implemented with the continuous injection of sinusoidal EM waves. Transient EMFI attacks are implemented with the injection of independent high energy electromagnetic pulses. Transient EMFI attacks are generally highly sensitive to timing as the electromagnetic pulses must be induced during specific computational operations, such as the execution of individual CPU instructions or specific portions of cryptographic operations in an FPGA or ASIC.

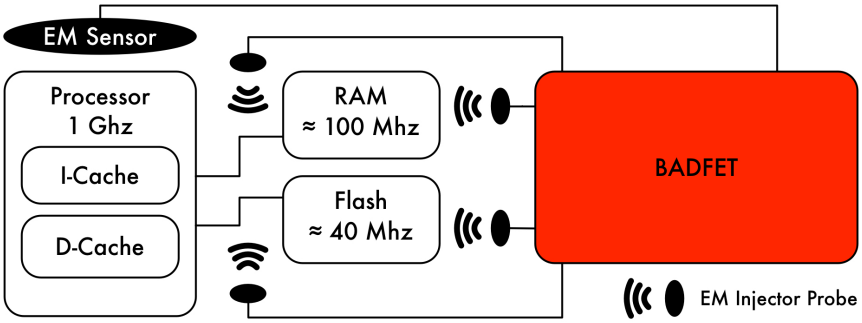

Figure 2: BADFET EMFI Attack Architecture

Electromagnetic Fault Injection (EMFI) has both advantages and disadvantages over the other above mentioned modalities. A notable advantage of electromagnetic fault injection (EMFI) is that it can be carried out non-invasively and without requiring physical contact with the target device. Most documented EMFI attacks do not require decapsulation of IC chips nor electrical connection to the device under attack. Furthermore, considering the permittivity and magnetic permeability of plastic enclosures commonly used on mobile and embedded device, numerous EMFI attacks can be leveraged against COTS devices non-invasively and in an air-gapped manner.

Unfortunately, the real-world effectiveness of EMFI attacks are constrained by a number of theoretical and practical limitations. Electromagnetic injectors must be engineered to induce sufficiently powerful EM field fluctuations in either transient or harmonic EM configurations. Practical implementations of such injectors typically discharge significant current through a conductive coil probe at both high voltage and high speed. EMFI attacks have been successfully demonstrated on low frequency, low density devices. However, the application of existing EMFI techniques against faster and denser digital circuits at greater distances requires EM injection hardware that must simultaneously have greater *temporal resolution*, greater *degree of isolation*, and greater *power*.

Temporal resolution: An increase in target device clock-rate requires an decrease in the pulse duration of the EM injector. We refer to this as the *temporal resolution* of the injector hardware. Intuitively, if the target device's clock-rate increases from 3Mhz to 30Mhz, the EM injector's temporal resolution will likely need to be increased proportionally. This is typically achieved by discharging the same amount of energy over a shorter period of time.

Spatial Resolution: An increase in target device density and complexity requires the EM injector to direct electromagnetic radiation at a smaller area of the target device without causing unwanted faults in neighboring circuitry. We refer to this as the degree of *spatial resolution* of the injector hardware.

Microprocessor	*Year*	*Spatial Resolution*	*Temporal Resolution*
8088	1973	0.11 mm	40μs
Pentium	1993	0.01 mm	17ns
Xeon Broadwell-E5	2016	0.06 μm	256ps

Table 1: A Survey of spatial and temporal resolutions needed to fault known processors

To place these challenges within some context, let us consider a number of examples and the required faulting resolution. Suppose a register requires 100 transistors on a chip die and the EMFI was to fault a read of the register value. The spatial resolution of the injector hardware would have to capable of faulting some subset of the 100 transistors. Through-out the past 50 years the number of transistors required for a register, or an operation, has remained relatively static while the distance between transistors has decreased by a factor of 1000. Therefore, both the spatial resolution necessary for faulting has also increased by a factor of 1000.

Distance and power: An increase in the distance between the target device and injector probe requires an increase of the EM radiation emitted by the injector hardware. In the near field, the increase in required energy can be approximated by the *inverse cube* relationship of [9].

In this paper, we aim to leverage EMFI against a modern multi-core 1GHz ARM-based VoIP phone; the Cisco 8861. Specifically, we aim to direct transient electromagnetic pulses towards the target device to non-invasively bypass its TrustZone-based secure boot process. This target device utilizes modern components (CPU, mem-

ory, NAND flash) that are significantly denser and have clock rates at least an *order of magnitude* faster than any hardware used in previous EMFI attacks. We posited that a direct application of previously proposed EMFI techniques will likely fail against this target device. Even if such EMFI techniques can be demonstrated on circuitry of this density and clock-rate, the cost of EM injection hardware of the necessary power, temporal resolution and spatial resolution will be prohibitively costly, and thus impractical. As an alternative, we present a novel EMFI technique that exploits *second-order* consequences of induced faults within modern computers and embedded systems. We discuss the theoretical motivation and a practical case-study of this second-order EMFI technique in Section 2. In order to reduce the cost of hardware required to conduct EMFI research, we designed and fabricated a capable EM injection device called BADFET. This device is capable of discharging in a low voltage mode (280 amps at 300 volts) or in a high-voltage (54 amps at 1100 volts), is microprocessor controlled, can be configured to carry out complex pulsed-excitation sequences with microsecond timing resolution and can be built for under $350. We present the hardware and software design of BADFET in Section 3. Experimental setup of second-order EMFI attacks against the Cisco 8861 is described in Section 4. Quantitative analysis of attack efficacy and BADFET performance is shown in Section 5. A brief survey of related work is shown in Section 6. Lastly, we present our concluding remarks in Section 7.

2 Second-Order EMFI Attack

EMFI-based attacks commonly exploit the effects of **first-order** induced faults. In other words, electromagnetic fields, generated in either transient or harmonic configurations are directed towards a specific component with the aim of introducing specific, deterministic and isolated faults in that same component. As Section 6 illustrates, first-order EMFI attacks have not been shown to be feasible against modern 1Ghz+ processors. We suspect that while such components are theoretically susceptible to first-order EMFI attacks, the level of temporal and spatial resolution required of the EM injection hardware is difficult to achieve reliably, even with prohibitively costly equipment.

We present an alternative application of EMFI that exploits the consequences of induced hardware faults of interdependent components and their associated interconnects within the same target device. This approach can significantly reduce the temporal and spatial resolution requirements of the EM injector hardware.

The main insight behind second-order EMFI attack techniques is a re-characterization of the device under attack. Instead of modeling the target device as a single computer, we model it as a collection of independent synchronous digital components. Such components often operate at different clock rates, communicate over network-like interconnects and utilize other components in timing sensitive ways.

Second-order EMFI attack techniques expose a spectrum of new attack vectors involving the interaction and interdependence of system components. Modeling the target device with this enhanced level of granularity allows the attacker to utilize multi-stage offensive strategies more commonly seen in traditional network-level exploitation and software concurrency attacks. Furthermore, second-order EMFI techniques can be used to attack system interconnects, such as i2c, SPI, and AXI (Advanced eXtensible Interface).

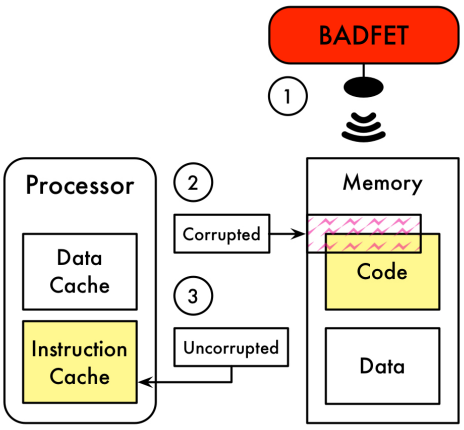

Figure 3: A Second-Order EMFI Attack

2.1 Data-Depedent Faults and the Instruction Cache

As a concrete example, we present a second-order EMFI attack that leverages non-specific data corruption in RAM to reliably cause software execution to enter into a data-dependent fault condition that is normally unreachable.

While non-specific RAM corruption indiscriminately effects both code and data, a sufficiently large working-set of the vulnerable bootloader *code* is stored in the CPU's instruction cache. As Figure 7 shows, the CPU is a separate component located approximately 19mm away from the DRAM chips. Thus, this EMFI attack can be carried out without significant spatial resolution requirement. Section 5 discusses the spatial and temporal resolution requirements of this attack in detail.

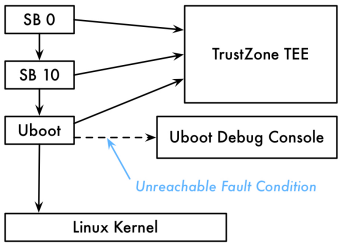

Figure 4: Exploiting vulnerabilities in the boot process

Label	*Function*	*Part*	*Speed*
A	DDR3L SDRAM	D9SFT	100 MHz
B	Processor	BCM11123	–
C	NAND Flash Memory	S34ML02G2	40 MHz

Table 2: EMFI Targets

The specific device under attack, shown in Figure 7, is a Cisco 8861 VoIP phone. A high-level diagram of the device's multi-stage process is shown in Figure 4, and is discussed in detail in Appendix A. uBoot is used in the secure boot process, and the typical uBoot debug CLI is included in the binary but configured to be unreachable via any user input. This secure boot bypass attack is carried out in the following two stages:

First Stage: First, the attacker reliably reaches the normally unreachable uBoot CLI using second-order EMFI. There are seven data-dependent fault conditions that cause uBoot to enter into this debug CLI state. Electromagnetic pulses are injected above the DRAM chips (Location B in Figure 7) a specific time after the device is powered on. A 300 Volt 10us pulse at 4.62 seconds past boot is emitted using the BADFET platform. This pulses indiscriminately corrupts data content within the DRAM chips as uBoot is executing. The application of EMFI causes a corruption of data in RAM, thus causing uBoot to enter a fault condition which executes the debug CLI. Note that the uBoot code content is also corrupted in RAM. However, since the necessary working-set of uBoot code, including the debug CLI, is stored in the processor's instruction cache, the corruption of in-memory code is *inconsequential*.

Second Stage: The attacker uses the uBoot CLI in order to load a second-stage binary that exploits a vulnerability within the device's TrustZone SMC implementation. The second stage exploit modifies TrustZone memory and allows the attacker to execute arbitrary code within the Trusted Execution Environment (TEE), thus bypassing secure boot. A full description of the SMC vulnerability is detailed in Appendix A.

3 BADFET Design and Implementation

3.1 Hardware Design

The BADFET system consists of multiple subsystems: an XYZ stage, a pulser, and an optional recording device.

The controller is responsible for reliably and accurately triggering the pulser from either an external trigger signal, or via its UART interface. It supports a SWD debug interface and UART shell for pulse configuration.

The pulser switches high-voltage and hig-current loads through a coil to emit EMFI. A bank of capacitors supplies the necessary power to emit EMFI through the probe tip. An SMA connector allows for probes to be swapped interchangably. A seperate high-voltage inverter is used to supply the capacitor bank with a low-current high-voltage source.

The XYZ stage is a modified 3d printer stage responsible for accurately positioning the pulser over the DUT. The controller interfaces with the stage and sends gcode commands to the stage for positioning.

3.1.1 EM Injector Design

The design and fabrication of precise electromagnetic injectors intended for high speed and high power pulsed excitation requires careful consideration. While a detailed discussion of electromagnetic injector probe design is out of the scope of this paper, we present an interative and empirical design process for the BADFET probe and invite the interested reader to see [12].

3.1.2 Pulser Design

The design of the BADFET EMFI pulser provided several unique challenges. Transient fault injection relies on the discharge of a capacitor through a coil to generate a transient EMP. The discharge of a capacitor is not itself a challenging problem; however, a design requirement was to develop a research platform capable of creating multiple transient pulses within microseconds of the first pulse. During the experimental design phase of the BADFET pulser subsystem, we identified a wealth of prior work relating to high-speed power switching and high-power capacitative discharge circuit design in the biological subfield of cellular electroperturbation[4] and the agricultural subfield of Pulsed Electric Field (PEF) food preservation[2].

Initially we devised an architecture consisting of several low-cost MOSFETs that would switch a bank of capacitors in series with our probe. Unfortunately, the Miller Effect [7] caused unwanted oscillations of the gate

driver circuitry and prevented this architecture from being as scalable as hoped. The BADFET switching architecture consists of an opto-isolated gate driver inline with a high current gate driver to provide proper isolation and the fast switching speeds necessary for EMFI generation.

We found that the PCB board design was extremely important for effective high-speed and high-current design. We required a highly symmetric design with an optimized discharge loop to limit trace inductance and resistance. Preventing generated EMF from re-triggering the BADFET pulse also proved challenging and required the use of an isolated gate driver. A large ground plane also assisted in assuaging this problem.

Another challenge we faced in the pulser design was the selection of appropriate current-limiting resistors for the switching circuit. We could not let the capacitor fully discharge through the selected MOSFETs as it would destroy the MOSFET internally and cause permanent latching. For this reason we selected low-inductance ceramic resistors. However, resistors are not explicitly related for high-current/high-voltage pulses. We selected "pulse-tolerant" resistors and used our empirical findings to select resistors that did not break down over time. We would also like to note the importance of a flyback diode aligned with the probe as the generated EM will quite easily render the device inoperable. See Appendix B for the BADFET schematic and Appendix C for the PCB layout.

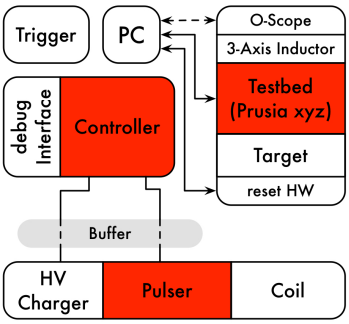

Figure 5: BADFET System Architecture

3.1.3 EM Injector Probe Design

Several design considerations were made during probe design:

- **Power Handling** - A large current and voltage flow through the probe tip for short durations. The probe needs to be able to handle these.
- **Internal Resistance** - The resistance of the probe needs to be minimal to not impede current flow and reduce magnetic flux.
- **Inductance** - Inductance in the coil causes ringing. This is an inevitable, but necessary consideration.
- **Shape** - The shape of the probe effects the directionality of the produced EM field and consequently effects what is isolated from the EM field.

Figure 6: Probe designs tested using BADFET

As Ordas et al. [13] noted in their paper, theoretical and simulated results from probe designs may vary in suprising ways. Instead of design from simulation, we take an iterative and empirical approach to our probe design process. We fabricated numerous probes of different designs, and used BADFET and a hand-made 3D inductor to create 4-dimensional recordings of each probe. See Section 5 for more detail about probe assessment. We then selected a probe with satisfactory power handling, directionality and radiation pattern that was appropriate for the size and shape of the RAM chip under attack.

Figure 6 illustrates three probe designs. Probe A is constructed using 13 turns of 54 Mil enamel coated solid copper wire. Probe B is constructed using 13 turns of the same copper wire with a cylindrical ferrite core with a diameter of 10mm and length of 25mm. Probe C is constructed using 8 turns of 25 Mil enamel coated copper wire, shaped into a 22mm x 17mm rectangle. A 3-dimentional electromagnetic radiation pattern is shown in Figure 8. All EM probes were terminated using female SMA connectors.

3.2 Software Design

The BADFET EMFI pulser is controlled by an on-board STM32 Microcontroller. This microcontroller controls pulse timing, charge voltage, and triggering setup. USB connectivity allows for simple configuration of these parameters. An onboard ADC measures capacitor voltage and disables charging when a set voltage is reached. Triggering can be achieved via USB or via the SMA triggering input. An input-to-output pulse delay can also be configured via the USB interface.

4 Experimental Setup

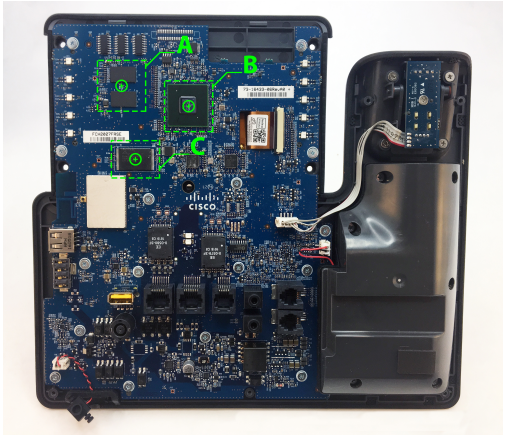

Figure 7: PCB of device under attack.

4.1 Attack Model

For the remainder of this paper, we adopt an attack model where the adversary is physically near the device under attack (DUA). Specifically, the attacker cannot make permanent modifications to the enclosure or the internal circuitry of the DUA. The attack must be carried out non-invasively, without causing permanent damage, and without leaving any detectable physical evidence. Furthermore, we assume that the DUA implements a typical TrustZone-based secure boot process. We also assume that the DUA's enclosure not electromagnetically shielded. Lastly, we assume that the attacker is familiar with the physical design of the DUA at a PCB level, and is familiar with the firmware running on the DUA. We assume that the attacker aims to disable or bypass the secure boot process.

As described in 3, the BADFET system consists of an XYZ stage, a recording device (e.g. oscilliscope), and a pulser. A Raspberry Pi 3+ is used as the primary device controller. It is connected via USB to a Prusia I3 V2 3D-printer platform. The BADFET python library supplied the necessary API for stage navigation and pulsing. A Raspberry PI GPIO is connected via SMA cable to the BADFET pulser input. The BADFET pulser is powered by a Keithely 2220-30-1 power supply. A high-voltage inverter is also powered by the Keithely supply and supplies the capacitor charging stage of the BADFET pulser. The high voltage inverter was set by hand to output 300V and verified with a Fluke 87 True RMS multimeter. The Raspberry Pi 3+ was used in conjunction with a relay board to cycle power to the DUT during testing.

For successful exploitation the Δtime from power-cycle to trigger pulse was determined to be 4.62 Seconds with an effective probe distance of 3mm.

4.2 Testing Model

Probe assessment required the Cisco 8861 IP phone to be removed from the BADFET stage. A custom 3-axis inductor was created for probe assessment. The recording device used for probe assessment was the Rigol DS104Z 4-Channel Oscilloscope. Three channels of the oscilloscope were connected to a custom 3-axis inductor placed on the XYZ stage. The final channel was connected to the external trigger for the BADFET pulser. The Oscilloscope was connected to the network to allow for waveform capture and post-processing. A four second delay was inserted between pulsing to ensure full charging of the capacitor bank.

5 Evaluation and Analysis

A number of experiments were devised to evaluate the performance of the EMFI injection device. A probe scanning procedure was devised to empirically test the performance of probe designs 5.0.1. A fault repeatability experiment was devised to determine confidence in our faulting technique. Finally, an experiment was devised to prove the attack is non-destructive.

5.0.1 Probe Scan

As described in section 3, we needed to gather empirical data from the sensors to test performance. The BADFET emitted EM pulses while stepping across the stage at increasing heights. The oscilloscope synchronized with the pulse input recorded waveforms from each inductor for each given pulse. These waveforms were captured over the network and saved for post processing. Post processing yielded plots of probe performance over a given area and height that allowed us to assess probe performance. Figure 8 is a sample of a spatial plot created from results acquired via this method.

5.0.2 Fault Repeatability

As XYZ found in their research, fault injection is not completely reliable, for this reason we performed a repeatability test to determine the fault confidence. For this specific test we placed the probe 3mm from the DUT. We pulsed the target DUT 100 times, recorded the serial console, and found we achieved a u-boot shell 72 of the 100 times.

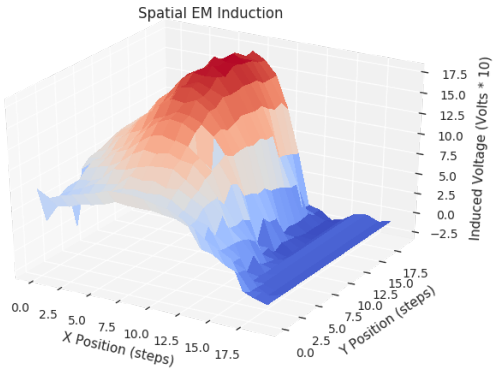

Figure 8: Sample spatial profile of an EMFI pulse

Platform	*Speed*	*Type*
ATmega128 [3]	3.57 MHz	MCU
Xilinx Spartan 3 [3]	–	FPGA
ARM Cortex-m3 [10]	56 MHz	MCU
Xilinx Spartan 7 [15]	100 MhZ	FPGA
SASEBO-G [5]	24 MHz	FPGA
Spartan 3-1000 [13]	max 100 Mhz	FPGA

Table 3: A Survey of EMFI Targets

6 Related Work

EMFI attacks have proven to be succesful on a number of select hardware platforms to weaken cryptographic protocols. In 2012 Dehaboui et al. [3] used trasient EMPs to fault an AES implementation on an ATmega128 and Xilinx Spartan 3 FPGA. Their attack was able to change individual bytes within the AES calculation in both the ATmega128 and FPGA. Similarly Hayashi et al. was able to utilize harmonic EMFI to fault AES calculations on a Side-Channel Attack Standard Evaluation Board (SASEBO). EMFI attacks have also proven to be useful in modifiying the control flow of processors. Moro et al. [10] were able to successfully modify the control flow of an ARM Cortex-M3 processor through both instruction modification and stepping. However, despite advances in EMFI technology, thus far EMFI attacks against modern gigahertz-speed are absent in literature. A survey of attacks and countermeasures suggests that 100 MHz is the state of the art in the field of EMFI attacks.

7 Conclusion

We demonstrate a reliable, non-invasive, remote attack that defeats a TrustZone-based secure boot implementation in a 1Ghz+ ARM-based VoIP phone using Electromagnetic Fault Injection (EMFI). This paper presents two contributions that enabled the reliable EMFI-based exploitation of a high clock-rate and high-density modern embedded system. First, we present *second-order* EMFI, a novel method of leveraging controlled electromagnetic pulses to exploit the consequences of induced hardware faults in interdependent components and their associated interconnects within the same target device. We demonstrate that second-order EMFI attacks can have significantly reduced temporal and spatial resolution requirements as compared to previously proposed EMFI techniques. Second, we present BADFET, a low-cost, high-performance pulsed EMFI platform that can be built under $350. We aim to share BADFET with the research community with the hope that it can help democratize future EMFI research by significantly reducing the cost of EM injection hardware. Additionally, we publicly disclose, two novel vulnerabilities within a widely used implementation of TrustZone SMC in Appendix A.

References

[1] AGOYAN, M., DUTERTRE, J.-M., MIRBAHA, A.-P., NACCACHE, D., RIBOTTA, A.-L., AND TRIA, A. Single-bit dfa using multiple-byte laser fault injection. In *Technologies for Homeland Security (HST), 2010 IEEE International Conference on* (2010), IEEE, pp. 113–119.

[2] CAMPBELL, D., HARPER, J., NATHAM, V., XIAO, F., AND SUNDARARAJAN, R. A compact high voltage nanosecond pulse generator. *Proc. ESA AME* (2008), 1–12.

[3] DEHBAOUI, A., DUTERTRE, J.-M., ROBISSON, B., AND TRIA, A. Electromagnetic transient faults injection on a hardware and a software implementations of aes. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on* (2012), IEEE, pp. 7–15.

[4] GUNDERSEN, M., KUTHI, A., GABRIELSSON, P., AND BEHRENDAND, M. Nanosecond pulse generator using a fast recovery diode. In *Power Modulator Symposium, 2004 and 2004 High-Voltage Workshop. Conference Record of the Twenty-Sixth International* (2004), IEEE, pp. 603–606.

[5] HAYASHI, Y.-I., HOMMA, N., MIZUKI, T., AOKI, T., AND SONE, H. Precisely timed iemi fault injection synchronized with em information leakage. In *Electromagnetic Compatibility (EMC), 2014 IEEE International Symposium on* (2014), IEEE, pp. 738–742.

[6] KARAKLAJIĆ, D., SCHMIDT, J.-M., AND VERBAUWHEDE, I. Hardware designer's guide to fault attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 21*, 12 (2013), 2295–2306.

[7] KAZIMIERCZUK, M. K. Class d voltage-switching mosfet power amplifier. In *IEE Proceedings B (Electric Power Applications)* (1991), vol. 138, IET, pp. 285–296.

[8] LEMKE, K. Embedded security: Physical protection against tampering attacks. In *Embedded Security in Cars*. Springer, 2006, pp. 207–217.

[9] MICHAUD, A. On the magnetostatic inverse cube law and magnetic monopoles.

[10] MORO, N., DEHBAOUI, A., HEYDEMANN, K., ROBISSON, B., AND ENCRENAZ, E. Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on* (2013), IEEE, pp. 77–88.

[11] O'FLYNN, C., AND CHEN, Z. D. Chipwhisperer: An open-source platform for hardware embedded security research. In *International Workshop on Constructive Side-Channel Analysis and Secure Design* (2014), Springer, pp. 243–260.

[12] OMAROUAYACHE, R., RAOULT, J., JARRIX, S., CHUSSEAU, L., AND MAURINE, P. Magnetic microprobe design for em fault attack. In *Electromagnetic Compatibility (EMC EUROPE), 2013 International Symposium on* (2013), IEEE, pp. 949–954.

[13] ORDAS, S., GUILLAUME-SAGE, L., AND MAURINE, P. Electromagnetic fault injection: the curse of flip-flops. *Journal of Cryptographic Engineering* (2016), 1–15.

[14] SKOROBOGATOV, S. P., AND ANDERSON, R. J. Optical fault induction attacks. In *International workshop on cryptographic hardware and embedded systems* (2002), Springer, pp. 2–12.

[15] ZUSSA, L., DEHBAOUI, A., TOBICH, K., DUTERTRE, J.-M., MAURINE, P., GUILLAUME-SAGE, L., CLEDIERE, J., AND TRIA, A. Efficiency of a glitch detector against electromagnetic fault injection. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014* (2014), IEEE, pp. 1–6.

8 Appendix A: Broadcom TrustZone Vulnerability

Multiple vulnerabilities exist within the Broadcom Trusted Execution Environment (TEE) implementation utilized by the BCM11123 SoC. The vulnerabilities described in this section allow an attacker from the non-secure world to reliably read and write TrustZone protected memory. Furthermore these vulnerabilities allow for reliable execution of arbitrary code in secure mode. This vulnerability was identified in the following Cisco 8861 IP Phone using firmware version *fbi88xx.BE-01-009.sbn*.

Specifically, the vulnerabilities are found within TEE service ID *0xE00013*, which implements RSA decryption. First, an invocation of *memcpy* without proper validation of user-supplied input allows the attacker to perform arbitrary reads of TrustZone-protected memory from the non-secure world. Second, a separate input validation vulnerability in the API's signature verification code allows the attacker to perform arbitrary writes to TrustZone-protected memory. The combination of these two vulnerabilities allows for execution of arbitrary code within the secure world. The following section describes the vulnerability discovery process.

```
u-boot> help rsa_decrypt
rsa_decrypt - call ssapi to decrypt rsa encrypted data

Usage:
rsa_decrypt Syntax: inBufAddr inBufSize keyBufAddr keyBufSize mod_len exp
 mod_len is 1/2 of modulus length in bytes, exponent=0x%x
Notes: the last 264 bytes of ABI image is the secondary key which
could be passed to this command
Example:
tftp 0x84000000 sig.bin
tftp 0x84000100 rsa_modulus.bin
rsa_decrypt 0x84000000 0x100 0x84000100 0x100 0x80 0x10001
when key includes mod_len and exp, they could be set to 0
rsa_decrypt 0x84000000 0x100 0x84000100 0x108 0x0 0
when using internal root or secondary key,
all the rest 3 parameters could be set to 0

using root key
rsa_decrypt 0x84000000 0x100 0 0 0x0 0
using secondary key
rsa_decrypt 0x84000000 0x100 1 0 0x0 0

u-boot> rsa_decrypt 0x84000000 0x100 0 0 0
input buffer @0x00000000 size=0x84000000
key buffer @0x00000100 size=0x0
mod_len = 0x00, exponent=0x0
ssapi_rsa_public_decrypt() return 9, decryptedDataSize=32
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
u-boot> rsa_decrypt
=== test RSA decrypt using hardcoded data ===
The decrypted signature matches.
ssapi_rsa_public_decrypt() return 1, decryptedDataSize=32
3d 54 68 65 20 79 65 6c 6c 6f 77 73 74 6f 6e 65 20 6e 61 74 69 6f 6e
61 6c 20 70 61 72 6b 3d 0a
```

Listing 1: Help for rsa_decrypt command

8.0.1 Vulnerability Discovery and Details

The Broadcom TEE vulnerabilities were identified on a Cisco 8861 IP phone. First, physical glitching of the phone's NAND flash chip during boot was used to interrupt the device's boot process, giving the attacker to the U-boot console before the phone's Linux kernel is loaded and executed. Although the U-boot console allows the attacker to read, write and execute arbitrary code in memory, the loader executes in non-secure mode, and is thus unable to access TrustZone-protected memory.

The U-boot console allows the attacker to invoke a device-specific function, *rsa_decrypt*, with arbitrary user-supplied parameters. The *rsa_decrypt* function likely to be remnant from a testing environment. This function is used to invoke the TEE service ID *0xE00013*, causing the processor to perform a context switch, transitioning from non-secure mode into secure-mode. During this process, the arbitrary user-supplied parameters are copied into TrustZone-protected memory, passed onto the Broadcom implementation of the RSA decryption function and executed within secure-mode.

The U-boot *rsa_decyrpt* command takes a series of arguments, including a flag to use default root and secondary keys, a memory address for user-provided public keys, and a modulus and exponent for RSA decryption. The output of the *rsa_decyrpt* command is shown in Listing 1.

A default validation test of the API is provided by this command, the disassembly of which can be found in Figure 10. The validation test function checks input arguments before passing them to the inner function _ssapi_public_decrypt. In addition to user-provided arguments, the outer test function passes hardcoded values including a pointer address to a malloc'd buffer and it's size as a "return size'.'

Figure 11 contains the code that actually branches into the secure world routine. We note, before the secure world context switch, the service ID value 0xE00013, used to select the requested crypto routine(s) from the SMC. Documentation for the SMC requests on this platform, known as *BCM_KONA*, can be found in the linux kernel source tree.

Two buffer sizes can be passed as arguments to the wrapper function which invokes the *_ssapi_public_decrypt* function: the size of the malloc'd buffer and a target size of the block that is being decrypted. The RSA-2048 routine can encrypt/decrypt up to a maximum of 0x100 (256) bytes, anything larger triggers the appropriate exception in the SMC exit status code. While it appeared that TEE service ID *0xE00013* was properly performing input validation, a closer examination revealed a vulnerability when the *_ssapi_public_decrypt* function is used to perform signature verification.

Signatures generated by the *_ssapi_public_decrypt* consist of a 256-byte block containing a SHA256 hash and PKCS#v1.5 padding.

The signature verification routine within secure world

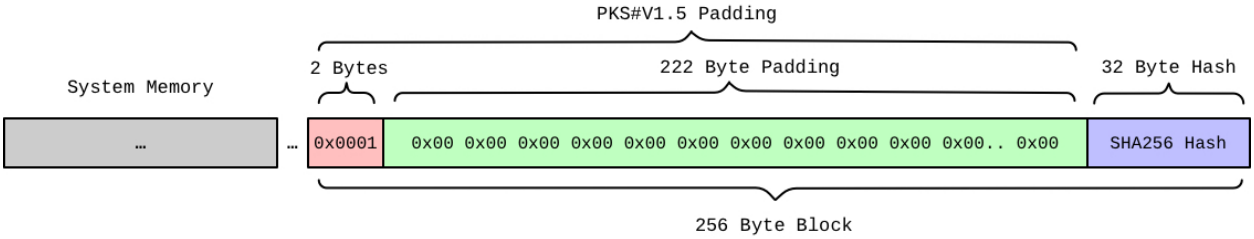

Figure 9: Exploitable Secure World Buffer

```
8E0177A4 LDR     R0, =aTestRsaDecrypt ; "=== test RSA decrypt using hardcoded da"...
8E0177A8 BL      SB2_PRINTF
8E0177AC LDR     R6, =SSAPI_KEY
8E0177B0 LDR     R3, =0x10001
8E0177B4 ADD     R1, SP, #0x58+returnSize
8E0177B8 MOV     R2, #0x80 ; '€'
8E0177BC MOV     R0, MALLOC_BUFFER
8E0177C0 STR     R6, [SP]
8E0177C4 STMFA   SP, {R2,R3}
8E0177C8 SUB     R2, R6, #0x100
8E0177CC MOV     R3, #0x100
8E0177D0 BL      _ssapi_public_decrypt ; r1 = returnSize
8E0177D0                               ; r2 = targetAddr
8E0177D0                               ; r3 = targetSize
8E0177D0                               ;
8E0177D4 ADD     R1, R6, #0x100
8E0177D8 LDR     R2, [SP,#0x58+returnSize]
8E0177DC MOV     R4, R0
8E0177E0 MOV     R0, MALLOC_BUFFER ; For this, r1=expected,
8E0177E4 BL      _strcmp ; r0 = s1
8E0177E4                 ; r1 = s2
8E0177E4                 ; r2 = Size
8E0177E4                 ;
8E0177E8 CMP     R0, #0
8E0177EC LDREQ   R0, =aTheDecryptedSi ; "The decrypted signature matches.\n"
8E0177F0 LDRNE   R0, =aErrorTheDecryp ; "ERROR: The decrypted signature does not"...
8E0177F4 BL      SB2_PRINTF
8E0177F8 B       loc_8E01791C
```

Figure 10: Wrapper for TZAPI function call

```
8E000C14
8E000C14 loc_8E000C14
8E000C14 LDR             R3, [SP,#0x40+Modulus]
8E000C18 MOV             R1, #0xF
8E000C1C STMEA           SP, {targetAddr,targetSize}
8E000C20 MOV             R2, R4
8E000C24 STR             ssapiKey, [SP,#8]
8E000C28 STR             R3, [SP,#0x40+__callMod]
8E000C2C LDR             R3, [SP,#0x48]
8E000C30 LDR             R0, =0xE000013
8E000C34 STR             R3, [SP,#0x10]
8E000C38 LDR             R3, =prtTo1004101_smcapi
8E000C3C LDR             R12, [R3]
8E000C40 ADD             R3, SP, #0x1C
8E000C44 BLX             R12
8E000C48 ADD             SP, SP, #0x2C
8E000C4C LDMFD           SP!, {R4-targetSize,PC}
8E000C4C ; End of function _ssapi_public_decrypt
8E000C4C
```

Figure 11: Call to SMC setup function

does not use the full 256-byte block for verification, but rather expects a default input "return size" of 32 bytes: the length of the SHA256 buffer (See Figure 9). The decryption function is passed the to-be-decrypted block containing both the padding and hash. It copies from the end of buffer, towards the beginning of the buffer the "return size" number of bytes. Instead of using the pre-allocated malloc buffer, or the SHA256 32-byte return size, attackers can abuse this by writing a wrapper to supply an arbitrary value. If the value supplied is larger than the 256 buffer size, secure world memory past the malloc'd signature block will be supplied into the user-supplied buffer. This memory disclosure can be used to find stack call signatures and can provide enough information to map the location of critical sections of code within the protected secure world address space.

An additional vulnerability is required in to leverage the knowledge of these critical code sections and to gain arbitrary code execution within the secure world. The previously described decryption routine also does not verify the return buffer address. This allows attackers to overwrite an address only accessible within the secure world. Using the same _ssapi_public_decrypt function, attackers can read both read and write into secure world code and data allowing for arbitrary code execution within the secure world.

8.0.2 PoC: Returning to U-Boot in secure mode

Recalling that U-Boot can operate in both secure or non-secure worlds, a custom shellcode payload that jumps to the beginning of the second stage bootloader from the secure mode will result in privilege escalation to the secure world.

```
u-boot> mw.l 0x8e007fb0 0x8fe81e2c
u-boot> mw.l 0x8e007fb4 0x00010001
u-boot> mw.l 0x8e007fb8 0x0e000013
u-boot>
u-boot> go 0x8e007eb0
## Starting application at 0x8E007EB0 ...


U-Boot 2011.06 (Dec 01 2014 - 14:17:24 CST) - bcm11125_be4_nand

...
0x35004020=0x00000022 0x35004024=0x0420c006
0x35004100=0x00000000 0x35001f18=0x00000006
Running in secure mode. <============   # We are now in secure mode
Card did not respond to voltage select!
MMC init failed
Auto-detected LDO daughtercard
...

u-boot> md.l 0x0
00000000: e59ff018 e59ff018 e59ff018 e59ff018
00000010: e59ff018 e7ffffff e59ff014 e59ff014
00000020: 00011aa8 000117c0 000117d0 000117e0
00000030: 000117f0 00011800 0001181c 00000000
00000040: 00000000 00000000 00000000 00000000
00000050: e9a5e225 fa000000 fa000022 e890a00a
```

Listing 2: Escaping the TEE Sandbox

8.1 Appendix B: BADFET Schematic

NOTE: BADFET is an experimental design and uses potentialy lethal voltages and current. Please replicate at your own risk.

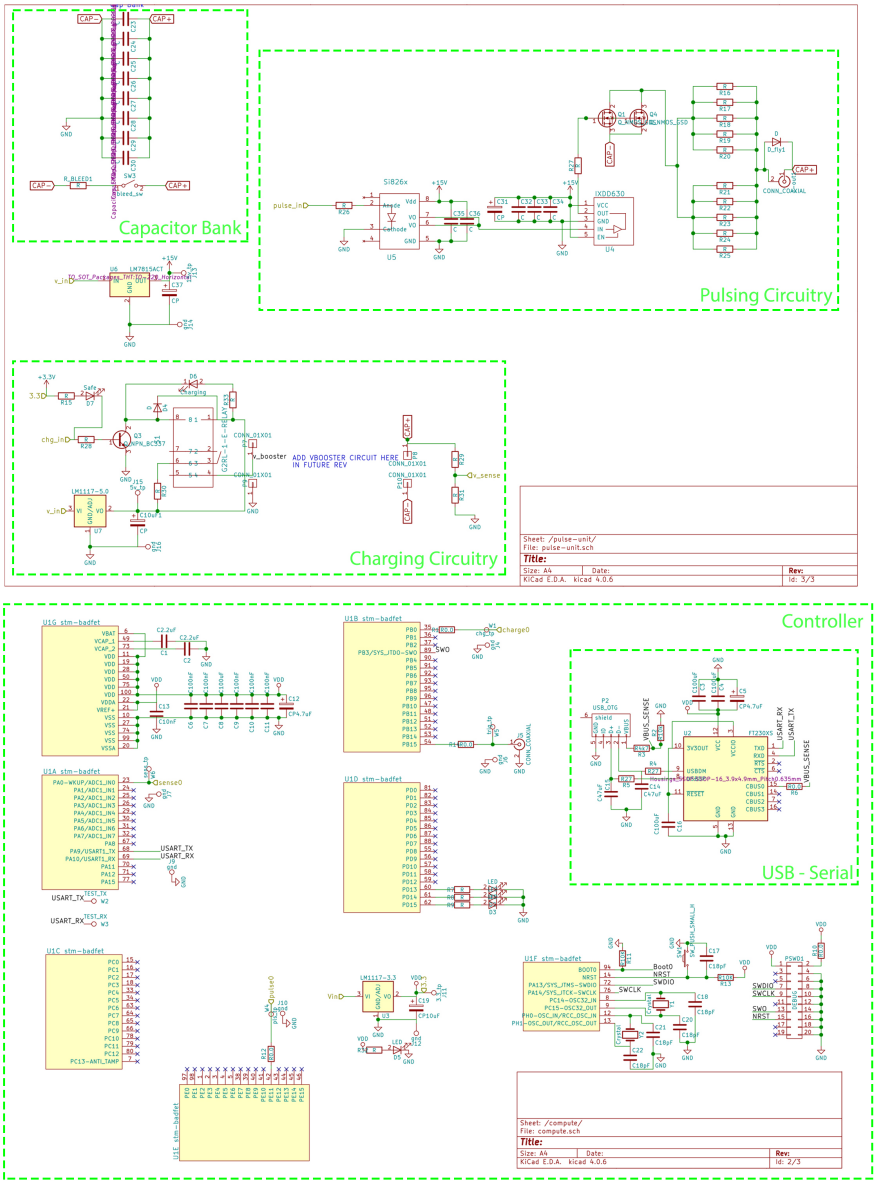

Figure 12: BADFET Schematic

8.2 Appendix C: BADFET PCB Layout

NOTE: BADFET is an experimental design and uses potentialy lethal voltages and current. Please replicate at your own risk.

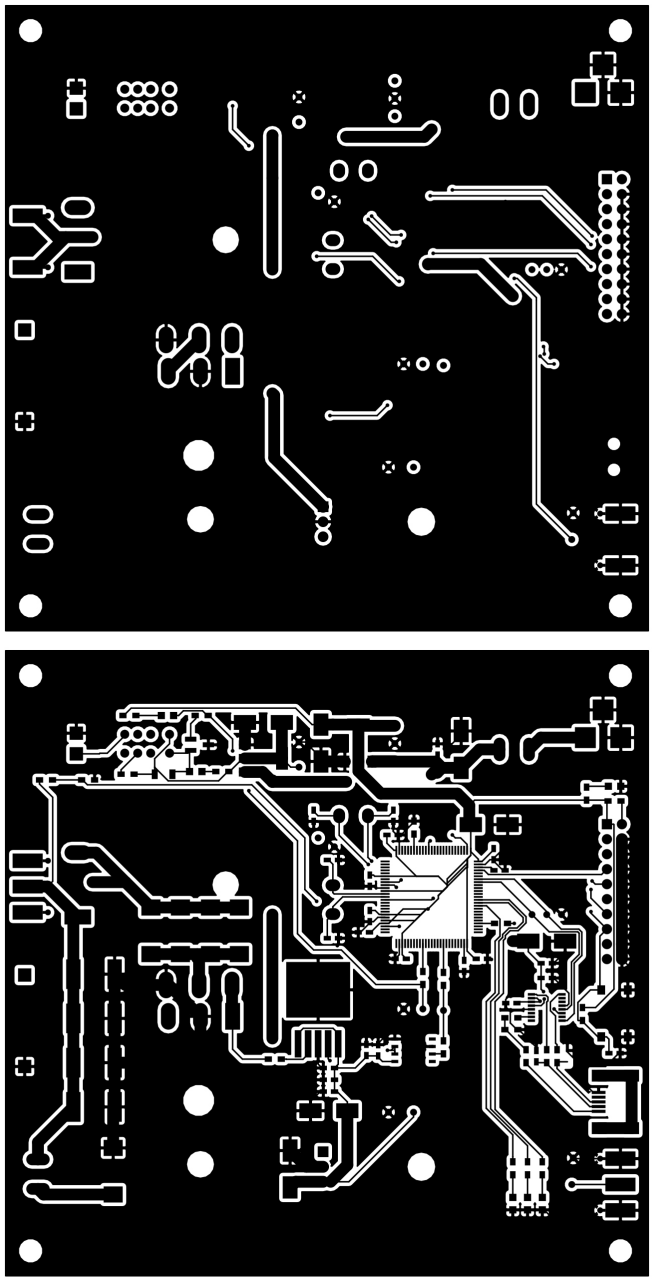

Figure 13: BADFET PCB